Permainan Strategi Battleship Menggunakan Backtracking Dan Depth First Search

Teguh Saputra[*1], Yohanes Aryo Bismo Raharjo[2]
[1,2]Jurusan Teknik Informatika; STMIK Pontianak. Jl. Merdeka No.372 Pontianak, 0561-735555
e-mail: [*1] teguhsaputra225@gmail.com, [2]aryo.bismo@stmikpontianak.ac.id

Abstrak

Battleship adalah permainan papan yang dimainkan dengan menembak kotak kosong di papan permainan sampai pemain atau komputer (AI) menang. Tujuan dari pembuatan permainan ini adalah perancangan game perang berbasis komputer yang memiliki Artificial Intelligence (AI) sehingga game ini tidak membutuhkan dua orang untuk bermain. Selain tidak membutuhkan dua orang untuk bermain, pemain juga dapat menikmati game battleship ini dengan artificial intelligence (AI) yang dilengkapi dengan algoritma. Algoritma untuk membuat game battleship ini adalah dengan menggunakan dua algoritma backtracking dan DFS. Permainan ini memungkinkan pemain dapat menjalankan aplikasi dengan memanfaatkan algoritma, yang dilengkapi dengan pencarian kapal pemain dan dengan menggunakan algoritma backtracking pemain tidak perlu lagi menempatkan kapal mereka terlebih dahulu. Deskripsi dari game perang ini seperti permainan perang, tetapi tema yang diambil dalam pembuatan aplikasi ini adalah apakah pencarian untuk kapal pemain menggunakan algoritma backtracking dan DFS adalah algoritma yang sangat efisien dalam aplikasi game battleship ini. Aplikasi game battleship ini dibuat dengan menggunakan metode waterfall dengan pemodelan Unified Modeling Language (UML) dan menggunakan python sebagai bahasa pemograman. Hasil penelitian diperoleh adalah permainan battleship yang meiliki artificial intelligence (AI) yang dapat disesuaikan dengan kemampuan pemain, sehingga dapat disimpulkan permainan ini dapat digunakan sebagai media hiburan. Selain sebagai media hiburan, aplikasi ini disarankan untuk dikembangkan dengan fitur yang lebih menarik.

Kata Kunci: *Battleship, Artificial Intelligence, Backtracking, DFS*

Abstract

Battleship is a board game. It can be play by shooting empty boxes on the game board until player or computer (AI) win. The purpose of this research is to create a computer-based battleship game that has Artificial Intelligence (AI) so this game does not require two people to play. Besides not requiring two people to play, players can also enjoy this battleship game with artificial intelligence (AI) which is equipped with algorithms. The algorithm to make this battleship game is by using two algorithm backtracking and DFS. This application is to allow players run this application with an algorithm, that is equipped with a search for the player ship and the player can play the game without placing their ship first. The description of this battleship game is like a destructive game, but the theme taken in making this application is whether the search for a the player ship using the backtracking and DFS algorithm is a incredibly efficient algorithm in this battleship game application. This battleship game application is made by using the waterfall method with Unified Modeling Language (UML) and python programming. The results of this research is a battleship games that have artificial intelligence (AI) that can be adapted to the player's ability, so that it can be used as an entertainment. Other than as an entertainment this application should be developed with more attractive features.

Keywords*: Battleship, Artificial Intelligence, Backtracking, DFS*

1. PENDAHULUAN

Game komputer merupakan salah satu aplikasi software yang saat ini banyak dikembangkan. Dengan jenis yang bermacam-macam dan tampilan yang menarik, game komputer termasuk software yang diminati oleh berbagai kalangan. Selain karena tampilan dan aplikasinya relatif menarik, game komputer juga disinyalir dapat menjadi salah satu sarana refreshing yang cukup menyenangkan terutama bagi orang yang telah terbiasa menggunakan komputer.

Permainan-permainan komputer juga bermacam-macam. Salah satu kelebihannya adalah kita tidak harus mencari orang untuk menjadi lawan tanding jika ingin bermain karena permainan berbasis komputer ini sudah mendukung *single-player mode* dimana kita dapat bermain sendiri melawan komputer yang dirancang untuk dapat berlaku seperti pemain manusia atau yang sering dikenal dengan *artificial inteligince* (AI)[1].

Backtracking dan *depth first search* merupakan salah satu contoh dari Kecerdasan Buatan. Prinsip kerja dari algoritma ini adalah pemangkasan simpul- simpul yang tidak mengarah ke solusi. Sehingga, setiap simpul yang tidak memenuhi suatu fungsi pembatas, tidak akan diproses di algoritma ini. Adapun media yang cocok untuk penerapan algoritma *backtracking* dan *depth first search* adalah permainan *battleship*, beberapa alasan mengapa *battleship* digunakan sebagai media penerapan kecerdasan buatan antara lain fungsi pembangkit pada kasus ini adalah semua kemungkinan bagian kapal baik secara vertikal maupun horizontal serta bagian air. Fungsi pembatas yang digunakan adalah perbandingan jumlah bagian kapal pada suatu kolom dan baris dengan angka pembanding yang berada dipinggir kanan (untuk baris) dan bawah (untuk kolom). Selain itu, fungsi pembatas lainnya adalah pengecekan ada atau tidaknya bagian kapal disekitar lokasi penempatan bagian kapal. Apabila ada, maka lokasi tersebut tidak dapat ditempatkan bagian kapal, sehingga dapat ditempatkan bagian air[2].

