Penerapan Algoritma *Minimax* Pada Game Macan-macanan

Kartika Imam Santoso[a,*], Farida Yunita[b], Nafi Projo Kusumo[c]

[a] Prodi Sistem Informasi, STMIK Bina Patria Magelang
[b] Prodi Teknik Informatika, STMIK Bina Patria Magelang
[c] Alumni Teknik Informatika, STMIK Bina Patria Magelang

Naskah Diterima : 25 Februari 2016; Diterima Publikasi : 30 Mei 2016

DOI: 10.21456/vol6iss1pp21-29

Abstract

Lots of traditional games, but now the game is becoming obsolete. Many games were replaced with the modern game technology products. Modern games are becoming more practical because it did not require the terrain and many friends. Quite alone in front of the screen was a person may engage in an exciting game. One of the efforts to preserve and disseminate traditional games one of which is macan-macanan is to adapt the game into a computer game. This study aims to apply artificial intelligence using minimax algorithms and programming language ActionScript 3 in the game with a macan-macanan research methods are prototyping. The design used in this study is an artificial intelligence approach for representing the state, science, human computer interaction for designing the user experience, as well as the UML for object-based design. Results from this study is that in order to determine the value of the evaluation algorithm *minimax* for the end node / terminal state in the game macan-macanan, required the calculation of the total step is valid for each piece, as well as to pawn macan, necessary calculations springboard to a higher value and the weight difference the appropriate type of pawns.

Keywords : ActionScript 3; Game; Macan-macanan; Minimax Algorithms.

Abstrak

Banyak sekali permainan-permainan tradisional namun sekarang permainan tersebut mulai ditinggalkan. Banyak permainan digantikan dengan permainan *modern* produk teknologi. *Game modern* menjadi lebih praktis karena tak memerlukan tanah lapang dan banyak teman. Cukup sendirian di depan layar pun seseorang bisa terjun dalam permainan yang mengasyikkan. Salah satu upaya untuk melestarikan serta menyebarluaskan permainan tradisional salah satunya adalah macan-macanan ini adalah dengan mengadaptasi permainan tersebut menjadi *game* komputer. Penelitian ini bertujuan untuk menerapkan kecerdasan buatan dengan menggunakan algoritma *minimax* serta bahasa pemrograman *ActionScript* 3 pada permainan macan-macanan dengan metode penelitian adalah *prototyping*. Perancangan yang digunakan dalam penelitian ini adalah pendekatan kecerdasan buatan untuk merepresentasikan keadaan, ilmu pengetahuan interaksi manusia dan komputer untuk perancangan *user experience*, serta *UML* untuk perancangan berbasis objek. Hasil dari penelitian ini adalah bahwa untuk menentukan nilai evaluasi algoritma *minimax* untuk *node* akhir / terminal *state* pada permainan macan-macanan, diperlukan perhitungan terhadap total langkah *valid* untuk setiap bidak, serta untuk bidak macan, diperlukan perhitungan loncatan dengan nilai yang lebih tinggi dan perbedaan bobot sesuai tipe bidak.

Kata Kunci : ActionScript 3, Game; Macan-macanan; Algoritma *Minimax*.

1. Pendahuluan

Dahulu banyak sekali permainan-permainan tradisional, sekarang permainan tersebut mulai ditinggalkan, banyak permainan digantikan dengan permainan *modern* produk teknologi. *Game modern* menjadi lebih praktis karena tak memerlukan tanah lapang dan banyak teman. Cukup sendirian di depan layar pun seseorang bisa terjun dalam permainan yang mengasyikkan (Dirgantara, 2012). Salah satu upaya untuk melestarikan serta menyebarluaskan permainan macan-macanan ini adalah dengan mengadaptasi permainan tersebut menjadi *game* komputer. Nama permainan tersebut adalah macan-macanan.

Bagaimana menerapkan permainan macan-macanan menjadi *game* komputer?. Maka tujuan penelitian adalah untuk mengetahui, apakah

*) Kartika Imam Santoso: kartikaimams@gmail.com

kecerdasan buatan dengan algoritma *minimax* dan *ActionScript3* dapat diterapkan untuk membuat permainan macan-macanan. Metodologi yang digunakan adalah *prototyping* (Pressman, 2011), untuk perubahan aplikasi secara cepat berdasarkan komunikasi atas kebutuhan *user*.

Algoritma *Minimax* dikembangkan pada 1928 oleh John von Neuman dan prinsip-prinsip yang dimilikinya masih merupakan salah satu yang paling dikenal dan banyak digunakan dalam *game* sekarang. Dalam *minimax*, MAX adalah sebutan bagi pemain (komputer) yang bertujuan untuk mendapatkan nilai maksimal dan MIN adalah sebutan bagi (lawan) yang bertujuan untuk mendapatkan nilai minimal. Fungsi evaluasi yang baik akan memperhitungkan dari banyak aspek yang berbeda, seperti posisi papan, jumlah bidak relatif terhadap lawan, dan mobilitas. Mobilitas menunjukkan kebebasan setiap pemain dalam memainkan bidak dan seberapa banyak langkah setiap bidak yang dimiliki oleh setiap pemain (Sutojo, 2011).

Permainan macan-macanan masuk ke kategori *board game* (Vebrina, 2008), dan permainan ini memiliki 2 versi (Dharmamulya, 2004), versi yang dibahas pada penelitian adalah versi dengan 2 bidak macan dan 10 bidak manusia. Bidak macan dapat memakan 2 bidak manusia yang sejajar. Dan tugas bidak manusia adalah mengurung pergerakan bidak macan.

