PERBANDINGAN ALGORITMA MINIMAX DAN NEGASCOUT PADA PERMAINAN CATUR SEDERHANA

David

Program Studi Teknik Informatika
Sekolah Tinggi Manajemen Informatika dan Komputer Pontianak
Jln. Merdeka No. 372 Pontianak, Kalimantan Barat
Email: david_Liauw@stmikpontianak.ac.id

ABSTRACT

Simple Chess is a board game played by two players using boards sized in 3 rows and 3 columns and each player has pawns with different colors (normally black and white). The player wins if he/she can arrange the pawns in one line (diagonal, vertical or horizontal). The writer makes the comparisons of negascout algorithm and minimax algorithm in the application of Simple Chess to maximize the more suitable algorithm and implement them on the pawn placements in the shortest time. This research applies experimental method in form of literature study. The aspects of this research cover required features such as interface, input validation and output design. The result of this design is the application of Simple Chess embedded in negascout algorithm and minimax algorithm. The tests used by White Box are Input, Output, program functions, and the speed of negascout algorithm and minimax algorithm in the pawn placements in Simple Chess. After being tested, the result shows that each algorithm has its own advantages and disadvantages, in which tests are done through pawn placements with the fastest time. After the comparison tests are finished, the writer concludes that in this research, minimax algorithm has a better pawn movement time in placements, whereas negascout algorithm has a better step decision.

Keywords : Negascout, Minimax, Searching, Game Playing.

PENDAHULUAN

Penulis merancang perbandingan algoritma pada permainan komputer yang berupa permainan jenis strategi Catur Sederhana. Penulis menggunakan Algoritma Negascout dan Algoritma Minimax, karena kedua Algoritma tersebut merupakan jenis Algoritma yang menggunakan teknik *branch-and-bound* (Russell&Norvig, 2003). Dimana penulis bertujuan untuk mencari algoritma manakah yang lebih cocok untuk di implementasikan kedalam permainan Catur Sederhana dan juga menghasilkan waktu dan peletakan bidak tercepat dan paling tepat dalam permainan Catur Sederhana. Catur Sederhana merupakan salah satu cabang permainan catur modern pada umumnya, dimana pada permainan Catur Sederhana ini, aturan permainan diambil dari permainan *tic tac toe* atau yang dikenal dengan nama *nought and crosses* [1]. (Russell&Norvig, 2003).

Permainan Catur Sederhana merupakan permainan papan yang dimainkan oleh dua pemain pada papan yang berukuran 3 baris dan 3 kolom dan setiap pemain memiliki bidak yang berbeda warna (biasanya hitam dan putih). Pemain dikatakan menang bila salah satu dari pemain dapat membuat bidak tersusun dalam 1 garis (*diagonal*, *vertical* maupun *horizontal*).

Permainan papan (*board game*) adalah sebuah permainan di mana bidak-bidak diletakkan, dipindahkan ataupun dimakan oleh bidak lawan yang dimainkan di atas papan yang bertanda sesuai dengan peraturan yang berlaku pada permainan tersebut. Permainan papan ada yang murni berbasis strategi, kesempatan ataupun gabungan dari kedua hal tersebut dan biasanya mempunyai suatu tahap kemenangan yang ingin dicapai oleh para pemain. Permainan papan juga mempunyai berbagai jenis yang dibedakan oleh ukuran papan dan jumlah pemain.

Penulis merancang perbandingkan pada algoritma negascout dan algoritma minimax untuk mendapatkan suatu perbandingan yang dapat dilihat dari peletakan bidak dengan waktu tercepat dalam permainan Catur Sederhana.

TINJAUAN PUSTAKA

Algoritma Negascout

Negascout memiliki kesamaan dengan algoritma *alpha-beta*. Algoritma ini didasarkan pada ide pencarian dimana satu cabang pohon pertama dicari dan di evaluasi nilainya. *Node* ini kemudian akan diasumsikan memiliki nilai paling baik dari *tree* tersebut. Pencarian *node* yang lain bisa lebih cepat karena dengan asumsi nilai terbaik yang telah di dapat maka dapat di lakukan pemotongan langsung terhadap cabang *tree* jika

node selanjutnya memiliki nilai lebih rendah dari asumsi nilai terbaik. Namun jika ternyata ditemukan *node* dengan nilai yang lebih baik dari asumsi nilai terbaik, maka akan dilakukan pencarian kembali terhadap asumsi nilai terbaik,[2].

```
Function Negascout (s,a,b, depth)
1:  if(depth==MAX_DEPTH) then return evaluate(s) ;
2:  m:=-∞;
3:  n:=b;
4:  for each {successor s_i of s}
5:     t:= -Negascout( s_i, -n, -max(a,m),depth+1);
6:     if(t>m)then
7:        if(n==b or depth>=MAX_DEPTH -2) then m:=t;
8:        else m:= -Negascout( s_i, -b, -t, depth +1);
9:     if(m≥b)then return m;
10:    n = max(a,m) +1;
11: return m;
```

Gambar 1. *Pseudocode* Algoritma *Negascout* [2].

