PENDEKATAN ALGORITMA *GREEDY* UNTUK MENENTUKAN LANGKAH BIDAK PADA PERMAINAN *CHECKERS*

Akhsani Taqwiym[1)]

[1)] *Komputerisasi Akuntansi STMIK Global Informatika MDP*
email : Akhsani.taqwiym@mdp.ac.id [1)]

Abstraksi

Checkers merupakan jenis permainan *board game*, yaitu jenis permainan yang memanfaatkan papan sebagai alat bantu permainannya. *Checkers* dipermainkan oleh 2 orang dengan akhir menghabiskan seluruh kepingna lawan. Dengan perkembangan teknologi informasi pada saat ini, sangat memungkinkan semua orang bisa memainkan permainan ini dimanapun dan kapanpun bahkan bisa dimainkan oleh 1 orang saja, dengan lawan bermain yang dikendalikan oleh komputer. Permainan *checkers* akan dikembangkan dengan menggunakan kecerdasan buatan dengan menerapkan algoritam greedy. Algoritma greedy sendiri diharapkan dapat menentukan langkah-langkah kepingan bidak dengan waktu yang lebih cepat sehingga dapat menyelesaikan permainan *checkers* lebih cepat.

***Kata Kunci* :** *Checkers, Algoritma Greedy, Artificial Intelligence*

Abstract

Checkers are a type of board game game, which is a type of game that uses the board as a play aid. Checkers are played by 2 people who end up spending all of their opponents. With the development of information technology at this time, it is possible for everyone to play this game wherever and whenever it can even be played by one person, with opponents playing who are controlled by a computer. Game checkers will be developed using artificial intelligence by applying greedy algorithm. The greedy algorithm itself is expected to be able to determine the steps of pieces of pieces with a faster time so that they can complete the game of checkers faster

***Keywords* :** *Checkers, Algoritma Greedy, Artificial Intelligence*

Pendahuluan

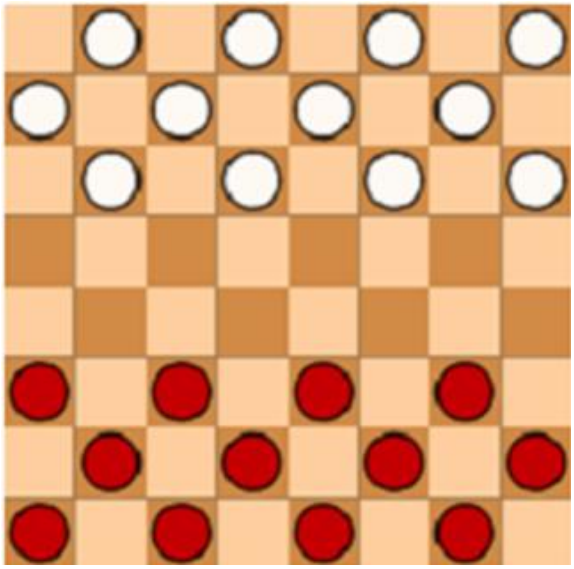
Permainan jenis *board games* juga sudah dikembangkan secara *virtual* pada ponsel. Kemajuan teknologi informasi sangat memungkinkan seseorang bisa memainkan *game* tertentu dimanapun dan kapanpun. Kecerdasan Buatan merupakan cabang dari ilmu komputer yang konsern dengan pengautomatisasi tingkah laku cerdas [1]. Dengan menggunakan kecerdasan buatan (AI) akan membuat komputer yang memainkan permainan tersebut seperti manusia. Algortima heuristik digunakan dalam ruang keadaan masalah secara slektif dan memiliki kemungkinan yang paling besar dalam menyelesaikan suatu masalah[2]. Salah satu algoritma heuristik yang pernah digunakan adalah algoritma MiniMax pada penelitian [3]. Pada penelitian tersebut didapat kelemahan algoritma MiniMax yang memerlukan waktu pencarian yang cukup lama, meskipun algoritma Minimax merupakan algoritma yang teruji dan dijamin dapat menemukan solusi terbaik, keterbatasan *memory* dan *cpu time* menyebabkan algoritma ini memiliki kelemahan, pada kondisi yang *end state* atau *goal state* berada pada lapisan pohon yang sangat dalam dimana untuk menghitungnya memerlukan *cpu time* yang sangat lama dan besarnya *memory* yang harus disediakan untuk menampung seluruh *node* yang akan dikomputasi [3]. Oleh karena itu, algoritma yang akan digunakan dalam penelitian ini adalah algoritma *Greedy*.

