IMPLEMENTASI ALGORITMA BACKTRACKING PADA APLIKASI PERMAINAN TRADISIONAL DAM-DAMAN BERBASIS JAVA DESKTOP

Funny Farady Coastera[1], Ernawati[2], Apni Nomansa[3]

[1,2,3]Program Studi Teknik Infomatika, Fakultas Teknik, Universitas Bengkulu.
Jl. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[1]ffaradyc@gmail.com,
[2]w_ier_na@yahoo.com,
[3]apninomansa@gmail.com

Penelitian ini bertujuan untuk membangun suatu aplikasi permainan tradisional dam-daman dengan mengimplementasikan algoritma *backtracking* berbasis *java desktop*. Algoritma *backtracking* adalah algoritma yang berbasis pada *Depth First Search* (DFS) untuk mencari solusi persoalan secara lebih mangkus. Dengan algoritma *backtracking*, kita tidak perlu memeriksa semua kemungkinan solusi yang ada, hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan, akibatnya waktu pencarian dapat dihemat. Aplikasi permainan tradisional *dam-daman* ini juga mendukung *multiplayer* dengan menggunakan pemrograman *socket.* Metode pengembangan sistem yang digunakan untuk membangun aplikasi ini adalah model *waterfall*. Sedangkan pada tahap analisa dan perancangan sistem dilakukan dengan menggunakan *Unified Modelling Language (UML)*. Hasil akhir dari penelitian ini adalah terciptanya sebuah aplikasi permainan tradisional dam-daman yang mendukung *single player* dan *multiplayer*.

Kata Kunci: Kecerdasan Buatan, Permainan Tradisional, Algoritma *Backtracking*, Pemrograman *socket*.

Abstract: **The aim of this research was to establish a traditional game aplication, dam-daman by implemented backtracking algorithm with java desktop based. Backtracking algorithm is the algorithm which based on the Depth First Search (DFS) to solve problem and give solution efectively. Through the backtracking algorithm, we do not need to check all possibility of providing solution, but only on the searching which aim at the oftenly considered solution, therefore the searching time will be efficient. The Dam-daman traditional game aplication also supports the multiplayer by using socket programming. The developing system method used for creating this aplication was the waterfall model. While the analysing phase and planning system were done by using Unified Modelling Language (UML). The final result of this research was a Dam-daman traditional game aplication which supports the single player and multiplayer.**

Keywords: Artificial Intelligence, Traditional Game, Backtracking Algorithm, Socket programming.

I. PENDAHULUAN

Permainan merupakan suatu bentuk hiburan yang sering digunakan manusia untuk menghibur diri sendiri atau kelompok dari suatu rutinitas sehari-hari. Permainan juga dapat digunakan sebagai alat hubungan atau kenyamanan sosial yang bersifat sangat menyenangkan, selain itu permainan juga bermanfaat bagi manusia, baik itu jasmani maupun rohani.

Permainan *Dam-Daman* merupakan salah satu permainan tradisional, permainan ini berasal dari Indonesia. Dua hal utama yang harus dipahami dalam permainan dam-daman adalah bidak dan papan main. Bidak biasa dibuat dari batu maupun kertas. Sedangkan papan main, bisa kertas atau tanah sekalipun yang digambar dengan motif tertentu. Alur permainan menghabiskan batu atau bidak dari lawan atau sampai lawan menyerah. Permainan ini juga bertujuan untuk melatih daya ingat dan kepekaan indera serta melatih ketelitian dan kecermatan dalam menentukan keputusan, karena dalam permainan membutuhkan strategi untuk menjadi pemenang. Jadi permainan *dam daman* ini sangat menarik untuk dimainkan.

Dalam permainan biasanya dikenal dengan istilah *single player* dan *multiplayer*. *Multiplayer* adalah dimana kita bermain melawan orang, sedangkan *single player* adalah dimana kita tidak harus mencari orang untuk menjadi lawan tanding jika ingin bermain atau dengan kata lain kita lawan agen cerdas. Agen cerdas merupakan komputer yang dirancang untuk dapat berpikir seperti pemain manusia dengan analogi *Artificial Inteligince* (AI). Untuk membuat AI dalam sebuah permainan, diperlukan suatu algoritma yang dapat membuat AI ini mampu mengambil keputusan yang terbaik agar dapat mengalahkan pemain atau setidaknya menghalau pemain menang.

Menurut Suyoto [1] AI merujuk pada satu bidang komputer yang berupaya melakukan tugas seperti manusia, dalam keadaan ini, komputer tersebut boleh dikatakan cerdas. Untuk melakukan hal tersebut digunakan suatu algoritma pencarian, seperti alnya algoritma *backtracking* (runut balik). Menurut Bakri [2] Algoritma ini merupakan perbaikan dari algoritma *bruteforce*, yang secara sistematis mencari solusi persoalan diantara semua kemungkinan solusi yang ada. Dengan metode ini, tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat.

