ISSN : 1411-1624
e-ISSN: 2621-8089

DESAIN *NON-PLAYER CHARACTER* PERMAINAN *TIC-TAC-TOE* DENGAN ALGORITMA MINIMAX

Gunadi Emanuel[1], R. Kristoforus J. Bendi[2], Arieffianto[3]
Universitas Katolik Musi Charitas[1,2,3]
Jalan Bangau No. 60 Palembang
Sur-el : gunadi@ukmc.ac.id[1], kristojb@ukmc.ac.id[2], arief_arieffianto@yahoo.com[3]

Abstract : *Tic-Tac-Toe is one of the board games. It is played by filling the columns on the board with X or O in such a way as to form parallel lines vertically, horizontally and diagonally. This study aims to design Non-Player Characters (NPC) in the tic-tac-toe game with the minimax algorithm. The Tic-tac-toe game will be designed with two game modes: easy and minimum random modes. While in minimax NPC mode will determine the best step. The game development process of the tic-tac-toe application is based on a linear sequence process model. In the analysis phase, the NPC will be designed based on the concept of minimax. Software modeling was designed using Unified Modeling Language (UML), and coded with Visual Basic programming. Our tests show that NPCs with the Minimax algorithm can work well.*

Keywords: *tic-tac-toe, minimax, non-player character*

Abstrak : *Tic-Tac-Toe adalah salah satu permainan papan. Itu dimainkan dengan mengisi kolom di papan tulis dengan X atau O sedemikian rupa sehingga membentuk garis paralel secara vertikal, horizontal dan diagonal. Penelitian ini bertujuan untuk merancang Karakter Non-Pemain (NPC) pada game tic-tac-toe dengan algoritma minimax. Game Tic-tac-toe akan dirancang dengan dua mode permainan: mode acak mudah dan minimum. Sedangkan pada mode minimax NPC akan menentukan langkah terbaik. Proses permainan pengembangan aplikasi tic-tac-toe didasarkan pada model proses linear urutan. Pada tahap analisis, NPC akan dirancang berdasarkan konsep minimax. Pemodelan perangkat lunak dirancang dengan menggunakan Unified Modeling Language (UML), dan diberi kode dengan pemrograman Visual Basic. Pengujian kami menunjukkan NPC dengan algoritma minimax dapat bekerja dengan baik.*

Kata kunci*: tic-tac-toe, minimax, non-player character*

1. PENDAHULUAN

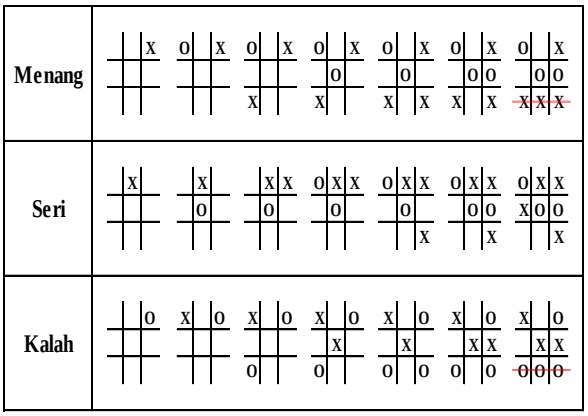
Perkembangan ilmu pengetahuan dan teknologi kian pesat. Salah satu jenis perkembangan ilmu pengetahuan dan teknologi adalah perkembangan pada dunia *game* [1]. Bermain *game* merupakan salah satu aktifitas yang sangat disukai oleh sebagian besar masyarakat. Dengan berkembangnya teknologi sekarang ini, game-game ini tidak hanya dapat kita jumpai pada kehidupan nyata, tapi juga dapat kita jumpai didalam permainan berbasis komputer. Salah satu jenis permainan komputer yaitu *board game* [2,3].

Tic-tac-toe merupakan permainan berjenis *board-game* berukuran matriks bujur sangkar. Diandaikan papan permainan berukuran 3 x 3, maka setiap pemain harus meletakkan bidak-bidanya pada sel-sel papan sedemikian hingga tiga bidak secara berturutan membentuk garis horisontal, vertikal, ataupun juga diagonal. Permainan ini harus dimainkan oleh 2 orang pemain [4,5,6]. Pada *Tic-tac-toe* berbasis komputer, salah satu pemain dapat diperankan

ISSN : 1411-1624
e-ISSN: 2621-8089

oleh komputer [3]. Penelitian ini bertujuan untuk mendesain *non-player character* (NPC) yang akan bertindak sebagai pemain yang diperankan oleh komputer.

Permainan papan (*board game*) merupakan jenis permainan yang dimainkan di atas papan dimana bidak diletakkan di atasnya. Bidak dalam permainan ini mampu berpindah tempat ataupun dimakan oleh bidak lawan, sesuai dengan peraturan yang berlaku pada permainan. Pada umumnya permainan ini berbasis strategi, keberuntungan ataupun gabungan dari kedua hal tersebut [7]. Permainan catur misalnya, untuk dapat memenangkannya diperlukan strategi dalam menempatkan bidak-bidaknya. Tetapi, permainan ular tangga, hanya mengandalkan keberuntungan angka dadu yang muncul. Sedangkan *Tic-Tac-Toe* digolongkan sebagai permainan papan yang melibatkan strategi untuk mencapai kemenangan.



