JURNAL INFOTEL
Informatika - Telekomunikasi - Elektronika

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Penerapan Algoritma Alpha Beta Pruning Sebagai Kecerdasan Buatan pada *Game* Pawn Battle

Ridho Rahman Hariadi[1], Imam Kuswardayan[2], Isye Arieshanti[3], Irooyan Alfi T.Z.[4]
[1,2,3,4]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
[1,2,3,4]Gedung Jurusan Teknik Informatika, Jl. Teknik Kimia Kampus ITS Sukolilo, Surabaya
Email korespondensi: imam@its.ac.id

Dikirim 16 Januari 2017, Direvisi 6 Maret 2017, Diterima 6 April 2017

Abstrak – Catur merupakan *game* strategi yang dimainkan oleh dua orang. Ada dua jenis warna bidak pada permainan ini, yaitu hitam dan putih. Agar dapat memenangkan sebuah permainan, pemain harus menguasai strategi-strategi dalam bermain catur. Ada banyak strategi yang hanya dapat dipahami dengan banyak bermain dan berlatih. Modul-modul tentang cara bermain catur pada umumnya hanya menjelaskan kejadian yang biasa terjadi dalam permainan catur sehingga berlatih merupakan satu-satunya cara yang dapat digunakan untuk meningkatkan kemampuan dalam bermain catur. Penelitian ini merupakan penelitian implementasi yang menggunakan Algoritma Alpha Beta Pruning sebagai kecerdasan buatan dalam permainan catur. Algoritma yang biasanya digunakan dalam permainan catur adalah algoritma Min-Max. Algoritma Min-mAx merupakan algoritma yang digunakan untuk menemukan langkah terbaik dalam permainan catur, sedangkan Algoritma Alpha Beta Pruning adalah algoritma yang digunakan untuk mencegah perluasan cabang/*node* untuk mendapatkan hasil pencarian langkah yang lebih baik dari sebelumnya. Penelitian ini diharapkan dapat membantu memberikan gambaran penerapan Algoritma Alpha Beta Pruning yang digunakan dalam membangun sebuah kecerdasan buatan pada permainan catur.

Kata kunci – Catur, Algoritma Alpha Beta Pruning, Kecerdasan Buatan

Abstract—Chess is strategic board game. Chess is played by two people. There are two colors in Staunton chess set. There are black and white. To become a winner in chess game, player needs to understand the strategy in playing chess. There are many modules which can help to learn chess. Those modules explain how to play chess and many strategies which are generally happened in playing chess. To increase the ability of playing chess, practice can be the only way to get more experience in playing chess. This research is implementation research using Alpha Beta Pruning Algorithm as artificial intelligent in playing chess. Min-Max Algorithm is the usual algorithm used in chess game. Min-Max Algorithm is algorithm used to find best move. This algorithm is different from Alpha Beta Pruning. Alpha Beta Pruning can prevent more extended node in its process to fing best move. This will cause the result of Alpha Beta Pruning better than Min-Max. This research aims to give description of how to implement Alpha Beta Pruning Algorithm as artificial intelligent in chess game.

Keywords – Chess, Alpha Beta Pruning Algorithm, Artificial Intelligence

I. PENDAHULUAN

Permainan catur adalah permainan kuno yang sampai saat ini masih popular dimainkan. Permainan ini berkembang sangat pesat dan diminati oleh semua kalangan, dari orang dewasa hingga anak-anak. Permainan catur diketahui dapat mengasah pemikiran kognitif yang tidak memerlukan banyak tenaga seperti olahraga lain pada umumnya.

Dalam bermain catur, dibutuhkan strategi-strategi untuk yang dapat secara serta merta dihafal agar bisa memenangkan suatu permainan. Strategi-strategi tersebut dapat ditemukan berdasarkan hasil pengalaman bermain seseorang. Modul-modul strategi serta cara bermain catur yang ada, pada umumnya hanya menjelaskan kejadian yang biasa terjadi dalam permainan catur. Kejadian-kejadian khusus hanya dapat diperoleh ketika pemain benar-benar bermain dan berlatih.

Ada banyak aplikasi permainan catur yang dikembangkan. Aplikasi-aplikasi tersebut dapat menjadi alternatif dalam belajar serta berlatih bermain catur. Banyak kecerdasan buatan yang diimplementasikan untuk menciptakan sebuah permainan catur yang lebih menantang.

Untuk permainan dengan jenis strategi seperti pada permainan catur, Algoritma Min-Max merupakan algoritma yang banyaj digunakan [7]. Algoritma ini merupakan algoritma pencarian solusi terbaik. Selain Algoritma Min-Max, terdapat Algoritma Alpha Beta Pruning. Algoritma Alpha Beta Pruning berbeda dengan Algoritma Min-max. Algoritma Alpha Beta Pruning mengoptimalisasi jumlah percabangan yang terbentuk. Algoritma ini mampu mengurangi jumlah percabangan dengan solusi yang kurang atau bahkan lebih buruk dari solusi yang ada sebelumnya sehingga algoritma ini mampu memberikan hasil yang lebih baik jika dibandingkan dengan Algoritma Min-Max.