Untuk fitur *multiplayer*,digunakan pemrograman *socket*, sehinga permainan bisa melibatkan interaksi antara manusia dan manusia. *Socket programming* merupakan pemrograman yang bertujuan agar satu program bisa berinteraksi dengan program lainnya dalam satu jaringan, biasa disebut juga dengan pemrograman jaringan. *Socket* adalah suatu abtraksi yang mana aplikasi dapat mengirim dan menerima data seperti sama halnya dengan membuka suatu *file* untuk dibaca dan ditulis pada tempat penyimpanan *file* [3]. *Socket* memungkinkan untuk masuk kedalam jaringan dan berkomunikasi dengan aplikasi lain yang juga masuk kedalam jaringan yang sama.

Melihat latar belakang diatas, penulis tertarik untuk membuat aplikasi permainan tradisional *dama-daman*, aplikasi permainan *dam-daman* ini merupakan permainan yang berbasis *java desktop* dan terdapat pengimplementasian algoritma *backtracking* (runut balik), sehingga memudahkan bagi masyarakat untuk mempelajari dan memainkan permainan ini. Aplikasi ini nantinya mendukung fitur *single player* dan *multiplayer* dengan bantuan pemrograman *socket.*

II. LANDASAN TEORI

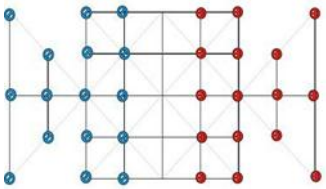
A. Permainan (Game)

Permainan merupakan salah satu cara untuk menghibur diri, karena permainan merupakan sebuah aktivitas rekreasi dengan tujuan bersenang-senang, mengisi waktu luang, atau berolahraga santai, permainan biasanya dilakukan sendiri atau bersama-sama. Menurut (Pradiyar dkk, 2007) [4] *game* adalah sebuah permainan komputer interaktif yang dikendalikan oeh mikroprosesor. Berikut ini adalah elemen dasar yang dimiliki setiap permainan digital (game) diantaranya [5]: grafis, *interface*, aktivitas pemain dan sebuah algoritma. Beberapa *game* (permainan) memiliki suatu aturan yang unik yang membedakan dengan *game-game* lainnya, aturan inilah basis pengelompokan beberapa *game* menjadi berbagai *genre*. Ada beberapa *game* berdasarkan jenis atau *genre* antara lain: *Maze Game*, *Board Game*, *Card Game*, *Trading Card Game*, *Quiz Game*, *Puzzle Game*, *Shooting Game*, *Shoot Them up*, *Adventure Game*, *Side Scroller Game*, *Fighting Game*, *Sport*

Game, *Racing Game*, *Simulation Game*, *Real Time Strategy* (RTS) *Game* dan *Role Playing Game* (RPG).

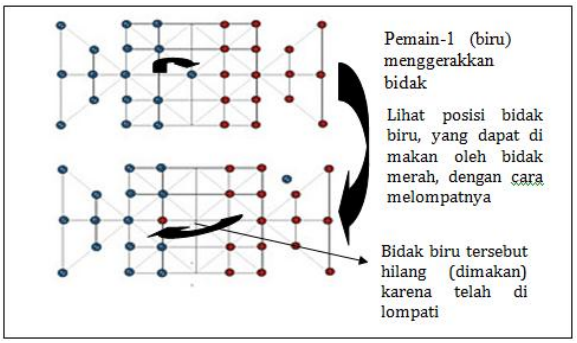
Permainan tradisional dapat diartikan sebagai satu kegiatan yang menyenangkan yang dilakukan menurut tradisi, sehingga menimbulkan rasa puas pada pelakunya [6]. Permainan *dam-daman* merupakan permainan tradisional yang berasal dari Indonesia. Dikutip dari http://www.urangkampoeng.com [7] berikut ini adalah peraturan dan petunjuk dalam permainan dam-daman:

1. Menggunakan papan permainan, berikut gambar dari papan permainan *dam-daman:*



Gambar 1 Papan Permainan *Dam-daman* [7]

2. Dalam sebuah papan permainan dam-daman, terdapat titik-titik yang digunakan pijakan pada masing-masing bidak. Diantara tiap-tiap titik terdapat garis horisontal, vertikal dan diagonal yang dijadikan jalur langkah bagi bidak.
3. Menggunakan bidak permainan. Pemain 1 dan 2 memiliki bidak yang berbeda dengan jumlah 16 bidak untuk masing-masing pemain.
4. Cara permaianannya adalah dengan cara mengerakan bidaknya secara bergantian seperti bermain Catur, ketika pemain A sudah mengerakan bidak maka giliran B, dan seterusnya.
5. Bidak hanya bisa dijalankan dengan satu langkah dengan bergerak maju atau ke samping mengikuti jalur (garis) papan permainan. Bidak tidak bisa bergerak mundur.
6. Bagi bidak yang bisa sampai pada kotak segitiga lawan pada baris terakhir, maka bidak itu menjadi bidak raja yang bebas berjalan (maju ataupun mundur), asalkan masih pada satu garis lurus.
7. Cara mengambil atau memakan bidak lawan dengan cara melompati bidak lawan dengan hanya jarak satu titik, dan berpeluang melompati kemungkinan yang ada dalam satu lompatan.



