Rancangan Permainan Othello Berbasis Android Menggunakan Algoritma *Depth-First Search*

[1]Mauza Saputri Handayani, [1]Dedy Arisandi, [1]Opim Salim Sitompul

[1]Program Studi S1 Teknologi Informasi
Fakultas Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara
E-mail: mauza.saputri@gmail.com | dedyarisandi@usu.ac.id | opim@usu.ac.id

***Abstrak*—Aplikasi *game* merupakan aplikasi yang banyak diminati oleh pengguna *mobile phone* untuk mendapatkan hiburan dan edukasi. Perkembangan *game* didukung oleh semakin canggihnya teknologi *mobile phone* yang ada baik secara model maupun *operating system*. Android adalah salah satu jenis sistem operasi *mobile phone* yang sedang berkembang saat ini. Salah satu jenis *game* bersistem operasi android pada *mobile phone* yang beredar luas antara lain *game* kecerdasan buatan, misalnya Othello. Othello merupakan salah satu permainan papan berbasis strategi yang dimainkan oleh dua pemain pada papan yang berukuran 8 baris dan 8 kolom. Setiap pemain memiliki bidak berbeda warna yaitu hitam dan putih. Penerapan kecerdasan buatan menggunakan algoritma DFS (*Depth First Search*) dengan algoritma Negamax yang dioptimasi dengan *Alpha Beta Pruning* ini dapat mengurangi ruang pencarian sehingga proses penelusuran dan evaluasi dapat dilakukan lebih cepat. Aplikasi ini dikembangkan dengan menggunakan bahasa pemrograman Java berbasis Eclipse IDE. Hasil yang diperoleh adalah sebuah aplikasi permainan Othello yang dapat diterapkan pada *mobile phone* berbasis Android.**

Kata Kunci—Othello, *Depth First Search*, Negamax, *Alpha Beta Pruning*, Android.

I. PENDAHULUAN

Aplikasi permainan (*game*) saat ini telah menjadi bagian yang tidak terpisahkan dari pengguna *mobile phone*. Dari berbagai aplikasi yang ada sampai saat ini aplikasi *game* merupakan aplikasi yang banyak diminati oleh *user* untuk mendapatkan hiburan dan edukasi. Sebagian besar waktu pengguna *mobile phone* dilakukan untuk bermain *game*. Hal ini terbukti dengan munculnya berbagai macam *game* yang berbeda-beda pada *mobile phone*. Perkembangan *game* didukung oleh semakin canggihnya teknologi *mobile phone* yang ada baik secara model maupun *Operating System*.

Perancangan aplikasi Othello ini difokuskan kepada bagaimana membuat agen cerdasnya. Agen adalah sesuatu yang dapat menerima kesan dari lingkungannya dan melakukan tindakan terhadap lingkungannya [1]. Dengan adanya agen cerdas pada aplikasi Othello, diharapkan aplikasi tersebut dapat berpikir dan menentukan pilihan langkah yang cerdas sehingga membuat para pemain terlibat untuk mengasah dan mengatur strategi untuk mengalahkan agen cerdas tersebut. Dalam perancangan agen cerdas penulis menggunakan algoritma *Depth-First Search*.

Algoritma *Depth-First Search* adalah algoritma pencarian pada sebuah pohon dengan menelusuri satu cabang sebuah pohon sampai menemukan solusi. Pencarian dilakukan pada satu node dalam setiap level dari yang paling kiri dan dilanjutkan pada node sebelah kanan Jika solusi ditemukan maka tidak diperlukan proses *backtracking* yaitu penelusuran balik untuk mendapatkan jalur yang diinginkan. Pada algoritma DFS pemakaian memori tidak banyak karena hanya node-node pada lintasan yang aktif saja yang disimpan. Selain itu, jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya secara cepat [2].

Untuk menentukan pilihan langkah selanjutnya agar memperkecil kemungkinan kehilangan nilai maksimal dalam permainan Othello ini, penulis akan menggunakan algoritma Negamax. Algoritma Negamax merupakan bentuk sederhana dari algoritma Minimax yang melakukan pencarian dengan menggunakan teknik algoritma DFS yang akan menelusuri setiap node untuk memperoleh hasil yang maksimum, namun jika kedalaman dan percabangan pohon terlalu besar maka algoritma Negamax akan memerlukan waktu yang sangat lama untuk mengambil keputusan. Untuk mempersingkat waktu pencarian sekaligus sebagai optimasi, maka digunakanlah algoritma *Alpha Beta*

Pruning. *Alpha Beta Pruning* merupakan algoritma yang akan mengurangi ruang pencarian Negamax.

