IMPLEMENTASI ALGORITMA ITERATIVE DEEPENING A* DAN METODE PRUNING PADA SOLUSI PERMAINAN PUZZLE FLOW FREE COLOR

Fahrurrozi

Jurusan Teknik Informatika FTIK UNIKOM
Jln. Depati ukur 112-114 Bandung
E-mail : rozymom86@gmail.com

ABSTRAK

Flow Free Color adalah sebuah permainan bertema *puzzle* yang cukup populer bisa dimainkan di komputer ataupun mobile. Permainan ini terdiri dari sebuah matriks n x n berisi beberapa lingkaran yang berwarna yang tersebar di beberapa grid. Pada permainan *flow free color* lintasan dianggap tidak *valid* jika tidak bertemu dengan pasangannya, dibutuhkan algoritma pencarian jalur solusi untuk menghubungkan titik-titik warna agar dapat terhubung sesuai dengan jenis warnanya.

Adapun algoritma yang dapat digunakan seperti algoritma *brute force*, *greedy* dan lain-lain. Pada penelitian sebelumnya menggunakan algoritma *brute force*, namun memiliki kelemahan karena waktu eksekusi yang dibutuhkan lebih lama. Terdapat salah satu algoritma untuk menyelesaikan permasalahan ini *yaitu algoritma Iterative Deepening A**.

Namun dalam penelitian ini akan digunakan algoritma IDA* dalam pencarian jalur solusi pada permainan *flow free color* dan Metode pruning akan mengurangi jumlah simpul yang dievaluasi oleh pohon pencarian.

Kata kunci : *Iterative Deepening* A*, algoritma A*, metode pruning, *flow free color*

1. PENDAHULUAN

Flow Free adalah sebuah permainan berjenis puzzle, game ini terdiri dari sebuah matriks n x n berisi beberapa pasangan lingkaran yang berwarna yang tersebar di beberapa grid [2]. Pada game ini pemain harus menghubungkan titik-titik warna sesuai dengan warnanya. Masalah yang terdapat pada game ini adalah cara menghubungkan titik-titik warna yang tersebar secara random agar dapat terhubung dengan warna yang sesuai. Dalam pemrosesan menghubungkan titik-titik warna, dibutuhkan algoritma pencarian. Algoritma pencarian yang diterapkan untuk menghubungkan titik-titik warna adalah algoritma brute force dan greedy [15].

Penerapan algoritma IDA* pada game flow free color yaitu untuk mendapatkan jalur solusi untuk menghubungkan tiap-tiap titik-titik warna sesuai dengan jenis warna. Algoritma IDA* mengambil kelebihan A* yaitu dalam biaya perhitungan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan untuk pemeriksaan setiap simpul-simpulnya, hal ini yang membuat algoritma IDA* complete dan optimal. Namun algoritma IDA* dilakukan secara iteratif, maka besar kemungkinan IDA* membangkitkan simpul-simpul yang berulang-ulang dan membutuhkan waktu yang lama. Untuk mengatasi masalah ini, akan menggunakan metode tambahan yaitu Metode Pruning. Dengan metode pruning ini mencoba untuk mengatasi kelemahan yang dimiliki algoritma IDA* yaitu dengan mengabaikan jalur yang kemungkinan tidak menghasilkan solusi, untuk mengetahui jalur tersebut menghasilkan solusi atau tidak menghasilkan solusi yaitu dengan membandingkan nilai alpha dan Beta sesuai dengan aturan dari metode pruning.

Dengan adanya masalah-masalah di atas, Algoritma *Iterative Deepening A** (IDA*) dan *pruning* diharapkan dapat menyelesaikan solusi permainan di atas

2. ISI PENELITIAN

2.1 *Game*

Game adalah permainan yang menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh piranti video. Permainan video umumnya menyediakan sistem penghargaan, misalnya skor yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan.