Gambar 2 Cara permainan Dam-daman

8. Terdapat sebuah aturan dimana sebuah bidak wajib memakan bidak lawannyaa jika ada peluang, seperti Gambar diatas, jadi bidak merah harus menggerakkan bidak yang ditengah untuk memakan bidak biru tidak bisa menggerakkan bidak lainnya.
9. Pemain yang kalah adalah pemain yang bidaknya sudah habis.

B. Kecerdasan Buatan

Kecerdasan buatan merupakan salah satu bidang ilmu komputer yang didefinisikan sebagai kecerdasan yang dibuat untuk suatu sistem dengan menggunakan algoritma- algoritma tertentu sehingga sistem tersebut seolah-olah dapat berpikir seperti manusia. Jadi tujuan ditanamkan kecerdasan buatan dalam suatu mesin atau aplikasi suapaya suatu sistem seolah-olah dapat berpikir seperti manusian, suatu system atau mesin tidaklah pintar jika tidak ditanamkan suatu sistem kecerdasan.

Kecerdasan buatan dalam permainan kompuer (*game*) telah lama dikenal. *Game* catur merupakan contoh penelitian AI paling awal, AI dalam *game computer* sebenarnya tidaklah benar-benar cerdas. *Game* pada dasarnya merupakan set intruksi kompeks yang dimasukkan ke program yang sudah memprediksi intelegensi pemakainya. Salin itu, isi program tersebu memuat prosedur pencarian. Menurut Suyoto [1], algoritma pencarian adalah sebuah teknik untuk pencarian sesuatu. Di dalam pencarian ada dua kemungkinan hasil yang akan kita dapatkan yaitu kita menemukan yang dicari atau kita tidak menemukan yang dicari. Oleh karena itu, kita tidak

dapat secara langsung mengatakan bahwa algoritma pencarian mewakili AI.

C. Algoritma Backtracking

Teknik runut balik (*backtracking*) merupakan salah satu teknik dalam penyelesaian masalah secara umum (*General Problem Solving*). Adapun dasar dari teknik ini adalah suatu teknik pencarian (Teknik *Searching*). Teknik pencarian ini digunakan dalam rangka mendapatkan himpunan penyelesaian yang mungkin. Dari himpunan penyelesaian yang mungkin ini akan diperoleh solusi optimal atau memuaskan [8].

Runut balik (*backtracking*) adalah algoritma yang berbasis pada *Depth First Search* (DFS) untuk mencari solusi persoalan secara lebih mangkus. Runut balik, yang merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan metode runut balik, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat. Saat ini algoritma runut balik banyak diterapkan untuk program permainan seperti permainan tic-tac-toe, menemukan jalan keluar dalam sebuah labirin, catur, *crossword puzzle*, *sudoku* dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*) [9].

Dalam penerapan algoritma *Backtracking* ada beberapa properti yang perlu dipertimbangkan, yaitu properti solusi persoalan, properti komponen vektor solusi dan properti kriteria pembatas. Berikut adalah penjabaran properti-properti tersebut:

1. Solusi persoalan.
   Solusi dinyatakan sebagai vektor dengan *n-tuple*: $X = (x_1, x_2, \ldots, x_n)$, $x_i \in S_i$ Mungkin saja $S_1 = S_2 = \ldots = S_n$.
   Contoh: $S_i = \{0, 1\}$, $x_i = 0$ atau 1
2. Fungsi pembangkit (nilai x_k)
   Dinyatakan sebagai predikat: $T(k)$
   $T(k)$ membangkitkan nilai untuk x_k, yang merupakan komponen vektor solusi.
3. Fungsi pembatas
   Dinyatakan sebagai predikat $B(x_1, x_2, \ldots, x_k)$, B bernilai *true* jika $(x_1, x_2, \ldots, x_k)$ mengarah ke solusi. Jika *true*, maka pembangkitan nilai untuk x_{k+1} dilanjutkan, tetapi jika *false*, maka $(x_1, x_2, \ldots, x_k)$ dibuang.

Menurut NST [8] dalam pencarian solusi algoritma *backtracking*, ada beberapa langkah yang dilakukan untuk mencapai solusi tersebut, yaitu sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan pencarian mendalam (DFS). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*Expand-node*).
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (*bounding function*). Simpul yang sudah mati tidak akan pernah diperluas lagi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut balik ke simpul hidup terdekat (simpul orang tua). Selanjutnya simpul ini menjadi simpul-E yang baru.

Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut balik.

D. Pemrograman Socket

Pengertian *socket* adalah *interface* pada jaringan yang menjadi titik komunikasi antarmesin pada Internet *Protocol*, dan tentunya tanpa komunikasi ini, tidak akan ada pertukaran data dan informasi jaringan. *Socket* terdiri dari elemen-elemen utama seperti protokol, *local* IP, *local port*, *remote* IP, dan *remote port* [9].

