Jurnal Komputasi

Vol 10 No. 1, 2022
©2022 Ilmu Komputer Unila Publishing Network all rights reserved

# Peningkatan Kemenangan Non-Playable Character dalam Permainan Triple Triad Menggunakan *Alpha-Beta Pruning*

**[1]Benedictta Dinda Permatasari, [2,*)]Hanny Haryanto, [3]Erna Zuni Astuti & [4]Erlin Dolphina**

[1,2,3,4] Teknik Informatika, Universitas Dian Nuswantoro, Jl. Imam Bonjol 207, Semarang, Indonesia

**Abstrak** — *Non-Playable Character* (NPC) merupakan salah satu elemen yang paling penting dalam video game. Umumnya, NPC menyediakan tantangan bagi pemain dalam menyelesaikan misi di dalam game, dimana NPC berarti bertindak sebagai musuh. Peran sebagai musuh menyebabkan tingkat kemenangan menjadi salah satu tujuan utama dari kecerdasan buatan yang diaplikasikan ke NPC. Tantangan yang disediakan NPC ini sangat penting untuk menjaga agar pemain tetap melanjutkan permainan. Untuk mendapatkan pengalaman yang sama menyenangkannya dengan saat bermain dengan orang lain, NPC harus dapat menyediakan tantangan yang seimbang layaknya manusia. Masalah yang terjadi adalah rendahnya tingkat kemenangan yang diraih oleh NPC, sehingga pemain dapat merasa bosan. Algoritma *alpha-beta pruning* merupakan salah satu algoritma pengambil keputusan yang sering diterapkan pada permainan yang memerlukan pemain lebih dari atau sama dengan dua. Karena itu algoritma ini cocok untuk diterapkan pada objek penelitian yaitu permainan Triple Triad. Permainan Triple Triad adalah permainan papan yang dimainkan oleh dua pemain. Permainan Triple Triad petama kali diperkenalkan sebagai sebuah mini-game yang terdapat dalam permainan Final Fantasy VIII. Permainan ini merupakan gabungan dari permainan kartu (*card game*) dan juga permainan papan (*board game*). Dalam penelitian ini, algoritma *alpha-beta pruning* terbukti dapat meningkatkan tingkat kemenangan (*win rate*) NPC. Hal ini ditunjukkan dengan hasil perbandingan *win rate* NPC yang memilih langkah secara acak yaitu sebesar 17.5% dengan NPC yang sudah diterapkan algoritma *alpha-beta pruning* yaitu sebesar 55%. Disini terlihat peningkatan *win rate* yang cukup signifikan.

**Kata Kunci:** *Alpha-beta pruning*; kartu; kecerdasan buatan; Non-Playable Character; permainan.

***Abstract*** *— Non-Playable Character (NPC) is one of the essential elements in video games. Generally, NPCs provide challenges for players in completing missions in the game, where NPCs mean acting as enemies. The role of the enemy causes the victory rate to be one of the main goals of artificial intelligence applied to NPCs. The challenges that these NPCs provide are significant to keep players going. NPCs must be able to provide a balanced challenge like humans to have an experience that is as enjoyable as when playing with other people. The problem is the low win rate achieved by NPCs so that players can feel bored. The alpha-beta pruning algorithm is one of the decision-making algorithms that is often applied to games that require more than or equal two players. Therefore, this algorithm is suitable for applying to the object of research, namely the* Triple Triad *game. The* Triple Triad *game is a board game played by two players. The* Triple Triad *game was first introduced as a mini-game in the Final Fantasy VIII game. This game is a combination of card games and board games. In this study, the alpha-beta pruning algorithm was proven to increase the win rate of NPCs. It is indicated by comparing the win rate of NPCs who choose a random step, which is 17.5%, with an NPC that has applied the alpha-beta pruning algorithm, which is 55%. Therefore, there is a significant increase in the win rate..*

*Keywords: Alpha-beta pruning; artificial intelligence; card; game; Non-Playable Character.*

** Corresponding author :*
*Hanny Haryanto*
*Universitas Dian Nuswantoro, Semarang, Indonesia*
*hanny.haryanto@dsn.dinus.ac.id*

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

## 1. PENDAHULUAN

Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan salah satu dari bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia [1]. Dijaman sekarang ini, kecerdasan buatan sudah diterapkan dalam berbagai bidang seperti industri, medis, pendidikan, bisnis bahkan dalam kehidupan sehari-hari. Beberapa contoh penerapan kecerdasan buatan adalah seperti *DeepFace* milik *facebook* yang berfungsi untuk mengenali wajah orang yang ada pada postingan foto, sistem rekomendasi pada *e-commerce*, asisten virtual seperti *Google assistant* dan juga *Siri*, serta dapat kita temukan juga pada *video game* yang umumnya diterapkan pada *non-player character* atau yang biasa disebut NPC.