Setiap permainan terdapat tingkat permainan yang dapat dipilih oleh pemain. Oleh karena itu penulis merancang permainan ini dengan tiga tingkat kesulitan yaitu *easy, normal, hell mode*. Setiap tingkat kesulitan dalam permianan menyesuaikan kebutuhan permainan seperti pemain yang masih pemula akan memilih tingkat permaianan *easy.* Dalam tingkat kesulitan easy, pemain diberi grid atau arena tempur yang lebih kecil sehingga pemain dapat lebih mudah mempelajari permainan *battleship.* Dalam tingkat kesulitan *normal* cocok untuk pemain yang sudah mengenal permainan *battleship.* Dalam tingkat kesulitan *hell mode* hanya cocok untuk pemain yang ingin tantangan, karena dalam tingkat kesulitan ini pemain benar-benar titangtang untuk dapat memenangkan permaianan dengan arena yang luas dan *turn* permainan yang dapat dibilang singkat.

Aplikasi permainan strategi *Battleship* dengan menggunakan VB.NET dapat menciptakan sebuah aplikasi sederhana yang mampu memberi hiburan dan membangun kemampuan logika pengurutan yang digabung dengan logika deduktif[3]. *Backtracking* dapat diterapkan dalam permainan *Math Maze*, dengan menerapkan algoritma *backtracking* dapat menghasilkan satu solusi untuk setiap *problem* yang dibangkitkan dan maze yang dihasilkan dengan algoritma backtracking akan menghasilkan maze yang tidak memiliki loop dan ruang terbuang[4]. Dengan adanya algoritma *bactracking* pada permainan *Caspa Banting* dapat mencari Solusi terbaik yang bernilai maksimum atau minimum dari sekumpulan solusi yang mungkin. Namun, adanya hal-hal tersebut sering dianggap oleh sebagian pengguna sebagai kelemahan yang membuat permainan tersebut menjadi kurang variatif dan terkesan membosankan[5].

Berdasarkan penjabaran permasalahan pada Pendahuluan diatas, maka Penulis tertarik untuk membuat jurnal yang berjudul "Permainan Strategi *Battleship* menggunakan Algoritma *Backtracking* dan *Depth First Search".*

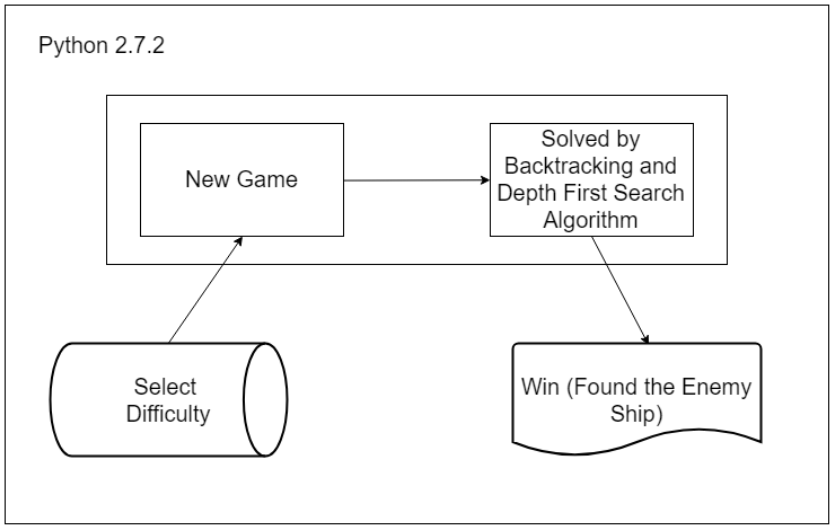
2. METODE PENELITIAN

Bentuk penelitian yang penulis gunakan dalam penelitian ini adalah studi literature, analisis, perancangan, pembangunan, dan pengujian[3]. Studi literatur merupakan strategi penelitian yang berusaha untuk memahami informasi pendukung yang ada dalam permainan komputer. Pada analisis akan diuraikan mengenai analisis permainan battleship itu sendiri. Analisis yang dilakukan yaitu berupa analisis aturan permainan dan analisis algoritma *backtracking* dan *depth first search* pada permainan *battleship*. Serta analisis kebutuhan sistem yang diperlukan adalah peletakan kapal-kapal perang yang akan diletakan dan papan permainan yang digunakan untuk bermaik permainan *battleship.* Dalam perancangan yang perlu di desain adalahproses permainan *battleship* berupa *use case* dari sistem tersebut, desain skenario permaian *battleship* dan desain tampilan permainan *battleship*. Pada tahapan pembangunan atau pengembangan ini digunakan tahapan pengembangan rekayasa perangkat lunak, dalam tahapan pembangunan dikembangkan dengan menggunakan python. Untuk menghasilkan suatu perangkat lunak yang bermutu, maka perlu dilakukan pengujian. Aplikasi permainan *Battleship* dengan menggunakan algoritma *backtracking* dan *depth first search* ini diuji dengan pengujian *Black-Box Testing* dan *White- Box Testing*.