Pemrograman *socket* adalah cara untuk menggunakan komponen/API (*Application Programming Interface*) *socket* untuk membuat sebuah aplikasi. Aplikasi *socket* umumnya terdiri dari dua kategori berdasarkan pengiriman datanya, yaitu datagram *socket* (menggunakan UDP) dan stream *socket* (menggunakan TCP). Terdapat perlakuan yang berbeda antara UDP dan TCP, walaupun sama-sama berfungsi sebagai protokol pertukaran data. UDP tidak memerlukan proses koneksi terlebih dahulu untuk dapat mengirimkan data, paket-paket data yang dikirimkan UDP bisa jadi melalui rute yang berbeda-beda, sehingga hasil yang diterima bisa jadi tidak berurutan.

Penggunaan *socket programming* memungkinkan adanya komunikasi antara *client* dan *server*. Salah satu contoh sederhana penggunaan *socket* programming adalah pembuatan program untuk *chatting*. Program tersebut sebenarnya merupakan bentuk aplikasi berupa komunikasi antara *client* dan *server*. Ketika seorang *user* (*client*) melakukan koneksi ke *chat server*, program akan membuka koneksi ke *port* yang diberikan, sehingga *server* perlu membuka *socket* pada *port* tersebut dan "mendengarkan" koneksi yang datang. *Socket* sendiri merupakan gabungan antara *host-address* dan port *address*. Dalam hal ini *socket* digunakan untuk komunikasi antara *client* dan *server*.

III. METODOLOGI

A. *Jenis Penelitian*

Jenis penelitian yang digunakan dalam penelitian ini adalah penelitian terapan.

B. *Teknik Pengumpulan Data*

Teknik pengumpulan data pada penelitian terapan ini menggunakan teknik studi pustaka (*Library research*).

C. *Jenis dan Sumber Data*

Jenis data yang digunakan dalam penelitian ini berasal dari data sekunder. Data *Sekunder* adalah data yang diperoleh atau dikumpulkan peneliti dari berbagai sumber yang telah ada (peneliti sebagai tangan kedua).

D. *Metode Pengembangan Sistem*

Metode perancangan sistem dalam penelitian aplikasi permainan dam-daman ini menggunakan model *waterfall.*

E. *Metode Pengujian Sistem*

Metode pengujian sistem dalam penelitian ini mengunakan *black-box testing* dan *white-box testing*.

F. *Pengujian Algoritma*

Pengujian algoritma *backtracking* dilakukan dengan membandingkan perhitungan manual atau pencarian manual pergerakan dari agen dengan pergerakan agen dalam sistem yang ditanamankan algoritma *backtarcking.*

IV. ANALISIS DAN PERANCANGAN

A. *Analisis Masalah*

Dari permasalahan yang dapat dilihat pada permainan dam-daman secara tradisional ada beberapaa kelemahan, diantaranya sebagai berikut :

1) Permainan masih dimainkan secara tradisional, yaitu dimainkan dengan papan permainan yang dibuat dikertas atau ditanah dan menggunakan batu atau kertas sebagai bidaknya.
2) Harus ada pemain penantang (pemain ke-2).

Untuk menyelesaikan permasalahan diatas, sehingga dilakukan sebuah penelitian yang nantinya dapat mengatasi permasalahan-permasalahan tersebut. Beberapa keunggulan yang didapat dari penelitian ini adalah:

1. Papan permainan dan bidak permainan dibuat lebih menarik dalam menampilkan *interface* aplikasi.
2. Bisa dimainkan tanpa harus ada pemain ke-2 (*single player*) dengan melawan AI.
3. Bisa juga dimainkan berdua (*multiplayer*) dengan bantuan pemrograman *socket* (jaringan).
4. Terdapat Petunjuk Permainan, bagi yang belum mengenai permainan ini.

B. *Analisis Fungsional*

Analisis fungsional merupakan paparan mengenai fitur-fitur yang akan dimasukkan ke dalam sistem yang dibuat. Adapun fitur-fitur aplikasi *game* ini adalah:

1) *Game* mampu mendukung s*ingle player* dan m*ultiplayer*
2) *Game single player* dapat dimainkan melawan komputer atau agen *game*
3) *Game multiplayer* dapat dimainkan dengan menggunakan bantuan jaringan lokal menggunakan pemrograman *socket*.
4) Fitur tambahan atau pelengkap pada aplikasi *game* ini adalah menampilkan informasi *help* dan *about*.

C. *Analisis Permainan Dam-Daman*

1) Analisis Papan Permainan dan Bidak: Papan permainan dam-daman ini dideklarasikan dengan *array* dua dimensi, dengan panjang kolom 9 dan panjang baris 5.