Pencarian Negascout menggunakan fungsi rekursif untuk mengeksplorasi *node-node* dari *successor* (variabel *s*) yang telah diberikan [3]. variabel *m* merupakan wadah dari nilai hasil evaluasi yang dilakukan. Pencarian akan dilakukan dengan mencari cabang *tree* pertama dari *s*. Negasi pada variabel dan fungsi dalam *for each* sendiri menandakan bahwa layaknya pada algoritma *minimax*, ada pengambilan langkah dari lawan (*min*) yang mana akan meminimalkan nilai dari langkah yang kita ambil sebelumnya (*max*).

Pertama, diberikan nilai *-INFINITY* terhadap variabel *m* yang menandakan bahwa pencarian berada pada *state* awal dimana nilai terbaik dari cabang pertama akan dijadikan asumsi nilai terbaik (variabel *m*). Selanjutnya dengan nilai rekursif akan di lakukan eksplorasi terhadap *node-node* lainnya, yang mana akan dicari apakah *node* yang di eksplorasi memiliki nilai lebih baik dari variabel *m*. Jika tidak ditemukan nilai yang lebih baik maka akan di kembalikan nilai *m*, namun jika di temukan *node* dengan nilai yang lebih baik dari nilai *m* maka akan di lakukan pencarian kembali asumsi nilai terbaik dari cabang *tree* tersebut.

Algoritma Minimax

"Algoritma Minimax adalah algoritma yang menggunakan *depth-first search* dengan kedalaman terbatas [4]. Minimax merupakan teknik *games* yang terkenal. Evaluasi yang digunakan dalam minimax ini adalah dengan menggunakan evaluasi statis,yang diasumsikan bahwa akan adanya kemungkinan dimana lawan dapat membuat langkah terbaik [5].

```
Algoritma Minimax :
    Minimax(StatusSaya, Kedalam, Pemain)
    IF (Kedalaman==Max)
       RETURN static (StatusSaya, Pemain)
    Bangkitkan Successor S[1..n]
       IF  (Pemain==Saya)
          RETURN max of Minimax (S [i],
kedalaman+1 , Lawan)
       ELSE
```

Gambar 2. *Pseudocode* Algoritma Minimax [4].

Catur Sederhana

Catur Sederhana merupakan salah satu jenis permainan berpapan yang diambil dari permainan catur pada umumnya. Dalam permainan Catur modern dimainkan oleh dua pemain pada papan yang berukuran 8 baris dan 8 kolom dan tiap pemain memiliki 16bidak yang warnanya berbeda (2kelompok biasanya kelompok 1 berwarna putih dan kelompok2 berwarna hitam), sedangkan pada permainan Catur Sederhana dimainkan oleh dua pemain pada papan yang berukuran 3 baris dan 3 kolom dan setiap pemain memiliki bidak yang berbeda warna (biasanya hitam dan putih). Pemain dikatakan menang bila salah satu dari pemain dapat membuat bidak tersusun dalam 1 garis(diagonal, vertikal maupun horizontal).

METODE PENELITIAN

Dalam penelitian ini, penulis menggunakan bentuk penelitian eksperimental. Metode eksperimental adalah metode yang dipergunakan oleh peneliti terhadap obyeknya dengan cara mengadakan eksperimen-eksperimen. Perangkat lunak yang telah dihasilkan akan diuji coba dengan cara terus-menerus untuk mendapatkan *bugs* atau *error* sehingga dapat diperbaiki dengan demikian akan didapatkan hasil yang lebih bagus.

Dalam metode perancangan penelitian ini penulis menggunakan rancangan sistem aplikasi pengembangan perangkat lunak dengan model air terjun dan model sistem formal dimana, Pada model air terjun sistem aplikasi pengembangan dilakukan pertahap dengan memberlakukan bahwa tahap berikutnya dapat dijalankan setelah tahap sebelumnya telah selesai maka dapat dikatakan semua tahap saling berhubungan satu dengan yang lainnya [6].