2. Kerangka Teori

2.1. Kecerdasan Buatan

Kecerdasan buatan adalah 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 yang sepenuhnya bisa menirukan beberapa fungsi otak manusia, seperti pengertian bahasa, pengetahuan, pemikiran, pemecahan masalah dan sebagainya (Kristanto, 2004:1).

Cerdas adalah memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Jadi, agar mesin bisa cerdas (bertindak seperti manusia), maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar (Sutojo, 2011:3).

2.2. Algoritma Minimax

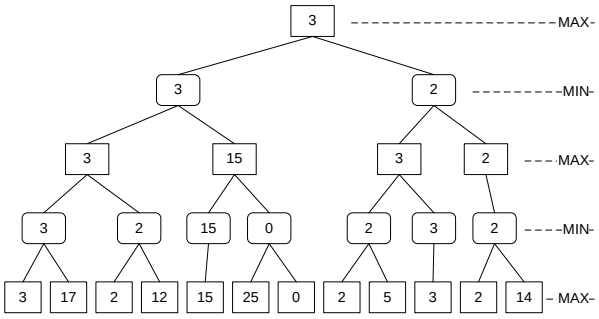
Algoritma *Minimax* dikembangkan pada 1928 oleh John von Neuman dan prinsip-prinsip yang dimilikinya masih merupakan salah satu yang paling dikenal dan banyak digunakan dalam *game* sekarang. Prinsip-prinsip ini hanya berlaku untuk *game* yang memiliki dua pemain yang bermain secara bergantian, yaitu manusia melawan komputer seperti dalam permainan catur, dam dan *tic-tac-toe*.

Menurut Sutojo (2011:440-445) untuk mengimplemen- tasikan algoritma *minimax*, kita memerlukan sebuah metode untuk mengukur seberapa baik sebuah posisi pada saat itu. Metode ini disebut sebagai fungsi evaluasi. Fungsi evaluasi yang baik akan memperhitungkan dari banyak aspek yang berbeda, seperti posisi papan, jumlah bidak relatif terhadap lawan, dan mobilitas. Mobilitas menunjukkan kebebasan setiap pemain dalam memainkan bidak dan seberapa banyak langkah setiap bidak yang dimiliki oleh setiap pemain.

Dalam *minimax*, MAX adalah sebutan bagi pemain (komputer) yang bertujuan untuk mendapatkan nilai maksimal dan MIN adalah sebutan bagi (lawan) yang bertujuan untuk mendapatkan nilai minimal.

Prosedur *minimax*:

1. Tandai masing-masing level dari ruang pencarian sesuai dengan langkahnya pada level itu.
2. Mulai pada *node* daun (node paling bawah), dengan menggunakan fungsi evaluasi, berikan nilai pada masing-masing *node*.
3. Arah menjalar ke atas : jika *node* orang tua adalah MAX, pilihlah nilai terbesar yang terdapat pada *node* anak dan berikan nilai pada MAX (*node* orang tua)
4. Arah menjalar ke atas : jika *node* orangtua adalah MIN, pilihlah nilai terkecil yang terdapat pada *node* anak dan berikan nilai MIN (*node* orang tua).



Gambar 1 Pohon *Minimax* (Sutojo, 2011)

Dalam Algoritma *Minimax* juga dikenal istilah fungsi evaluasi, fungsi evaluasi adalah menentukan nilai dari representasi suatu keadaan, yang digunakan sebagai acuan perhitungan suatu *node*. Berikut adalah fungsi evaluasi dari permainan catur, secara matematis diekspresikan sebagai berikut:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s) \qquad (1)$$

Keterangan : w adalah bobot tiap jenis bidak catur dan $f(s)$ adalah jumlah jenis bidak catur putih dikurangi dengan jumlah jenis bidak catur hitam (jumlah kuda putih – jumlah kuda hitam).

Algoritma *minimax* memiliki karakteristik *Completeness* (jika pohon pencariannya berhingga walaupun besar), Optimalitas (algoritma *minimax* akan optimal jika melawan pemain dengan langkah yang optimal pula), Kompleksitas Waktu (lamanya eksplorasi pencarian), Kompleksitas Ruang (banyaknya ruang pencarian) (Sutojo,2011:442).

2.3. ActionScript 3.0 (*AS3*)

ActionScript 3.0 (AS3) diperkenalkan pada 2006 dan menjadi bahasa pemrograman primer untuk *Flash* setelahnya. *Flash* 5 pada tahun 2000 ditambahkan dengan pengenalan bahasa baru, yaitu *ActionScript* 1.0. Bahasa *script* ini memiliki semua bagian dari pengembangan bahasa berbasis web.

Flash MX 2004, yang juga dinamai *Flash* 7, membawa versi *ActionScript* 2.0, versi yang lebih *powerful*, dan pembuatan *object-oriented programs* menjadi lebih mudah. Bahasanya lebih mirip dengan ECMA *Script*, standar pengembangan bahasa pemrograman dari *European Computer Manufaturers Association*. *Javascript* juga berbasis ECMA *Script*.