Gambar 3. Analisis Papan Permainan Dam-daman

Untuk bidak permainan terbagi menjadi dua warnanya yaitu bidak biru dan bidak merah, disini bidak biru adalah milik dari agen cerdas sedangkan bidak merah milik pemain (*user*). Masing-masing bidak dibagi menjadi bidak normal dan bidak raja. Bidak normal (*bluenormal* dan *rednormal*) adalah bidak yang diberikan pada awal permainan yang tidak bisa bergerak mundur hanya bisa bergerak maju serong atau lurus atau ke samping kiri atau kanan. Bidak raja (*redking* dan *blueking*) adalah bidak normal yang berubah menjadi bidak raja karena sudah mencapai garis akhir pertahanan lawan, bidak ini bisa berjalan ke segala arah. Garis akhir pertahan lawan adalah jika bidak tersebut berada pada i = 0 untuk bidak biru dan i = 8 untuk bidak merah.

2) *Analisis Status Berjalan Dam-Daman (Move) :* Dalam menjalankan bidak, terdapat ketentuan-ketentuan seperti berjalan harus sesuai dengan *grid*, berjalan hanya dengan satu langkah dan berjalan tidak bisa mundur kecuali bidak raja. Sehingga dibutuhkan suatu status kondisi (*Boolean*) yang mengecek kondisi berjalan bidak itu benar atau salah. Masing-masing bidak akan dianalisis berjalannya sesuai dengan ketentuan yang ada.

3) Analisis Status Loncat Dam-Daman: Meloncat atau menangkap bidak adalah cara bagaimana memakan bidak lawan untuk menghabisi bidak lawan agar memenangkan permainan. Dalam meloncati bidak lawan, ada beberapa syarat yang wajib terpenuhi, diantaranya : (1) Harus begeser dua titik (titik pertama bidak yang diloncati, titik yang kedua titik setelah meloncat) dan ada kemungkinan loncat lebih dari satu dalam satu giliran. (2) Titik setelah meloncat harus kosong tidak ada bidak yang mengisinya.

4) *Analisis Status Permainan:* Status permainan disini bertujuan untuk mengetahui apakah permainan sudah berakhir atau belum serta untuk mengetahui pemenang dari permainan ketika permainan sudah berakhir.

5) *Analisis Algoritma Backtracking dalam Permainan Dam-Daman*: Pada algoritma *Backtracking*, ada beberapa properti yang perlu dipertimbangkan dalam pencarian solusi yaitu properti solusi persoalan, properti fungsi pembangkit dan properti kriteria pembatas.

1. Properti solusi persoalan, solusi persoalan dari permainan ini berupa *array-array* yang terdapat pada papan permaina, yaitu [0][0], [0][2], [0][4], [1][1], [1][2], [1][3], [2][0], [2][1], [2][2], [2][3], [2][4], [3][0], [3][1], [3][2], [3][3], [3][4], [4][0], [4][1], [4][2], [4][3], [4][4], [5][0], [5][1], [5][2], [5][3], [5][4], [6][0], [6][1], [6][2], [6][3], [6][4], [7][1], [7][2], [7][3], [8][0], [8][2] dan [8][4].
2. Properti fungsi pembangkit, properti fungsi pembangkit dalam permainan dam-daman ini berupa *Generate Move* yaitu menghidupkan kemungkinan pergerakan yang bisa dilakukan oleh bidak sesuai dengan peraturan permainan. *Generate Move* menghasilkan *Move list* yang akan memproses kemungkinan pergerakan dengan kedalaman 4.
3. Properti fungsi pembatas, dalam analisis permainan dam-daman ini terdapat beberapa fungsi pembatas berdasarkan warna bidak yang sedang berjalan.

a. Bidak biru (*bluenormal* atau *rednormal*):
   1. *Score* > *Bestscore*, jika tidak maka penelusuran *backtrack* ke simpul anak yang lain. Jika ya maka *Bestscore* = *Score*.
   2. *Bestscore* > *Bluebest* dan *Bestscore* < *Redbest*. Jika tidak maka penelusuran *backtrack* ke simpul anak yang lain. Jika ya maka nilai *Bluebest* = *Bestscore*.

b. Bidak merah (*rednomral* atau *redking*):
   1. *Score* < dari *Bestscore*, jika tidak maka penelusuran *backtrack* ke simpul anak yang lain. Jika ya maka *Bestscore* = *Score*.
   2. *Bestscore* > *Bluebest* dan *Bestscore* < *Redbest*. Jika tidak maka penelusuran *backtrack* ke simpul anak yang lain. Jika ya maka nilai *Bluebest* = *Bestscore*.