Adapun tujuan dari penulisan ini adalah mengimplementasikan algoritma DFS dengan menggunakan algoritma Negamax yang dioptimasi dengan *Alpha Beta Pruning* pada permainan Othello dan dapat menerapkannya pada *mobile phone* berbasis Android.

Sedangkan, manfaat dari penulisan ini adalah mampu memberikan pemahaman kepada penulis dan pembaca tentang bagaimana cara kerja kecerdasan buatan pada permainan Othello yang menggunakan algoritma DFS dengan algoritma Negamax yang dioptimasi dengan *Alpha Beta Pruning*. Selain itu, aplikasi yang dibangun pada *mobile phone* berbasis Android ini diharapkan dapat digunakan masyarakat secara luas.

II. IDENTIFIKASI MASALAH

Pencarian langkah agen cerdas dalam permainan Othello tidak dapat dilakukan dengan cara penilaian manual, karena adanya pertimbangan dan strategi yang harus diketahui oleh agen sebelum mengambil keputusan langkah terbaik. Oleh karena itu, penulis akan membuat sebuah aplikasi *game* Othello pada *mobile phone* berbasis Android dengan menerapkan algoritma DFS menggunakan algoritma Negamax yang dioptimasi dengan *Alpha Beta Pruning* yang diharapkan dapat membantu mencari solusi pada permainan Othello tersebut.

Permasalahan yang dibahas dalam skripsi ini adalah bagaimana meciptakan sebuah agen cerdas pada permainan Othello yang dapat menemukan solusi secara tepat.

III. PENELITIAN TERDAHULU

Beberapa metode penelitian yang pernah dilakukan untuk menyelesaikan permainan Othello antara lain:

1) Implementasi Algoritma *Greedy* pada Permainan Othello.
   Algoritma *Greedy* dapat memecahkan masalah optimum, namun tidak selalu menghasilkan solusi yang optimum. Prinsip algoritma *greedy* pada setiap langkah adalah mengambil pilihan terbaik yang dapat diperoleh saat itu tanpa memperhatikan konsekuensi ke depan, dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan menghasilkan optimum global pada akhir proses [3].
2) Pengembangan Permainan Othello Menggunakan Algoritma Minimax, Pencarian Heuristik dan Logika Samar.
   Algoritma Pencarian Heuristik merupakan metode yang paling efektif diantara ketiga algoritma yang telah dipilih untuk menjadi basis perekayasaan sebuah kecerdasan buatan tanpa kemampuan belajar yang dirancang untuk pemain pemula [4].
3) Penerapan Konsep Algoritma Minimax dengan Menggunakan Algoritma *Breadth-First Search* (BFS) pada Permainan Reversi.
   Algoritma ini akan memungkinkan terjadinya beberapa *cut-off* sehingga waktu eksekusi untuk algoritma ini akan lebih efisien [5]. Namun *Breadth-First Search* (BFS) membutuhkan memori yang cukup banyak, karena harus menyimpan semua *node* yang pernah dibangkitkan dalam satu pohon [1].

IV. METODOLOGI PENELITIAN

A. *Othello*

Permainan Othello adalah permainan yang dimainkan oleh dua orang pemain. Permainan ini dimainkan diatas papan Othello persegi yang terdiri dari 8 baris dan 8 kolom kotak-kotak kecil. Pada awal permainan akan diletakkan dua koin hitam dan dua koin putih pada tengah-tengah papan.

Aturan permainan Othello secara umum adalah sebagai berikut [6]:

1) Setiap pemain memulai permainan dengan masing-masing memiliki 32 koin, salah satu sisi pada setiap koin berwarna hitam dan sisi koin lainnya berwarna putih
2) Pemain yang menggunakan koin hitam akan bermain terlebih dahulu.
3) Bila pemain koin hitam yang akan bermain, maka koin hitam harus diletakkan di kotak kosong yang berdekatan dengan koin putih. Dan begitu juga kondisinya untuk pemain yang menggunakan koin putih.
4) Pemain harus memainkan langkah dalam posisi "mengepung" koin lawan. Dengan kondisi bahwa saat pemain melakukan langkah maka koin dengan warna yang berlawanan berada diantara koin dengan warna yang dimainkan pemain. Koin yang "terkepung" akan berbalik menjadi koin yang sedang dimainkan pemain.
5) Apabila salah satu pemain tidak bermain karena tidak ada kotak yang sesuai dengan aturan nomor 3, maka pemain yang satunya lagi yang bermain.
6) Apabila kedua pemain sama-sama tidak dapat mengambil langkah lagi, maka permainan berakhir.