2.2 *Genre Game*

Game memiliki banyak tipe atau genre. Masing-masing genre memiliki karakteristik berbeda. Genre game [7] diantaranya yaitu:

a. Action Game

Game ini dikenali dari aksi didalamnya yang berupa pergerakan, attack, reaksi dan pergerakan-pergerakan lainnya.

b. Adventure Game
Game tipe ini menuntut player untuk menyelesaikan misi yang diberikan sepanjang petualangan di dalam game.
c. Education Game
Game edukasi memberikan penekanan pada unsur pendidikan dan pembelajaran.
d. Puzzle Game
Puzzle game memiliki tujuan yang sangat sederhana untuk menyelesaikan puzzle yang disediakan.
e. Role-Playing Game
Pada game bertipe ini memiliki dunia yang luas untuk dieksplorasi oleh player, baik untuk mencari harta karun, objek dan status (leveling), sambil menghancurkan monster dan objek yang menghalanginya.
f. Simulation Game
Simulation Game atau sims adalah suatu bentuk permainan yang menggambarkan situasi dunia nyata, baik dalam rupa simulasi driving maupun dalam hubungannya dengan lingkungan interaksi antara manusia.
g. Strategi Game
Dalam game tipe ini dibutuhkan pemikiran dan perencanaan yang benar untuk dapat memenangkan permainan dan biasanya menggunakan sudut pandang overhead sehingga player dapat melihat seluruh area permainan.

2.3 Metode-metode Pencarian

Metode pencarian secara umum dibagi menjadi 2 kelompok, yaitu blind atau uninformed search (pencarian buta atau tidak dibekali informasi) dan heuristic atau informed search (pencarian dengan panduan atau berbekal informasi). Setiap metode memiliki karakteristik yang saling membedakan satu sama lain dengan kelebihan dan kekurangan masing-masing, yang bisa diukur performansinya dengan empat kriteria [4] berikut:
a. Completeness, yaitu apakah metode pencarian tersebut menjamin ditemukannya solusi jika solusinya memang ada.
b. Time complexity, yaitu berapa lama waktu yang diperlukan selama proses pencarian.
c. Space complexity, yaitu berapa banyak memori yang diperlukan selama proses pencarian.
d. Optimality, yaitu apakah metode pencarian tersebut menjamin ditemukannya solusi yang terbaik jika ada beberapa solusi yang berbeda.

2.4 Algoritma Iterative Deepening A* (IDA*)

Algoritma Iterative Deepening A* merupakan gabungan dari algoritma IDS dan A*. Algoritma Iterative Deepening Search adalah suatu teknik yang menggabungkan keuntungan pencarian dengan menggunakan teknik Breadth First Search (BFS) dalam hal complete dan optimal serta keuntungan dari teknik pencarian Depth First Search (DFS) dalam hal Space Complexity [3]. Sedangkan Algoritma A*(A Star) adalah algoritma pencarian graf/pohon yang mencari jalur dari satu titik awal ke sebuah titik akhir yang telah ditentukan [4]. Kelebihan algoritma A* yaitu biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Sedangkan kelebihan algoritma IDS menggabungkan kelebihan Breadth First Search (complete dan optimal) dan kelebihan Depth First Search (membutuhkan sedikit memori) [10]. IDS merupakan algoritma Blind Search yang melakukan pencarian secara iteratif menggunakan penelusuran Depth Limited Search (DLS) dimulai dengan batasan limit 0 [5].

Jika jalur solusi belum ditemukan, setiap iterasi mengembalikan nilai f-limit baru yang akan digunakan sebagai batasan pencarian untuk iterasi berikutnya. Nilai f-limit itu sendiri diperoleh dari gabungan biaya sebenarnya dan biaya perkiraan seperti terdapat pada persamaan (1) , yang secara matematika dinyatakan sebagai :

$f(n) = g(n) + h(n)$ ...(1)

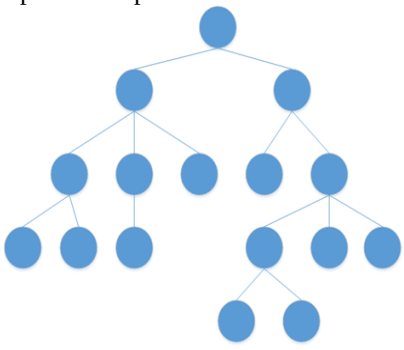
Dengan :
n = simpul
g(n) = Jarak asli dari lokasi awal ke lokasi ke n
h(n) = Jarak dari lokasi awal ke node lokasi n
f(n) = Nilai evaluasi
Sedangkan untuk melakukan pencarian h(n) seperti terdapat pada persamaan (2) dilakukan dengan cara sebagai berikut :

$$h(n) = \sqrt{(x - x1)^2 + (y - y1)^2} \quad ...(2)$$

Dengan :
h(n) = perkiraan biaya minimum dari simpul n ke simpul tujuan
X = posisi X simpul awal
Y = posisi Y simpul awal
X1 = posisi X dari simpul lokasi ke simpul tujuan
Y 1= posisi Y dari simpul lokasi ke simpul tujuan