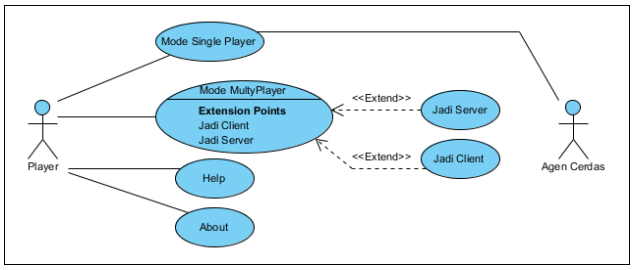
6) *Analisis Fungsi Evaluasi Permainan Dam-Daman:* Dengan menggunakan fungsi evaluasi, maka akan dapat mengetahui solusi yang tepat dalam pohon solusi. Dalam suatu permainan yang berbasis giliran (turn based games), tugas dari fungsi evaluasi adalah untuk melihat kondisi permainan dan memberikan nilai dari sudut pandang satu pemain. Berikut hasils analisis dari fungsi evaluasi permainan dam-daman :

1. Bidak Nomal Merah (*rednormal*)
   $$score = score - 100 - (i)^2 \quad (1)$$
2. Bidak Raja Merah (*redking*) yang berada dipinggir
   $$score = score - 200 + 10 \quad (2)$$
3. Bidak Raja Merah yang tidak berada dipinggir
   $$score = score - 200 \quad (3)$$
4. Bidak Nomal Biru (*bluenormal*)
   $$score = score + 100 + (8-i)^2 \quad (4)$$
5. Bidak Raja Biru (*blueking*) yang berada dipinggir
   $$score = score + 200 - 10 \quad (5)$$
6. Bidak Raja Biru yang tidak berada dipinggir
   $$score = score + 200 \quad (6)$$

D. Perancangan Sistem

Perancangan sistem bertujuan untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap, sehingga nantinya mengurangi resiko kesalahan pada saat implementasi atau *coding*, supaya mengurangi resiko terbuangya biaya dan tenaga secara sia-sia yang diakibatkan dari kesalahan analisis dalam pembangunan / implementasi program.

1) Perancangan UML: Aplikasi permainan ini menggunakan UML sebagai bahasa pemodelan, diagram UML yang digunakan dalam perancangan aplikasi ini menggunakan UML 2.0 yang terdapat beberapa diagram, yaitu *class*, *object*, *package*, *deployment*, *component*, *activity*, *sequence*, *communication*, *interaction overview*, *timing*, *behavior state machine*, *protocol state machine*, dan *use case* diagram.



Gambar 4. Diagram Use Case Permainan Dam-Daman

V. HASIL DAN PEMBAHASAN

A. Implementasi Sistem

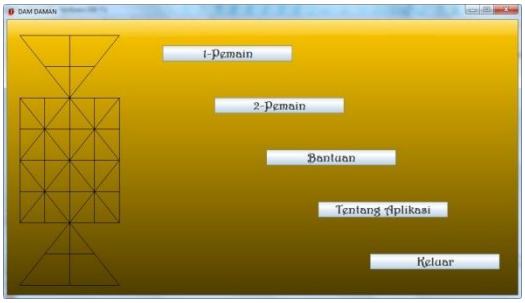
Pembuatan program (*Coding*) pada penelitian ini dilakukan dengan membuat beberapa *package* yang terdapat didalamnya kelas, gambar, dan efek suara. Ada dua *package* yang didalamnya kelas-kelas, yaitu *package* damdaman52.v1 dan *package* multiplayer, berikut adalah kelas kelas yang telah dikelompokan didalam masing-masing *package* tersebut:

1. *Package* damdaman52.v1 : kelas DamFrame, kelas Dam, kelas Help, kelas About, kelas DamMove, kelas Dpanel, kelas GameEngine, kelas GameWin, kelas PanelDam, kelas PlaySound, kelas StartPanel,
2. *Package* multiplayer: kelas DamBoardPanel, kelas DamModel, kelas DefaultSocket, kelas GameFrame, kelas LocalClientSocket, kelas LocalServerSocket, kelas Model, kelas Player, kelas Square, kelas SquareID, kelas Stone,

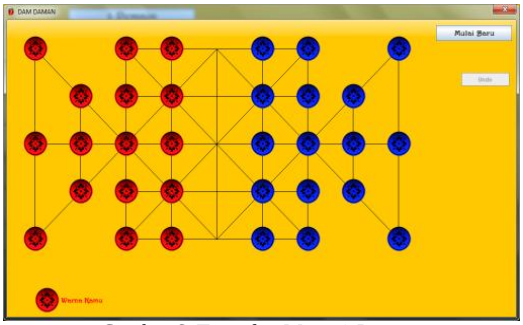
kelas Stone2D, kelas StartWindowFrame, kelas StatusConnection, dan kelas Trasporter.