3. HASIL DAN PEMBAHASAN

Dalam proses perancangan menggunakan *waterfall* terdapat beberapa fase perancangan, yaitu fase perencanaan yang didalam hal ini penulis harus mengetahui terlebih dahulu apa yang diinginkan dari pemilik termasuk masalah awal lalu kemudian kebutuhan yang diperlukan, lalu fase perancangan dalam fase ini penulis sudah mulai merancang hal apa saja yang seharusnya dibuat seperti arsitektur, *use case* dan *activity diagram*, selanjutnya fase pengkodean dalam fase ini penulis sudah menampilkan hasil dari rancangan seperti awal permainan, permainan sedang berlangsung dan tampilan menang/kalah dalam permainan, dan pada fase terakhir pengujian di fase ini semua hasil rancangan di uji apakah sudah lancar proses kerjanya ataupun masih ada yang kurang. Bahasa pemograman yang akan digunakan adalah python.

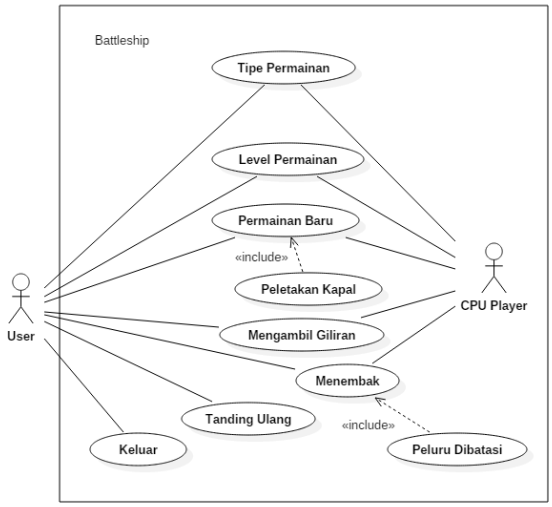
Perancangan arsitektur bertujuan mempresentasi proses bagaimana sistem perangkat lunak yang dibangun berjalan sesuai dengan keinginan penulis[6].



Gambar 1. Arsitektur permainan *battleship*

Pada gambar 1 terdapat arsitektur permainan *battleship*. Pada Arsitektur permainan *battleship* ini proses awalnya adalah di *player* (pemain) menentukan tingkat kesulitan permainan, setelah itu permintaan pemilik masuk ke permainan baru dan di letak kapal telah tersusun secara otomatis agar permainan siap dimainkan, setelah kapal pemain sudah ditemukan maka algoritma yan digunakan langsung berkerja. Disini peranan algoritma *backtracking* dan *depth first search* adalah mengisi kotak pada baris pertama dan kolom pertama dengan bagian kapal yang memenuhi fungsi pembatas. Misalnya, pada baris pertama serta kolom pertama, komponen yang dapat dibangkitkan adalah komponen yang memenuhi angka pinggir kanan baris tersebut yaitu satu dan pinggir bawah kolom tersebut yaitu tiga. Komponen yang memenuhi kriteria tersebut adalah bagian air, area yang sudah ditembak dan kapal.

Use case diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada di luar sistem atau actor[7]. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dari bagaimana sistem berinteraksi dengan dunia luar. Perancangan proses yang terjadi dalam permainan *battleship* ini dengan *Use Case Diagram* adalah sebagai berikut:



Gambar 2. *Use case diagram* permainan *battleship*

Pada gambar 2 *use case diagram* permainan *battleship* terdiri dari *user* dan CPU *player*. Actor *user* bertugas memulai dan menjalankan permainan. CPU *player* bertugas untuk merespon setiap tindakan *use* dalam permainan *battleship.*

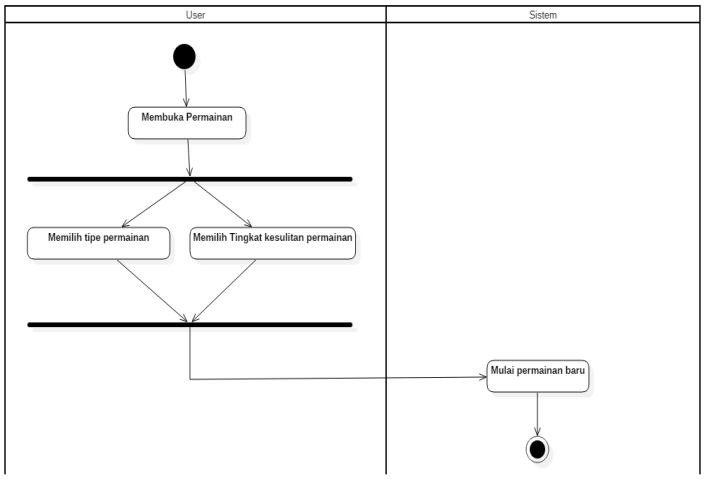
3.1 Activity Diagram

Activity Diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity Diagram* merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing)[8].