Algoritma A* merupakan algoritma pencarian solusi terbaik yang menggabungkan teknik dari Algoritma Djikstra dan DFS. Algoritma ini menggunakan *graph* berbobot tidak berarah sebagai dasar pencarian jejak [10]. Algoritma A* telah menerapkan nilai-nilai heuristik untuk menemukan solusi yang paling optimal [11]. Algoritma A* juga dapat digunakan dalam permainan strategi seperti catur untuk pencarian solusi terbaik. Penelitian yang dilakukan Fahrurrozi [12] mengimplementasikan Algoritma Iterative Deepening A* pada permainan Flow Free Color (permainan strategi sejenis catur dengan menggunakan papan ukuran tertentu). Algoritma Iterative Deepening A* yang diimplementasikan pada penelitian ini memberikan hasil yang baik pada pencarian solusi. Namun, dengan menggunakan pohon pencarian *Depth First Search*, aka nada banyak simpul yang terbentuk pada setiap pola jalur pencarian sehingga jika dibandingkan dengan Algoritma Alpha Beta Pruning, Algoritma A* memiliki tingkat komputasi yang lebih tinggi. Algoritma Alpha Beta Pruning adalah lebih baik jika dibandingkan dengan Algoritma A* pada pengukuran tingkat komputasi yang dihasilkan karena pada Algoritma Alpha Beta Pruning, jalur-jalur simpul yang tidak menghasilkan solusi akan diiabaikan.

Penelitian ini mengimplementasikan Algoritma Alpha Beta Pruning sebagai kecerdasan buatan dalam permainan catur. Permainan catur yang dibangun dengan mengimplementasikan algoritma tersebut bernama Pawn Battle. Pawn Battle dibangun dengan tujuan untuk membantu melatih seseorang dalam bermain catur sehingga dengan kecerdasan buatan yang baik dapat menciptakan suatu permainan catur yang lebih menantang. Pawn Battle merupakan permainan catur dengan satu pemain (*single player*) yang dapat digunakan untuk berlatih melawan kecerdasan buatan pada komputer. Pawn Battle membantu pemain untuk melatih pemikiran yang strategis dan cermat dalam bermain catur. Selain itu, Pawn Battle dibangun sebagai media hiburan.

Permainan Pawn Battle dibangun dengan menggunakan kakas bantu berupa Unity dan Blender. Bahasa pemrograman yang digunakan dalam pengembangan Pawn Battle adalah C#. Pawn Battle merupakan aplikasi yang dapat berjalan di komputer dengan *platform* Windows.

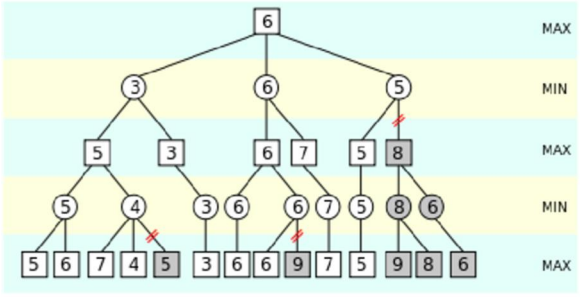
Penelitian ini bertujuan untuk mendapatkan gambaran bagaimana mengimplementasikan Algoritma Alpha Beta Pruning dalam permainan catur.

II. METODE PENELITIAN

A. Algoritma Alpha Beta Pruning

Algoritma Alpha Beta Pruning merupakan algoritma pencarian yang digunakan untuk mengurangi jumlah *node* melalui proses evaluasi. Sama halnya dengan Algoritma Min-Max, Algoritma Alpha Beta Pruning merupakan algoritma untuk mencari solusi yang optimal dari sebuah permasalahan, namun perbedaannya ada pada proses evaluasi yang digunakan untuk mengurangi jumlah *node* solusi yang dibentuk pada iterasi sebelumnya.

Algoritma Alpha Beta Pruning memiliki dua nilai, yaitu alpha dan beta [5]. Alpha merupakan nilai maksimum yang dimiliki oleh pemain, sedangkan nilai beta adalah nilai minimum yang dimiliki oleh lawannya (NPC). Nilai alpha adalah negatif tak terhingga pada kondisi awal permainan dan beta adalah positif tak terhingga. Pemilihan *node* kemungkinan pergerakan langkah akan selalu diperbarui pada setiap kali pemain selesai memindahkan pionnya. Jika terdapat sebuah *node* dengan nilai beta kurang dari sama atau sama dengan alpha, maka *node* tersebut tidak akan dipilih untuk perhitungan langkah berikutnya sehingga cabang di bawah pada *node* itu akan dipangkas (*prune*). Adapun ilustrasi dari Algoritma Alpha Beta Pruning tersebut digambarkan pada Gambar 1.