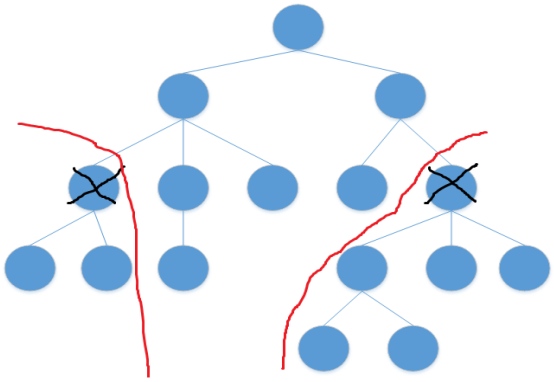
2.5 Metode Pruning

Pruning adalah metode yang mengabaikan state yang tidak mungkin menghasilkan solusi [11]. Misalkan tree di bawah merepresentasikan ruang pencarian yang harus ditelusuri untuk menemukan solusi dapat dilihat pada Gambar 1.



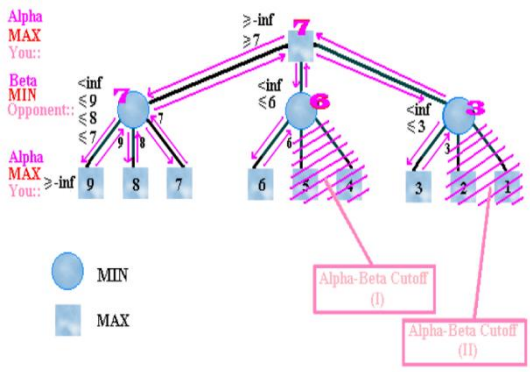
Gambar 1 Contoh Pohon Ruang Pencarian

Dengan *pruning* dapat dibuktikan secara matematis, dapat memangkas pencarian dengan tidak menelusuri state pencarian yang tidak perlu dapat dilihat pada Gambar 2.



Gambar 2 Contoh pruning pada pohon ruang pencarian.

Karekteristik dari metode pruning ini adalah :
a. Semakin tinggi state atau semakin kecil kedalaman state yang kita pruning maka semakin banyak state pencarian yang kita pangkas
b. Dengan pruning yang tepat, keuntungan dari pemangkasan ini dapat mengurangi waktu pencarian secara eksponensial.
Dengan metode ini, waktu penelusuran dapat dikurangi. Salah satu aplikasi dari metode pruning yang terkenal adalah penggunaan Alpha-Beta pruning pada komputer catur [5]. Alpha-beta pruning mengeliminasi kemungkinan langkah ke depan dari himpunan hasil analisis jika mereka terbukti tidak lebih baik dari langkah yang telah ditemukan. Pengembang dari komputer catur Deep Blue menggunakan algoritma ini dan menghasilkan inovatif teori yang disebut sebagai "singular extensions" [5].



Gambar 3 Contoh Algoritma Alpha-Beta Pruning

Algoritma akan mendata dua nilai : alpha dan beta, yang akan merepresentasikan nilai minimum untuk MAX, dan nilai maksimum untuk MIN. Keadaan awal nilai alpha adalah –infinity, dan beta bernilai awal +infinity. Sedangkan alpha-beta Cutoff adalah node alpha-beta yang tidak perlu dievaluasi. Selama jalannya rekursi, perbedaan nilai alpha dan beta akan semakin mengecil. Ketika nilai beta menjadi lebih kecil dari alpha, berarti keadaan saat ini bukan merupakan hasil terbaik dari kedua pemain, dengan demikian tidak perlu dieksplorasi lebih jauh seperti terdapat pada Gambar 3.
Langkah-langkah metode pruning :
1. Hampiri node pertama dengan nilai 9, naik ke node parent level min masukkan nilai 9, hampiri nilai 8, karena 9<8 maka ganti node parent 8, hampiri 7, karena 8<7 maka nilai 7 naik k level min.
2. Setelah ketiga node pertama dihampiri, naik ke level max masukkan nilai 7.
3. Hampiri node keempat pada dengan nilai simpul 6, naik ke node parent pada level min masukkan nilai 6. Jika menghampiri node berikutnya, maka mencari nilai yang lebih tinggi dari 6, sementara pada level max mencari yang lebih kecil dari 7, maka node 5 dan 4 di pruning (tidak dihampiri).
4. Setelah itu hampiri node ketujuh pada nilai simpul 3, naik ke node parent pada level min masukkan nilai 3. Jika menghampiri node berikutnya, maka akan mencari nilai yang lebih tinggi dari 3, sementara pada level max mencari lebih kecil dari 7, maka node 2 dan 1 di pruning (tidak dihampiri). Berdasarkan langkah-langkah diatas, adapun flowchart dari pruning terdapat pada