Algoritma *Greedy* adalah algoritma yang dapat menyelesaikan sebuah masalah dengan waktu yang singkat[4]. Algoritma ini diharapkan mampu mengambil keputusan dalam menentukan langkah bidak dengan waktu yang lebih cepat sehingga mampu menyelesaikan permainan. Jenis permainan papan akan diselesaikan dengan algoritma greedy ialah *checkers.*

Tinjauan Pustaka

Permainan Checkers

Permainan papan *checkers* merupakan permainan yang dimainkan oleh 2 orang dengan cara memilih salah satu kepingan. Setiap pemain akan memilih kepingan yang berbeda sebagai penanda saat permainan pada papan *checkers.* Dalam memainkan permaian *checkers* terdapat beberapa komponen, diantaranya : pemain, papan permainan, koin, posisi awal, cara bergerak, raja, dan permainan selesai.



Gambar 1. Posisi Awal Permainan Checkers

Kecerdasan Buatan (AI)

Kecerdasan Buatan merupakan cabang dari ilmu komputer yang konsern dengan pengautomatisasi tingkah laku cerdas. Keberhasilan suatu sistem salah satunya ditentukan oleh kesuksesan dalam pencarian dan pencocokan.

Konsep pencarian yang dilakukan dengan AI akan menjadikan hasil yang didapatkan lebih unggul dibandingkan dengan metode lainnya. Prinsip kontribusi kecerdasan buatan untuk ilmu pengetahuan dari pencarian ini merupakan konsep basis pengetahuan (*knowledge based*) heuristik [5].

Game playing dan pemecahan teorema (*theorem solving*) adalah dua aplikasi yang paling tua dari kecerdasan buatan, dan keduanya memerlukan heuristik untuk memangkas ruang dari kemungkinan solusi [1].

Algoritma MiniMax

Pada algoritma Minimax, lawan direpresentasikan dengan min (*minimize*). *Player* sebagai lawan dari min di representasikan dengan max (*maximize*). Max merupakan singkatan dari *maximize* yang memiliki arti *player* sebagai pemain menginginkan nilai yang semaksimal mungkin untuk meraih kemenangan. Min atau singkatan dari *minimaze* adalah lawan *player* yang akan mencoba segala cara untuk meminimalkan nilai yang dapat *player* peroleh yang dapat menyebabkan *player* memenangkan permainan.

Dalam mengimplementasikan algoritma Minimax, *Tree* di pilih sebagai struktur data[2]. Setiap level pada *tree* merepresentasikan sebagai pergerakan dan di beri label min atau max, tergantung pada pihak mana yang melangkah. Selain melabel setiap level pada *tree*, algoritma Minimax memberikan nilai pada setiap *node* untuk memberikan informasi bahwa *node* tersebut menguntungkan untuk max atau min. Biasanya informasi tersebut berupa 1 atau 0. Arti dari 1 atau 0 tersebut adalah *node* yang diberi keterangan 1 merupakan *node* yang menguntungkan untuk max. Sedangkan *node* yang diberi keterangan 0 merupakan *node* yang menguntungkan min. Setelah seluruh proses penilaian selesai, algoritma Minimax memilih *node* yang memiliki nilai max terbesar dengan asumsi *node* tersebut merupakan *node* yang paling menguntungkan bagi *player* dengan memaksimalkan dan meminimalkan nilai lawan [2].

Algoritma Greedy

Algoritma *greedy* adalah algoritma yang cara kerjanya mirip dengan salah satu sifat buruk manusia, yaitu rakus. Algoritma greedy ini praktis, ringkas dan *fleksibel* (dapat digunakan pada berbagai persoalan dengan hasil yang cukup memuaskan) [6].

Setiap langkah yang menjadi pilihan terbaik dapat diperoleh saat itu (tidak bisa diubah, diulang, serta tidak perlu memperhatikan keputusan selanjutnya) dan berharap bahwa dengan membuat pilihan yang terlihat seperti solusi terbaik dengan membuat solusi optimum lokal, akan mencapai optimum global. Algoritma *greedy* mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global. Prinsip algoritma *greedy* adalah: "*take what you can get now*!" [6].