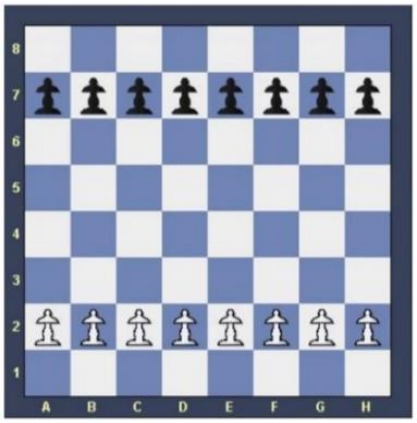
Gambar 1. Ilustrasi Algoritma Alpha Beta Pruning

Permainan dengan menggunakan komputer mengandalkan kemampuan komputer dalam memproses komputasi dimana komputer dituntut memberikan respon yang cepat pada pengguna. Pengimplementasian algoritma yang tepat dapat membantu komputer dalam dua hal tersebut. Algoritma Min-Max merupakan algoritma yang

banyak digunakan dalam permainan seperti catur, Othello serta tic tac toe [9]. Namun, diketahui bahwa algoritma ini masih memiliki tingkat komputasi yang tinggi [8] sehingga dalam penelitian ini diterapkan Algoritma Alpha Beta Pruning untuk membantu mengurangi komputasi tersebut.

B. Permainan Pawn Battle

Permainan yang dibangun pada penelitian ini diberi nama Pawn Battle. Pawn Battle memiliki aturan yang sama dengan permainan catur pada umumnya. Perbedaannya hanya pada jumlah bidak yang dimainkan. Pwn Battle memiliki 8 buah bidak pada setiap pemainnya. Delapan buah bidak tersebut adalah bidak pion yang terletak pada baris ke-2 dan baris ke-8 bidang catur seperti pada Gambar 2.



Gambar 2. Tata Letak Pion Pada Kondisi Awal Permainan

Untuk aturan permainan dari Pawn Battle adalah sama dengan aturan permainan pada permainan Pawn Battle pada umumnya. Aturan tersebut dapat diperoleh dari literatur [4].

C. Unity 3D dan Blender

Untuk membangun permainan Pawn Battle digunakan Unity dan Blender. Unity 3D merupakan sebuah kakas bantu yang dapat digunakan untuk membangun sebuah aplikasi permainan dalam berbagai macam *platform* yang berbeda seperti Android, iOS, XBox, Windows, Mac, Linux Wii serta Aculst Rift (Unity Game Development 2015), sedangkan Blender adalah kakas bantu yang dapat digunakan untuk merancang model 3 dimensi. Blender merupakan aplikasi yang dapat digunakan di berbagai macam *platform* yang berbeda seperti Linux, Windows maupun Macintosh [6]. Antarmuka Blender dibangun menggunakan OpenGL. Hal ini bertujuan agar Blender kompatibel pada ketiga *platform* tersebut. Blender memiliki fitur yang dapat mendukung keseluruhan model 3D mulai dari pembuatan model, *rigging*, pembuatan animasi, simulasi, *rendering*, *compositing*, dan pelacakan gerakan. Blender juga mendukung fitur pengeditan video serta pengembangan aplikasi permainan [2].

Unity 3D digunakan dalam membangun aplikasi Pawn Battle, dengan Bahasa pemrograman C# sehingga aplikasi Pawn Battle dikembangkan pada komputer dengan *platform* Windows dan Blender digunakan untuk memodelkan visualisasi 3D pada permainan.

Proses perancangan Pawn Battle dimulai dengan studi pustaka terkait tentang hal-hal yang berhubungan dengan Algoritma Alpha Beta Pruning, Unity 3D, Blender, dan konsep tentang kecerdasan buatan. Dari pustaka-pustaka tersebut kemudian dirancang kebutuhan permainan seperti aturan permainan, perancangan antarmuka, tahap level permainan, dan hal-hal lainnya yang diperlukan untuk membangun perangkat lunak. Setelah tahap perancangan, Pawn Battle kemudian diimplementasikan dalam Bahasa Pemrograman C# pada komputer Windows.

D. Perancangan Kasus Pengguna

Pada tahap perancangan kasus pengguna, dianalisis kebutuhan-kebutuhan pada permainan catur, seperti jumlah bidak yang dimainkan. Jumlah bidak yang dimainkan pada penelitian ini adalah delapan buah bidak pion, dimana letak bidak-bidak seperti ditunjukkan pada Gambar 2. Aturan permainan ini sama dengan aturan permainan catur tradisional, namun bidak pion hanya dapat melangkah ke depan dan tidak dapat mundur. Aplikasi ini dibangun dengan tujuan untuk menciptakan sebuah permainan yang variatid dan otomatis antara pemain dan kecerdasan buatan pada komputer.