*Video Game* adalah salah satu sarana hiburan berupa aplikasi yang dimana pengguna akan berinteraksi dengan aplikasi tersebut dengan mengikuti sejumlah standar aturan, yang nantinya aksi pengguna dapat menghasilkan nilai kuantitatif yang berupa kesuksesan atau kegagalan. *Game* secara umum dikategorikan dalam lima kategori yaitu: *action* (aksi), *adventure* (petualangan), *role-playing*, *first-person shooter*, dan strategi, akan tetapi dalam beberapa tahun terakhir banyak *video game* yang telah mengkombinasikan *gameplay* dari beberapa *genre*, sehingga klasifikasi umum tersebut saat ini dianggap tidak informatif [2].

Permainan Triple Triad petama kali diperkenalkan sebagai sebuah *mini-game* yang terdapat dalam *game Final Fantasy VIII*, permainan ini merupakan gabungan dari permainan kartu (*card game*) dan juga permainan papan (*board game*). Permainan ini dilakukan dengan dua pemain yang saling melawan satu sama lain. Satu sisi bermain sebagai sisi "biru" dan sisi lain nya adalah sisi "merah". Permainan ini dilakukan diatas papan 3x3 dengan kartu sebagai bidaknya. Setiap pemain masing-masing memiliki lima kartu diawal permainan dan tujuan utama dalam permainan ini adalah menangkap atau meng-*capture* kartu milik pemain lain yang nantinya pemain yang memilliki kartu paling banyak akan dianggap sebagai pemenang dalam permainan. Pada permainan yang memerlukan dua orang atau lebih untuk dapat memulai permainan, meskipun sudah ada mode permainan yang memungkinkan *player* untuk bermain bersama baik itu secara lokal maupun online, mode permainan *singleplayer* masih banyak digunakan oleh *player* baik itu untuk memahami permainan tersebut, atau *player* memiliki kendala untuk dapat bermain secara *multiplayer*. Pada saat seseorang bermain dalam mode permainan *singleplayer* pemain akan melawan *non-player character* (NPC) yang dikendalikan oleh komputer.

Kecerdasan Buatan diterapkan pada NPC lawan untuk meningkatkan kemampuan NPC dalam permainan agar pemain tidak merasa permainan terlalu mudah atau terlalu sulit pada saat melawan NPC dikarenakan kesulitan yang tidak seimbang. Kecerdasan buatan di dalam *game* dibutuhkan untuk meningkatkan tantangan dalam permainan dan membuat permainan menjadi lebih dinamis dan terarah. Terdapat banyak metode yang dapat digunakan untuk menerapkan kecerdasan dalam *game* salah satunya adalah algoritma *minimax.*

Algoritma *minimax* adalah salah satu teknik pencarian menggunakan *decision tree* dan juga *breadth-first search* atau *depth-first search* dengan kedalaman (*depth*) terbatas, yang didesain untuk memaksimalkan kemenangan dan meminimalkan kekalahan dalam skenario terburuk dalam *game* [3], [4]. Menurut [5], dalam penelitiannya yang menerapkan algoritma *minimax* pada permainan *tic-tac-toe,* algoritma *minimax* algoritma yang sangat bagus dan cocok untuk pengambilan keputusan oleh *AI*, terutama dalam permainan *n player* ($n >= 2$). Meskipun dianggap baik, akan tetapi algoritma ini tidak cocok untuk dterapkan pada permainan yang memiliki kemungkinan yang besar seperti catur, karena pohon solusi akan sulit terbentuk. Untuk mengatasi hal ini, terdapat metode yang dapat diterapkan untuk memangkas pohon pencarian dari algoritma *minimax*, yang salah satunya adalah metode *alpha-beta pruning*. Algoritma *alpha-beta pruning* ekuivalen dengan algoritma *minimax* dalam hal, keduanya sama-sama mencari langkah terbaik dari posisi *p* dan keduanya akan memberikan nilai yang sama berdasarkan manfaat yang diharapkan, yang membedakannya dengan *minimax* adalah algoritma ini lebih cepat karena dalam proses pencarian, *alpha-beta* tidak menelusuri cabang yang tidak akan

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

memberikan pengaruh terhadap nilai yang telah tersimpan [6]. *Alpha-beta pruning* diterapkan karena algoritma ini didesain untuk melakukan pencarian dengan lebih cepat tanpa hilangnya informasi [7]