3.1.1 *Activity Diagram* Permainan Baru

Untuk memulai permainan *user* harus memilih tipe permain apabila pemain memilih *player vs player* permainan langsung mulai, dan yang terakhir saat memilih user memilih *player vs*

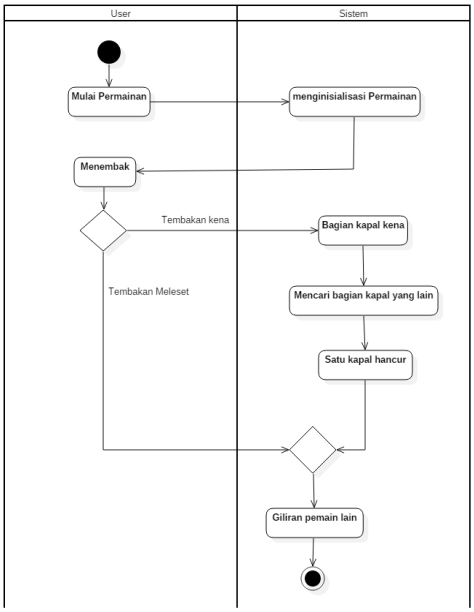
computer user harus memilih tingkat kesulitan permainan. Terdapat tiga tingkat kesulitan yang user pilih yaitu *easy, normal,* dan *hard.* Gambar 3 Merupakan diagram aktivitas yang dilakukan user untuk memulai permainan yang telah user setting terlebih dahulu



Gambar 3. Activity Diagram Memulai Permainan Baru

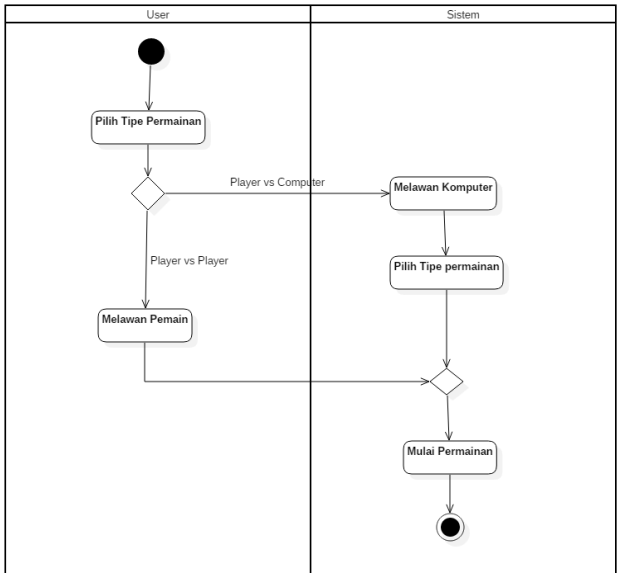
3.1.2 *Activity Diagram* Mengambil Giliran

User dan komputer hanya memliki satu langkah dalam setiap langkah. Apabila pemain salah atau menentukan koordinat yang kedua kali maka system akan mengingati pemain bahwa langkah yang ditentukan salah. Gambar 4 Merupakan diagram aktivitas yang dilakukan *user* atau komputer telah menggunakan langkahnya.



3.1.3 *Activity Diagram* Tipe Permainan

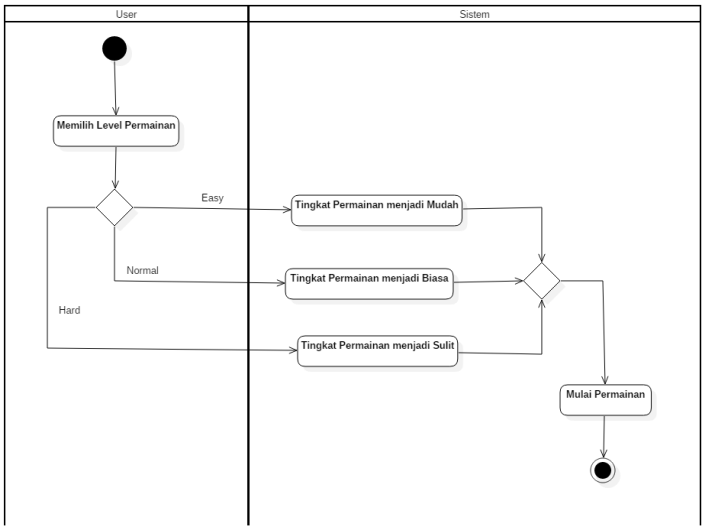
Merupakan diagram aktivitas yang dilakukan *user* untuk menentukan tipe permainan yaitu sendiri ataupun berdua dengan *user* lain. Apabila user memilih untuk melawan komputer maka *user* langsung dibawa ke level permainan, sedangkan apabila pelayar memilih untuk bermain berdua maka permainan langsung dimulai.



Gambar 4. *Activity Diagram* tipe permainan

3.1.4 *Activity Diagram* tingkat kesulitan

Apabila pemain memilih untuk bermain sendiri yaitu melawan computer maka *user* harus memilih tingkat kesulitan yang sesuai dengan kemampuan pemain easy untuk *user* yang baru pertama kali memainkan permaian *battleship, normal* untuk *user* yang sudah berpengalaman , dan *hard* untuk *user* yang sudah mahir dalam memainkan permainan *battleship* Gambar 5 Merupakan diagram aktivitas yang dilakukan *user* untuk menentukan tingkat kesulitan permainan yaitu *easy*, *normal*, ataupun *hard*.