Pada permainan ini terdapat tiga level kesulitan, yaitu *esay*, *medium*, dan *hard*. Yang membedakan masing-masing level tersebut adalah kedalaman *node* (kemungkinan langkah yang diambil). Pada level *easy*, pemain akan melawan NPC dengan kemampuan prediksi kemungkinan yang terjadi pada dua langkah ke depan, sedangkan pada level *medium*, NPC mengetahui prediksi kemungkinan yang terjadi pada empat langkah ke depan dan pada level *hard*, NPC dapat memprediksi kemungkinan terjadi pada lima langkah ke depan.

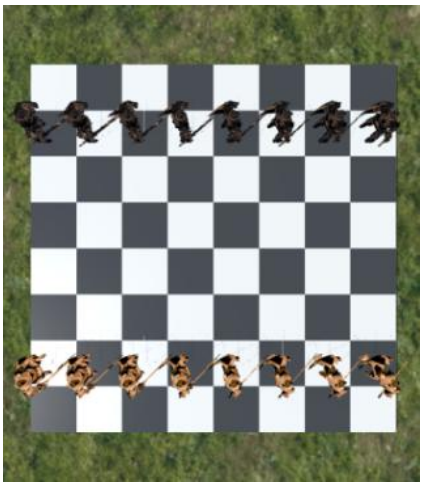
Permainan ini mengimplementasikan kecerdasanj buatan yang menggunakan komputasi dari komputer/PC yang digunakan sehingga dibutuhkan spesifikasi *Random Access Memory* (RAM) untuk menjalankan aplikasi ini dikarenakan perhitungan variasi langkah dari bidak pion bergantung dari spesifikasi RAM yang digunakan. Selain itu, aplikasi ini juga memiliki kebutuhan grafis agar visualisasi permainan dapat ditampilkan dengan baik tanpa efek *drop rate fps* yang menyebabkan visualisasi melambat (*lag*).

E. Perancangan Antarmuka

Antarmuka halaman pada Permainan Pawn Battle dimulai dengan tampilan halaman menu utama (Gambar 3). Pada Halaman tersebutm terdapat pilihan menu permainan, yaitu *easy*, *medium*, dan *hard*. Selanjutnya, ketika pemain telah memilih salah satu dari menu tersebut, akan ditampilkan halaman Main.

Pada halaman ini, posisi bidak catur ada pada posisi awal. Selain antarmuka halaman tersebut, terdapat juga antarmuka halaman kalah dan menang. Pada antarmuka halaman kalah atau menang tersebut terdapat pilihan untuk kembali ke menu utama dan mengulang permainan. Antarmuka kondisi awal permainan ditunjukkan pada Gambar 4, sedangkan antarmuka *Game Over* ditunjukkan pada Gambar 5.

Gambar 3. Antarmuka Halaman Menu



Gambar 4. Antarmuka Kondisi Awal Permainan

Gambar 5. Antarmuka Kondisi *Game Over*

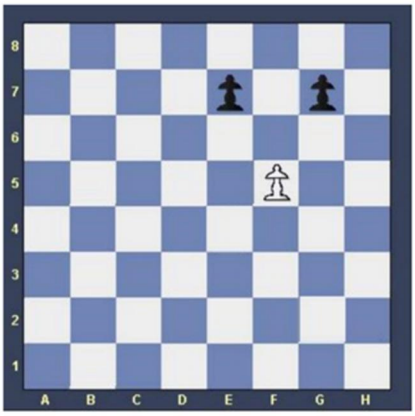
F. Perancangan Nilai Heuristik pada Kecerdasan Buatan

Alur permainan merupakan serangkaian proses yang dapat diikuti pemain dalam memainkan permainan. Peraturan permainan Pawn Battle sama persis dengan permainan catur dimana pion hanya bisa dijalankan untuk maju selangkah dan memakan pion lawan dengan melangkah miring pada diagonal di depan pion tersebut. Pion yang berada pada barisan paling depan dapat dijalankan dua langkah maju. Terdapat kondisi tertentu, yaitu *En Passant*, yaitu kondisi dimana pemain melangkah dua langka ke depan tanpa mempedulikan kondisi di samping kiri (jika ada pion lawan). Kondisi menang ditentukan dengan habisnya semua pion lawan atau pemain berhasil sampai ke seberang (posisi daerah lawan). Selain itu, terdapat kondisi *Stalemate* dimana pemain lawan dan NPC sama-sama tidak memiliki pion untuk dijalankan pada langkah selanjutnya.

Dalam permainan Pawn Battle ada beberapa komponen penting yang diimplementasikan. Komponen tersebut merupakan nilai heuristic (*heuristic value*) yang merupakan penentu dari tingkat kecerdasan dari NPC. Ada sepuluh nilai heuristic yang ditetapkan dalam permainan ini, antara lain sebagai berikut.

a) Pawn en Prise

Merupakan kondisi dimana pion NPC barada pada baris ke-5 dan pion pemain berada pada kanan dan kiri kolom di baris awal serta di depan pion tersebut tidak ada pion pemain (lawan) yang menghadang. Kondisi ini digambarkan pada Gambar 6.