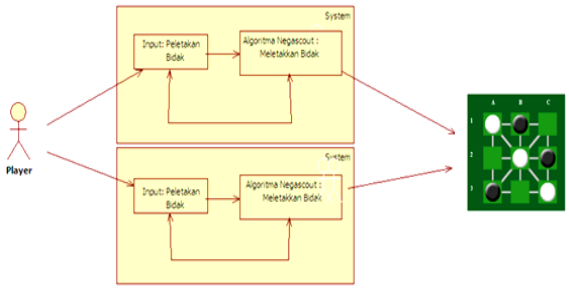
Kemudian pada model sistem formal didasari pada transformasi matematis dari spesifikasi program [6]. Dimana pada setiap proses yang telah dikerjakan disisipkan program yang ekuivalen sehingga membuat transformasi cukup dekat. Maka jika menganggap tidak adanya verifikasi yang error implementasi program akan benar dari spesifikasi yang diharapkan.

Pada tahap ini penulis mendefinisikan persyaratan pengembangan sistem formal sebagai berikut:

a. Aplikasi dapat membuktikan algoritma mana yang lebih unggul dan tepat untuk diimplementasikan kedalam aplikasi permainan Catur Sederhana.
b. Aplikasi dapat menghasilkan perbandingan yang berupa waktu yang digunakan untuk peletakan bidak pada ruang peletakan.
c. Pada aplikasi permainan catur sederhana ini, player hanya melakukan peletakan bidak pada ruang peletakan sesuai dengan garis yang telah ditentukan.
d. Pada Aplikasi permainan catur sederhana ini, player menang jika berhasil membuat bidak tersusun dalam 1 garis (Diagonal, Vertikal, maupun Horizontal).

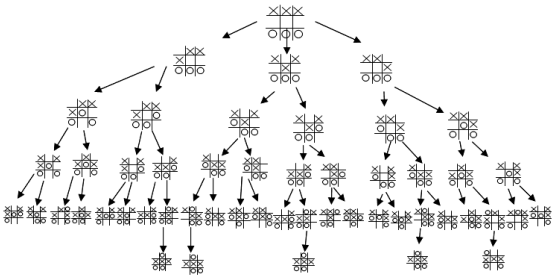
HASIL DAN PEMBAHASAN

Proses perancangan merupakan deskripsi dari kebutuhan yang direpresentasikan ke dalam perangkat lunak sehingga dapat diperkirakan kualitasnya sebelum dimulai pembuatan kode (*coding*). Aplikasi yang dibangun dalam penelitian ini adalah Aplikasi perbandingan algoritma negascout dan algoritma minimax dalam permainan catur sederhana. Arsitektur Aplikasi perbandingan algoritma negascout dan algoritma minimax dalam permainan catur sederhana.



Gambar 3. Arsitektur *Flow Diagram* dari Aplikasi Permainan Sederhana

Aplikasi perbandingan algoritma negascout dan algoritma minimax dalam permainan catur sederhana dapat dilihat pada gambar 3 dimana saat program dijalankan, dapat dilihat terdapat inputan yang berupa peletakan bidak terhadap papan sederhana. Selain peletakan bidak tersebut, terdapat juga Algoritma (peletakan bidak yang diproses oleh algoritma negascout dan algoritma minimax), sehingga permainan lebih mudah dimengerti dan dimainkan.



Gambar 4. *Graph* Minimax dan Negascout Berkedalaman Enam yang Diterapkan Pada Langkah Pembukaan Permainan Catur Sederhana

Gambar 4 menunjukkan *graph* minimax dan Negascout berkedalaman *level* yang diterapkan pada algoritma minimax dan negascout. Untuk memulai proses pencarian, kita hanya akan 'turun' ke kedalam pertama dan menerapkan evaluasi heuristik kita pada suatu keadaan dan semua sibling-nya. Katakanlah pada simpul-simpul MIN. Nilai maksimum dari MIN ini kemudia di-*backed-up* ke parentnya (simpul MAX, seperti dalam minimax). Nilai ini kemudian ditawarkan pada parent berikutnya dari MIN yang potensial.