B. *Kecerdasan Buatan*

Artificial Intelligence atau Kecerdasan Buatan merupakan bagian dari ilmu pengetahuan komputer yang khusus ditujukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Sistem memperlihatkan sifat-sifat khas yang dihubungkan dengan

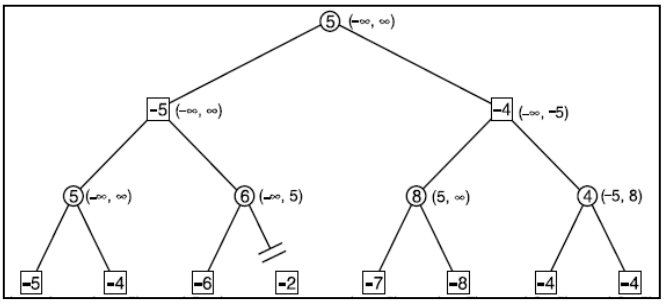
kecerdasan dalam kelakuan atau tindak-tanduk yang sepenuhnya bisa menirukan beberapa fungsi otak manusia, seperti pengertian bahasa, pengetahuan, pemikiran, pemecahan masalah dan lain sebagainya [7].

C. Algoritma

Depth-First Search adalah algoritma pencarian pada sebuah pohon dengan menelusuri satu cabang sebuah pohon sampai menemukan solusi. Pencarian dilakukan pada satu *node* dalam setiap level dari yang paling kiri. Jika pada level yang paling dalam, solusi belum ditemukan, maka pencarian dilanjutkan pada *node* sebelah kanan.

Untuk mengatasi banyaknya *node* yang ditelusuri oleh Negamax, perlu dibuat optimasi yang dapat mereduksi kemungkinan yang akan dianalisis. *Alpha Beta Pruning* merupakan optimasi dari algoritma Negamax yang mengurangi jumlah *node* yang dievaluasi oleh pohon pencarian. Dengan algoritma ini hasil optimasi dari algoritma Negamax tidak akan berubah.

Alpha Beta Pruning menelusuri pohon permainan dengan meletakkan 2 nilai pada setiap *node*, yaitu alpha (α) dan beta (β). Nilai α ditetapkan sama dengan -∞ sedangkan nilai β sama dengan +∞. Jika α < β, maka kesempatan untuk mencari langkah terbaik masih ada dan pencarian akan tetap dilanjutkan. *Node* akan melakukan maksimalisasi dan memperbaiki nilai α dari nilai anak-anaknya, kemudian nilai yang telah memperbaiki nilai α tersebut akan dibandingkan dengan nilai β sementara. Jika α > β, maka evaluasi dihentikan [8].



Gambar 1. Cara kerja algoritma *Alpha Beta Pruning* [9]

Sama halnya seperti algoritma Negamax, algoritma ini juga melakukan penelusuran dengan menggunakan *Depth First Search* untuk menelusuri pohon permainan sampai batas kedalaman yang telah ditentukan dan berhenti mengevaluasi langkah ketika terdapat minimal satu langkah yang lebih buruk daripada langkah yang dievalusi sebelumnya, sehingga langkah berikutnya tidak perlu dievaluasi lebih jauh.

D. Analisis Sistem

Penulis menggunakan array untuk menyimpan nilai-nilai yang diberikan pada setiap kotak dalam permainan. Penulis memberikan nilai index yang dimulai dari 0 s/d 63 sesuai dengan ukuran papan Othello yang terdiri dari 64 kotak. Nilai-nilai tersebut akan dipanggil sesuai dengan posisi index bidak yang akan kita letakkan.

Dalam permainan Othello pengambilan posisi strategis lebih penting daripada jumlah keping yang diambil sehingga untuk bisa membuat kecerdasan buatan yang kompeten dalam permainan Othello sangat dibutuhkan sistem penilaian yang dinamis dan adaptif karena dalam kasus-kasus berbeda, nilai posisinya juga berbeda [4]. Penulis menggunakan strategi nilai posisi sebagai perubah langkah bidak dalam permainan. Penulis menentukan pemberian nilai sesuai dengan strategi dalam permainan Othello, dimana pada posisi tertentu merupakan posisi yang lebih menguntungkan dari pada posisi yang lain. Gambar 2 berisi nilai-nilai posisi di setiap kotak permainan.