Gambar 6. Kondisi *Pawn en Prise*

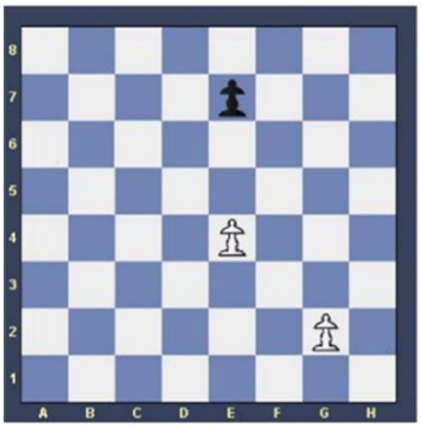
Jika kondisi ini dicapai oleh pemain, maka nilai *node* pemain akan ditambah (+3). Namun, jika kondisi ini dicapai oleh NPC (lawan), maka nilai *node* pemain akan ditambah (-3).

b) Menghitung Jumlah Pion

Jumlah pion merupakan selisih dari pion hitam dan pion putih dimana masing-masing pion akan dikalikan 5.

c) Menghitung Jumlah Pion Terlewat

Jumlah pion yang terlewat merupakan kondisi dimana sisi kanan dan kiri pion kosong (tidak ada pion lawan yang menghadang). Adapun kondisi ini digambarkan pada Gambar 7.

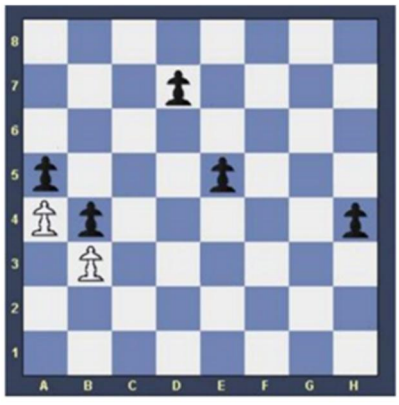


Gambar 7. Kondisi Pion Terlewat

Jika terjadi kondisi ini pada permainan (seperti pada Gambar 5), maka nilai *node* pion putih akan dikalikan (+5) ditambah dengan *node* pion hitam dikalikan.

d) Kondisi Akhir Permainan

Ada dua kondisi akhir dalam permainan, yaitu kondisi kalah dan kondisi seri. Kondisi menang adalah ketika salah satu pemain berhasil menghabiskan pion lawannya atau pion pemain berhasil menjangkau lokasi (daerah) pion lawan, sedangkan kondisi seri (*stalemate*) adalah kondisi dimana salah satu atau kedua pemain sudah tidak dapat lagi menggerakkan pionnya. Kondisi seri juga bisa terjadi ketika salah satu pemain hanya memiliki satu pion dan yang lainnya memiliki lebih dari satu pion. Gambar 8 mengilustrasikan kondisi seri tersebut.



Gambar 8. Kondisi Seri (*Stalemate*)

Pada kondisi akhir permainan, nilai heuristik akan diubah menjadi maksimal pada pemain yang menang dan akan diubah menjadi minimal untuk pemain yang kalah, sedangkan untuk kondisi *stalemate* atau seri, nilai heuristik akan diubah menjadi 0.

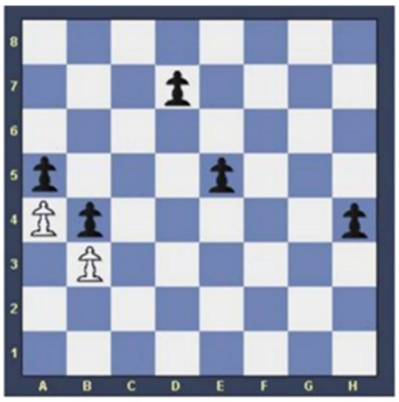
e) Jumlah Mayoritas Pion

Merupakan selisih dari total jumlah teman pada sisi kanan kiri pion. Jika pada satu baris yang sama, pion hitam beerjumlah 3 dan pion putih berjumlah 2, maka nilai mayoritas pion pada baris tersebut adalah 3-2 atau jumlah pion hitam dikurangi pion putih pada baris yang sama.

f) Jumlah Mayoritas yang Lumpuh

Jumlah mayoritas yang lumpuh merupakan kondisi ketika pion berada pada garis depan dan posisi pion tersebut tidak menguntungkan pergerakan dari pion musuh. Adapun ilustrasi dari kondisi ini digambarkan pada Gambar 9.

Jika terjadi kondisi seperti Gambar 9, maka jarak pion dengan garis akhir akan dikalikan (+2) ditambah dengan pion lawan yang dikalikan (-2).



Gambar 9. Kondisi Mayoritas Yang Lumpuh

g) Kombinasi *Backthrough*

Merupakan kondisi dimana masing-masing pion hitam dan putih sejajar dan berhadapan. Jika terjadi kondisi *backthrough*, maka pion yang menang adalah pion yang berada pada garis paling depan. Gambar 10 menggambarkan kondisi ini.