Selanjutnya, algoritma 'turun' lagi ke child berikutnya dan mengakhiri eksplorasi parent-nya jika ada nilainya yang sama dengan atau lebih besar dari nilai beta ini. Prosedur yang sama dapat dilakukan pada pemangkasan alfa untuk child simpul MAX. Dua aturan yang didasarkan pada nilai *alfa* dan *beta* dalam mengakhiri pencarian adalah:

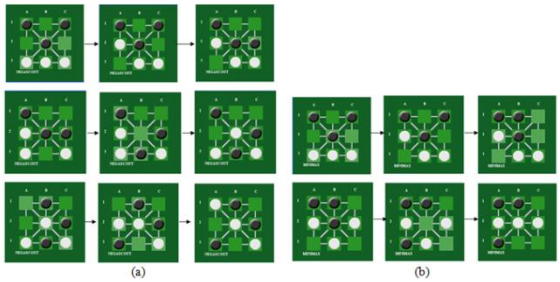
a. *Search* dapat dihentikan di bawah setiap simpul MIN yang memiliki nilai beta lebih kecil dari atau sama dengan nilai *alfa* dari asal muasal (*ancestor*) MAX-nya.
b. *Search* dapat dihentikan dibawah setiap simpul MAX yang memiliki nilai *alfa* lebih besar dari atau sama dengan nilai *beta* dari asal muasal (*ancestor*) MAX-nya.

Transformasi Formal Pengembangan Sistem Formal

Algortima Negascout

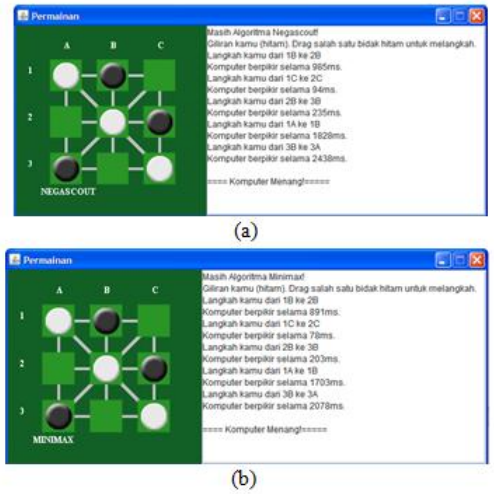
Algoritma Negascout dan Minimax didasari pada ide pencarian dimana satu cabang pohon pertama dicari dan dievaluasi nilainya. Algoritma Negascout dan Minimax akan dihubungkan dengan metode evaluasi yang berguna untuk menentukan nilai posisi yang ditemukan oleh kedua algoritma Negascout dan Minimax. Dengan adanya metode evaluasi tersebut, maka akan diasumsikan nilai posisi yang terbaik dari beberapa cabang pohon yang kemudian akan dilakukan pemotongan langsung terhadap cabang pohon jika terdapat nilai yang lebih rendah dari asumsi nilai terbaik [7]. Namun jika ternyata didapatkan cabang pohon yang memiliki nilai lebih baik, maka akan

dilakukan pencarian kembali terhadap asumsi nilai terbaik [8]



Gambar 5. Tampilan Pengujian Permainan Catur Sederhana a) Negascout, b) Minimax

Dapat dilihat pada gambar 5a, pengujian yang dilakukan terhadap Aplikasi perbandingan algoritma negascout dalam permainan catur sederhana. *Player* melakukan gerakan pertama kali yang kemudian di ikuti oleh Algoritma Negascout, dan berulang seterusnya, sehingga didapatkan 1garis lurus (diagonal, vertikal, dan horizontal). Dapat dilihat pada gambar 5b, pengujian yang dilakukan terhadap Aplikasi perbandingan algoritma Minimax dalam permainan catur sederhana. *Player* melakukan gerakan pertama kali yang kemudian di ikuti oleh Algoritma minimax, dan berulang seterusnya, sehingga didapatkan 1garis lurus (diagonal, vertikal, dan horizontal).