Penelitan yang dilakukan oleh [5], yang menerapkan algoritma *minimax* tanpa optimasi pada permainan *tic-tac-toe* 3x3 dan menyimpulkan bahwa algoritma *minimax* cocok untuk diterapkan untuk pengambilan keputusan pada permainan *tic-tac-toe* atau permainan lainnya yang memiliki jumlah pemain lebih dari dua, akan tetapi pohon solusi akan sulit terbentuk apabila permainan tersebut memiliki kemungkinan yang sangat besar, contohnya seperti pada catur. Penelitian terkait lainnya adalah penelitian yang dilakukan oleh [8] yang menerapkan kecerdasan buatan pada permainan *checker* menggunakan algoritma *minimax*, dengan menambahkan nilai *alpha-beta* dan menghasilkan kesimpulan bahwa penggunaan nilai *alpha-beta* dapat memangkas pohon pencarian sehingga waktu yang diperlukan menjadi lebih sedikit, dan pada penelitian yang dilakukan oleh [9] yang membandingkan penerapan dua metode yaitu algoritma *minimax* dan *alpha-beta pruning* pada permainan *connect-4* menyimpulkan bahwa algoritma *alpha-beta pruning* melakukan lebih sedikit iterasi jika dibandingkan dengan *minimax.* Pada penelitian lainnya yang dilakukan oleh [10] yang menerapkan algoritma *negascout* sebagai pencari solusi dalam permainan congklak dan menghasilkan kesimpulan bahwa algoritma *negascout* tidak lebih baik dari algoritma *minimax* dalam faktor tingkat kemenangan, tetapi bekerja lebih cepat dari algoritma *minimax* dengan perbandingan waktu pencarian 2:5. Permasalahan *search* pada penelitian ini masuk pada *game playing* [11], menggunakan *alpha-beta pruning* [12] untuk meningkatkan *win rate.*Algoritma *alpha-beta pruning* adalah algoritma yang dirancang untuk mempercepat proses pencarian tanpa ada hilangnya informasi [7]. Algoritma ini ekuivalen dengan algoritma *minimax* dalam hal yaitu keduanya sama-sama mencari langkah terbaik dari posisi *p* dan keduanya akan memberikan nilai yang akan memberikan keuntungan, akan tetapi *alpha-beta pruning* lebih cepat dari *minimax* karena algoritma ini tidak menelusuri cabang-cabang yang tidak akan memberikan pengaruh terhadap nilai evaluasi yang telah tersimpan [6]. Algoritma ini diterapkan dengan menambahkan nilai *alpha* dan *beta* dalam fungsi *minimax,* yang dalam penelitian ini diterapkan pada triple triad berjenis *board game*. *Board game* melibatkan *counter* atau bidak yand dipindahkan atau ditempatkan pada papan khusus permainan tersebut, berdasarkan aturan-aturan permainan yang telah ditentukan. Permainan papan ada yang murni berbasis strategi, *chance* (kesempatan), atau gabungan dari kedua hal tersebut [13]. Permainan ini umumnya dimainkan oleh dua orang atau lebih, pemain yang memainkan permainan papan harus menyusun siasat dan rencana yang tepat sebelum menggerakkan bidak agar dapat mengalahkan lawan. Kekeliruan dalam menyusun strategi dapat mengakibatkan kekalahan dalam permainan [14]. Kelemahan *minimax* diungkapkan oleh [15] dan [16] dalam menangani permasalahan yang memiliki ruang lingkup kecil namun membutuhkan banyak waktu.

Permainan Triple Triad tidak memiliki kemungkinan sebesar permainan catur, tetapi memiliki kemungkinan yang lebih besar daripada permainan *tic-tac-toe*. Walaupun memiliki ukuran papan yang sama dengan permainan *tic-tac-toe* yaitu 3x3, permainan ini memiliki lebih banyak aspek yang perlu dipertimbangkan untuk menentukan langkah, mulai dari nilai kartu yang beragam, serta *state* papan yang dinamis (dapat berubah) bergantung pada keadaan, karena itulah dipilih metode *alpha-beta pruning* untuk diterapkan karena algoritma ini didesain untuk melakukan pencarian dengan lebih cepat tanpa hilangnya informasi sehingga memiliki tingkat akurasi yang ekuivalen dengan algoritma *minimax.*

## 2. METODOLOGI PENELITIAN

### 2.1. *Pengambilan Data*

Pengambilan data dilakukan dengan bermain Triple Triad melawan NPC yang mengambil langkah secara acak (*random generated movement*) sebanyak *n* kali (*n* = jumlah permainan) dan mencatat hasil akhir dari setiap permainan (pada sisi NPC) baik itu berupa kemenangan ataupun kekalahan.