Gambar 10. Kondisi *Backthrough*

Nilai *node* pada kondisi ini dihitung dengan mengurangi jarak awal pion dengan jarak akhir pion dan hasilnya dikalikan (+2).

h) Jarak Maksimal Pion

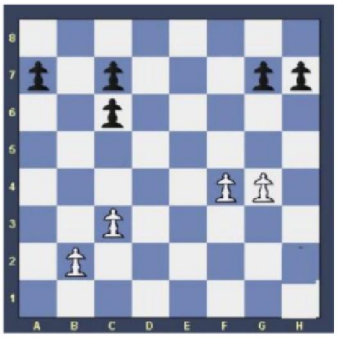
Jarak maksimal pion dihitung dengan mengurangi jarak terjauh pion pemain (jarak maksimal yang dicapai pion hitam dikurangi jarak maksimal pion putih).

i) Pion yang Terisolasi

Merupakan kondisi dimana salah satu pion hanya berjumlah satu, sedangkan lawannya telah mencapai garis depan. Adapun perhitungan nilai *node* untuk kondisi ini adalah dengan mengurangi jarak posisi awal dan akhir pion putih dikali 2 dengan jarak posisi awal dan akhir pion hitam dikali 2.

j) Pion Bertumpuk

Kondisi ini merupakan kondisi dimana terdapat dua atau lebih pion yang bertumpuk. Jumlah banyaknya pion tidak memberi pengaruh yang besar. Kondisi ini menyebabkan pergerakan dari masing-masing pemain kurang maksimal. Gambar 11 mengilustrasikan kondisi ini.



Gambar 11. Kondisi Pion Bertumpuk

Nilai *node* pada kondisi ini dihitung dengan mencari selisih dari pion bertumpuk hitam dan pion bertumpuk putih.

k) Tabel Nilai Heuristik

Tabel 1 merupakan rangkuman nilai heuristik dari uraian penetapan sepuluh nilai heuristik.

Tabel 1. Rangkuman Nilai Heuristik

No	Nama Strategi	Heuristic Value
1	Pawn en prise (pp)	±3
2	Menghitung pion (np)	±5
3	Pion yang terlewat (pp)	±5
4	Kondisi Akhir (fc)	±∞
5	Mayoritas (m)	±1
6	Mayoritas yang lumpuh (dm)	±2
7	Kombinasi backthrough (bc)	±2
8	Jarak maksimal pion (md)	±1
9	Pion yang terisolasi (ip)	±2
10	Pion yang betumpuk (sp)	±1

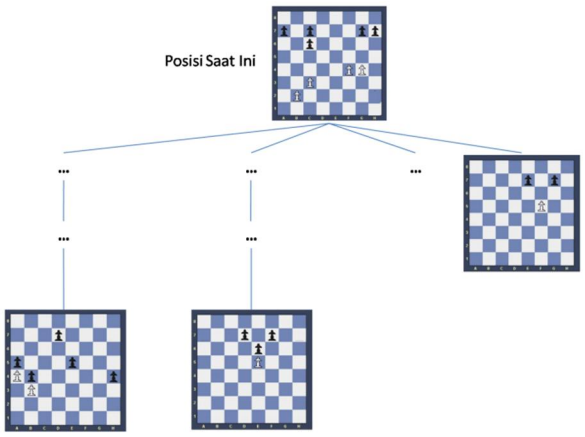
G. *Some Common Mistakes*

Algoritma Alpha Beta Pruning adalah algoritma pencarian tingkat kedalaman (*Depth First Search*) yang memodelkan data ke dalam bentuk *tree* (pohon). Level kedalaman dari pencarian masing-masign *node* adalah bergantung dari nilai *node* yang ditentukan dari nilai heuristik. Pada algoritma ini, terdapat nilai alpha dan beta. Nilai alpha merupakan nilai heuristik dari NPC dan nilai beta merupakan nilai heuristik dari pemain. Perluasan suatu *node* adalah berdasarkan aturan-aturan yang telah ditetapkan (*Rule Based*). Adapun pseudocode dari Algoritma Alpha Beta Pruning adalah sebagai berikut.

```
function alphabeta(node, depth, α, β, maximizingPlayer)
    if depth = 0 or node is a terminal node
        return the heuristic value of node
    if maximizingPlayer
        v := -∞
        for each child of node
            v := max(v, alphabeta(child, depth - 1, α, β, FALSE))
            α := max(α, v)
            if β ≤ α
                break (* β cut-off *)
        return v
    else
        v := ∞
        for each child of node
            v := min(v, alphabeta(child, depth - 1, α, β, TRUE))
            β := min(β, v)
            if β ≤ α
                break (* α cut-off *)
        return v
```

Gambar 12. Pseudocode Algoritma Alpha Beta Pruning

Gambar 13 menunjukkan visualisasi dari Algoritma Alpha Beta Pruning dalam mencari solusi terbaik pada permainan catur.