2000	300	10	10	10	10	300	2000
300	300	10	10	10	10	300	300
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
300	300	10	10	10	10	300	300
2000	300	10	10	10	10	300	2000

Gambar 2. Nilai-nilai posisi

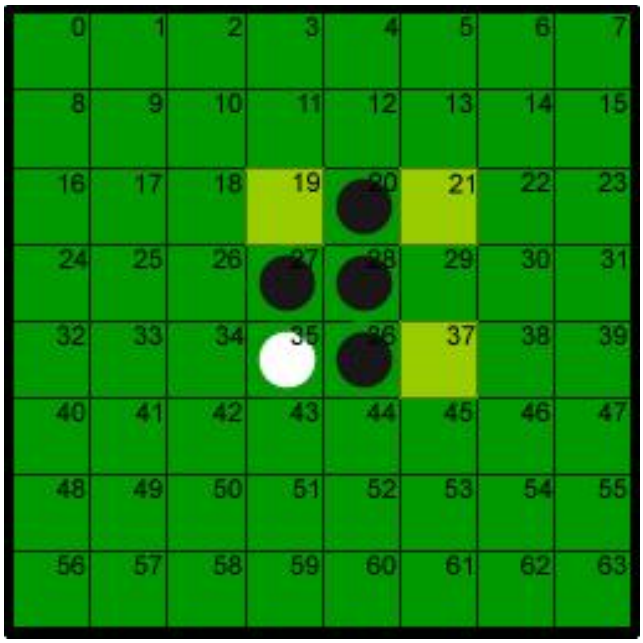
Setiap kotak dalam permainan diberi suatu nilai yang digunakan untuk mengevaluasi langkah bidak. Penulis membagi nilai-nilai tersebut menjadi 3 bagian, yaitu:

1) Nilai Posisi Pojok
   Posisi pojok merupakan posisi yang paling menguntungkan dalam permainan Othello. Bidak yang diletakkan pada posisi tersebut tidak akan dapat dibalikkan sampai akhir permainan. Oleh karena itu, penulis memberikan nilai 2000 untuk nilai di posisi pojok.
2) Nilai Posisi Setelah Pojok
   Posisi setelah pojok atau posisi pinggir yang berbatasan dengan pojok adalah posisi yang sangat menguntungkan jika dapat dimanfaatkan dengan baik menjelang akhir permainan. Namun, posisi ini sangat berbahaya jika dimainkan pada awal permainan karena pojok yang berbatasan dengannya akan diambil oleh lawan. Penulis memberikan kerugian senilai 300 untuk nilai posisi setelah pojok, dengan adanya pertimbangan

tambahan kerugian senilai 300 makn dapat menghindari keputusan untuk mengambil langkah tersebut, khususnya pada awal permainan dimana perhitungan evaluasi nilai masih sedikit sehingga nilai 300 sudah merupakan nilai yang cukup besar.

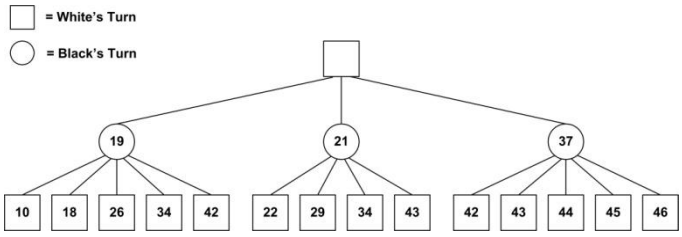
3) Nilai Posisi Dasar
Penulis memberikan nilai 10 karena nilai tersebut tidak memberikan pengaruh yang sangat signifikan pada permainan.

Berikut adalah salah satu kondisi permainan Othello pada aplikasi ini.



Gambar 3. Kondisi permainan pada Othello

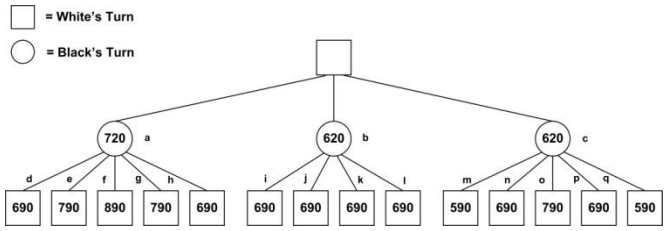
Pada Gambar 3 penulis dapat membuat suatu pohon permainan yang dapat merepresentasikan contoh permainan. Pohon permainan harus sesuai dengan segala kondisi permainan yang berlaku mulai dari langkah-langkah yang boleh dilakukan sampai aturan-aturan permainan. Gambar 4 merupakan pohon permainan yang diperoleh dari kondisi permainan pada Gambar 3.