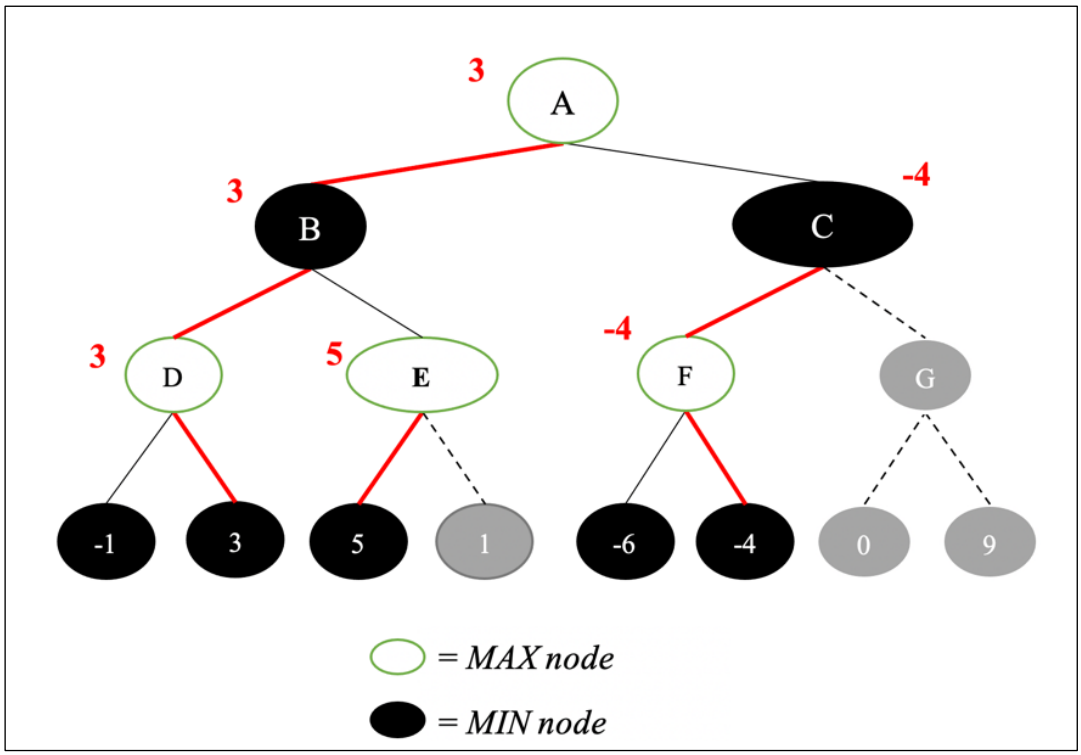
©2022 Ilmu Komputer Unila Publishing Network all rights reserved

### 2.2. *Pengolahan Data*

Data hasil dari setiap permainan akan dimasukan kedalam excel kemudian dihitung total kemenangan dari keseluruhan permainan, yang kemudian nanti akan di rata-rata untuk mendapatkan persentase *win rate* dari NPC yang belum diterapkan kecerdasan buatan.

### 2.3. *Metode alpha-beta pruning*

Metode *alpha-beta pruning* merupakan metode optimasi untuk algoritma *minimax*. Dengan menerapkan metode *alpha-beta pruning* pencarian dapat memperoleh hasil yang ekuivalen dengan *minimax* tetapi dengan waktu yang lebih sedikit. Metode *alpha-beta pruning* diterapkan ke algoritma *minimax* dengan menambahkan dua parameter baru yaitu *alpha* dan *beta*, yang dimana *Alpha* adalah nilai terbaik yang telah didapatkan setelah menelusuri pohon pencarian, dan *Beta* adalah nilai terendah yang telah didapatkan setelah menelusuri pohon pecarian. Di inisiasi awal nilai *Alpha* adalah *-infinity*, yang adalah skenario terburuk dalam pencarian nilai terbaik (*best value*) dan nilai *Beta* adalah *+infinity* yang adalah skenario terburuk dalam pencarian nilai terburuk (*worst value*). Dengan menambahkan parameter *Alpha* dan *Beta*, maka proses pencarian tidak lagi menelusuri seluruh *node*, karena nilai *Alpha* dan *Beta* akan membentuk sebuah pertimbangan bahwa apakah sebuah cabang akan dapat memengaruhi nilai yang sudah tersimpan, jika sebuah cabang dianggap tidak akan memengaruhi nilai yang sudah tersimpan (berdasarkan pertimbangan parameter *alpha* dan *beta*) maka cabang tersebut akan dipangkas (*pruned*) dan tidak telusuri.




Gambar 1. *Alpha-beta pruning* memangkas cabang yang tidak perlu ditelusuri

Cara kerja algoritma *alpha-beta* pada Gambar 1 adalah dengan membawa nilai *alpha* dan *beta* untuk dapat menjamin *best value* untuk *MAXIMIZER* dan *worst value* untuk *MINIMIZER* ke *leaf node* paling kiri bawah. Nilai awal $alpha = -infinity$ dan nilai awal $beta = +infinity$, saat mencapai *node* paling bawah maka *node* tersebut akan mengembalikan nilai -1, karena *node* D adalah *node MAX* maka nilai *alpha* pada *node* D adalah $max(-infinity, -1)$, maka $alpha = -1$, untuk memutus apakah cabang D yang sebelah kanan perlu untuk ditelusuri maka dilakukan pengecekan apakah nilai $beta <= alpha$, jika *true*, maka cabang disebelah kanan tidak perlu ditelusuri, akan tetapi nilai $beta = +infinity$ maka cabang ini tidak bisa di*prune*.