Gambar 13. Ilustrasi Algoritma

III. HASIL DAN PEMBAHASAN

Setelah tahap implementasi, pengembangan permainan Pawn Battle dilanjutkan dengan proses pengujian. Pada tahap pengujian, perangkat lunak akan diujicobakan berdasarkan fitur-fiturnya atau yang dikenal dengan sebutan *Usability Testing* [3]. Adapun rangkuman hasil pengujian tersebut ditunjukkan pada Tabel 2.

Pada fitur memilih menu utama, pemain diminta untuk memilih level atau tingkat kesulitan dalam permainan. Ada tiga level kesulitan dalam permainan ini, yaitu *easy*, *medium*, dan *hard*. Pengujian pemilihan level pada menu utama ini berjalan dengan baik dan pemain dapat bermain sesuai dengan level yang dpilih.

Pada level *easy*, pengujian dimulai dengan proses perhitungan kemungkinan langkah terbaik serta perhitungan nilai heuristik. Dalam hal ini, pencocokan dilakukan secara manual dengan mencocokkan nilai dari hasil perhitungan sistem dan nilai hasil perhitungan manual. Pada level *easy*, NPC akan menghitung kemungkinan terbaik pada 2 langkah selanjutnya sehingga dilakukan *expand* terhadap *node* sebanyak dua kali (dua level) karena hanya melakukan *expand node* sebanyak dua kali, komputer menghasilkan nilai perhitungan kemungkinan yang sama. Dalam algoritma yang digunakan, jika *node* memiliki nilai yang sama, makan digunakan *node* yang pertama kali dihitung.

Pada level *medium* dan level *hard*, pengujian juga dilakukan dengan mekanisme yang sama, dimana yang berbeda adalah jumlah kemungkinan langkah yang diprediksi pada masing-masing level. Level *medium* akan menghitung 4 langkah kemudian sehingga komputer akan meng-*expand node* sebanyak 4 (ke dalam levelnya adalah 4). Pada level ini, kemungkinan gerakan pion akan bermacam-macam.

Pada level *hard*, jumlah kemungkinan langkah yang dihitung adalah 5 langkah kemudian sehingga dengan jumlah ke dalam sebanyak 5, *node* akan di-*expand* dan menghasilkan lebih banyak variasi langkah pada permainan. Namun, dari langkah yang dilakukanm dapat diperkirakan bahwa gerakan pion akan lebih banyak menghindar pion termakan oleh pion lawan dan saling menjaga agar pion lainnya tetao aman.

Dari lima kali pengujian pada masing-masing level *easy*, *medium*, dan *hard*, maka hasilnya dapat dirangkum pada Tabel 2. Tabel tersebut menjelaskan performa NPC dalam menghadapi pemain pada permainan Pawn Battle.

Tabel 2. Hasil pengujian NPC pada pemain

Level	1	2	3	4	5	Menang
Easy	O	O	-	X	-	2
Medium	O	-	O	-	-	2
Hard	O	O	-	O	O	4

Pada tabel 2, terdapat tanda “O”, “-“, dan “X”. Tanda “O” berarti menang, tanda “-“ berarti seri, dan tanda “X” berarti kalah. Berdasarkan Tabel 3, dijelaskan bahwa NPC memiliki jumlah kalah yang kecil. Pada level *easy*, NPC kalah satu kali melawan pemain, sedangkan pada level *medium* dan *hard*, NPC memiliki jumlah menang lebih banyak jika dibandingkan dengan pemain.

IV. PENUTUP

A. Kesimpulan

Pawn Battle merupakan permainan catur yang diharapkan dapat membantu pemain dalam berlatih bermain catur melawan komputer (NPC). Kecerdasan buatan yang diimplementasikan pada NPC di permainan ini adalah Algoritma Alpha Beta Pruning. Dari hasil pengujian yang telah dilakukan, Algoritma Alpha Beta Pruning dapat diimplementasikan dengan baik pada permainan Pawn Battle. NPC memiliki jumlah menang lebih banyak dibandingkan dengan pemain sehingga kecerdasan buatan yang dibangun dengan menggunakan algoritma ini dapat digunakan sebagai alternatif dalam berlatih bermain catur.

B. Saran

Saran dari penelitian ini ditujukan bagi para pengembang permainan catur. Ada banyak algoritma optimasi yang dapat digunakan dalam membangun kecerdasan buatan dalam permainan catur. Salah satunya adalah Algoritma Alpha Beta Pruning. Algoritma Alpha Beta Pruning memodelkan setiap kemungkinan pergerakan menjadi cabang-cabang pohon untuk mencari solusi terbaik dari masing-masing kemungkinan pergerakan yang terjadi. Semakin banyak pergerakan langkah yang dicari kemungkinan terbaiknya, maka akan semakin besar tingkat komputasi dari komputer yang digunakan. Hal ini dapat ditunjukkan oleh kemampuan komputer dalam menggerakkan pion NPC pada tiap levelnya. Pada level *easy*, komputer memiliki tingkat visualisasi yang lebih cepat jika dibandingkan pada level medium ataupun *hard*. Hal ini dikarenakan pada level *easy*, jumlah kemungkinan pergerakan langkah terbaik yang di-*expand* hanya sebatas dua level kedalaman cabang pohon, sedangkan pada level *medium* dan *hard*, jumlah kemungkinan langkah terbaik yang di-*expand* adalah empat hingga lima level kedalaman cabang pohon sehingga kemungkinan pergerakan pion pada level *medium* dan *hard* lebih banyak jika dibandingkan dengan kemungkinan pergerakan pion pada level *easy*. Hal inilah yang menyebabkan tingkat komputasi pada level *medium* dan *hard* meningkat.