Gambar 4 *Flowchart Pruning*

sedangkan pseudocode untuk *pruning* sebagai berikut:

```
evaluate (node, alpha, beta)
    if node is a leaf
        return the heuristic value of
    if node is a minimizing node
        for each child of node
            beta = min (beta, evaluat
            if beta <= alpha
                return beta
        return beta
    if node is a maximizing node
        for each child of node
            alpha = max (alpha, evalu
            if beta <= alpha
                return alpha
        return alpha
```

2.6 Analisis Permainan Flow Free Color

Permainan Flow Free Color adalah game yang bergenre puzzle yang telah dikembangkan oleh big duck game untuk perangkat mobile dan ios, pada game ini terdapat tantangan, tantangan yang diberikan pada game ini sangat menarik dengan ditantang berbagai tingkat kesulitan yang ditampilkan dalam berbagai grid permainan yaitu mulai dari 5x5 sampai dengan tingkat tertinggi 14 x 14.

Aturan permainan flow free color sebagai berikut :

1. Diasumsikan terdapat grid 4 x 4 yang berisi 6 titik warna yang terdiri dari 2 warna yang sama pada setiap titik warna
2. 6 titik warna ini terdiri dari 2 warna merah, 2 warna kuning, dan 2 warna hijau
3. Warna –warna ini akan tersebar di dalam grid 4 x 4 dan warna – warna ini harus dihubungkan sesuai dengan warna yang sama
4. Dalam menghubungkan warna kita harus membentuk sebuah lintasan dan menelusuri grid agar warna yang akan dihubungkan dapat terhubung.
5. Dalam menelusuri grid tidak diperbolehkan memotong lintasan yang telah terbentuk, dan jika terdapat grid yang kosong atau tidak dilalui maka lintasan dianggap tidak valid.

2.7 Analisis Data Masukan

Analisis Data masukan yang dilakukan dalam penelitian ini adalah menganalisis inputan dalam algoritma IDA* pada permainan flow free color. Data masukan parameter yang diinputkan oleh user yaitu titik-titik warna yang diwakili dengan angka dari 0-6 yang terdiri dari warna kosong, green, red, blue, yellow, orange, purple. Titik-titik warna tersebut dilakukan secara random dalam penyebaran disetiap grid. Adapun tahapan untuk melakukan random sebagai berikut:

Jumlah grid - Jumlah warna x 2



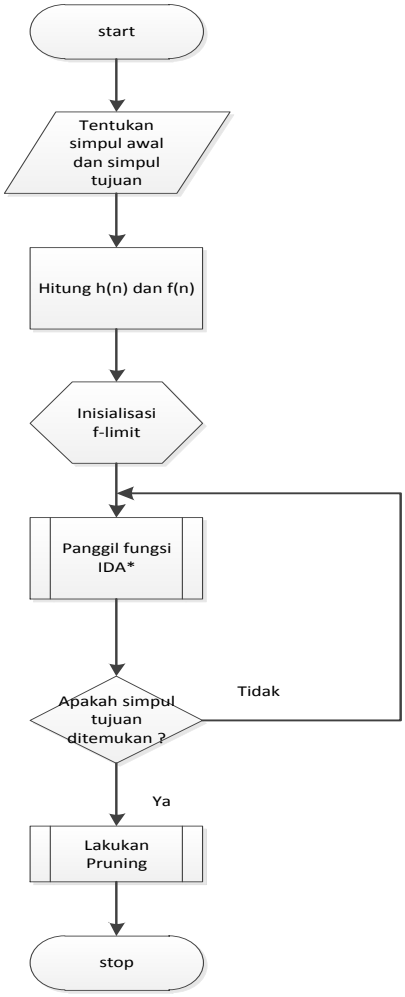
Jumlah grid – 1

Adapun pseudocode random adalah sebagai berikut :

```
function genColor (seed) {
   color = Math.floor((Math.abs(Math.sin(seed) *
16777215)) % 16777215);
   color = color.toString(16);
   // pad any colors shorter than 6 characters with
leading 0s
   while(color.length < 6) {
      color = '0' + color;
   }
   return color;
}
for (chr in chars) {
   if (chars.hasOwnProperty(chr)) {
      color = genColor(chars[chr].charCodeAt(0));
      colorloop.append('<div style="background:#'
+ color + '">' + chars[chr] + ' = #' + color +
'</div>');
      ccount[color] = ccount[color] === undefined
? 1 : ccount[color] + 1;
   }
}
for(itm in ccount) {
   if (ccount.hasOwnProperty(itm)) {
      if (ccount[itm] > 1) {
         repeatCount += 1;
      }
   }
}
```