Gambar 1. Kondisi akhir permainan untuk bidak X

Papan permainan *tic-tac-toe* berupa sel-sel matriks berukuran n x n. Pada kondisi awal permainan, papan permainan kosong. Setelah itu pemain saling bergantian meletakkan bidaknya pada sel-sel papan permainan Kondisi akhir permainan adalah: menang, seri, dan kalah (Gambar 1). Misalkan papan permainan berukuran 3 x 3. Kondisi menang (untuk bidak X) akan tercapai apabila menjadi pemain pertama yang dapat meletakkan 3 bidak (X) dalam posisi sejajar (vertikal, horisontal, diagonal). Kondisi seri akan tercapai apabila hingga seluruh sel-sel tersisi, kedua pemain tidak dapat meletakkan 3 bidaknya pada posisi sejajar. Sedangkan kondisi kalah (untuk bidak X) tercapai apabila pemain lawan yang pertama kali meletakkan 3 bidak (O) pada posisi sejajar [3].

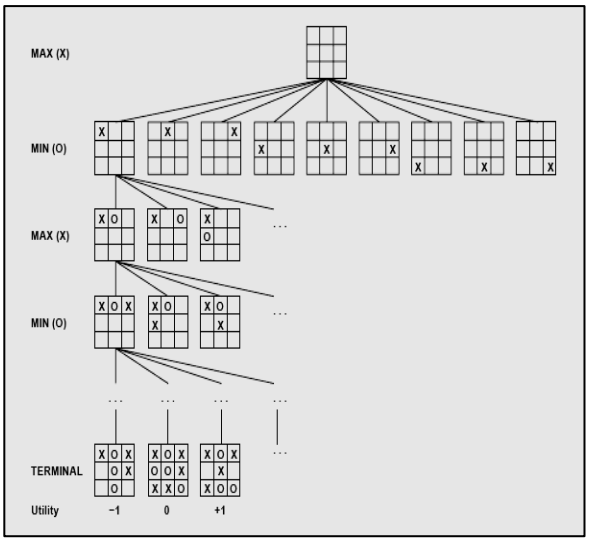
Agar komputer dapat berperan sebagai pemain, diperlukan sebuah algoritma yang berfungsi untuk mencari langkah terbaik dari situasi sekarang. Salah satu algoritma yang paling banyak digunakan adalah algoritma Minimax. Algoritma ini sangat cocok untuk permainan dengan dua pemain. Algoritma ini digunakan untuk memilih langkah terbaik yang diberikan, dimana kedua pemain saling berusaha untuk memenangkan permainan [7].

Minimax merupakan salah satu teknik permainan yang terkenal. Minimax menggunakan teknik pencarian *depth-first search* dengan kedalaman terbatas, dan fungsi evaluasi yang digunakan adalah fungsi evaluasi statis, dengan mengasumsikan bahwa lawan akan membuat langkah terbaiknya yang dapat dilakukan, algoritma minimax cocok digunakan untuk permainan catur, *Othello*, *checkers*, dan *Tic-Tac-Toe* [4,8].

Penerapan algoritma Minimax dalam tic-tac-toe dibuat berdasarkan prosedur Minimax untuk mendapatkan langkah terbaik dari posisi yang ada. Setiap posisi memiliki nilai yang dapat dihasilkan dari langkah terbaik, dengan

ISSN : 1411-1624
e-ISSN: 2621-8089

berasumsi bahwa pemain akan selalu memaksimalkan nilai, ketika lawan akan mencoba untuk meminimalkannya, sehingga akan menghasilkan pohon permainan yang berisi semua kemungkinan tersebut (Gambar 2). Keuntungan penggunaan algoritma Minimax adalah mampu menganalisis semua kemungkinan posisi permainan untuk menghasilkan keputusan terbaik dengan mencari langkah yang akan membuat lawan mengalami kerugian. Fungsi evaluasi yang digunakan adalah fungsi evaluasi statis dengan asumsi lawan akan melakukan langkah terbaik yang mungkin. Pada Minimax dikenal adanya istilah gerakan pemain MAX melawan pemain MIN [9].