Perbandingan *ActionScript* 3.0 atas *ActionScript* 2.0:

1. Kecepatan : AS3 mengeksekusi lebih cepat daripada AS2. *Flash virtual machine* yang baru dibuat untuk menjalankan aplikasi AS3, dan dioptimalisasi dengan *just-in-time compiler*. Kecepatan eksekusi 2 : 5 lebih cepat daripada AS2 (tergantung tes)
2. *Error handling/debugging* : pada AS2, penanganan *error* dan *debug* masih proses *trial and error* yang membutuhkan *trace statement*. AS3 ditambahkan dengan *compiler* yang mencari *error* pada *logic* sebaik sintak dan *runtime error system* untuk membantu mendiagnosa permasalahan pada kode yang terkompile secara sempurna namun tidak bekerja saat dijalankan.
3. *Advanced event model* : Saat in banyak *game* bukan lagi program prosedural yang menunggu *input* dari *user*, tetapi juga membuat lingkungan dimana *event* dapat terjadi dan harus ditangani oleh program. Pada AS3, *event* termasuk dalam sistem, yang membuatnya terlihat lebih natural dan pada saat bersamaan membantu *developer* mendesain *software* yang lebih logis dan mudah dipelihara (Fulton, 2010:7)

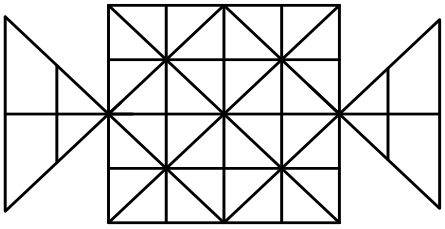
2.4. Game Macan-macanan

Permainan Macan-macanan ini merupakan *board game*, *board game* adalah permainan dengan kepingan-kepingan yang ditempatkan di atas, dipindahkan dari atau digerakkan di atas suatu permukaan khusus, permukaan khusus itu disebut papan permainan.

Permainan macan - macanan ini merupakan permainan tradisional yang menggunakan strategi dalam memainkannya, serta hanya dapat dimainkan oleh dua orang.

Menurut Dharmamulya ada 2 jenis tipe permainan macan-macanan. Yang pertama adalah dengan menggunakan 21 bidak manusia, dan 1 bidak macan, bidak macan dapat memakan bidak manusia berdasarkan jumlah ganjil (1, 3, 5) dalam garis sejalur, dan tugas bidak manusia adalah mengurung pergerakan macan, papan permainan hanya memiliki satu gunungan. Yang kedua dengan 8 bidak manusia dan 2 bidak macan, serta satu gunungan, bidak macan hanya bisa memakan 2 (dua) bidak manusia dalam garis sejalur, tugas bidak manusia mengurung pergerakan bidak macan.

Obyek penelitian permainan macan-macanan yang didapat lebih menyerupai permainan macan-macanan tipe kedua yang dijabarkan di atas. Namun dengan 2 gunungan (segitiga) dan tambahan 2 bidak manusia.



Gambar 2.Papan permainan Macan-macanan (Dharmamulya, 2004)

Berikut ini adalah beberapa ketentuan dan peraturan dalam permainan macan-macanan:

1. Papan memiliki 37 pijakan yang dapat ditempati oleh bidak macan maupun bidak manusia.
2. Permainan memiliki 2 bidak macan, dan 10 bidak manusia.
3. Papan permainan dalam keadaan kosong.
4. Setiap satu pijakan hanya bisa ditempati oleh satu bidak.
5. Bidak macan adalah pemain yang melakukan langkah pertama, yaitu meletakkan di salah satu pijakan.
6. Bidak macan dan bidak manusia melangkah bergiliran masing-masing di pijakan yang kosong.
7. Setelah semua pijakan terisi masing-masing bidak hanya bisa bergeser.
8. Masing-masing bidak hanya bisa melangkah di pijakan yang kosong, serta mengikuti jalur / garis yang menghubungkan pijakan satu dengan yang lain.
9. Bidak macan bertugas untuk memakan bidak manusia, cara memakannya adalah jika 2 bidak manusia yang berada pada satu jalur dan pijakan ketiga kosong.
10. Bidak manusia bertugas untuk mengurung bidak macan, agar tidak dapat melangkah (mati langkah).

11. Bidak macan menang jika bidak manusia hanya tersisa 4 buah.
12. Bidak manusia memenangkan permainan jika bidak macan mati langkah (tidak dapat berpindah pijakan).

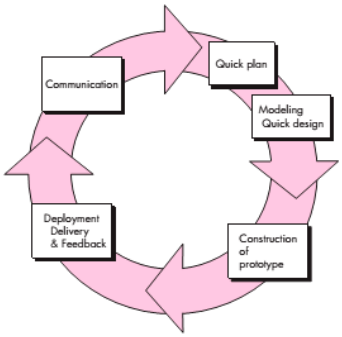
3. Metode

3.1. Metode Penelitian

Metode penelitian yang dipakai adalah metode *prototyping*.Tahapan yang dilakukan dalam metode *prototyping* pada penelitian kali ini adalah :

1. C*ommunication* : komunikasi dan identifikasi kebutuhan user, terhadap permainan macan-macanan yang akan dirancang.
2. *Quick plan & modeling quick design* : perancangan dan pemodelan cepat, lebih ke sisi *human interface*, merancang tampilan agar sesuai dengan kebutuhan user dan permainan.
3. *Construction of prototype* : perancangan dan pembuatan prototipe permainan macan-macanan, serta penerapan algoritma kecerdasan buatan dan penggunaan bahasa pemrograman AS3.
4. *Deployment delivery & feedback* : evaluasi dan pengujian *prototype* oleh pengguna.