Perbandingan Algoritma MiniMax dan Algoritma Greedy

Meskipun algoritma Minimax merupakan algoritma yang teruji dan dijamin dapat menemukan solusi terbaik, keterbatasan *memory* dan *cpu time* menyebabkan algoritma ini memiliki kelemahan, pada kondisi yang *end state* atau *goal state* berada pada lapisan *tree* yang sangat dalam dimana untuk menghitungnya memerlukan *cpu time* yang sangat lama dan besarnya *memory* yang harus di sediakan untuk menampung seluruh *node* yang akan dikomputasi [3].

Jika dibandingkan dengan algoritma Minimax dalam permasalahan waktu, algoritma *Greedy* lebih unggul. Algoritma *Greedy* tetap menjadi pilihan utama untuk permasalahan yang sederhana, karena ini adalah metode yang paling cepat dibanding metode yang lain, menggunakan memori sedikit/kecil, dan dapat memberikan solusi hampiran atau aproksimasi terhadap nilai optimum yang diinginkan, serta hasil yang diberikan masih merupakan solusi yang layak (feasible solution) [7].

Dengan keunggulan Algoritma *Greedy*, diharapkan nantinya kecerdasan buatan dapat menentukan langkah bidak permainan *Checkers* dengan waktu perhitungan yang lebih singkat.

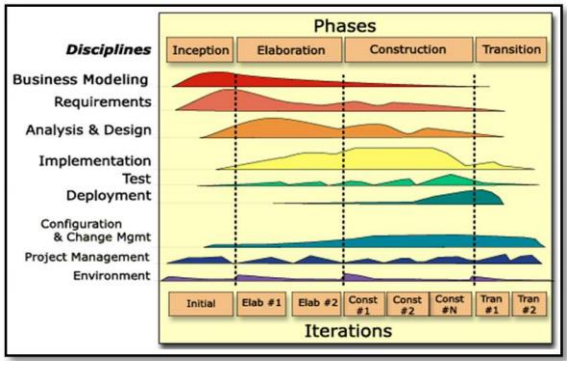
Rational Unified Process (RUP)

Metode pengembangan perangkat lunak berorientasi *object* memiliki beberapa keunggulan yaitu, pada metode ini titik berat pengembangan perangkat lunak ada di tahap analisis, tetapi mempermudah tugas pada bagian implementasi karena model-

model yang dihasilkan lebih mudah diadaptasi ke dalam program [8].

Pada tahap implementasi, program yang dihasilkan akan lebih modular. Metode pengembangan perangkat lunak berorientasi *object* merupakan proses pengembangan berkesinambungan dimana setiap hasil dari fase sebelumnya akan digunakan pada fase berikutnya.

Berikut ini adalah *workflow* dari konteks analis dan desain dengan menggunakan proses pengembangan perangkat lunak *Rational Unified Process* (RUP) [9].



Gambar 2. *Workflow* RUP [9]

Unified Modeling Languange (UML)

Unified Modeling Language (UML) dibuat berdasarkan grafik untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis *Object Oriented* (OO). Diagram UML seperti *Use Case* Diagram, *Class* Diagram, dan *Sequence* Diagram [10].

Metode Penelitian

Perangkat lunak yang dibangun merupakan perangkat lunak berbasis *mobile* (*phone based*) berupa permainan *Checkers* yang dapat digunakan pada ponsel yang berbasis Java MIDP. Perangkat lunak (*game*) memiliki 3 tingkat kesulitan pada komputer (*Artificial Intelligence*) yang meliputi : *easy* (mudah), *medium* (menengah), dan *hard* (susah). Sebelum bermain, *player* harus memilih terlebih dahulu tingkat kesulitan mana yang akan diambilnya untuk dimainkan.

Game dimainkan pada bidang berukuran 8x8 dengan kotak-kotak kecil berwarna hitam-putih seperti papan catur. *Game* ini menggunakan koin sebagai bidaknya yang berbentuk silinder datar seperti koin pada umumnya, yang berwarna gelap dan terang. Masing-masing pemain memiliki 12 koin yang diletakkan pada 3 baris pertama pada bidang yang terdekat dengan pemain dan diletakkan pada bidang yang berwarna hitam.