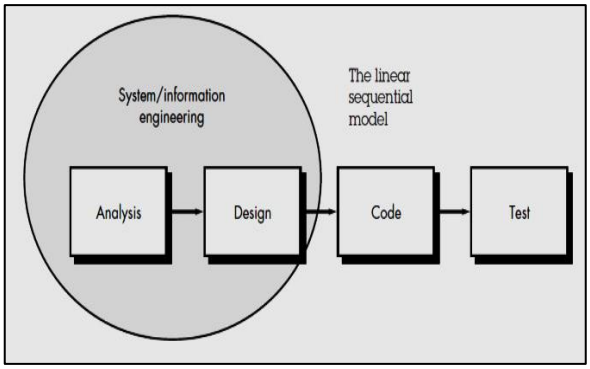


Gambar 2. Pohon permainan tic-tac-toe

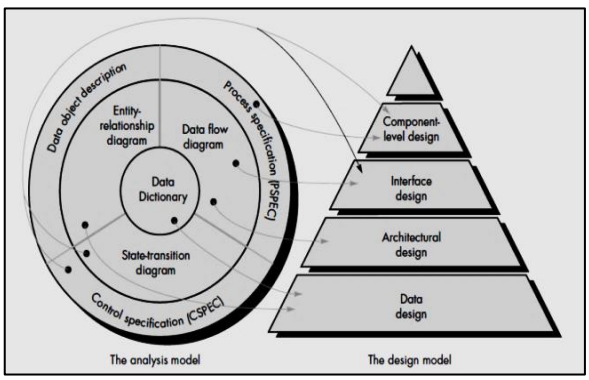
Pohon (*tree*) merupakan graf yang masing-masing *node*-nya (kecuali *root*) hanya memiliki satu induk (*parent*), dengan kata lain tidak memiliki *cycle*. *Node* yang tidak memiliki *child* disebut *terminal node*. Metode pencarian yang umumnya digunakan pada pohon pencarian adalah *breadth-first search* (BFS) atau *depth-first search* (DFS). Karena ruang pencarian pohon permainan *Tic-Tac-Toe* cukup besar, maka penggunaan metode BFS dirasa tidak tepat, sebab metode ini membutuhkan kapasitas memori yang besar. Berbeda dengan metode DFS yang hanya membutuhkan memori relatif kecil.

2. METODOLOGI PENELITIAN

Metode penelitian ini mengacu pada model proses sekuensial linier [10]. Dalam model ini (Gambar 3), pengembangan perangkat lunak dilakukan secara sistematik dan linier, yang dimulai dari aktivitas analisis, desain, pengkodean dan pengujian perangkat lunak. Aktivitas analisis dan desain merupakan bagian dari aktivitas rekayasa sistem. Setiap model analisis yang dihasilkan akan diterjemahkan ke bentuk desain yang bersesuaian (Gambar 4).



Gambar 3. Model sekuensial linier



Gambar 4. Hubungan tahap analisis dan desain

ISSN : 1411-1624
e-ISSN: 2621-8089

Secara umum tahapan-tahapan dalam penelitian ini mengacu pada tahapan dalam model sekuensial linier seperti yang diuraikan pada Tabel 1.

Tabel 1. Tahapan penelitian

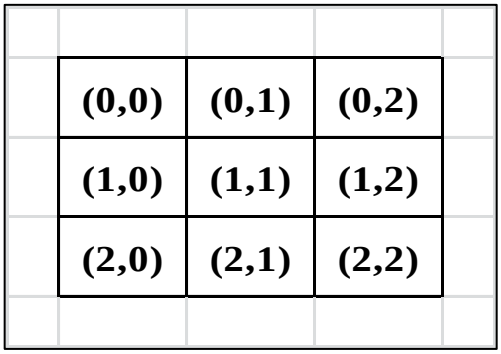
Tahap	*Kegiatan*
Analysis	Menentukan aturan permaian Analisis pergerakan NPC
Design	Desain algoritma minmax Desain perangkat lunak dengan menggunakan *unified modeling language*
Code	Membangun aplikasi dengan bahasa pemrograman Visual Basic
Test	Melakukan uji perangkat lunak Melakukan uji statistik

3 HASIL DAN PEMBAHASAN

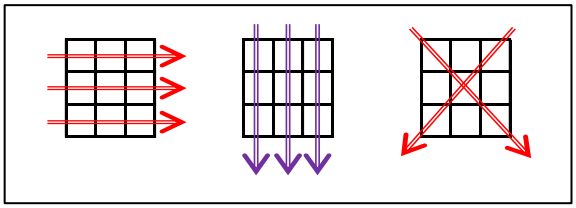
3.1 Aturan Permainan

Aturan permainan *tic-tac-toe* yang diterapkan sebagai berikut.

a. Besar papan permainan berukuran 3 x 3 dengan deret kemenangan 3 bidak.
b. Pemain dalam permainan ini dibatasi untuk satu lawan satu, dilakukan bergiliran antara manusia dan komputer (NPC).
c. Pemain manusia (*player*) akan dihalangi oleh NPC untuk mencapai tujuan (goal).
d. Untuk memberikan langkah, setiap pemain harus mengisi bidak dengan simbol pemain masing-masing, biasanya X atau O.
e. Setiap pemain hanya mempunyai satu kali kesempatan pada setiap giliran.
f. Bidak yang sudah terisi tidak bisa ditimpa oleh bidak lain pada langkah berikutnya.
g. Langkah yang sudah diambil tidak dapat dibatalkan atau diganti dengan langkah yang lain.
h. Tujuan dari permainan ini adalah untuk mendapatkan deret dengan tiga 3 bidak yang sama secara horizontal, vertikal atau diagonal.
i. Pemenang ditentukan oleh pemain yang pertama kali menyusun deret tersebut.
j. Permainan akan dihentikan jika sudah ada pemenang.