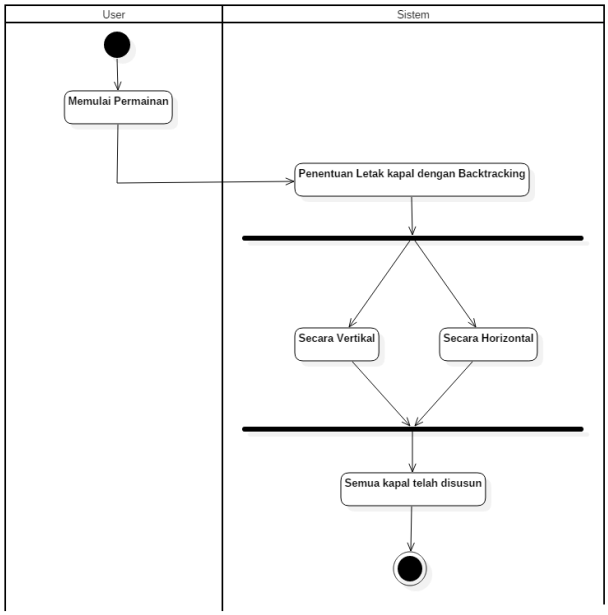


Gambar 5. *Activity Diagram* tingkat kesulitan

3.1.5 *Activity Diagram* Peletakan Kapal

Dalam permainan *battleship* ini user tidak perlu lagi menentukan letak kapalnya masing-masing Sehingga pemain tidak perlu lagi menentukan letak kapal yang sesuai. Kapal yang diletak tidak akan bertabrakan dan tersusun secara vertical dan horizontal. Gambar 6

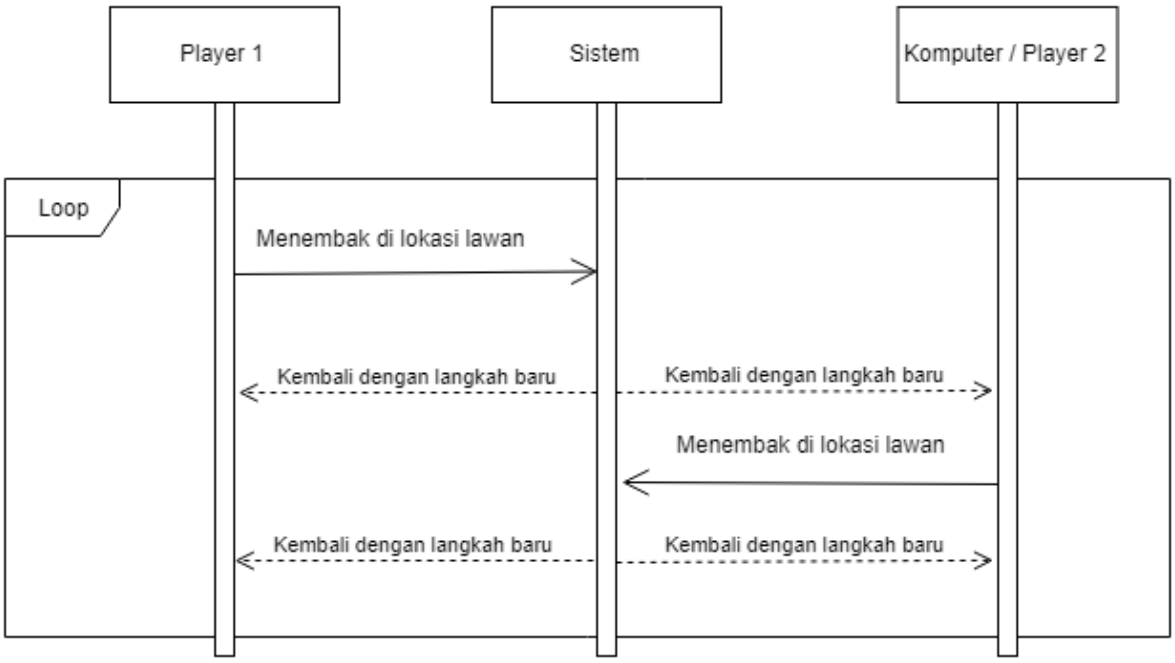
Merupakan diagram aktivitas yang komputer untuk menentukan kapal pemain maupun lawan dengan menggunaan algoritma *backtracking.*



Gambar 5. *Activity Diagram* Peletakan Kapal

3.2 Sequance Diagram

Sequance Diagram menunjukan interaksi yang terjadi antara objek di dalam dan di sekitar (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu[9]. Interaksi *user* dan aplikasi pembentukan permainan *battleship* ini dapat diilustrasikan sebagai berikut



Gambar 6. Sequance diagram permainan *battleship*

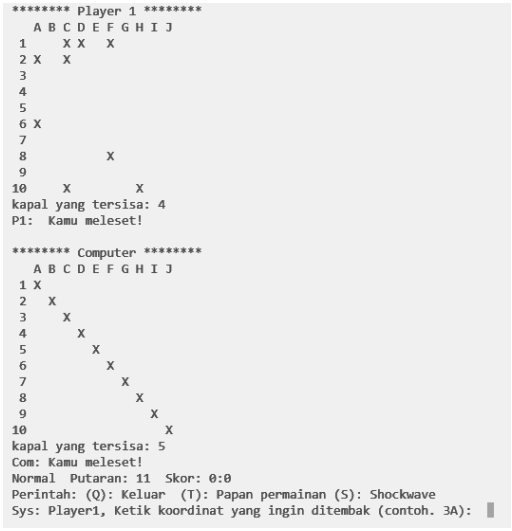
Pada gambar 6 terlihat ketika *user* menjalankan aplikasi ini, pertama *user* akan menentukan terlebih dahulu. Lalu *user* lain atau komputer harus membalas langkah yang telah pemain tentukan agar dapat memenangkan permainan. *Looping* akan terus terjadi sampai salah satu pemain telah berhasil menghancurkan semua kapal lawanya.