Gambar 4. Pohon permainan pada Othello

Gambar 4 menjelaskan giliran bermain pada sisi putih memiliki 3 kemungkinan langkah yaitu 19, 21, dan 37. Jika sisi putih memilih langkah 19, maka ketika giliran sisi hitam bermain akan memiliki 5 kemungkinan langkah yang sesuai dengan aturan aturan permainan yaitu 10, 18, 26, 34, dan 42, demikian seterusnya.

Setelah membangun suatu pohon permainan, maka penulis memberikan nilai evaluasi pada setiap langkah yang terjadi pada pohon permainan. Nilai evaluasi dihitung dari sudut pandang satu pemain, dalam kondisi ini penulis mengevaluasi langkah dari sisi AI (putih).



Gambar 5. Pohon dengan nilai evaluasi pada setiap langkah

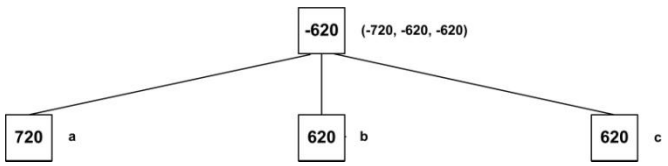
Pada Gambar 5 penulis mengganti setiap langkah pada pohon permainan, dengan nilai evaluasi pada langkah tersebut sesuai dengan nilai-nilai evaluasi yang telah dijelaskan pada pembahasan sebelumnya. Tabel 1 merupakan penjelasan bagaimana nilai-nilai tersebut diperoleh.

Tabel 1. Proses Menghitung Nilai Evaluasi

Node	Langkah (p=putih, h=hitam)	Score	Langkah Legal	Nilai Lawan	Jumlah Bidak Player	Total Bidak	Nilai Bidak	Nilai Akhir
a	p(19)	0	7	100	4	6	10	0+7*100+(2*4-6)*10 = 720
b	p(21)	0	6	100	4	6	10	0+6*100+(2*4-6)*10 = 620
c	p(37)	0	6	100	4	6	10	0+6*100+(2*4-6)*10 = 620
d	h(10)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
e	h(18)	0	8	100	3	7	10	0+8*100+(2*3-7)*10= 790
f	h(26)	0	9	100	3	7	10	0+9*100+(2*3-7)*10= 890
g	h(34)	0	8	100	3	7	10	0+8*100+(2*3-7)*10= 790
h	h(42)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
i	h(22)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
j	h(29)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
k	h(34)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
l	h(43)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
m	h(42)	0	6	100	3	7	10	0+6*100+(2*3-7)*10= 590
n	h(43)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
o	h(44)	0	8	100	3	7	10	0+8*100+(2*3-7)*10= 790
p	h(45)	0	7	100	3	7	10	0+7*100+(2*3-7)*10= 690
q	h(46)	0	6	100	3	7	10	0+6*100+(2*3-7)*10= 590

Dari nilai-nilai evaluasi tersebut maka akan diketahui bagaimana AI memperoleh nilai maksimum dengan mempertimbangkan nilai maksimum dari lawan.

Penulis akan menjelaskan cara kerja algoritma Negamax dari kondisi permainan pada Gambar 3, dengan menggambarkan sebuah pohon pencarian sebagai berikut:



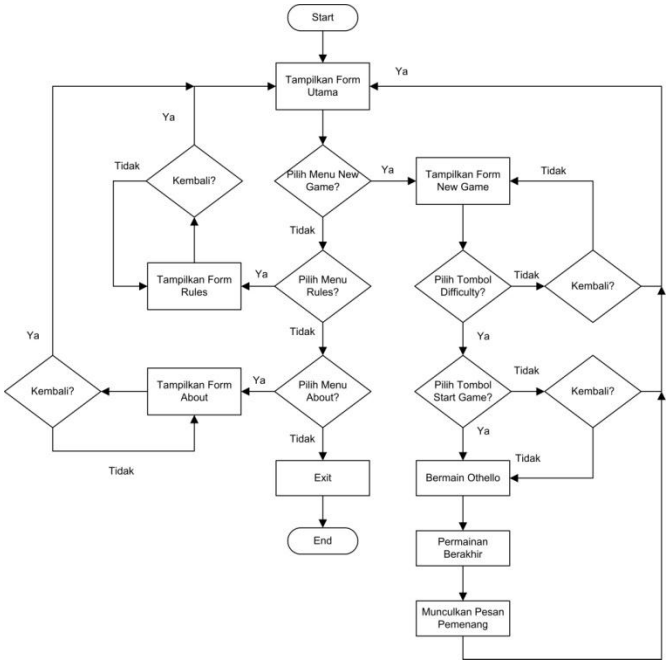
Gambar 6. Proses perubahan nilai *node* dengan algoritma Negamax