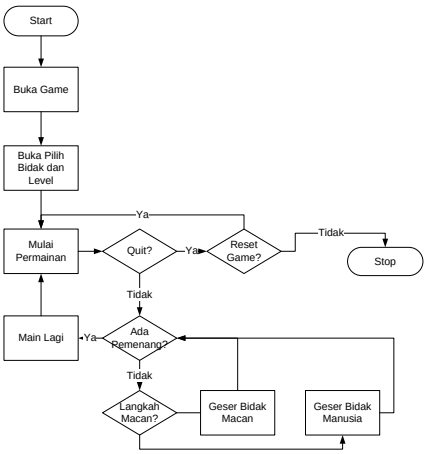
Diagram tahapan metode *Prototyp*ing bisa dilihat pada gambar 2.



Gambar 3. Diagram *Prototyping* (Pressman, 2011)

3.2. Analisis Kebutuhan Pengguna (User)

Analisis kebutuhan pengguna dapat dilihat pada Gambar 4 Berikut ini.



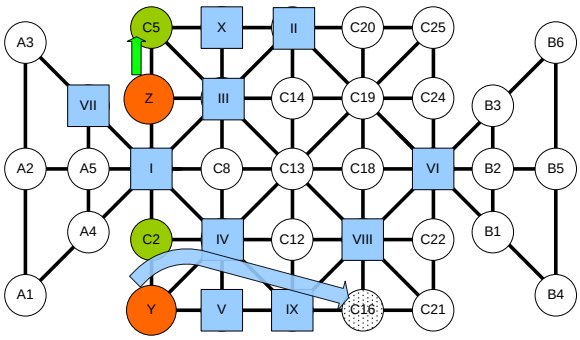
Gambar 4. *Flowchart* Kebutuhan *User*

Dari gambar 4 di atas, diketahui analisis kebutuhan pengguna meliputi :

1. Mengakses tampilan permainan macan-macanan
2. *User* bisa memilih tipe bidak dan *level* permainan sesuai yang diharapkan
3. *User* bisa bermain permainan macan-macanan melawan AI dengan memanfaatkan papan permainan, dan tampil pemenang dalam permainan.
4. Serta yang *user* bisa melakukan *reset* permainan

3.3. Representasi papan permainan

Papan permainan macan-macanan dapat direpresentasikan sebagai berikut:



Gambar 5. *Representasi* Papan Permainan

Tidak seperti pada papan permainan catur atau *tic-tac-toe*, yang memiliki jumlah baris dan kolom yang dapat dihitung, dan kemudian dapat diasumsikan ke dalam sebuah *array*. Papan permainan macan-macanan sangat berbeda, sehingga untuk memudahkan tiap pijakan diberi kode, A dan B untuk gunungan, serta C untuk papan tengah, masing-masing pijakan diberi nomor urut. Sedangkan bidak direpresentasikan dengan huruf Y dan Z untuk bidak macan, dan angka romawi I sampai X untuk bidak manusia. Pada masing-masing pijakan yang ditempati oleh bidak, ada keterhubungan antar pijakan, yaitu C1 yang ditempati oleh bidak Y dengan pijakan C2 keterhubungan ini disebut "koneksi", serta "loncatan" keterhubungan pijakan C1 dengan C16. Sehingga dapat dihitung jumlah koneksi dan jumlah loncatan tiap pijakan yang ditempati oleh bidak.

Berdasarkan representasi papan permainan tersebut di atas, dapat dinotasikan ruang keadaannya, dengan notasi : (tipe bidak, jumlah bidak, total koneksi, total loncatan), contoh: (O, 10, 33, 0) dan (M, 2, 3, 1), dimana O = bidak manusia dan M = bidak macan. Serta untuk masing-masing bidak dapat direpresentasikan menjadi (kode bidak, kode pijakan, jumlah koneksi, jumlah loncatan), contoh : (Y, C1, 1, 1) atau (VIII, C19, 6, 0). Berikut ini adalah tabel jumlah langkah bidak dan macan pada tiap pijakan :

Tabel 1. Jumlah Langkah Bidak Manusia

Bidak / Pijakan	∑ koneksi
I / C3	4
II / C19	3
III / C14	3
IV / C7	4
V / C6	0
VI / C23	7
VII / A6	3
VIII / C17	6
IX / C11	2
X / C10	1
Total	33

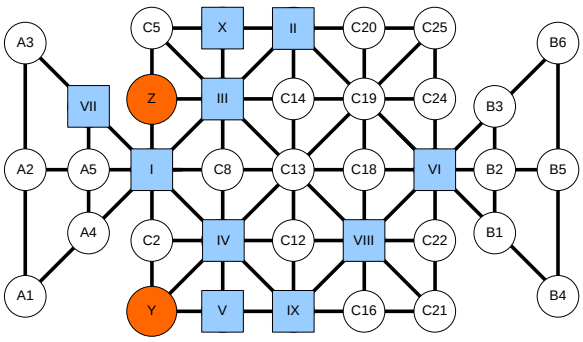
Tabel 2. Jumlah Langkah Bidak Macan

Bidak / Pijakan	∑ koneksi	∑ Loncatan
Y / C1	1	1
Z / B1	1	0
Total	3	1

3.4. Representasi ruang keadaan

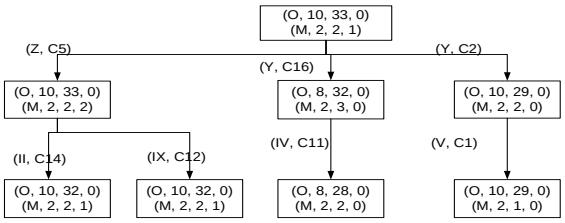
Terdapat beberapa cara untuk merepresentasikan ruang keadaan, yaitu dengan *Graph* Keadaan, Pohon Pelacakan dan Pohon *AND/OR*.