Gambar 5. Papan permaian dan nilai index



Gambar 6. Kemungkinan *goal*

Papan permainan tic-tac toe terdiri atas 9 kotak. Untuk memudahkan proses eksekusi algoritma, setiap kotak akan diberi indeks yang terdiri atas indeks baris (i) dan indeks kolom (j). Gambar 5 memperlihatkan papan permainan dan nilai indeksnya. Goal atau tujuan permainan tercapai apabila salah satu pemain telah menempatkan 3 bidak dalam deret sejajar. Ada 8 kemungkinan deretan yang terbentuk (Gambar 6), yakni: 3 deret dalam posisi horisontal, 3 deret dalam posisi vertikal dan 3 deret dalam posisi diagonal

ISSN : 1411-1624
e-ISSN: 2621-8089

3.2 Analisis Pergerakan NPC dan Desain Algoritma

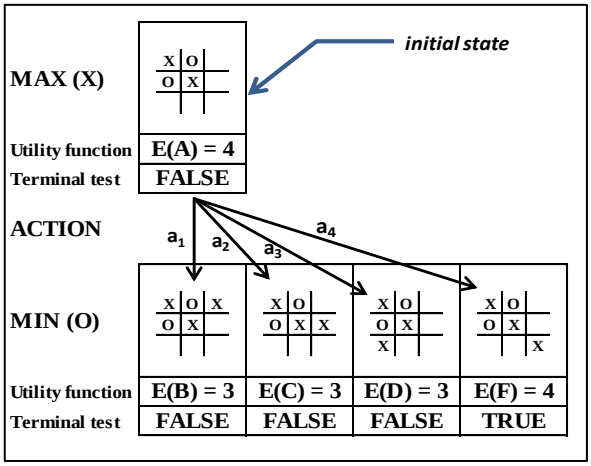
Pergerakan NPC akan dirancang dalam dua mode, yakni mode random dan mode minimax. Pada mode random, NPC akan meletakkan bidak secara acak pada posisi yang kosong (null). Sedangkan pada mode minimax, NPC akan menentukan langkah terbaik berdasarkan algoritma minimax. Algoritma minimax memiliki parameter sebagai berikut.

a. *Initial state*; merupakan keadaan papan sebelum algoritma dijalankan
b. *Player*; pemain mana yang mendapat giliran meletakkan bidak pada sebuah state
c. *Action*; merupakan himpunan pergerakan yang dapat dilakukan player
d. *Result*; merupakan keadaan papan setelah sebuah action dilakukan.
e. *Terminal test*; merupakan fungsi pengecekan tujuan. Apakah kondisi kemenangan telah terpenuhi?
f. *Utility function*; merupakan nilai evaluasi kemungkinan kemenangan.

Selain itu, setiap level pada pohon pencarian akan ditandai sebagai level MAX dan level MIN. Level MAX mewakili giliran NPC untuk meletakkan bidak, sedangkan level MIN mewakili giliran pemain lawan.

Sebagai ilustrasi, dapat diperhatikan Gambar 7. Misalkan NPC mengendalikan bidak X dan pemain lawan mengendalikan bidak O. *Initial state* (disimbolkan dengan node A, sebagai node induk) merupakan keadaan papan setelah pemain lawan meletakkan bidak O pada kotak (1,0). Giliran main berikutnya adalah NPC yang meletakkan bidak. Terdapat 5 *action* yang mungkin dilakukan oleh NPC, yakni

a. a_1: meletakkan bidak X pada kotak (0,2),
b. a_2: meletakkan bidak X pada kotak (1,2),
c. a_3: meletakkan bidak X pada kotak (2,0),
d. a_4: meletakkan bidak X pada kotak (2,2).
e. a_5: meletakkan bidak X pada kotak (2,1) (tidak digambarkan dalam ilustrasi).



Gambar 7. Pohon pencarian tic-tac-toe

Setelah salah satu aksi dilakukan keadaaan papan akan berubah (*result*). *Terminal test* diperoleh dengan memeriksa posisi bidak pada papan permainan. *Terminal test* akan bernilai *TRUE* jika salah satu kondisi berikut terpenuhi:

a. nilai indeks baris dari ketiga bidak adalah sama atau,
b. nilai indeks kolom ketiga bidak adalah sama atau,
c. untuk setiap bidak, nilai indek baris sama dengan nilai indeks kolomnya atau,
d. untuk setiap bidak, jumlah nilai indeks baris ditambah nilai indeks kolom sama dengan 2.

Permainan akan dihentikan jika *terminal test* telah bernilai *TRUE*. Gambar 8 memperlihatkan desain *pseudocode* untuk mengecek *terminal test*.

ISSN : 1411-1624
e-ISSN: 2621-8089

```
Algoritma terminaltest
  Input: intial state, piece
  Output: boolean value
  Set x,y
  Value ← FALSE
  Valrow ← 1
  Valcol ← 1
  Valright ← 1
  Valleft ← 1
  for x ← 0 to 2 do
    for y ← 0 to 2 do
      if board(x,y) = piece then
        valrow←valrow*1
        valcol←valcol*1
      else
        valrow←valrow*0
        valcol←valcol*0
      endif
      if x = y then
        if board(x,y) = piece then
          valright←valright*1
        else
          valright←valright*0
        endif
      endif
      if x + y = 2 then
        if board(x,y) = piece then
          valleft←valleft*1
        else
          valleftt←valleft*0
        endif
      endif
    next y
    if valrow = 1 then value←TRUE
    if valcol = 1 then value←TRUE
  next x
  if valright = 1 then value←TRUE
  if valleft = 1 then value←TRUE
  return value
end
```