B. *Pengujian White Box*

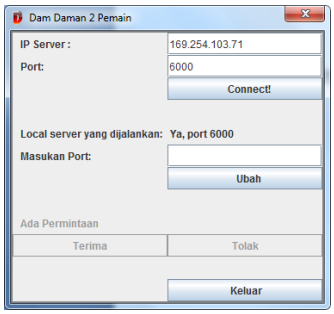
Pengujian *white box* dilakukan dengan menguji *atribut* dan *method* yang ada pada kelas-kelas yang dibangun. Pengujian dilakukan dengan mengecek semua *statement* pada program telah dieksekusi paling tidak satu kali. Hasil dalam pengujian *white box* yaitu semua *statement*, *atribut* dan *method* yang ada pada kelas-kelas dalam aplikasi semua berjalan sesuai dengan analisis perancangan. Berikut hasil dari tempilan aplikasi dari hasil kelas-kelas yang telah diuji:



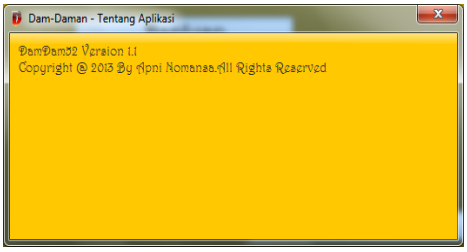
Gambar 5. Tampilan Menu Utama



Gambar 6. Tampilan Menu 1-Pemain



Gambar 7. Tampilan Pengaturan Menu 2-Pemain



Gambar 8. Tampilan Tentang Aplikasi

Gambar 9. Tampilan Bantuan, Peraturan Permainan

Gambar 10. Tampilan Bantuan, Bantuan Aplikasi

C. *Pengujian Black Box*

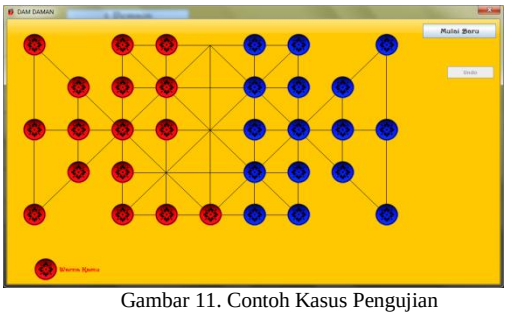
Tabel 1. Pengujian *Black Box* ke 18

NO	Aktivitas Pengujian	Hasil Pengujian
1	Klik tombol menu “1-Pemain”	Sukses
2	Klik tombol menu 2-Pemain	Sukses
3	Klik tombol menu Bantuan	Sukses
4	Klik tombol menu Tentang Aplikasi	Sukses
5	Klik tombol menu Keluar	Sukses
6	Klik tombol Mulai	Sukses
7	Menjalankan bidak yang dipilih sesuai dengan peraturan	Sukses
8	Memakan bidak lawan yang dilompati sesuai dengan peraturan	Sukses
9	Memindahkan bidak ke garis pertahanan lawan	Sukses
10	Memenangkan permainan	Sukses
11	Menjalankan bidak tidak sesuai dengan peraturan	Sukses
12	Memakan bidak lawan tidak sesuai peraturan	Sukses
13	Mengaktifkan Server Permainan	Sukses
14	Klik tombol Ubah	Sukses
15	Klik tombol Connect	Sukses
16	*Client* Mengirim Permohonan ke	Sukses

	Server	
17	Klik tombol Terima	Sukses
18	Klik tombol Tolak	Sukses
19	Mengkoneksikan permainan *server* dan *Client*	Sukses
20	Klik tombol Connect dengan IP dan Port yang salah	Sukses
21	Tidak ada client yang mengirim permohonan	Sukses
22	Menutup permainan di komputer server	Sukses
23	Memilih *Combo Box* Pilihan Peraturan Permainan	Sukses
24	Memilih *Combo Box* Pilihan Batuan Aplikasi	Sukses
25	Melihat informasi tentang aplikasi	Sukses

D. Pengujian Algoritma Backtracking

Pengujian Algoritma *Backtracking* adalah perbandingan pengujian algoritma dengan perhitungan manual dengan pengujian algoritma yang telah ditanam didalam aplikasi permainan. Pada gambar dibawah ini agen bermain sebagai bidak biru dan mendapat giliran melangkah.

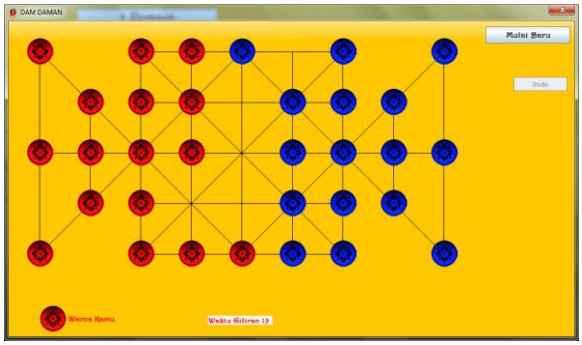


Gambar 11. Contoh Kasus Pengujian

Pada gambar diatas bidak merah sebagai pemain berada pada *array* [0][0], [0][2], [0][4], [1][1], [1][2], [1][3], [2][0], [2][1], [2][2], [2][3], [2][4], [3][0], [3][1], [3][2], [3][4] dan [4][4] sedangkan bidak biru sebagai agen berada pada *array* [5][0], [5][1], [5][2], [5][3], [5][4], [6][0], [6][1], [6][2], [6][3], [6][4], [7][1], [7][2], [7][3], [8][0], [8][2] dan [8][4]. Dalam pengujian algoritma ini dibuat pohon permainan yang menggambarkan pencarian solusi pergerakan dari kondisi papan permainan diatas. Karena terdapat percabangan begitu banyak, sehingga pohon permainan dibagi menjadi beberapa pohon bagian berdasarkan kemungkinan pergerakan pertama bidak. Terdapat 7 kemungkinan pergerakan pertama bidak, yaitu [5][0] ke [4][0], [5][1] ke [4][0], [5][1] ke [4][1], [5][1] ke [4][2], [5][2] ke [4][2], [5][3] ke [4][2] dan [5][3] ke [4][3].