Pada Gambar 6 terdapat 3 langkah legal bagi AI yang memiliki nilai a=720, b=620, dan c=620. Pada Negamax, nilai-nilai tersebut merupakan bentuk asumsi bahwa nilai yang ada merupakan nilai maksimal bagi lawan, dan AI

akan mengevaluasi nilai yang merupakan negasi dari nilai tersebut, dengan tujuan untuk menghindari lawan mengambil nilai maksimal. Maka dapat diketahui nilai maksimal untuk AI adalah -620. Dalam kondisi ini terdapat dua nilai maksimal yang sama yaitu -620. Oleh karena itu AI akan memilih nilai yang pertama kali ditelusuri atau didapatkan.

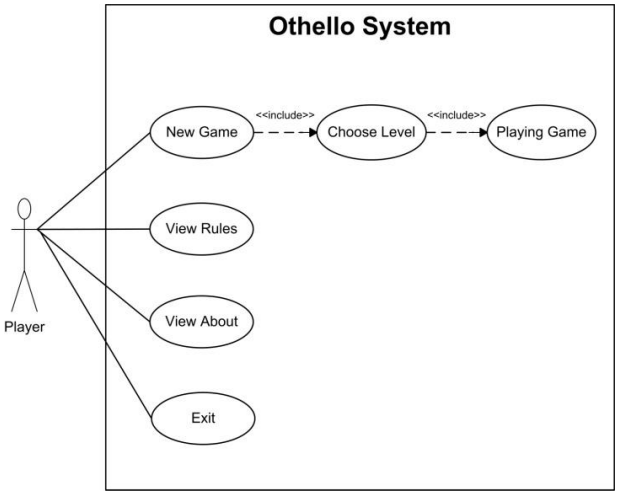
E. Perancangan Sistem

Aplikasi ini dimulai dengan menampilkan *form* Utama yang berisikan menu pilihan *New Game*, *Rules*, *About*, dan *Exit*. Pada setiap pilihan ini, akan diberikan tampilan *form-form* berikutnya. Aplikasi akan berhenti jika pengguna memilih tombol *Exit*. Untuk lebih jelasnya dapat dilihat pada Gambar 7 yang mendeskripsikan proses perancangan aplikasi permainan Othello.



Gambar 7. *Flowchart* aplikasi permainan

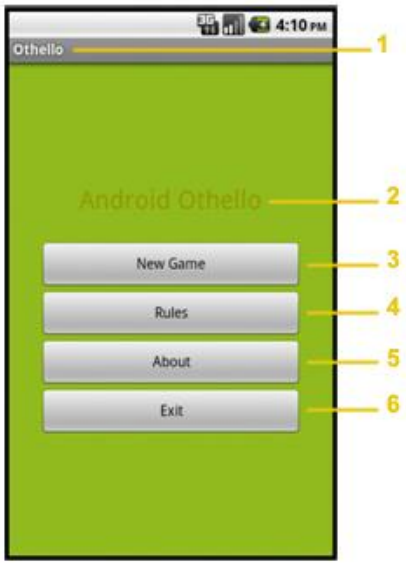
Berikut ini adalah diagram *use case* aplikasi Othello.



Gambar 8. *Use case diagram*

V. HASIL DAN PEMBAHASAN

Pada *form* utama terdapat menu pilihan yang dapat dipilih oleh pemain, seperti *New Game*, *Rules*, *About*, dan *Exit*.