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

Kemudian pencarian dilanjutkan ke cabang D sebelah kanan, dan mengembalikan nilai 3, maka nilai *alpha* pada *node* D adalah $max(-1,3), alpha = 3$. Kemudian *node* D mengembalikan nilai yang disimpan ke *node* B, karena *node* B adalah *node MIN*, maka nilai *beta* pada *node* B adalah $min(+infinity, 3)$.

Lalu pencarian berlanjut ke cabang dari B yang belum ditelusuri yaitu cabang E pada cabang E nilai (*alpha*, *beta*) adalah $(-infinity, 3)$, *nilai beta* diturunkan oleh *node* B ke *node* E. setelah turun ke *node* E, maka pencarian berlanjut ke cabang sebelah kiri, dan didapatkan nilai 5 karena E adalah *node MAX*, nilai *alpha* pada *node* E adalah $max(-inf, 5)$, maka $alpha = 5$, setelah mendapatkan nilai *alpha* maka sekarang $nilai\ (alpha, beta)$ pada *node E* adalah $(5,3)$ untuk memutuskan apakah cabang E yang disebelah kanan perlu ditelusuri, maka dilakukan pengecekan apakah nilai *beta*<=*alpha* pada *node* E, dan nilai $3 <= 5$ adalah *true*, maka cabang E yang disebelah kanan tidak perlu ditelusuri dan dapat dipangkas (*prune*).

Karena *node* B adalah *node MIN* maka nilai node tersebut tetap 3, dan nilai 3 dikembalikan ke *node* A. Setelah nilai 3 dikembalikan ke *node* A, maka nilai (*alpha, beta*) pada *node* A adalah $(3, +infinity)$, kemudian dari *node* A , pencarian bergerak turun ke *node* C yang memiliki $nilai\ (alpha, beta) = (3, +infinity)$ yang diturunkan dari *node* A, kemudian ke *node* F yang memiliki nilai *alpha-beta* yang sama yang diturunkan dari C.

Dari *node* F, ditelusuri *child node* pertama dari *node* F dan mengembalikan nilai -6. Nilai *alpha* di *node* F tidak berubah karena $-6 < 3$, karena nilai *beta* masih belum berubah dari +*infinity*, maka dilanjutkan untuk menelusuri cabang kedua dan mengembalikan nilai -4, nilai F adalah $max(-6, -4)$ maka nilai $F = -4$, tetapi nilai *alpha* di *node* F tidak berubah karena $-4 < 3$. Kemudinan nilai pada *node* F dikembalikan ke *node* C, maka nilai *beta* pada *node* C adalah $min(+infinity, -4), beta = -4$, dari *node* C untuk memutuskan apakah perlu menelusuri cabang *G*, dilakukan pengecekan apakah nilai $beta <= alpha$ pada *node* C, dan $-4 <= 3$ adalah *true* maka cabang G tidak perlu ditelusuri, dan nilai C adalah -4. Setelah nilai *node* B dan C didapatkan maka nilai *node* A adalah $max(3, -4)$ maka nilai yang dikembalikan ke *node* A adalah 3.

### 2.4. *Cara pengujian metode*

Untuk menguji apakah metode *alpha-beta pruning* dapat meningkatkan *win rate* NPC dalam permainan Triple Triad, maka peneliti akan bermain permainan Triple Triad dengan NPC yang sudah diterapkan AI menggunakan metode *alpha-beta pruning* sebanyak *n* kali (*n* = jumlah permainan), sesuai dengan banyak permainan yang sudah dilakukan dengan NPC sebelum diterapkan algoritma. Kemudian akan dilakukan analisa perbandingan untuk mendapatkan hasil perbandingan *win rate* dari kedua tipe NPC. *Win rate* akan dihitung menggunakan rumus (1) sebagai berikut:

$$\frac{Total\ wins + (0.5 \times Total\ Draws)}{Total\ Matches} \times 100\% \qquad (1)$$

## 3. HASIL DAN PEMBAHASAN

Permainan Triple Triad adalah permainan yang dilakukan oleh dua orang, dimana kedua pemain dilambangkan dengan sisi “biru” dan sisi “merah”, dengan menggunakan kartu dan papan berukuran 3x3. Diawal permainan kedua pemain masing-masing memiliki 5 kartu, setiap kartu memiliki properti nilai kiri, nilai kanan, nilai atas dan nilai bawah seperti ditunjukkan di Gambar 2.