Game ini dimainkan oleh seorang *player* dengan komputer (*Artificial Intelligence*) yang bertindak sebagai musuh. *Player* menjalankan koin dengan memilih koin dari kiri maupun kanan dan menetapkan koin pilihannya untuk dijalankan. Selanjutnya giliran komputer yang menjalankan koin. Koin bergerak secara diagonal engikuti warna bidang hitam, satu di tiap langkahnya.

Cara lain adalah dengan melangkahi satu buah koin lawan. Keadaan itu mungkin dapat dilakukan jika pada diagonal setelah koin lawan, merupakan bidang kosong. Jika langkah kedua itu terjadi, koin lawan yang dilangkahi mati, dan harus keluar dari bidang permainan. Koin dengan pangkat "biasa" hanya dapat bergerak maju. Namun, koin dengan pangkat "raja", dapat bergerak maju maupun mundur. Permainan selesai ditandai oleh habisnya koin lawan pada bidang permainan atau koin sudah tidak dapat bergerak kemanapun.

Fitur Utama Perangkat Lunak

Perangkat lunak yang akan dibangun memiliki fitur-fitur yang terdiri dari kebutuhan fungsional dan non-fungsional.

Tabel 1. Kebutuhan Fungsional

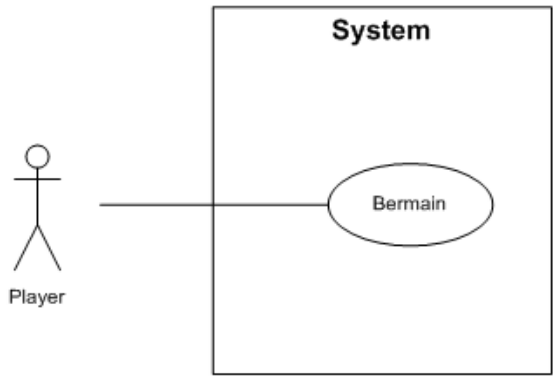
No	Kebutuhan
1	Dapat melakukan proses pemilihan level yang akan dimainkan
2	Dapat melakukan pemilihan, penetapan, dan pembatalan penetapan koin
3	Dapat melakukan pemilihan jalan, dan penjalanan koin.
4	Dapat menampilkan hasil yang diraih, baik kemenangan atau kekalahan.

Tabel 2. Kebutuhan Non-Fungsional

No	Kebutuhan
1	Bidang permainan beserta isinya dihasilkan dari grafis standar, bukan merupakan bentuk *image*, sehingga lebih menghemat *memory*.
2	Sistem mudah dipelajari dan digunakan karena itu sistem harus tidak membingungkan, ditunjang dengan fasilitas *Rule.*

Diagram *Use Case*

Pada *use case* ini digunakan oleh aktor (*player)* dalam menjalankan permainan mulai dari memilih tingkat kesulitan permainan yang terdiri dari 3 level, yaitu *easy*, *medium*, dan *hard*, memilih koin, memilih jalan koin, menjalankan koin dan menetapkan penetapan koin.