Gambar 9. Tampilan awal aplikasi permainan Othello

Penjelasan mengenai penomoran pada Gambar 9 adalah sebagai berikut:

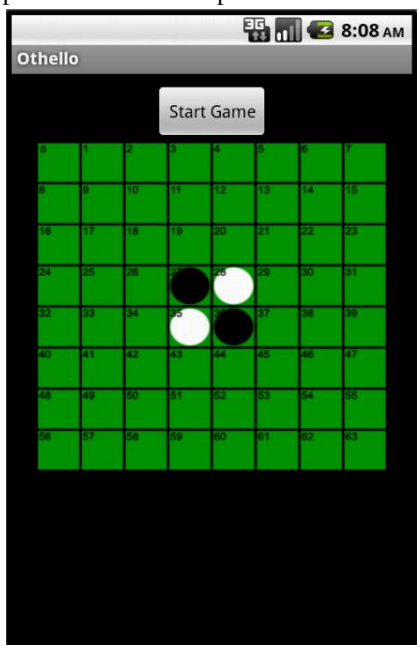
1) Nama aplikasi (Othello).
2) Judul aplikasi (Android Othello).
3) Tombol *new game* akan menampilkan halaman *difficulty* jika diklik.
4) Tombol *rules* akan menampilkan halaman aturan bermain Othello jika diklik.
5) Tombol *about* akan menampilkan halaman tentang Othello jika diklik.
6) Tombol *exit* untuk keluar dari aplikasi permainan Othello.

Tombol *new game* akan menampilkan halaman *difficulty* yang berfungsi untuk menentukan tingkat kesulitan sesuai dengan keinginan pemain. Berikut adalah tampilan halaman *difficulty*.

Gambar 10. Tampilan halaman *difficulty*

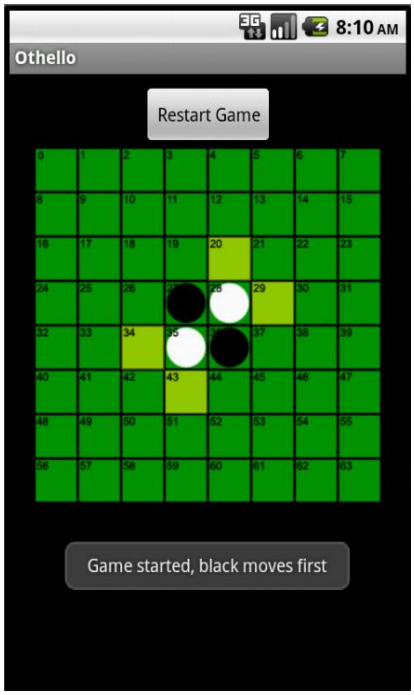
Gambar 10 memperlihatkan tampilan halaman *difficulty* yang dapat dipilih oleh pemain. Setelah pemain

memilih tingkat kesulitan yang diinginkan, kemudian akan ditampilkan halaman arena permainan yang dapat dilihat pada Gambar 11. Berikut adalah tampilan halaman arena permainan sebelum permainan dimulai.



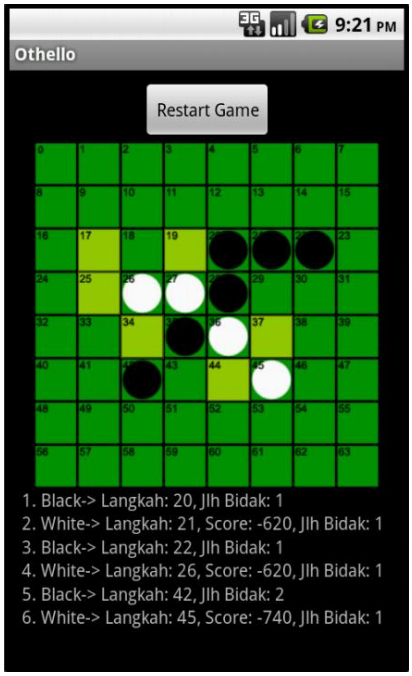
Gambar 11. Tampilan halaman arena permainan sebelum permainan dimulai

Pada halaman arena permainan, terdapat tombol *start game* yang berfungsi untuk memulai permainan. Jika tombol *start game* diklik, maka tombol *start game* tersebut berubah menjadi *restart game* dan kemudian akan muncul pesan permainan dimulai oleh bidak hitam. Selain itu, pada papan Othello akan muncul kotak legal bagi bidak hitam untuk meletakkan bidaknya. Kemudian dibawah papan permainan akan muncul pesan pemberitahuan bahwa permainan telah dimulai dan permainan dimulai pertama kali oleh bidak hitam seperti yang terlihat pada Gambar 12. Berikut adalah tampilan halaman arena permainan saat permainan dimulai.



Gambar 12. Tampilan halaman arena permainan saat permainan dimulai

Saat permainan berlangsung, dibawah papan permainan terdapat *message history* yang akan menampilkan langkah, *score*, dan jumlah bidak yang diambil oleh setiap pemain seperti yang terlihat pada Gambar 13. Berikut adalah tampilan halaman arena permainan saat permainan berlangsung.