Gambar 8. ***Pseudocode terminal test***

Untuk memilih sebuah aksi, NPC akan membangun pohon pencarian sedalam 1 level (*one-ply search*). Setelah itu NPC akan mengevaluasi nilai *utility function* dari setiap *node* anak. Nilai *utility function* (E) diperoleh dengan cara menghitung kemungkinan kemenangan yang dapat dicapai NPC (B1) dikurangi kemungkinan kemenangan yang dapat dicapai pemain lawan (B2). Sehingga secara matematis dapat dituliskan seperti pada persamaan (1).

$$\boldsymbol{E = B1 - B2} \qquad (1)$$

```
Algoritma WinPosibility
  Input: state, piece
  Output: number of win posibility
of piece
  Set x,y
  Value ← 0
  Valrow ← 1
  Valcol ← 1
  Valright ← 1
  Valleft ← 1
  for x ← 0 to 2 do
    for y ← 0 to 2 do
      if board(x,y) = piece then
        valrow←valrow*1
        valcol←valcol*1
      else
        valrow←valrow*0
        valcol←valcol*0
      endif
      if x = y then
        if board(x,y) = piece then
          valright←valright*1
        else
          valright←valright*0
        endif
      endif
      if x + y = 2 then
        if board(x,y) = piece then
          valleft←valleft*1
        else
          valleftt←valleft*0
        endif
      endif
    next y
    if valrow=1 then value←value+1
    if valcol=1 then value←value+1
  next x
  if valright=1 then value←value+1
  if valleft=1 then value←value+1
  return value
end
```

Gambar 9. ***Pseudocode*** **kemungkinan kemenangan**

Gambar 9 menampilkan *pseudocode* untuk menghitung kemungkinan kemenangan setiap pemain. Sedangkan Gambar 10

ISSN : 1411-1624
e-ISSN: 2621-8089

menampilkan *pseudocode* untuk mendapatkan nilai *Utility Function*.

Sebagai ilustrasi, dimisalkan pada Gambar 7, A adalah *node* induk dengan *node* anaknya adalah B, C, D, dan F. Nilai E dihitung mulai dari *node-node* tersebut. Nilai *utility function* untuk *node* B dapat dihitung sebagai berikut. Kemungkinan kemenangan NPC adalah deret [(0,0), (1,1), (2,2)], [(0,2), (1,1), (2,0)], [(0,2), (1,2), (2,2)], dan [(2,0), (2,1), (2,2)]. Dengan demikian ada 4 kemungkinan menang bagi NPC (B1 = 4). Sedangkan bagi pemain lawan, hanya ada 1 kemungkinan menang (B2 = 1), yakni pada deret [(2,0), (2,1), (2,2)]. Dengan demikian E(B) = 4 – 1 = 3. Dengan cara yang sama, nilai *utility function* untuk *node* C, D, dan F dapat dihitung. Setelah seluruh nilai *utility function* pada node anak dihitung, maka nilai utility function pada node induk dapat dihitung.

```
Algoritma UtilityFunction
  Input: state
  Output: utility value
  B1←WinPosibility(state,X)
  B2←WinPosibility(state,O)
  Return (B1 – B2)
end
```

Gambar 10. *Pseudocode utility function*

Misalkan, E(B) = 3, E(C) = 3, E(D) = 3, dan E(F) = 4. Karena node induk (node A) berada pada posisi MAX, maka E(A) = MAX[E(B), E(C), E(D), E(F)] = MAX[3,3,3,4] = 4. Dengan demikian, NPC akan memilih action a4 sebagai langkah terbaik. Gambar 11 menampilkan algoritma FindMax untuk menentukan langkah terbaik.

```
Algoritma FindMax
  Input: state, piece
  Output: MaxValue, BestMove
  Set k ← 0
  Set MaxValue ← -10
  For x ← 0 to 2 do
    For y ← 0 to 2 do
      If board(x,y) = null then
        Board(x,y)←piece
        E(k) ← UtilityFunction(state)
        If E(k) > MaxValue then
          MaxValue ← E(k)
          Bestrow ← x
          Bestcol ← y
        Endif
        Inc(k)
      Endif
      Borad(x,y)←null
    Next y
  Next x
  Return MaxValue
  Return BestMove(bestrow,bestcol)
end
```

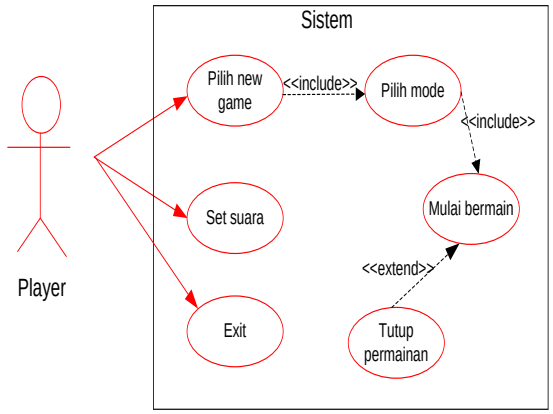
Gambar 11. *Pseudocode FindMax*

3.3 Desain Perangkat Lunak dan Kode

Perangkat lunak dirancang dengan menggunakan *Unified Modeling Language* (UML). Dalam hal ini kami menggunakan empat diagram UML yakni *use case diagram*, *class diagram*, *activity diagram*, dan *sequence diagram*. Pada Gambar 12 diperlihatkan *use case* diagram untuk perangkat lunak yang akan dibangun. *Use case diagram* dilengkapi dengan skenario untuk menjelaskan aliran aksi yang dijalankan. Terdapat tiga sekenaio yang dirancang yakni skenario *new game*, skenario option dan skenario exit. Pada Tabel 2 tersaji contoh skenario untuk jika aktor (*player*) memilih aksi *new game*.