Dalam penelitian kali ini digunakan pendekatan dengan pohon pelacakan. Gambar 6 di bawah merupakan keadaan awal dari kondisi papan permainan, yang kemudian direpresentasikan ruang keadannya.



Gambar 6. Keadaan Awal Kondisi Papan Permainan

Dari keadaan awal tersebut, kemudian dihitung *heuristik*, dan kemudian dihasilkan pohon pelacakan seperti gambar 7 di bawah ini, tidak semua *node* ditampilkan dalam pohon pelacakan berikut ini.



Gambar 7. Pohon Pencarian Representasi Keadaan

3.5. Perancangan Algoritma Minimax

Dalam perancangan sistem dengan algoritma *minimax*, akan dijabarkan kemungkinan langkah yang akan diambil oleh *player* maupun AI, sehingga kemudian dapat dihitung nilai tiap *node* daun, berdasarkan fungsi evaluasi untuk permainan macan-macanan. Gambar papan permainan sama dengan gambar 8 dengan giliran langkah selanjutnya oleh bidak macan (MAX), dan hanya akan dibahas dengan kedalaman 2 node.

Tabel 3. Kemungkinan Langkah dalam Permainan

	Kemungkinan Langkah		Representasi Keadaan	
0	Kondisi Papan Permainan {(Z, C4), (Y, C1), (I, C1), (II, C15), (III, C9), (IV, C7), (V, C6), (VI, C23), (VII, A6), (VIII, C17), (IX, C11), (X, C10)}		(M, 2, 2, 1)	(O, 10, 33, 0)
1	Z, C5	: bidak Z ke pijakan C5	(M, 2, 2, 2)	(O, 10, 33, 0)
1.1	II, C14	: bidak II ke pijakan C14	(M, 2, 2, 1)	(O, 10, 34, 0)
1.2	IX, C12	: bidak IX ke pijakan C12	(M, 2, 2, 1)	(O, 10, 34, 0)
1.3	I, C4	: bidak I ke pijakan C4	(M, 2, 1, 2)	(O, 10, 31, 0)
2	Y, C2	: bidak Y ke pijakan C2	(M, 2, 2, 0)	(O, 10, 33, 0)
2.1	V, C1	: bidak V ke pijakan C1	(M, 2, 1, 0)	(O, 10, 34, 0)
2.2	X, C5	: bidak X ke pijakan C5	(M, 2, 1, 0)	(O, 10, 34, 0)
2.3	I, C8	: bidak I ke pijakan C8	(M, 2, 4, 0)	(O, 10, 33, 0)
3	Y, C16	: bidak Y ke pijakan C16	(M, 2, 3, 0)	(O, 8, 34, 0)
3.1	IX, C11	: bidak IX ke pijakan C11	(M, 2, 2, 0)	(O, 8, 30, 0)
3.2	X, C5	: bidak X ke pijakan C5	(M, 2, 2, 0)	(O, 8, 35, 0)
3.3	VI, C22	: bidak VI ke pijakan C22	(M, 2, 3, 0)	(O, 8, 29, 0)

Dari pencarian kemungkinan langkah tersebut, ditemukan *node* daun, yaitu langkah 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, 3.1, 3.2, dan 3.2. Dari langkah tersebut juga telah diketahui jumlah langkah dan jumlah loncatan semua bidak, kemudian langkah perancangan selanjutnya adalah menentukan fungsi evaluasi pada tiap *node* daun tersebut. Fungsi evaluasi dalam permainan macan-macanan ini yang digunakan adalah:

$$Eval(s) = w_1 m_1(s) + w_2 m_2(s) + \cdots + w_n m_n(s) = \sum_{i=1}^{n} w_i m_i(s) \quad (2)$$

1. w adalah bobot bidak, (bidak macan = 5, bidak manusia = 1).
2. Sementara $m(s)$ adalah jumlah langkah *valid* (*valid move*) dari bidak, untuk loncatan dihitung dengan pengali 8. Serta jumlah bidak yang terloncati akan menambah nilai dari bidak macan (jumlah bidak terloncati dikali pengali) yaitu (2 x 8).

Diambil contoh untuk *node* / langkah (1.3), dengan representasi keadaan {(M, 2, 1, 2) (O, 10, 31, 0)}. Maka dapat dihitung

$$\sum_{i=1}^{2} w_i m_i(s)_{max} = 5.(1+1.8) + 5.(0+1.8) = 5.9 + 5.8 = \mathbf{85} \quad (3)$$

$$\sum_{i=1}^{10} w_i m_i(s)_{min} = 1.1 + 1.3 + 1.5 + 1.4 + 1.0 + 1.7 + 1.3 + 1.6 + 1.2 + 1.0 = 1 + 3 + 5 + 4 + 0 + 7 + 3 + 6 + 2 + 0 = \mathbf{31} \quad (4)$$