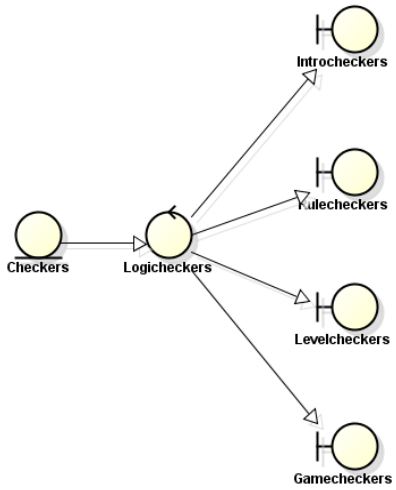


Gambar 2. Diagram *Use Case*

Kelas Analisis

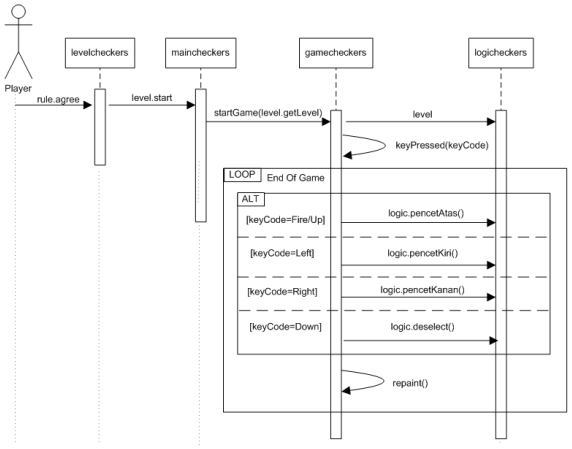
Tabel 3. Daftar Kelas Analisis

No	*Nama Kelas*	*Jenis*	*Keterangan*
1	Checkers	Entitas	Merupakan kelas MIDlet untuk menjalankan aplikasi berbasis Java MIDP
2	Logicheckers	Control	Merupakan fungsi utama pengendali permainan yang terdapat juga *Algoritma Greedy* didalamnya.
3	Introcheckers	Interface	Antarmuka tampilan awal
4	Rulecheckers	Interface	Antarmuka tampilan *rule* untuk permainan
5	Levelcheckers	Interface	Antarmuka untuk memilih level permainan
6	Gamecheckers	Interface	Antarmuka tampilan bidang permainan yang bersesuaian dengan logicheckers. Antarmuka ini juga menampilkan pesan menang atau kalah bagi *player*.



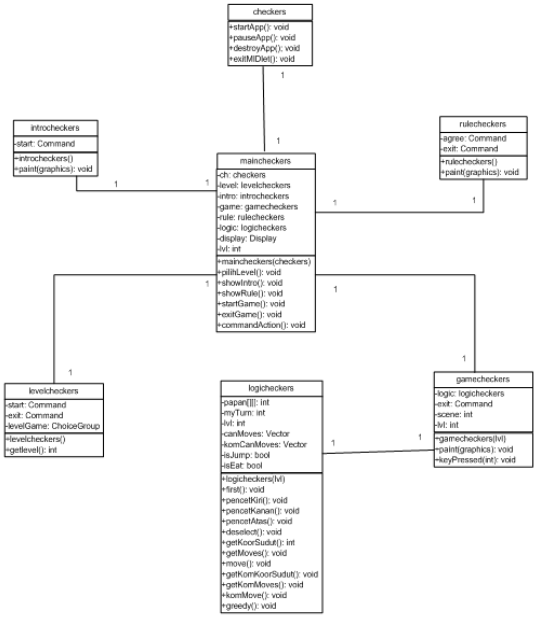
Gambar 3. Kelas Analisis

Sequence Diagram



Gambar 4. *Sequence Diagram* Bermain

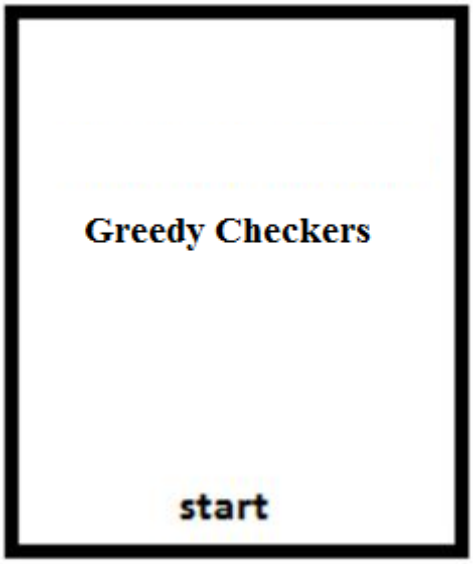
Kelas Diagram Keseluruhan



Gambar 5. Kelas Diagram Keseluruhan

Perancangan Antarmuka

Interface ini hanya berupa tampilan pembuka dari *game*. Berisi nama *game*, gambar papan ditengah, dan tombol *start*. Jika menekan tombol *start* maka akan masuk ke *interface rule.*