Pada Gambar 13 disajikan *class diagram* yang dirancang untuk palikasi tic-tac-toe. Diagram aktivitas (*activity diagram*) juga dirancang untuk tiga aktivitas utama, yakni *new game*, diagram aktivitas option, dan diagram aktivitas proses exit. Pada Gambar 14 disajikan contoh diagram aktivitas.

ISSN : 1411-1624
e-ISSN: 2621-8089

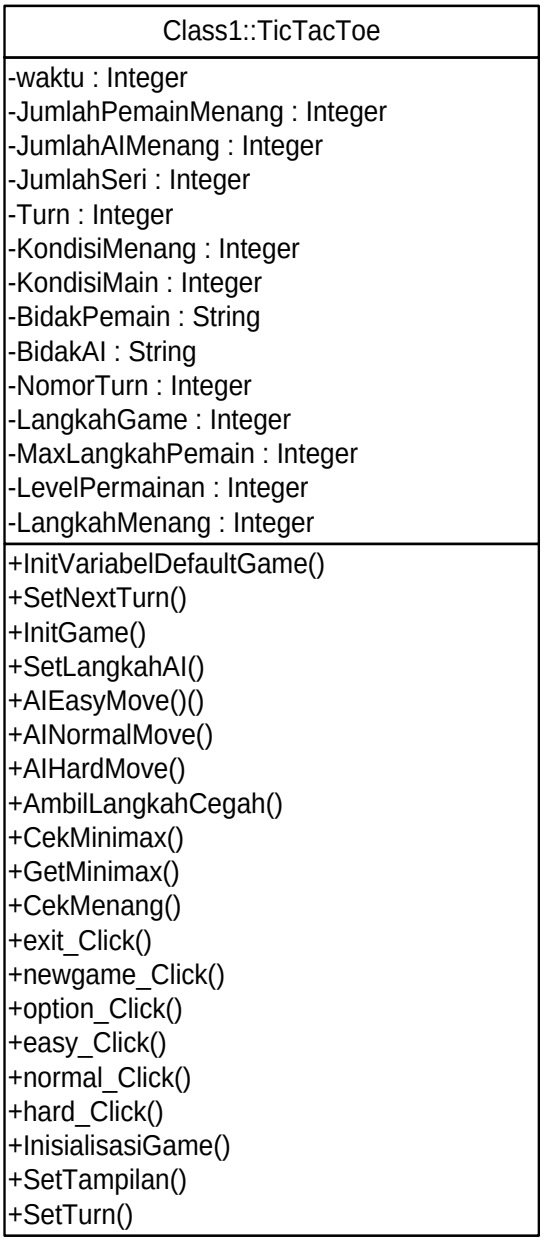


Gambar 12. ***Use case diagram aplikasi tic-tac-toe***

Tabel 2. Skenario *new game*

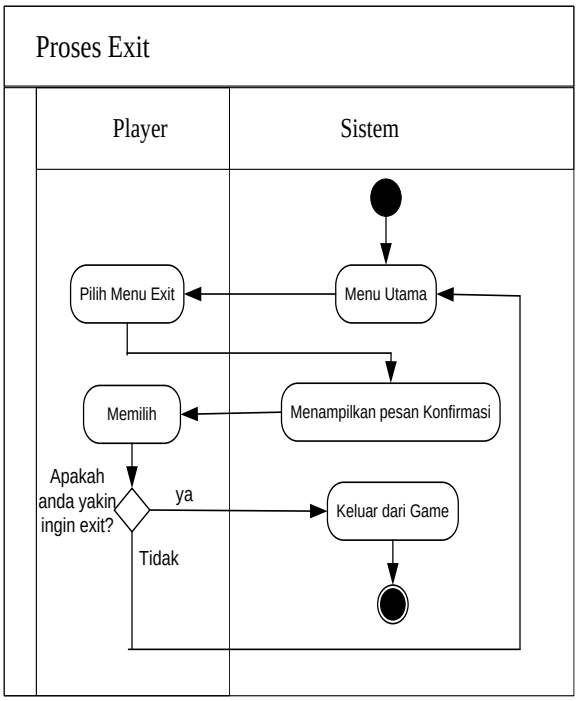
Aktor yang Menggunakan	**Pengguna**
Fungsi	Melakukan permainan *tic-tac-toe* pada awal permainan
Deskripsi	Memainkan permainan *tic-tac-toe*
Kondisi Awal	**Halaman menu utama**
Mekanisme Kerja	
Aktor	**Sistem**
	1 Tampil *form* menu utama
2 Klik tombol *new game*	
	3 Tampil *form level*
4 *Klik level* yang diinginkan	
	5 Tampil *form* permainan
	6 Menentukan pemain pertama
	7 Aktor jalan terlebih dahulu
8 Aktor mengisi kotak kosong	
	9 Memeriksa dan mengisi kotak kosong
Alternatif 1	
	7 Sistem jalan terlebih dahulu
	8 Sistem mengisi kotak kosong
9 Aktor mengisi kotak kosong	
	10 Selalu bergiliran sampai ada yang menang
Kondisi Akhir	**Tampil Pemberitahuan sistem menang atau aktor menang**