3.3 Hasil Penelitian

Dalam proses perancangan menggunakan *waterfall* terdapat beberapa fase yaitu fase analisis, fase perancangan, fase pembangunan, dan fase pengujian[10]. Berdasarkan hasil analisis dan perancangan sistem yang telah dilakukan, maka implementasi ke dalam bentuk program computer dapat dilakukan. Implementasi merupakan tahap menerjemahkan hasil perancangan perangkat lunak secara rinci kedalam bahasa pemograman. Implementasi dilakukan dengan menggunakan bahasa pemograman python, sehingga hanya dapat dijalankan pada komputer dengan sistem operasi *Windows*. Material permainan *battleship* yang berupa kapal *(ship)* dipresentasikan berdasarkan kedua papan medan pertempuran.

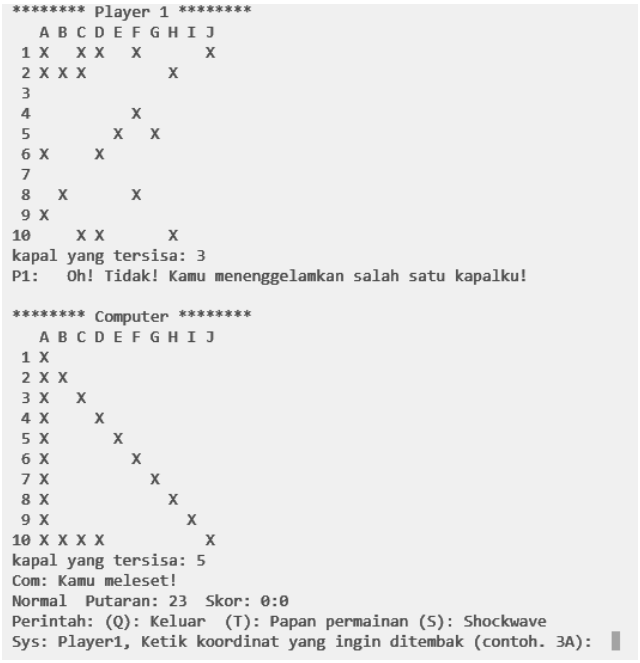
Gambar 1. Tampilan medan permainan

Pada gambar 1 yang merupakan tampilan papan permainan battleship. Pada form ini digunakan untuk menaruh kapal pemain yang sudah tersusun secara otomatis. Fungsi yang digunakan adalah klik kanan pada mouse digunakan untuk menembak kapal musuh yang terletak di medan musuh.



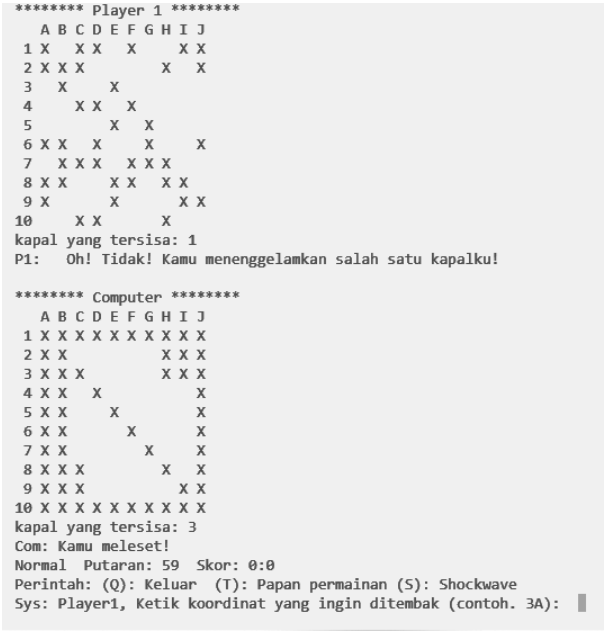
Gambar 2. Permainan sedang berlangsung

Pada tampilan gambar 2, komputer atau *artificial inteligince* (AI) berhasil menembakan salah satu badan kapal milik pemain.



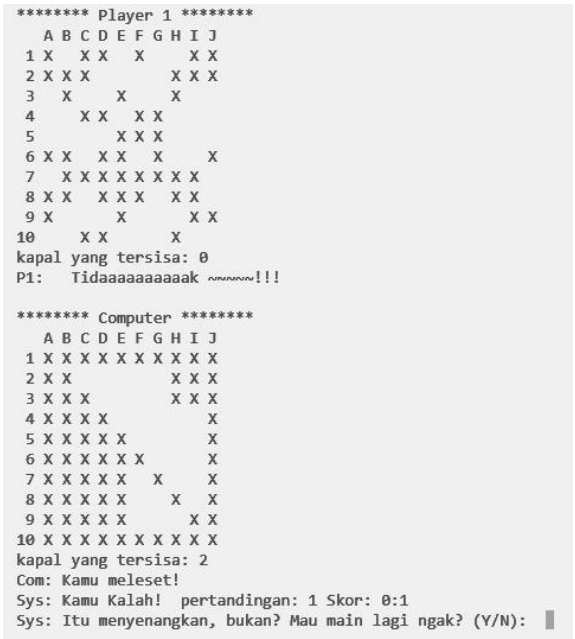
Gambar 3. Komputer mulai mencari bagian kapal yang lain