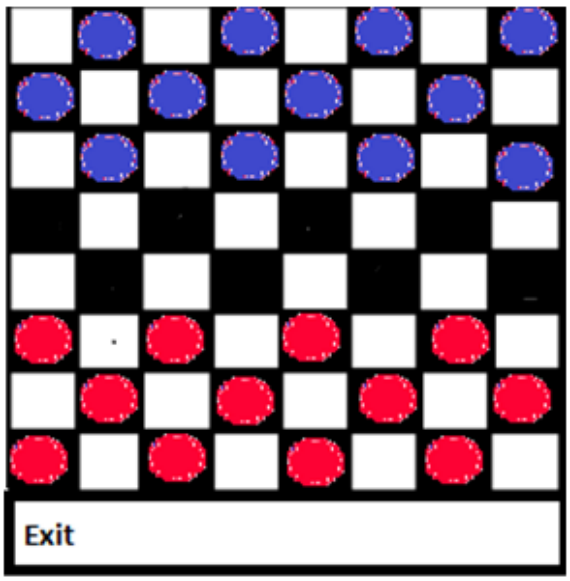
Gambar 6. Rancangan Tampilan Utama

Interface ini berisi tulisan tentang petunjuk tombol-tombol yang dapat digunakan, tombol *agree,* dan tombol *exit*. Jika menekan tombol *agree* maka akan masuk ke *interface* level, sedangkan jika menekan tombol *exit* maka akan langsung keluar dari aplikasi.

Gambar 7. Rancangan Tampilan Rule

Gambar 8. Rancangan Tampilan Level

Interface ini berisi pilihan level yang dapat dipilih oleh *player*, yaitu *easy*, *medium*, dan *hard*. Dan tombol-tombol yang dapat digunakan yaitu tombol *start,* dan tombol *exit*. Jika menekan tombol *start* maka akan masuk ke *interface game* (papan permainan), sedangkan jika menekan tombol *exit* maka akan langsung keluar dari aplikasi



Gambar 9. Rancangan Tampilan Board Game

Jika diakhir permainan *player* berhasil memenangkan permainan maka akan ditampilkan pesan sebagai berikut :

Gambar 10. Rancangan Tampilan Menang

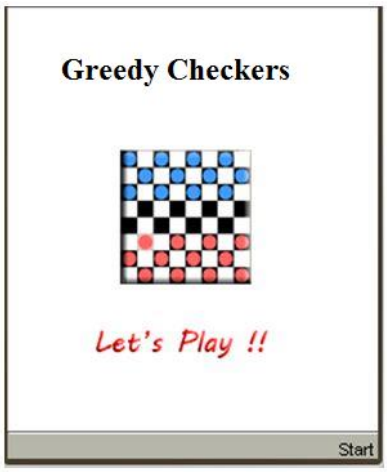
Tetapi jika diakhir permainan *player* dinyatakan kalah, maka pesan yang akan tampil sebagai berikut :

Gambar 11. Rancangan Tampilan Kalah

Hasil dan Pembahasan

Implementasi Antarmuka

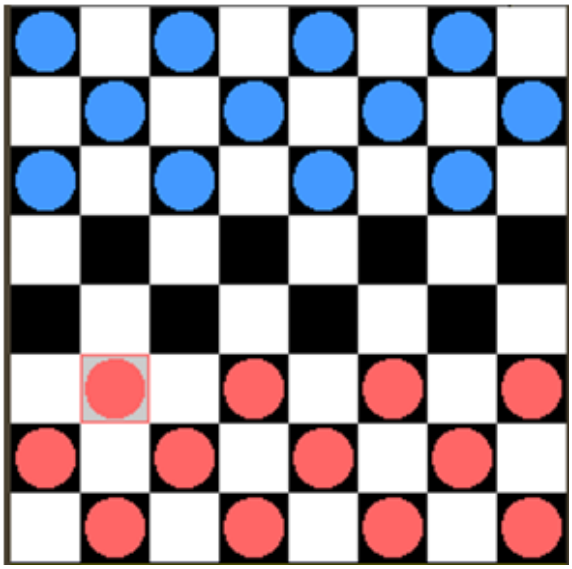
Hasil implementasi antar muka perangkat lunak yang telah dibangun adalah sebagai berikut