Selain menggunakan diagram *use case* dan diagram aktivitas untuk menggambarkan perilaku pengrakat lunak, kami juga merancang perilaku interaksi perangkat lunak dengan menggunakan sequence diagram. Mengacu pada tiga aktivitas utama, sequence diagram juga dirancang untuk tiga aktivitas tersebut. Gambar 15 menampilkan contoh sequence diagram yang dirancang.

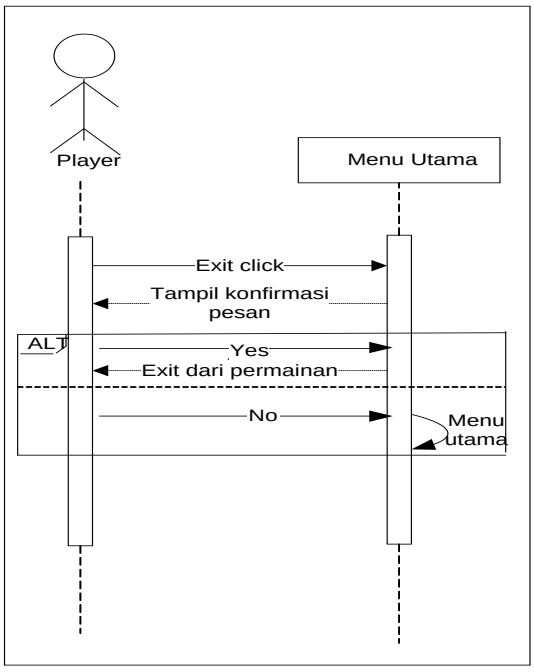


Gambar 13. ***Class diagram aplikasi tic-tac-toe***

ISSN : 1411-1624
e-ISSN: 2621-8089

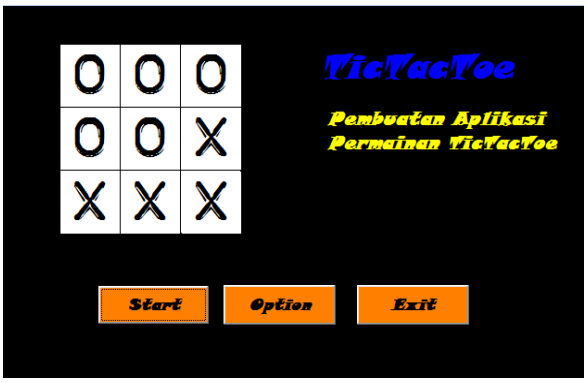


Gambar 14. *Activity diagram* proses *exit*



Gambar 15. *Sequence diagram* proses *exit*

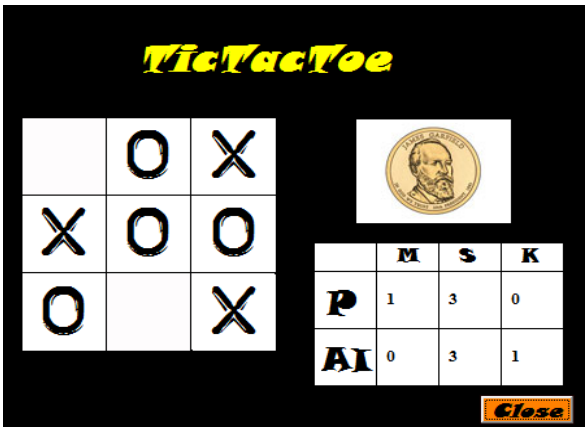
Aplikasi *tic-tac-toe* ini dibangun dengan menggunakan pemrograman Visual Basic dan dapat dijalankan pada platform Windows. Gambar 16 dan Gambar 17 menampilkan contoh antarmuka pemakai dari aplikasi yang dibangun.



Gambar 16. *Tampilan awal aplikasi*

3.4 Pengujian Perangkat Lunak

Pengujian perangkat lunak dilakukan dengan teknik *whitebox testing* dan *blackbox testing*.



Gambar 17. *Antarmuka aplikasi*

Pengujian *whitebox* dilakukan untuk memastikan bahwa semua jalur independen dari suatu modul telah dilalui paling sedikit satu kali. Pada aplikasi ini pengujian *whitebox* yang dilakukan adalah pengujian *basis path* dan kemudian menghitung nilai *cyclomatic complexity*. Berdasarkan *flowgraph* yang disusun, diperoleh jumlah *node* (N) sebanyak 23 dan jumlag *edge* (E) sebanyak 30. Dengan demikian diperoleh nilai V(G) = E – N + 2 = 9. Hal ini berarti bahwa aplikasi yang dibangun memiliki prosedur yang terstruktur baik, stabil , dan resiko rendah.