©2022 Ilmu Komputer Unila Publishing Network all rights reserved


Gambar 2. Properti kartu Triple Triad

Pemain yang akan menjadi giliran pertama akan dipilih secara acak. Pemain akan meletakan kartu secara bergantian diatas papan pada kotak yang belum terisi, hingga sembilan kotak yang terdapat pada papan terisi seluruhnya. Gambar 3 menunjukkan state awal dari permainan Triple Triad.


Gambar 3. State awal permainan Triple Triad

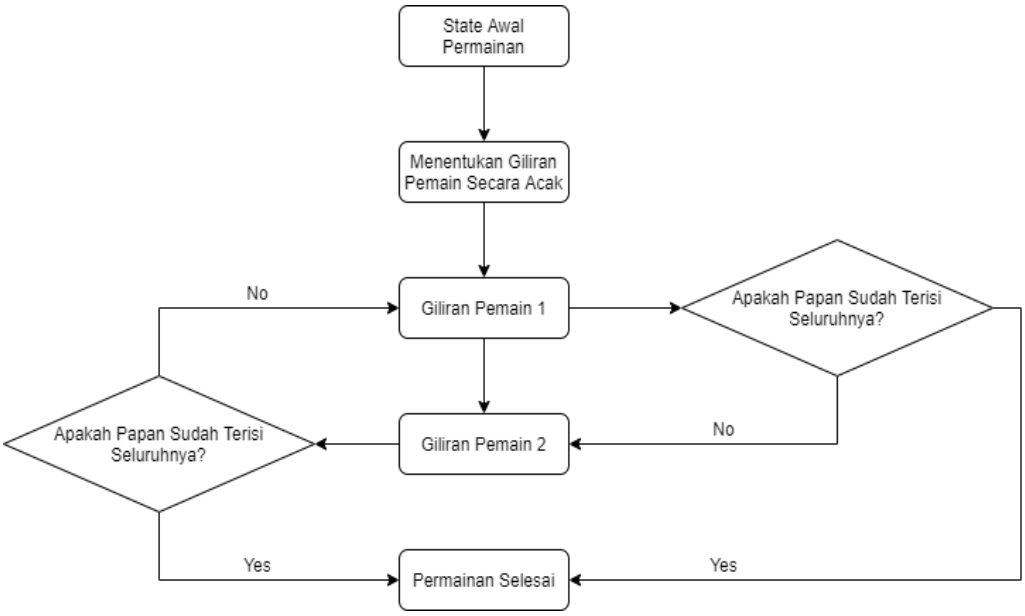
Tujuan utama dalam permainan ini adalah untuk mendapatkan kartu paling banyak, dengan menangkap (*capture*) kartu pemain lain. Proses capture ini ditunjukkan pada Gambar 4.


Gambar 4. Proses capture dalam permainan Triple Triad

Untuk meng-*capture* kartu milik pemain lain dilakukan dengan cara meletakkan kartu di kotak yang bersinggungan dengan kartu bermain lawan, kemudian nilai dari sisi yang bersinggungan akan bandingkan, jika kartu milik lawan memiliki nilai yang lebih kecil, maka kartu berhasil di-*capture* dilambangkan dengan berubahnya warna kartu menjadi warna pemain yang berhasil menangkap kartu tersebut. Kartu yang telah menjadi milik pemain lain, bisa di-*capture* kembali oleh pemilik sebelumnya.

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

Pada akhir permainan, pemain yang memiliki kartu paling banyak akan menjadi pemenang. Gambar 5 menunjukkan alur permainan Triple Triad.




Gambar 5. Alur permainan Triple Triad

Penerapan alpha beta pruning dalam permainan ini terdapat pada fungsi alpha_beta. Fungsi alpha_beta memerlukan beberapa parameter tambahan yang harus digunakan untuk memanggil fungsi tersebut seperti *board state*, *state* kartu NPC, *state* kartu pemain, *depth* atau jumlah langkah selanjutnya yang akan diprediksi, **`boolean isMaximizing`*`Player`*** yang menandakan giliran *player* dalam fungsi dan kemudian parameter alpha dan beta. Fungsi dieksekusi secara rekursif dengan nilai parameter yang selalu berubah. Di bagian awal fungsi terdapat kondisi **`if(depth==0)`** yang menunjukan sudah tidak ada langkah selanjutnya yang dapat diprediksi, maka fungsi akan mengevaluasi *score* pada papan dan mengembalikan nilai akhir *score* pada state tersebut. Bagian selanjutnya adalah kondisi dimana depth tidak sama dengan nol, yang terbagi menjadi dua kondisi yang ditentukan oleh variabel **`boolean isMaximizingPlayer`** yang dimana *maximizing player* dianggap sebagai NPC dan *minimizing player* atau ditulis sebagai **`!isMaximizingPlayer`** dianggap sebagai pemain yang melawan NPC. Pada kondisi **`isMaximizingPlayer`** dan **`!isMaximizingPlayer`** terdapat perulangan yang hampir sama pada fungsi **`AI_move`**, yang dimana program akan mensimulasikan langkah selanjutnya yang dapat diambil oleh *player* pada giliran tersebut yang membedakan kedua kondisi ini adalah **`isMaximizingPlayer`** akan mengembalikan **`best`*`Score`*** dengan nilai paling tinggi, sedangkan **`!isMaximizingPlayer`** akan mengembalikan nilai terendah. Selain perbedaan lain juga terdapat juga penggunaan parameter **`alpha`** dan **`beta`** disetiap kondisi, yang dimana pada isMaximizing*Player* nilai *score* memengaruhi nilai **`alpha`** dan pada **`!isMaximizing`*`Player`*** nilai *score* memengaruhi nilai beta, *loop* pada kedua kondisi akan berhenti jika nilai *beta* kurang dari atau sama dengan *alpha*. Kode program 1 berikut adalah *pseudocode* dari fungsi **`alpha_beta`** tersebut.