Gamabr 12. Antarmuka Tampilan Utama

Gambar 13. Antarmuka Tampilan Rule

Gambar 14. Antarmuka Tampilan



Gambar 15. Antarmuka Papan Permainan Checkers

Gambar 16. Antarmuka Tampilan Menang

Gambar 17. Antarmuka Tampilan Kalah

Pengujian Perangkat Lunak

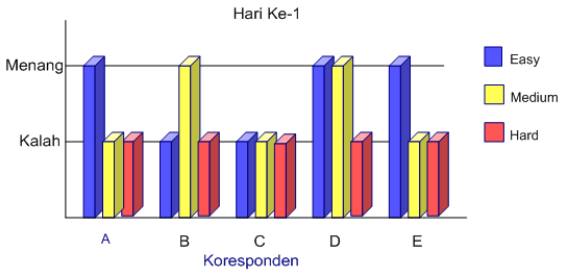
Pada algoritma MiniMax, *tree* memiliki pohon binary sebagai pohon pencarian. Perbandingan kecepatan algoritma *greedy* dengan algoritma MiniMax dapat dilihat pada tabel berikut:

Tabel 4. Perbandingan Kecepatan Algoritma Greedy dan MiniMax

Level	Algoritma MiniMax	Algoritma Greedy
Level *Easy*	1 detik	0,17 detik
Level	3 detik	0,20 detik
Level *Hard*	7 detik	0,24 detik

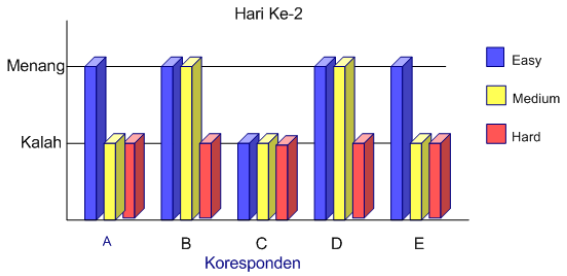
Berdasarkan Tabel 4 terlihat jelas perbandingan kecepatan antara Algoritma *Greedy* dan Algoritma MiniMax dalam permainan *Checkers* dimana Algoritma *Greedy* jauh lebih cepat. Sehingga dapat dibuktikan Algoritma *Greedy* dapat menentukan langkah bidak permainan *Checkers* dengan waktu perhitungan yang lebih singkat.

Selain itu dilakukan juga pengujian program kepada 5 orang koresponden. Kelima koresponden tersebut menjalankan program dan bermain dalam tiga tingkatan level yang telah disediakan selama 7 kali percobaan pada hari yang berbeda.



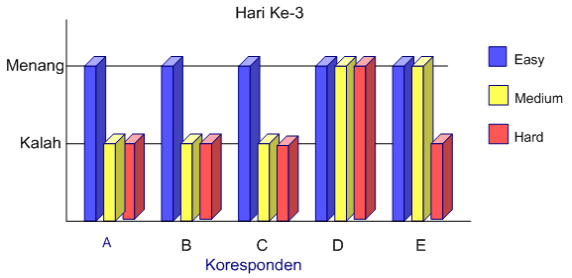
Gambar 18. Grafik Percobaan Pertama

Koresponden dapat memenangkan permainan : 60% pada level *easy*, 40% pada level *medium*, dan 0% pada level *hard.*



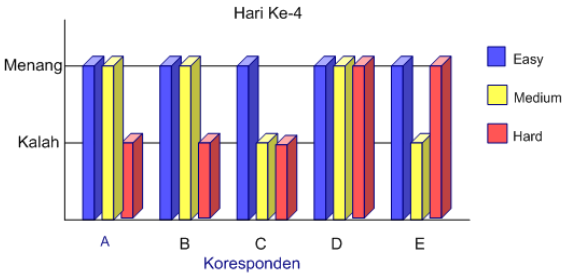
Gambar 19. Grafik Percobaan Kedua

Koresponden dapat memenangkan permainan : 80% pada level *easy*, 40% pada level *medium*, dan 0% pada level *hard.*



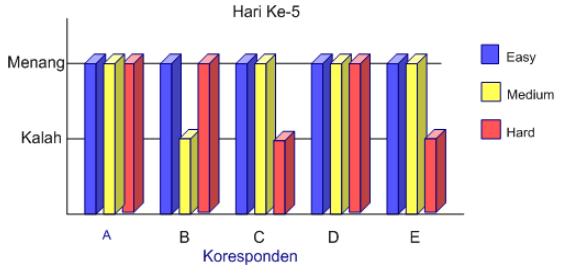
Gambar 20. Grafik Percobaan Ketiga

Koresponden dapat memenangkan permainan : 100% pada level *easy*, 40% pada level *medium*, dan 20% pada level *hard*