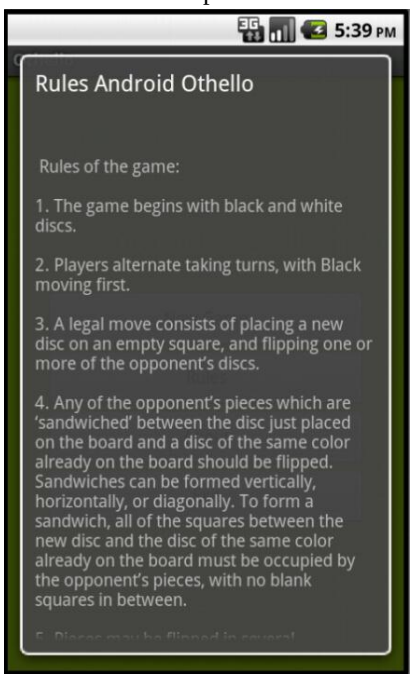


Gambar 13. Tampilan halaman arena permainan saat permainan berlangsung

Akhir permainan dari permainan Othello adalah pada saat kedua pemain sudah tidak dapat mengambil langkah lagi. Akhir permainan juga dapat terjadi pada saat papan

telah terisi penuh maupun pada saat papan belum terisi penuh.

Tampilan halaman *rules* dapat dilihat pada Gambar 14. Tombol *rules* akan menampilkan halaman *rules* yang berfungsi untuk menampilkan aturan bermain Othello. Berikut adalah tampilan halaman *rules*.



Gambar 14. Tampilan halaman *rules*

Tampilan halaman *about* dapat dilihat pada Gambar 15. Tombol *about* akan menampilkan halaman *about* yang berfungsi untuk menampilkan informasi tentang Othello. Berikut adalah tampilan halaman *about*.

Gambar 15. Tampilan halaman *about*

VI. KESIMPULAN

Aplikasi permainan Othello yang dirancang mempunyai bagian yang penting, yaitu: fungsi evaluasi, algoritma penelusuran pohon permainan, dan algoritma pencari nilai optimal. Fungsi evaluasi yang digunakan adalah strategi peletakan koin pada papan permainan. Algoritma Negamax tidak efisien apabila digunakan secara tunggal pada permainan Othello ini karena ruang pencarian yang terlalu besar, sehingga perlu dilakukan pemotongan dengan *Alpha Beta Pruning*. Dari hasil pengujian aplikasi yang telah dilakukan dapat disimpulkan bahwa aplikasi permainan Othello dapat berjalan dengan baik pada Samsung Galaxy Mini GT-S5570.

Selanjutnya penelitian ini dapat dikembangkan dengan memberikan nilai random pada fungsi evaluasi agar jenis permainan dari agen lebih bervariatif untuk setiap permainan. Selain itu, aplikasi ini dapat dikembangkan dengan menambahkan animasi untuk membuat tampilan antarmuka aplikasi permainan menjadi lebih menarik.

DAFTAR PUSTAKA

[1] Russell, S. J. dan Norvig, P. 2010. *Artificial Intelligence: A Modern Approach.* Third Edition. New Jersey: Pearson Education.

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., dan Stein, C. 2009. *Introduction to Algorithms*. Third Edition. Massachusetts: MIT Press.

[3] Rachmah, N. F. 2008. *Implementasi Algoritma Greedy pada Permainan Othello*. Skripsi. Bandung: Institut Teknologi Bandung.

[4] Leandro, J. 2009. *Pengembangan Permainan Othello Menggunakan Algoritma Minimax, Pencarian Heuristik dan Logika Samar*. Skripsi. Jakarta: Universitas Bina Nusantara.

[5] Wijaya, S. 2010. *Penerapan Konsep Algoritma Minimax dengan Menggunakan Algoritma Breadth-First Search (BFS) pada Permainan Reversi*. Skripsi. Medan: Universitas Sumatera Utara.

[6] Rose, B. 2004. *Othello: A Minute to Learn A Lifetime to Master*. Connecticut: Anjar Co.

[7] Kristanto, A. 2004. *Kecerdasan Buatan.* Yogyakarta: Graha Ilmu.

[8] Heineman, G. T., Pollice, G., dan Selkow, S. 2009. *Algorithms in a Nutshell*. California: O'Reilly Media.

[9] Millington, I. dan Funge, J. 2009. *Artificial of Intelligence for Games*. Second Edition. Massachusetts: Morgan Kaufmann Publishers.