Kemudian dihitung nilai untuk masing-masing player. Dengan persamaan di bawah ini, sehingga ditemukan nilai total, yang nantinya akan menjadi perhitungan AI dalam pengambilan langkah :

$$Nilai_{max} = \mathbf{85} \quad (5)$$

$$Nilai_{min} = \mathbf{31} \quad (6)$$

$$Nilai_{total} = Nilai_{max} - Nilai_{min} = \mathbf{54} \quad (7)$$

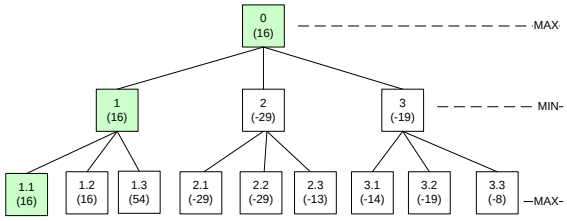
Apabila ditemukan adanya *node* dimana salah satu MAX atau MIN memenangkan pertandingan, maka persamaan yang dipakai adalah :

$$Nilai_{max} WIN = 100 \quad (8)$$

$$Nilai_{min} WIN = -100 \quad (9)$$

Setelah diketahui fungsi evaluasi untuk mencari nilai *node* daun, maka dapat dihitung nilai untuk masing-masing langkah yang akan ditempuh oleh *player* maupun AI.

Dari hasil fungsi evaluasi, maka didapatkan nilai total tiap *node* daun, kemudian dibuat pohon pencarian dan selanjutnya akan diterapkan algoritma *minimax* yang bisa dilihat pada gambar 8 berikut ini :



Gambar 8. Pohon *Minimax* kemungkinan langkah

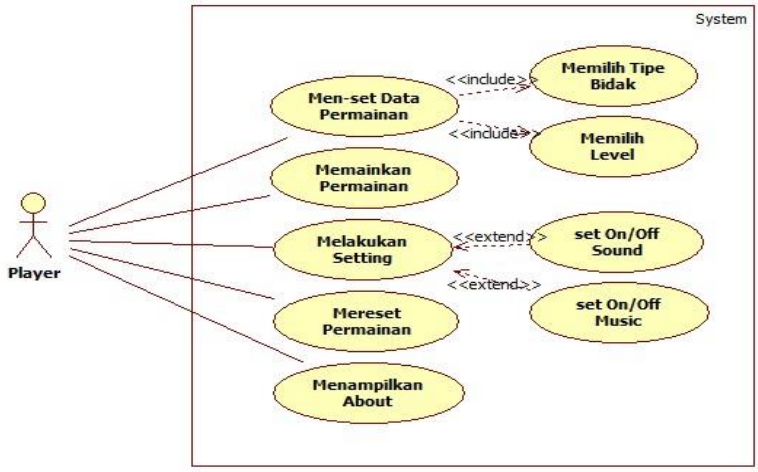
Penjabaran Pohon *minimax* :

1. AI akan menandai *node* (0) sebagai *node* awal, dan mencari kemungkinan langkah selanjutnya.
2. AI memprediksi langkah selanjutnya adalah *node* (1), karena *node* (1) bukan merupakan *terminal state* (salah satu bidak menang) ataupun tingkat kedalaman *node* = 2, maka prediksi langkah berlanjut ke *node* selanjutnya, (4.1)
3. Diperoleh *node* (1.1) dengan kedalaman = 2, maka *node* (1.1) dihitung nilainya dengan fungsi evaluasi (lihat tabel 4.3) menghasilkan nilai (16). Nilai tersebut disimpan di *node* awal / *parent* (1). Karena masih ada kemungkinan langkah lain, maka dibuat prediksi langkah.
4. *Node* (1.2), dihitung dan dibandingkan dengan nilai dari *node parent* (16), karena *node* (1) berfungsi sebagai MIN, maka diambil nilai terkecil antara nilai *node parent* dan *node* (1.2), karena sama maka diambil nilai sebelumnya (-14). (4.2)
5. *Node* (1.3) adalah kemungkinan langkah lain dengan nilai (54), karena lebih besar dari *node* (1) maka diambil nilai *node* (1). Nilai node (1) disimpan di *node root* (0).
6. Masih ada prediksi langkah yaitu *node* (2), seperti poin 2, karena bukan terminal state, maka dibuatlah *node child*, seperti langkah pada poin 2, 3 dan 4. Yaitu *node* (2.1, 2.2, 2.3), *node* (2) memiliki nilai -29. (4.3)
7. *Node root* (0) berfungsi sebagai MAX, yang membandingkan nilai terbesar antara nilai awal yang didapat dari *node* (1) dan *node* (2), karena 16 > -29, maka nilai yang dipakai adalah 16.
8. Sama seperti poin (6), maka dihitung *node* (3) dengan nilai -19, dan sama seperti poin 7, karena 16 > -19, maka yang dipakai adalah nilai *node* (1). Dan langkah yang diambil selanjutnya adalah langkah (1), (lihat tabel 4) langkah tersebut adalah memindah bidak Z ke C5.