Gambar 21. Grafik Percobaan Keempat

Koresponden dapat memenangkan permainan : 100% pada level *easy*, 60% pada level *medium*, dan 40% pada level *hard.*



Gambar 22. Grafik Percobaan Kelima

Koresponden dapat memenangkan permainan : 100% pada level *easy*, 80% pada level *medium*, dan 60% pada level *hard.*

Berdasarkan pengujian program kepada 5 orang koresponden didapat hasil bahwa sampai percobaan pada hari ke-3 level *easy* sudah dapat dimenangkan oleh semua koresponden. Sampai percobaan pada

hari ke-5 level *medium* hanya 80% yang dapat dimenangkan oleh semua koresponden, tetapi untuk level *hard* hanya 60% koresponden yang dapat memenangkannya.

Hasil ini dirasa cukup memuaskan karena penggunaan Algoritma *Greedy* yang cepat dalam mengambil tindakan dapat memberikan tingkat kesulitan yang cukup baik pada level *medium* maupun *hard*. Tetapi jika untuk memberikan tingkat kesulitan yang maksimal Algoritma *Greedy* dirasa masih kurang handal.

Kesimpulan dan Saran

Kesimpulan

Kesimpulan yang diperoleh setelah melakukan penelitian in yaitu pengembangan permainan *Checkers* menggunakan Algoritma *Greedy* telah berhasil dilakukan, namun masih memiliki kelemahan, yaitu algoritma ini kurang bekerja maksimal. Hal ini dikarenakan Algoritma *Greedy* mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi kedepannya

Saran

Saran yang ingin penulis sampaikan untuk pengembangan lebih lanjut dari penelitian ini diantaranya sebagai berikut:

1. Penggunaan jenis ponsel agar dapat digunakan pada *game Checkers* ini diusahakan memiliki resolusi lebar layar lebih kecil dari tinggi layar. Jika sebaliknya maka gambar bidang dan koin permainan akan menjadi terlalu lebar.
2. Untuk memperoleh hasil yang lebih maksimal, maka perlu dilakukan pengembangan lebih lanjut mengenai pendekatan Algoritma *Greedy* pada permainan *Checkers* ini.

[10] Daftar Pustaka

[1] A. Desiani and M. Arhami, *Konsep kecerdasan buatan*. Yogyakarta: Andi Offset, 2006.

[2] J. Setiawan, F. A. Famerdi, D. Udjulawa, and Yohannes, "Perbandingan Performa Algoritma Minimax dan Breadth First Search Pada Permainan Tic-Tac-Toe," *JuTISI*, vol. 4, no. 1, 2018.

[3] D. Syapnika and E. R. Siagian, "Penerapan Algoritma Minimax Pada Permainan Checkers," *J. Ris. Komput.*, vol. 2, no. 6, pp. 28–32, 2015.

[4] A. Sirait, "Perancangan Dan Pembuatan Game Edukasi Right Thinking Dengan Mentode Greedy," *J. Teknol. Inf.*, vol. 1, no. 2, pp. 174–184, 2017.

[5] F. I. Saputra and A. P. Dewi, "Penerapan Algoritma Minimax Untuk Game Tic Tac Toe," in *Seminar Nasional Teknologi Informasi dan Bisnis (SENATIB)*, 2017, no. November, pp. 3–10.

[6] A. M. Herli, I. K. Raharjana, and Purbandini, "Sistem Pencarian Hotel Berdasarkan Rute Perjalanan Terpendek Dengan Mempertimbangkan Daya Tarik Wisata Menggunakan Algoritma Greedy," *J. Inf. Syst. Eng. Bus. Intell.*, vol. 1, no. 1, pp. 9–16, 2015.

[7] A. Ambarwari and U. T. Indonesia, "Penerapan Algoritma Greedy Pada Permasalahan Knapsack Untuk Optimasi Pengangkutan Peti Kemas Penerapan Algoritma Greedy Pada Permasalahan," no. January, 2016.

[8] M. Salahuddin and A. S. Rosa, *Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: Modula, 2011.

[9] P. Kruchten, *The rational unified process: an introduction*. United States of America: Addison Wesley Professional, 2004.

[10] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition*. USA: Addison Wesley Professional, 2004.