Dari hasil pencarian pohon permainan pada kasus diatas terdapat 475 kemungkinan pergerakan yang dapat dilakukan oleh bidak biru dengan 4 kedalaman pohon pencarian. Dengan mengunakan algoritma *backtracking* dari 475 kemungkinan pergerakan hanya 166 kemungkinan pergerakan yang ditelusuri karena kebanyakan pergerakan melawati fungsi pembatas dari algoritma *backtracking* sehingga pergerakannya dibatalkan penelusurannya. Dari 166 kemungkinan pergerakan yang ditelusuri, didapatkan pergerakan terbaik dengan *score* -2 dari pergerakan [5][0] ke [4][0] pada kemungkinan pergerakan ke 6.

Setelah didapatkan hasil pergerakan yang dilakukan agen cerdas, selanjutnya pengujian dilakukan pada aplikasi permainan yang telah dibuat. Setelah dilakukan pengujian pada aplikasi permainan ternyata hasilnya sama yaitu agen cerdas akan menggerakan bidak pada [5][0] kemudian bergerak ke [4][0], berikut gambar perubahan posisi yang terjadi:



Gambar 12. Perubahan posisi bidak setelah bidak biru bergerak

VI. KESIMPULAN

Dari analisis perancangan serta hasil implementasi program aplikasi yang dilakukan, dapat ditarik kesimpulan sebagai berikut:

1. Aplikasi permainan dam-daman dengan mengimplementasikan algoritma *backtracking* berhasil dibuat dengan menggunakan bahasa pemrograman *java* dan dibantu oleh *Netbeans IDE*.

2. Algoritma *backtracking* dapat diimplementasikan kedalam permainan dam-daman sebagai agen cerdas (komputer) penantang pemain (*user*).
3. Algoritma *Backtracking* memiliki pencarian yang baik karena memilki fungsi pembatas, sehingga ketika pencarian itu dianggap tidak akan menemukan solusi lebih baik maka pencari diberhentikan.

VII. SARAN

Berdasarkan hasil pengerjaan yang diperoleh maka penulis mempunyai saran-saran untuk meningkatkan kinerja dan pengembang aplikasi sebagai berikut:

1. Menambahkan fungsionalitas yang baru, seperti terdapatnya tingkatan permainan.
2. Membandingkan algoritma *backtracking* dengan algoritma lainnya untuk mengetahui optimasi algoritma ini sebagai kecerdasan buatan pada permainan dam-daman.
3. Mengembangkan aplikasi yang dapat digunakan pada sistem operasi *mobile*, seperti *android, iphone* dan *blackberry*.

REFERENSI

[1] Suyoto. *Intelengensi Buatan : Teori dan Pemrograman.* Yogyakarta: Gava Media. 2004.

[2] Bakri, Addhal Huda. *Analisis dan Implementasi Algoritma Backtracking pada Permainan Congklak*. [Online]. Available : http://repository.usu.ac.id

[3] Kurniawan. Agus. 2011. *Dasar Pemrograman Socket dengan Java*. [Online]. Available : http://blog.aguskurniawan.net/post/ Dasar-Pemrograman-Socket-Dengan-Java.aspx

[4] Pradiyar, dkk, 2007. *Analisa dan Perancangan Game 2d Strategi Pada Pc Berbasiskan Engine Torque Game Builder*. [Online] Available : http://library.binus.ac.id.

[5] Zain, Moh.Fatoni. 2009. *Game Simulasi Out-Boud Jelajah Malang*. [Online]. Available : http://lib.uin-malang.ac.id

[6] Sartika, Wida. 2010. *Perancangan Media Informasi Permainan Tradisional Jawa Barat*. [Online]. Available : http://elib.unikom.ac.id/.

[7] *Permainan Sunda: Cara Bermain Dam Daman*. *[Online]*. *Available : http://www.urangkampoeng.com/*.

[8] NST, Muhammad Rival Anggi. 2010. *Analisis dan Implementasi Algoritma Runut Balik (Backtracking) pada Permainan Magic Square*. [Online]. Available : http://repository.usu.ac.id.

[9] Munir, Rinaldi. 2013. *Algoritma Runut-Balik*. [Online]. Available : http://informatika.stei.itb.ac.id~rinaldi.munirStmik2012-2013Algoritma%20Runut-balik.ppt.