Gambar 6. Tampilan Pengujian
a) Algoritma Negascout; b) Algoritma Minimax

Pengujian Negascout

Pada gambar 6a menampilkan pengujian aplikasi pada algoritma Negascout. Pada tahap ini, *user* mencoba menguji kecepatan peletakan bidak pada ruang kosong peletakan didalam aplikasi permainan catur sederhana. Dapat dilihat pada gambar 6, Algoritma Negascout berhasil memenangkan permainan dengan 5 langkah (*step*), dimana kecepatan waktu per*step* berbeda-beda. Pada langkah awal algoritma Negascout menghabiskan waktu 985*ms*, pada langkah ke dua menghabiskan waktu 94*ms*, pada langkah ke tiga menghabiskan waktu 235*ms*, pada langkah ke empat menghabiskan waktu 1828*ms* dan langkah ke lima menghabiskan waktu selama 2438*ms* dengan total 5580*ms*. Dari pengujian terhadap algoritma negascout yang telah dilakukan oleh penulis pada aplikasi permainan catur sederhana, maka didapatkan hasil yang telah dirangkum dalam table 1. Berdasarkan pada tabel 1 dan dapat juga dilihat pada gambar 7 dapat dilihat pengujian kecepatan dalam peletakan bidak pada algoritma algoritma dalam satuan *Ms*(*MiliSecond*).

Pengujian Minimax

Pada gambar 6b menampilkan pengujian aplikasi pada algoritma minimax. Pada tahap ini, *user* mencoba menguji kecepatan peletakan bidak pada ruang kosong peletakan didalam aplikasi permainan catur sederhana. Dapat dilihat pada tabel 1, Algoritma Minimax berhasil memenangkan permainan dengan 5 langkah (*step*), dimana kecepatan waktu per*step* berbeda-beda. Pada langkah awal algoritma minimax menghabiskan waktu 891*ms*, pada langkah ke dua menghabiskan waktu 78*ms*, pada langkah ke tiga menghabiskan waktu 203*ms*, pada langkah ke empat menghabiskan waktu 1703*ms* dan langkah ke lima menghabiskan waktu selama 2078*ms* dengan total 4953*ms*. Dari pengujian terhadap algoritma Minimax yang telah dilakukan oleh penulis pada aplikasi permainan catur sederhana, maka didapatkan hasil yang telah dirangkum dalam tabel 1. Berdasarkan pada tabel 1 dan dapat juga dilihat pada gambar 7 dapat dilihat pengujian kecepatan dalam peletakan bidak pada algoritma algoritma dalam satuan *Ms*(*MiliSecond*).

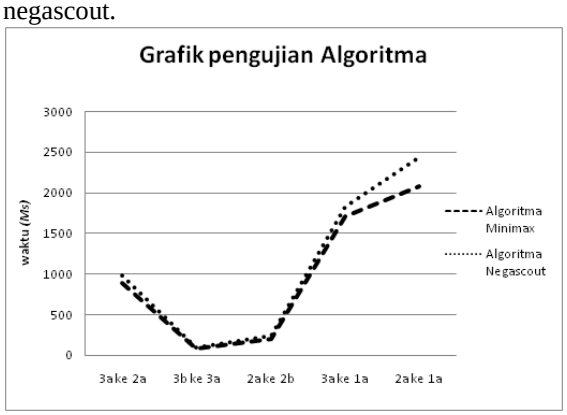
Analisis Pengujian

Analisis pada Algoritma Negascout dan Algoritma Minimax merupakan analisis perbandingan yang akan dilakukan dengan menjalankan kedua aplikasi permainan pada suatu perangkat keras yang sama dan dengan spesifikasi yang sama pula. Pada aplikasi permainan catur sederhana, akan dilakukan analisis berupa kecepatan pada proses peletakan dengan waktu tercepat hingga permainan berakhir (ada yang mencapai 1 garis lurus Diagonal, Vertikal, Horisontal).

Tabel 1. Hasil Pengujian Perbandingan Algoritma Negascout dan Algoritma Minimax Berdasarkan Kecepatan Pada Proses Peletakan

Pengujian	Langkah (*Step)*	Algotima Minimax	Algoritma Negascout
		Waktu (*Ms*)	
1.	3a melangkah ke 2a	891	985
2.	3b melangkah ke 3a	78	94
3.	2a melangkah ke 2b	203	235
4.	3a melangkah ke 2a	1703	1828
5.	2a melangkah ke 1a	2078	2438
Total waktu (*Ms*)		**4953**	**5580**

Berdasarkan pada tabel 1 atau dapat juga dilihat pada gambar 7 dapat dilihat perbandingan kecepatan dalam peletakan bidak pada algoritma negascout dan algoritma minimax dalam satuan *Ms*(*MiliSecond*). Dengan adanya table diatas, dapat disimpulkan bahwa kecepatan peletakan bidak pada ruang peletakan yang masih kosong dalam aplikasi permainan catur sederhana, algoritma minimax lebih ungul dibandingkan dengan algoritma negascout.