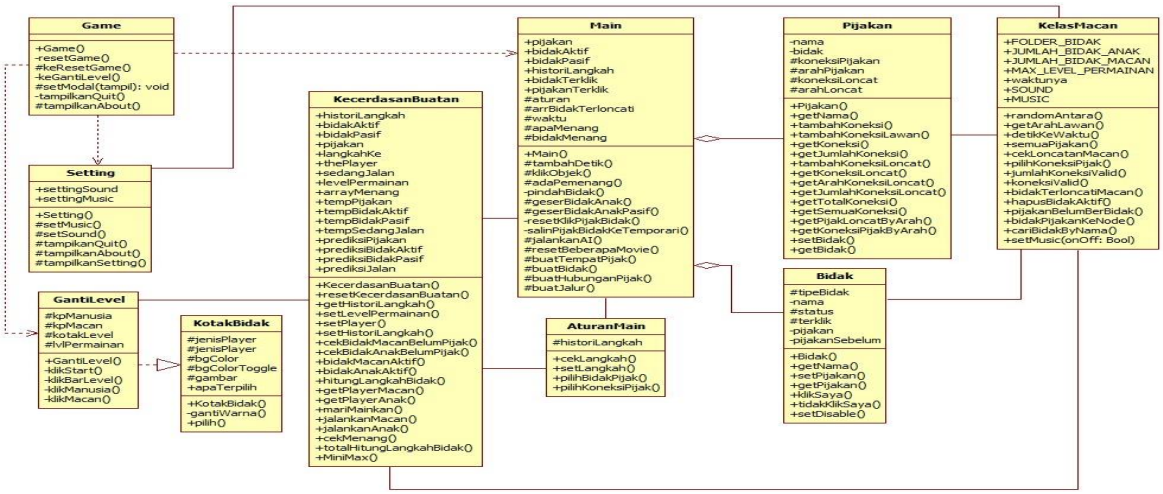
Tabel 4. Kemungkinan Langkah dalam Permainan

Langkah		Representasi Keadaan		Nilai Max	Nilai Min	Nilai Total
1.1	II, C14	(M, 2, 2, 1)	(O, 10, 34, 0)	50	34	16
1.2	IX, C12	(M, 2, 2, 1)	(O, 10, 34, 0)	50	34	16
1.3	I, C4	(M, 2, 1, 2)	(O, 10, 31, 0)	85	31	54
2.1	V, C1	(M, 2, 1, 0)	(O, 10, 34, 0)	5	34	-29
2.2	X, C5	(M, 2, 1, 0)	(O, 10, 34, 0)	5	34	-29
2.3	I, C8	(M, 2, 4, 0)	(O, 10, 33, 0)	20	33	-13
3.1	IX, C11	(M, 2, 2, 0)	(O, 8, 30, 0)	16	30	-14
3.2	X, C5	(M, 2, 2, 0)	(O, 8, 35, 0)	16	35	-19
3.3	VI, C22	(M, 2, 3, 0)	(O, 8, 29, 0)	21	29	-8

3.6. Use Case Diagram



Gambar 9. *Use Case Diagram*



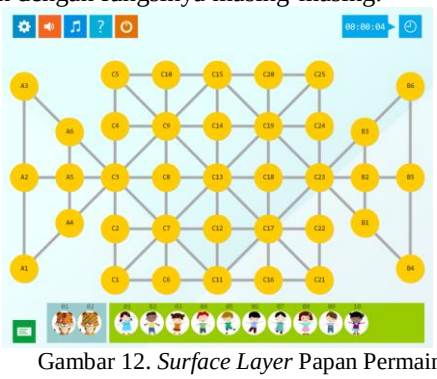
Gambar 10. *Class diagram*

4. Hasil dan Pembahasan

Hasil penelitian ini adalah *game* macan-macanan yang mempunyai tampilan awal seperti pada gambar 11.

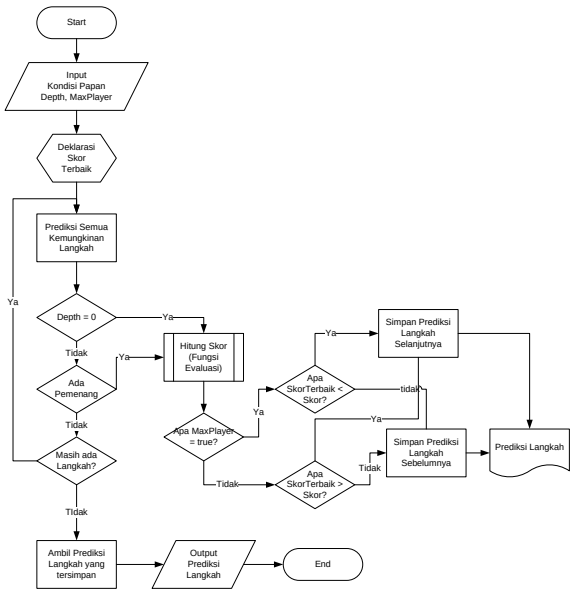
Gambar 11. Tampilan Awal *Game*

Pada tampilan ini, disajikan beberapa *icon*, dan warna yang cukup kontras, sehingga dapat dibedakan untuk masing-masing fungsinya. Terdapat pilihan *level*, yang berbeda warna. Serta fungsi tombol *setting* yang yang ketika diklik akan memunculkan icon dengan fungsinya masing-masing.



Gambar 12. *Surface Layer* Papan Permainan

Pada gambar 12 ditampilkan pijakan ditandai dengan kode di tengah, dan bidak dengan kode di atasnya. Pijakan dibuat semirip mungkin dengan permainan fisiknya.