Pada gambar 3, *backtracking* dan *depth first search* mulai jalan secara otomatis, karena ada bagian badan kapal pemain yang tertembak. Logika pencarian solusi-nya dengan menggunakan algoritma *backtracking* dan *depth first search.* Dengan menggunakan logika x-1. Setelah logika x-1 di jalankan, maka logika x-2 berjalan secara otomatis, pencarian solusi dilakukan pada arah kiri. Setelah logika x-2 di jalankan, maka logika x-3 berjalan secara otomatis, pencarian solusi dilakukan pada arah kiri. Pencarian solusi dijalankan pada logika x-4. Setelah pada posisi ini, maka pencarian solusi akan dijalankan pada posisi ke kanan.



Gambar 4. Komputer mulai mencari bagian kapal yang lain

Pada gambar 4 merupakan lanjutan dari gambar 3, setelah pencarian solusi pada sisi kiri telah selesai, maka pencarian solusi akan dilanjutkan pada logika x+1, atau pada sisi kanan. Pencarian solusi dilanjutkan pada koordinat x+2. Pencarian solusi hanya berhenti sampai x+2, karena telah mencapai sisi papan permainan. Jika tidak terkena sisi papan, maka pencarian akan dilanjutkan sampai koordinat x+4. Pencarian solusi dilanjutkan kearah bawah dengan koordinat y-1, dilanjutkan pada logika y-2. Pencarian solusi dilanjutkan pada logika y-3. Walaupun ada terkena tembakan badan kapal yang lain, pencarian solusi tetap dijalankan dengan urutan logika, sehingga pencarian solusi tidak menjadi acak.



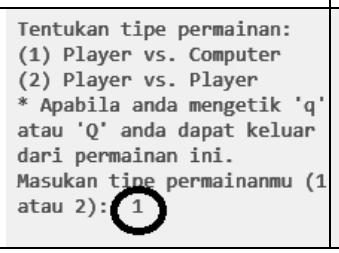
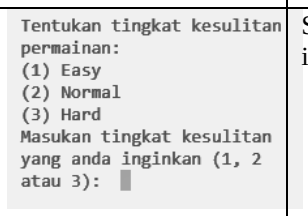
Gambar 5. Komputer menang melawan pemain

Setelah antara salah satu sisi menang, akan tampil seperti pada gambar 5, baik player yang menang maupun komputer yang menang dalam permainan *battleship* maka sistem akan meminta tanding ulang, apabila pemain menolak maka permainan berakhir.

3.4 Pengujian

Pengujian software sangat diperlukan untuk memastikan software/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan[11]. Pengembang atau penguji software harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Untuk menghasilkan suatu perangkat lunak yang baik, maka perlu dilakukan pengujian. Aplikasi permainan *Battleship* dengan menggunakan algoritma *backtracking* dan *depth first search* ini diuji dengan pengujian *Black-Box Testing.* Bagian-bagian yang akan diuji adalah *form* menu utama dan *form* permainan. Dari hasil pengujian yang dilakukan dari item pengujian yang ada di *form* menu utama dan *form* permainan didapatkan hasil sesuai yang diharapkan dengan yang dimplementasikan.

Tabel 3.1 Tabel Pengujian Tipe Permainan

No	Skenario Pengujian	Use Case	Aktor	Path		
1	Memilih Tipe permainan	Tipe Permainan	Player	Home/Battleship		
	Deskripsi : Pengujian ini dilakukan untuk menguji Boolean 1 atau 2					
	Aktifitas	Interface Eksekusi	Tahap Eksekusi	Hasil yang diharapkan	Interface Setelah dieksekusi	Status
1a	Memilih Tipe permainan		Ketik "1"	Permainan akan masuk ke tingkat kesulitan permainan		Sesuai

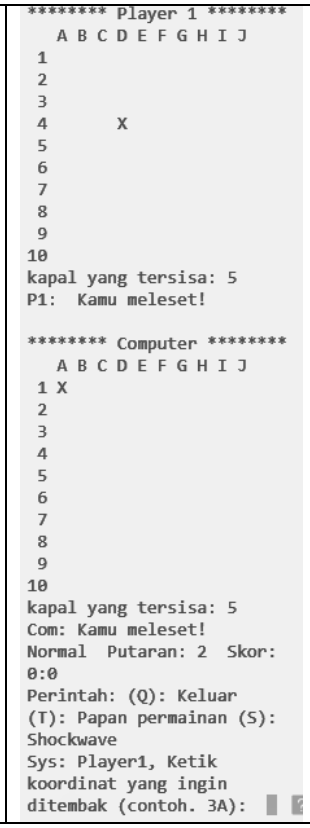
Tabel 3.2 Tabel Pengujian Level Permainan

No	Skenario Pengujian	Use Case	Aktor	Path		
2	Memilih tingkat kesulitan permainan	Level Permainan	Player	Home/Battleship		
	Deskripsi : Pengujian ini dilakukan untuk menguji integer 1 , 2 atau 3					
	Aktifitas	Interface Eksekusi	Tahap Eksekusi	Hasil yang diharapkan	Interface Setelah dieksekusi	Status