Gambar 7. *Diagram* Perbandingan Waktu Antara Algoritma Minimax dengan Algoritma Negascout

Algoritma minimax hanya membutuhkan waktu 4953*ms*, sedangkan algoritma negascout membutuhkan waktu 5580*ms*, yang berarti algoritma negascout membutuhkan waktu lebih lama untuk menyelesaikan permainan dibandingkan dengan algoritma minimax. Sehingga penulis dapat mengasumsikan bahwa algoritma minimax lebih cocok untuk diimplementasikan kedalam aplikasi permainan catur sederhana.

KESIMPULAN

Berdasarkan analisis dan hasil penelitian dari bab sebelumnya, maka penulis dapat mengambil kesimpulan bahwa kedua algoritma yang menjadi subjek perbandingan pada penelitian ini yakni Algoritma Negascout dan Algoritma Minimax dalam permainan catur sederhana mempunyai keunggulan dan kelemahannya masing-masing. Seperti yang terlihat pada hasil uji perbandingannya, pada Algoritma Minimax lebih unggul dalam waktu peletakan bidak pada ruang peletakan, sedangkan Algoritma Negascout lebih unggul dalam pengambilan langkah terbaik. Sehingga dapat disimpulkan bahwa algoritma yang cocok untuk diimplementasikan ke dalam aplikasi permainan catur sederhana adalah algoritma Minimax. Penulis mengharapkan pada penelitian selanjutnya dapat mencoba menerapkan algoritma lain atau pun dapat memodifikasi algoritma menjadi lebih kompleks untuk diujikan.

KESIMPULAN

Berdasarkan analisis dan hasil penelitian dari bab sebelumnya, maka penulis dapat mengambil kesimpulan bahwa kedua algoritma yang menjadi subjek perbandingan pada penelitian ini yakni Algoritma Negascout dan Algoritma Minimax dalam permainan catur sederhana mempunyai keunggulan dan kelemahannya masing-masing. Seperti yang terlihat pada hasil uji perbandingannya, pada Algoritma Minimax lebih unggul dalam waktu peletakan bidak pada ruang peletakan, sedangkan Algoritma Negascout lebih unggul dalam pengambilan langkah terbaik. Sehingga dapat disimpulkan bahwa algoritma yang cocok untuk diimplementasikan ke dalam aplikasi permainan catur sederhana adalah algoritma Minimax. Penulis mengharapkan pada penelitian selanjutnya dapat mencoba menerapkan algoritma lain atau pun dapat memodifikasi algoritma menjadi lebih kompleks untuk diujikan.

DAFTAR PUSTAKA

[1] Russell, Stuart J.; Norvig, Peter, 2003. *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, pp. 163–171, ISBN 0-13-790395-2

[2] Mandziuk, 2010. *Knowledge-Free and Learning-Based Methods in Intelligent Game Playing*. Computational Intelligence 276 : Springer, (diakses 19 Mei 2012).

[3] Effendi, Aditya Kurniawan., Delima, Rosa., Antonius R. C., 2012. Implementasi Algoritma Negascout Untuk Permainan Checkers, Informatika Jurnal Tekhnologi Komputer dan Informatika, Vol.8, No. 1, April 2012

[4] Kusumadewi, Sri, 2003. *Artificial Intelligence*, Graha Ilmu, Yogyakarta.

[5] Gilbert, E. N., 1985. An optimal minimax algorithm, *Annals of Operations Research*, Volume 4, Issue 1, pp 103-121

[6] Sommerville, Ian., 2012. *Software Engineering*.9th edition, Pearson, United State.

[7] Perry, William E., 2006. *Effective Methods For Software Testing*, Edisi Ketiga, Wiley, Canada.

[8] Plaat, Aske., Schaeffer, Jonathan., Pijls, Wim., Bruin, Arie de., 1995. An Algorithm Faster than NegaScout and SSS* in Practice, *Computer Strategy Game Programming Workshop*.

[9] Gunawan, Kristian, Andika, *Game playing untuk Othello dengan menggunakan algoritma negascout dan MTDF* (*SNATI 2009)*, http://journal.uii.ac.id/index.php/Snati/article/viewFile/1278/1088, (diakses 19 Mei 2012).

[10] Hasibuan, Zainal A. 2007. Metodologi Penelitian Pada Bidang Ilmu Komputer Dan Teknologi Informasi, Konsep: Metode Teknik dan Aplikasi, Depok.