Gambar 13. *Flowchart* Implementasi Algoritma *Minimax* ke Permainan

Ketika agen / AI ingin mengambil langkah selanjutnya, AI akan mencari perhitungan dengan menggunakan algoritma *minimax*, tahapannya adalah :

1. Deklarasi *variable* skor terbaik, dengan nilai – *INFINITY* untuk MAX *player*, dan *INFINITY* untuk min.
2. Kemudian AI memprediksi beberapa langkah selanjutnya.
3. Jika kedalaman (*depth*) = 0, maka hitung skor sementara dengan fungsi evaluasi dari kondisi papan permainan. Jika *depth* != 0 lanjut ke poin d.
4. Jika ada pemenang dalam kondisi papan permainan, maka hitung skor sementara dengan fungsi evaluasi. Jika belum ada pemenang, lanjut ke poin *h*.
5. Setelah perhitungan skor dengan fungsi evaluasi, skor sementara akan disimpan dan dibandingkan dengan *variable* skorTerbaik.
6. Jika *Max Player* (bidak macan), maka bandingkan, apakah skor sementara > skorTerbaik? Jika ya, simpan langkah dan update skor terbaik = skor sementara. Jika tidak, simpan prediksi langkah sebelumnya.
7. Jika Min Player (bidak manusia), maka bandingkan, apakah skor sementara < skor terbaik? Jika ya, simpan langkah dan update skor terbaik = skor sementara. Jika tidak, simpan prediksi langkah sebelumnya.
8. Kemudian Apakah masih ada langkah selanjutnya? Jika ada maka menuju poin b. Jika tidak ada, maka ambil prediksi langkah yang tersimpan, dan kirim parameter ke pemroses langkah.

Tabel 5. Pengujian *Interface* Sistem

Test Case	Hasil		Keterangan
	Diharapkan	Didapatkan	
Pengujian fungsi *Music*	*Music off* ketika diklik tombol *Music*, dan *on* ketika diklik ulang	*Music off* ketika diklik tombol *Music*, dan *on* ketika diklik ulang	Sesuai
Pengujian fungsi *Sound*	Jika tombol *sound* aktif, ketika memilih bidak akan terdengar suara, jika *non-aktif*, suara tidak terdengar	Saat tombol *sound* aktif, terkadang suara tidak terdengar	Kurang Sesuai
Pengujian fungsi pemilihat tipe bidak	Tombol *start non-aktif* jika *player* tidak memilih tipe bidak	Tombol *start* aktif, ketika *player* memilih salah satu bidak	Sesuai
Pengujian fungsi aturan main	Yang pertama melangkah adalah bidak macan	Yang dapat melangkah pertama adalah bidak macan	Sesuai
	Langkah ganjil adalah bidak macan, dan genap adalah bidak manusia	Langkah ganjil adalah langkah macan, dan genap untuk manusia	Sesuai
	Seluruh bidak tiap tipe harus terletak di papan permainan, sebelum dapat bergeser antar pijakan	Bidak aktif pada kotak bidak, harus habis terlebih dahulu, sebelum bidak dapat bergeser ke pijakan lain	Sesuai
	Bidak hanya dapat bergeser ke piakan yang kosong	Tidak dapat bergeser ke pijakan yang memiliki bidak	Sesuai
	Player macan bisa loncat jika terdapat 2 bidak manusia sejajar	*Player* macan dapat meloncati 2 pemain sejajar.	Sesuai
	Tampil peringatan pemenang, jika ada pemenang	Ada peringatan pemenang, jika salah satu bidak menang	Sesuai

5. Kesimpulan

Berdasarkan pembahasan dan penelitian yang telah dipaparkan pada bab-bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut :

1. Algoritma *minimax* dan algoritma turunannya masih menjadi pilihan untuk *board game* yang berbasis pada 2 *player*. Namun banyaknya pohon pencarian akan menambah lamanya waktu bagi komputer untuk mengambil langkah.
2. Untuk mendapatkan bobot nilai *node* daun, diperlukan fungsi evaluasi terhadap papan permainan, yang berdasar pada jumlah langkah *valid* yang dapat dilakukan oleh tiap-tiap bidak disertai bobot untuk masing-masing tipe bidak.
3. *ActionScript 3* yang digunakan untuk pemrograman berbasis objek, sangat membantu untuk penerapan permainan macan-macanan ke *game* komputer, AS3 dapat digunakan untuk membuat game yang cukup menarik, serta implementasi algoritma *minimax* ke AS3 dapat dilakukan dan menambah interaksi dengan AI.
4. Pada pengujian masih terdapat kekurangan yang tidak terlalu esensial, seperti penggunaan AS3 untuk *sound*, yang terkadang dapat dibunyikan dan terkadang tidak. Serta *context menu* yang belum diimplementasikan ketika pembuatan aplikasi ini.

Daftar Pustaka

Dharmamulya, S., 2004. Permainan Tradisional Jawa. Yogyakarta : Kepel Press.

Dirgantara, Y.A., 2012. Pelangi Bahasa Sastra dan Budaya Indonesia Kumpulan Apresiasi dan Tanggapan. Yogyakarta : Garudhawaca Digital Book and POD.

Fulton, J. dan Fulton, S., 2010. The Essential Guide to *Flash* Games. New York : USA. Springer-Verlag.

Garrett, J.J., 2011. The Elements Of User Experience (2nd ed.), Berkeley : USA, Pearson Education.

Kristanto, A., 2004. Kecerdasan Buatan. Yogyakarta : Graha Ilmu.

Pressman, R. S,. 20011. Software Engineering A Practitioner's Approch (7th ed). New York, USA : McGraw-Hill.

Sutojo, T., Mulyanto, E., Suhartono, V., 2011. Kecerdasan Buatan. Yogyakarta : Penerbit Andi.

Vebrina, Y.G., 2008. *Penerapan Q Learning pada Aplikasi Board Game*. Skripsi. Bandung : Institut Teknologi Bandung.