ISSN : 1411-1624
e-ISSN: 2621-8089

Pengujian *blackbox* dilakukan untuk memastikan bahwa semua fungsi-fungsi suatu perangkat lunak telah berjalan dengan benar. Pada aplikasi ini pengujian *blackbox* dilakukan dengan memeriksa kesalahan antarmuka pemakai. Tabel 3 memperlihatkan hasil pengujian *blackbox* pada antarmuka permainan (form permainan).

Pengujian secara statistik juga dilakukan untuk melihat seberapa baik kinerja algoritma yang telah dirancang. Pengujian dilakukan dengan melibatkan 99 kali percobaan memainkan aplikasi tic-tac-toe. Hasilnya (Tabel 4) menunjukkan bahwa pada mode random, *player* dapat memenangkan seluruh sesi permainan (100%). Sedangkan pada mode minimax, player tingkat kemenangan player menurun menjadi 67,6%. Hal ini menunjukkan bahwa algoritma minimax dapat bekerja dengan baik sebagai NPC.

Tabel 3. Hasil pengujian *balckbox* form permainan

No	*Item Pengujian*	*Hasil yang diharapkan*	*Hasil Implementasi*	*Keterangan*
1.	Koin Putar	Sistem Aktif.	Tampilan Pesan “AI *First*”	Sesuai yang diharapkan
2.	Bidak yang dikeluarkan.	Sistem Aktif.	*User* klik kolom kosong dan bidak muncul	Sesuai yang diharapkan
3.	Papan *score*	Sistem Aktif.	Tampil *score player* dan AI	Sesuai yang diharapkan
4.	Pilih *Back*.	Sistem Aktif.	Tampil *form Level*.	Sesuai yang diharapkan
5.	Pilih Keluar	Sistem Aktif.	Tampil *form* Menu Utama.	Sesuai yang diharapkan

Tabel 4. Hasil pengujian *statistik*

Player	***Mode***	
	Random	Minimax
Menang	100%	67,6%
Seri	0%	30,3%
Kalah	0%	2,02%

4 KESIMPULAN

Berdasarkan hasil penelitian dapat diambil kesimpulan berikut.

a. Algoritma minimax dapat digunakan sebagai NPC dalam permainan tic-tac-toe secara khusus dan secara umum permainan papan yang melibatkan dua *player*.
b. Dalam penelitian ini, tingkat kedalaman pohon pencarian hanya sebesar satu tingkat, pada penelitian lanjutan tingkat kedalaman pohon pencarian dapat ditingkatkan.

DAFTAR PUSTAKA

[1] R. P. Kusrachmadi, Rizky Yuniar Hakkun, and Idris Winarno, "Pengenalan Jaringan Komputer Berbasis Role Playing Game Menggunakan RMXP dan RGSS," Surabaya, Tugas Akhir 2013.

[2] Muhammad Kurniawan, Afib Pamungkas, and Salman Hadi, "Algoritma Minimax Sebagai Pengambila Keputusan Dalam Game Tic-Tac-Toe," in *Seminar Nasional Teknologi dan Multimedia*, Yogayakarta, 2016.

[3] R. Kristoforus Jawa Bendi, "Penggunaan Fungsi Heuristik Sederhana Pada Permainan Tic-Tac-Toe," in *Prosiding Seminar Nasional Inovasi Dan Aplikasi Teknologi Di Industri*, Malang, 2017, pp. A18.1-A18.7.

[4] Umair Z Ahmed, Krishnendu Chatterjee, and Gulwani Sumit, "Automatic generation of alternative starting positions for simple traditional board games," in *National Conference on Artificial Intelligence*, Austin, Texas USA, 2015, pp. 1-8.

[5] S. Karamchandani, P. Gandhi, O. Pawar, and S. Pawaskar, "A simple algorithm for designing an artificial intelligence based Tic Tac Toe game," in *Proceeding of International Conference on Pervasive Computing*, 2015, pp. 1-4.

[6] S Kosasi, "Permainan Papan Strategi Menggunakan Algoritma Minimax," in

ISSN : 1411-1624
e-ISSN: 2621-8089

Seminar Nasional Teknologi Informasi, Komunikasi dan Industri, 2014, pp. 105-112.

[7] E Jayadi, MAF Rachman, and M Yuliansyah, "Aplikasi Game Tic Tac Toe 6x6 Berbasis Android Menggunakan Algoritma Minimax Dan Heuristic Evaluation," in *Seminar Nasional Teknologi Informasi dan Multimedia*, 2016, pp. 91-96.

[8] Dicky Herman Firmansyah, Nana Juhana, and Irfan Maliki, "Implementasi Algoritma Minimax Pada Permainan Tic-Tac-Toe Skala 9x9," Bandung, Skripsi 2009.

[9] Nur Jannah, "Analisis dan Implementasi Algoritma Minimax dengan Optimasi Alpha-Beta Pruning pada Permainan Five In Row," Medan, Skripsi 2010.

[10] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*.: McGraw-Hill, 2001.