Algoritma Alpha Beta Pruning merupakan perbaikan dari Algoritma Min-Max. pada awal perkembangannya, algoritma ini banyak digunakan sebagai solusi dalam penyelesaian berbagai permainan strategi seperti catur, Amazon Breakthrough serta Havannah [13][14]. Namun, setelah dikenal paradigm paradigm algoritmik Monte-Carlo Tree Search (MCTS), banyak penelitian beralih. MCTS merupakan algoritma pencarian yang menerapkan konsep nilai sampel distribusi kemungkinan dari masing-masing solusi yang dihasilkan. MCTS terbukti lebih baik menghasilkan solusi dibandingkan Algoritma Min-max. Selain itu, MCTS juga mungkin dipadukan dengan Algoritma Min-Max dan Algoritma Alpha Beta Pruning. Algoritma perpaduan ini dikenal dengan MCTS-EPT (MCTS dengan Early Playout Termination atau kondisi akhir permainan yang diketahui lebih awal. Hal ini adalah berdasarkan nilai kemungkinan yang dihitung). MCTS menjadi solusi yang lebih baik pada permainan yang memiliki factor percabangan yang besar [15][16]. Untuk penelitian selanjutnya, diharapkan MCTS dapat diterapkan untuk mendapatkan hasil yang lebih baik.

DAFTAR PUSTAKA

[1] U. Technologies, "Unity - unity - overview," Unity, 2015. [Online]. Available: http://www.unity3d.com/unity. Accessed: Jun. 2, 2015.

[2] B. Foundation, "About," blender.org, 2015. [Online]. Available: http://www.blender.org/about. Accessed: Jul. 2, 2016.

[3] "Methods - usability testing," 2002. [Online]. Available: http://www.usabilityfirst.com/usability-methods/usability-testing/. Accessed: Jan. 23, 2016.

[4] "Pawn Battle Strategies," in Kenil Worth Chess Club. [Online]. Available: http://www.kenilworthchessclub.org/media/Pawn_Battle_Strategies.pdf. Accessed: Jun. 23, 2015.

[5] S. J. Russell and P. Norvig, Artificial intelligence: A modern approach ; [the intelligent agent book], 2nd ed. United States: Prentice Hall/Pearson Education, 2010.

[6] Blender Foundation, "Blender Supported Platforms". 26 April 2015. [Online]. Available:http://wiki.blender.org/index.php/Dev:Ref/Supported_platforms. [Accessed 3 June 2015]

[7] "Currently known best algorithm(s) for computer chess?," 2010. [Online]. Available: http://stackoverflow.com/questions/2026262/currently-known-best-algorithms-for-computer-chess. Accessed: Jun. 3, 2015.

[8] Ilham, Anwari 2008, Penerapan Algoritma Minimax dengan Optimasi MTD(f) pada Permainan Catur, Jurnal, ITB, Bandung.

[9] Wikipedia, "Wikipedia Minimax", 4 April 2015. [Online]. Available: https://en.wikipedia.org/wiki/Minimax. [Accessed 3 June 2015]

[10] Hermawan L, Bendi RK. Penerapan Algoritma A* Pada Aplikasi Puzzle. Information Technology. 2013;4:23.

[11] Panggabean, I. B. T., Suryadharma, Y., & Nugroho, P. (2006). Penyelesaian Permasalahan 8 Puzzle dengan Menggunakan Algoritma A*(A Star). Dalam Makalah Mahasiswa Tahun, 2005-2006.

[12] Fahrurrozi, Implementasi Algoritma Iterative Deepening A8 dan metode Pruning Pada Solusi Permainan Puzzle Flow Free Color, 2010

[13] D.E Knuth, R.W. Moore, An Analysis of Alpha Beta Pruning, Artificial Intelligence, Vol6(4), 1975, pp. 293-326

[14] J. Schaeffer, A.Plaat, New Advances in Alpha Beta Searching

[15] C. Browne, D. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothariks, C. Colton, A Survey of Monte Carlo Tree Search Methods, IEEE Trans. Comput. Intell. AI Games, Vol(1), 2012, pp. 1-49

[16] R. Coulom, Efficient Selectivity and Backup Operators in Monte Carlo Tree Search, Computer and Games, 5th International Conference, CG 2006, Italy, Turin, Italy, May 29-31, pp. 72-83