Pengujian akan dilakukan dengan melakukan permainan Triple Triad sejumlah *n* kali masing – masing dengan NPC yang sudah diterapkan algoritma *alpha-beta pruning* dengan NPC yang memilih langkah secara acak, kemudian yang *win rate* NPC *alpha-beta* akan dibandingkan dengan *win rate* NPC yang memilih langkah secara acak (*random*).

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

```
Function: alpha_beta(board: array of cells, aiCards: array of card,
playerCards: array of card, isMaximizingPlayer: boolean, depth:
int, alpha: int, beta: int)
score,bestScore: int
if (depth == 0) then
    bestScore ← evaluateScore()
    return score
else if (depth != 0) then
       if (isMaximizingPlayer) then
           for each row:
               for each column:
                   for each aiCards.card:
                         board[row][column] = card
                         aiCards.card.remove()
                         score ← alpha_beta(board,aiCards,
                         playerCards,isMaximizingPlayer, depth-
1,alpha,beta)
                         bestScore = max(score,bestScore)
                         alpha = max(score, alpha)
                         if (beta <= alpha)
                             break
           return bestScore
       if (!isMaximizingPlayer) then
           for each row:
               for each column:
                   for each playerCards.card:
                         board[row][column] = card
                          playerCards.card.remove()
                         score ← alpha_beta(board,aiCards,
                         playerCards,isMaximizingPlayer, depth-
1,alpha,beta)
                         bestScore = min(score,bestScore)
                         beta = beta(score, alpha)
                         if (beta <= alpha)
                             break
           return bestScore
```

Kode Program 1. Fungsi alpha_beta

Pengujian akan dilakukan dengan ketentuan – ketentuan sebagai berikut:

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

a. Kartu *NPC* dan *player* akan ditentukan secara acak
b. Pengujian akan dilakukan sebanyak dua puluh kali untuk masing – masing tipe NPC
c. Nilai *depth* (kedalaman) yang akan digunakan adalah tiga, yang artinya yang akan diprediksi adalah tiga langkah berikutnya.
d. Peraturan Triple Triad yang digunakan adalah *open* yang dimana kedua pemain meletakan kartu menghadap ke atas (terbuka), dan *random* dimana kartu yang digunakan oleh kedua pemain dipilih secara acak.

Hasil pengujian akan disajikan dalam bentuk tabel, yang terdiri dari hasil pertandingan dari kedua tipe NPC, yang akan digunakan untuk menghitung *win rate* dan membandingkan *win rate* dari kedua tipe NPC. *Win rate* akan dihitung menggunakan rumus (1). Setiap win akan memberikan nilai satu (1) pada total kemenangan, dan setiap draw akan membilikan nilai setengah (0.5) pada total kemenangan.

Tabel 1. Hasil pengujian

| **NPC** | ***Win*** | ***Lose*** | ***Draw*** | ***Win rate* (%)** |
|---|---|---|---|---|
| *Random* | 2 | 15 | 3 | 17,5% |
| *Alpha-beta pruning* | 7 | 5 | 8 | 55% |

Hasil yang diperlihatkan oleh Tabel 1 menunjukan perbedaan nilai *win rate* yang cukup signifikan yang dimana *win rate* NPC alpha-beta 37.5% lebih tinggi jika dibandingkan dengan *win rate* NPC yang memilih langkah secara *random* (acak).

## 4. KESIMPULAN

Berdasarkan hasil pengujian pada Tabel 1 maka dapat diambil kesimpulan bahwa algoritma *alpha-beta pruning* dapat diterapkan pada NPC untuk meningkatkan tingkat kemenangan (*win rate*). Hal ini ditunjukan dengan nilai *win rate* NPC yang meningkat menjadi 55% atau 37.5% lebih tinggi dibandingkan dengan *win rate* NPC yang memilih langkah secara acak yaitu 17.5%.