2a	Memilih Tingkat kesulitan permainan	Tentukan tingkat kesulitan permainan: (1) Easy (2) Normal (3) Hard Masukan tingkat kesulitan yang anda inginkan (1, 2 atau 3): 2	Ketik "2"	Permainan akan masuk ke permainan baru dengan tingkat kesulitan normal	******** Player 1 ******** A B C D E F G H I J 1 2 3 4 5 6 7 8 9 10 kapal yang tersisa: 5 P1: Tembak Komandan! ******** Computer ******** A B C D E F G H I J 1 2 3 4 5 6 7 8 9 10 kapal yang tersisa: 5 Com: Coba saja kalau berani!! Normal Putaran: 1 Skor: 0:0 Perintah: (Q): Keluar (T): Papan permainan (S): Shockwave Sys: Player1, Ketik koordinat yang ingin ditembak (contoh. 3A):	Sesuai

Tabel 3.3 Tabel Pengujian Menembak

No	Skenario Pengujian	Use Case	Aktor	Path		
3	Memilih koordinat letak kapal musuh	Menembak	Player	Home/Battleship		
	Deskripsi : Pengujian ini dilakukan untuk menguji String 1A – 10J					
	Aktifitas	Interface Eksekusi	Tahap Eksekusi	Hasil yang diharapkan	Interface Setelah dieksekusi	Status

3a	Memilih koordinat letak kapal musuh		Ketik "1A"	Koordinat medan musuh AI akan tertembak		Sesuai

4. KESIMPULAN

Setelah menyelesaikan perancangan perangkat lunak permainan strategi *Battle Ship* dengan menggunakan algoritma *backtracking* dan *depth first search (DFS)*, penulis menarik kesimpulan sebagai berikut:

a. Aplikasi ini sangat bermanfaat bagi user karena dapat mengasah otak user untuk menentukan strategi yang tepat dalam menentukan koordinasi tembakan yang tepat dan tidak sia-sia.
b. Algoritma *backtracking* dan *depth first search* merupakan algoritma yang sangat bagus dan cocok untuk pengambilan keputusan oleh kecerdasan buatan.
c. Algoritma *backtracking* dan *depth first search* dapat diterapkan di dalam aplikasi permainan *Battleship*.
d. Semakin tinggi tingkat kesulitan permainan, semakin sulit juga bagi pemain untuk memenangkan permainan.
e. Perangkat lunak dapat dimainkan sendiri maupun dua orang dengan satu komputer.
f. Dengan menggunakan algoritma *backtracking* dan *depth first search untuk kecerdasan buatan dalam permainan* battleship, user (manusia) akan kesulitan untuk menang melawan kecerdasan buatan tersebut.

5. SARAN

Saran-saran yang dapat Penulis berikan dalam pengembangan program aplikasi game memberikan beberapa saran sebagai berikut:

a. Perangkat lunak dapat dikembangkan untuk *user* yang lebih banyak (lebih dari 2 orang).
b. Untuk menambahkan fitur-fitur yang lebih interaktif untuk pemain sehingga pemain tidak merasa kesulitan untuk memainkan permainan *battleship* tersebut.
c. Memperindah tampilan grafik agar pemain tidak merasa bosan dengan tampilan permainan.

d. Sebaiknya aspek multimedia perlu ditambahkan, misalnya grafik ataupun suara untuk mendukung tampilan aplikasi.
e. Menambahkan algoritma selain *backtracking* dan *depth first search* agar kemampuan kecerdasan buatanya lebih bertambah.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada dosen pembimbing, keluarga, dan teman-teman yang telah memberikan dukungan dalam menyelesaikan jurnal ini.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2009) *Diktat Kuliah IF 2251 Strategi Algoritmik*. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.
[2] Sumarsono, Abraham Ranardo. (2009). Penerapan Algortima Runut Balik dalam Pencarian Solusi Teka Teki Battleship. Institut Teknologi Bandung.
[3] Oktavianto, H. (2016). Perancangan Aplikasi Permainan Strategi Battle Ship Menggunakan VB. NET. *INTEKSIS*, *1*(1).
[4] Teneng, T., Purwadi, J., & Kurniawan, E. (2011). Penerapan Algoritma Backtracking Pada Permainan Math Maze. *Jurnal Informatika*, *6*(2).
[5] Fahrudin, B. (2016). Penerapan Algoritma Backtracking Pada Permainan Capsa Banting. *JURIKOM (Jurnal Riset Komputer)*, *3*(6).
[6] Roger.S, 2011. *Software Engineering : A Practioner's Approach*. 5th .McGrawHill.
[7] Haviluddin. 2011. *Memahami Penggunaan UML*. Unified Modelling Language. Jurnal Informartika Mulawarman. Vol 6 No.1 Februari 2011. Samarinda.
[8] Nugroho, Adi, 2010, Rekayasa Perangkat Lunak Menggunakan UML dan Java, Edisi 1,Andi Publisher, 2010.
[9] Tripathy, A., & Mitra, A. (2013). Test case generation using activity diagram and sequence diagram. In *Proceedings of International Conference on Advances in Computing* (pp. 121-129). Springer, New Delhi.
[10] Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26-30.
[11] Rouf, A. (2012). Pengujian perangkat lunak dengan menggunakan metode white box dan black box. *Himsyatech*, *8*(1).