Berdasarkan dari pengkajian hasil penelitian maka pengembangan penelitian ke depan adalah sebagai berikut. Triple Triad memiliki peraturan permainan yang bervariasi, yang dimana pada penelitian ini hanya diterapkan dua peraturan, sesuai dengan yang sudah ditetapkan pada batasan masalah, yaitu open dan *random.* Dalam Penelitian selanjutnya, dapat memilih untuk menerapkan kombinasi *rule* yang berbeda dari keseluruhan delapan *rule* yang tersedia. Kartu yang digunakan dalam triple triad terbagi menjadi beberapa level dari level satu hingga sepuluh, peneliti pada penelitian selanjutnya dapat merancang sistem level pada NPC mulai dari *easy, intermediate, hard, expert* dan sebagainya. Pada permainan triple triad original, terdapat sistem reward atau lebih familiar disebut dengan sistem trade yaitu dimana diakhir permainan, pemain yang menang dapat mengambil kartu milik pemain yang kalah, sesuai dengan trade *rule* yang diterapkan. Penelitian selanjutnya dapat menerapkan sistem reward berdasarkan trade sistem yang dimiliki permainan triple triad original.

## DAFTAR PUSTAKA

[1] H. Jaya, Sabran, M. M. Idris, Y. A. Djawad, A. Ilham, and A. S. Ahmar, “Kecerdasan Buatan,” *Fak. MIPA Univ. Negeri Makassar*, 2018.

[2] S. M. Doherty, J. R. Keebler, S. S. Davidson, E. M. Palmer, and C. M. Frederick, “Recategorization of video game genres,” 2018, doi: 10.1177/1541931218621473.

©2022 Ilmu Komputer Unila Publishing Network all rights reserved

[3] E. Jayadi, M. Aziz, F. Rachman, and M. Yuliansyah, "Aplikasi Game Tic Tac Toe 6X6 Berbasis Android Menggunakan Algoritma Minimax Dan Heuristic Evaluation," pp. 6–7, 2016.

[4] S. D. H. Permana, "Implementation of Min Max Algorithm as Intelligent Agent on Card Battle Game," *IJISTECH (International J. Inf. Syst. Technol.*, vol. 2, no. 2, p. 53, 2019, doi: 10.30645/ijistech.v2i2.20.

[5] M. Kurniawan, A. Pamungkas, and S. Hadi, "Algoritma Minimax Sebagai Pengambil Keputusan Dalam Game Tic-Tac-Toe," *Semin. Nas. Teknol. Inf. dan Multimed. 2016 STMIK AMIKOM Yogyakarta, 6-7 Februari 2016*, pp. 37–42, 2016.

[6] S. H. Fuller, J. G. Gaschnig, and J. Gillogly, "Analysis of the alpha-beta pruning algorithm.pdf," *Dep. Comput. Sci. Carnegie-Mellon Univ.*, 1973.

[7] D. E. Knuth and R. W. Moore, *An analysis of alpha-beta pruning*, vol. 6, no. 4. 1975.

[8] D. Syapnika and E. R. Siagian, "Penerapan Algoritma Minimax Pada Permainan Checkers," *J. Ris. Komput.*, 2015.

[9] R. Nasa, R. Didwania, S. Maji, and V. Kumar, "Alpha-Beta Pruning in Mini-Max Algorithm-An Optimized Approach for a Connect-4 Game," *Int. Res. J. Eng. Technol.*, 2018.

[10] R. Darmawan and G. Hermawan, "SOLUSI LANGKAH PADA GAME CONGKLAK MENGGUNAKAN METODE NEGASCOUT," *Komputa J. Ilm. Komput. dan Inform.*, 2016, doi: 10.34010/komputa.v5i2.2462.

[11] R. E. Korf, "Artificial Intelligence Search Algorithms," in *In Algorithms and Theory of Computation Handbook*, Los Angeles: CRC Press, 1996.

[12] J. Mańdziuk, "Knowledge-free and learning-based methods in intelligent game playing," *Stud. Comput. Intell.*, 2010, doi: 10.1007/978-3-642-11678-0.

[13] C. Crawford, *Chris Crawford on Game Design*. United States Of America: New Riders, 2003.

[14] J. McGonigal, "What exactly is a game?," *Real. is broken why games make us better how they can Chang. world*, 2011.

[15] T. Lin, C. Jin, and M. I. Jordan, "Near-Optimal Algorithms for Minimax Optimization," in *Proceedings of Machine Learning Research*, 2020, vol. 125, pp. 1–42.

[16] K. K. Thekumprampil, P. Jain, P. Netrapalli, and S. Oh, "Efficient algorithms for smooth minimax optimization," *Adv. Neural Inf. Process. Syst.*, vol. 32, no. NeurIPS, pp. 1–12, 2019.