2.8 Analisis Algoritma IDA*

Algoritma IDA* tidak mengingat jalur terpendek saat ini dan biaya simpul-simpul yang pernah dikunjungi, tetapi hanya mengingat satu path tunggal dalam satu waktu. Pada akhirmya algoritma IDA* akan menemukan jalur dengan biaya minimum ke simpul tujuan karena hanya meng-cache (menyimpan dalam memori) simpul yang berada dalam jalur pencarian. Oleh karena itu jumlah memori yang dibutuhkan menjadi jauh lebih sedikit. Berikut merupakan flowchart langkah-langkah pencarian (IDA*),



Gambar 5 *Flowchart Algoritma* IDA*

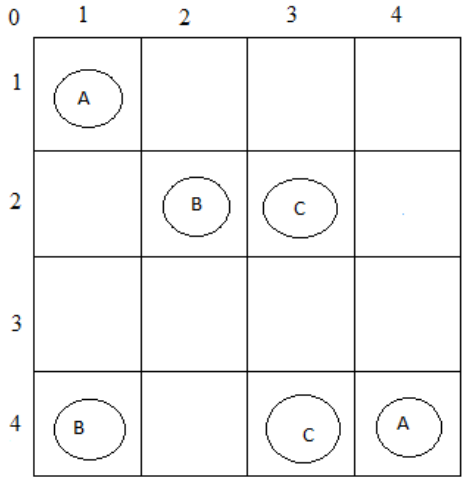
Sedangkan pseudo- code dari algoritma IDA* adalah sebagai berikut :

```
function depth_first_search (n, limit)
    If f(n) > limit
    then return f(n)
    if h(n) = 0 then BERHASIL
    return nilai terendah dari depth_first_search (ni;,limit) bagi semua successor n; dari n
end
prosedure IDA* (n)
    limit ≥ h(n)
repeat
    limit = depth_first_search (n,limit)
until BERHASIL
end
```

2.9 Penerapan Heuristic Iterative Deepening A*

Pada permainan ini mempunyai grid 4 x 4 yang terdapat 6 titik pola warna yang terdiri dari 2 warna yang sama yaitu 2 warna merah, 2 warna kuning, 2 warna hijau. Dimana setiap warna diwakili oleh huruf yaitu :

A = warna merah
B = warna kuning
C = warna hijau



Gambar 6 contoh kasus *flow free color*

Pada permainan flow free color ini terdapat beberapa pola warna di dalam setiap grid-grid. Pola-pola warna yang terdapat dalam grid sebagai berikut :

Tabel 1 Tabel Pola-pola warna

Pola titik warna permainan *flow free color*			
No	Nama Pola	Koordinat Awal	Koordinat Akhir
1.	Pola warna A	1,1	4,4
2.	Pola warna B	2,2	1,4
3.	Pola warna C	3,2	3,4

2.10 Pola Pencarian Warna A

Tabel 2 Pola Pencarian Warna A

Pencarian Pola Warna A		
Simpul Awal	Perpindahan akhir	Iterasi
1,1	4,4	1
1,1	2,1	2
2,1	3,1	3

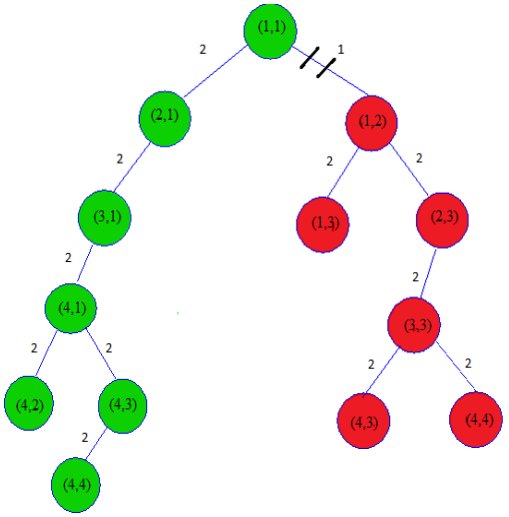
3,1	4,1	4
4,1	4,2	5
4,2	4,3	6
4,3	4,4	7
1,1	4,4	8
1,1	1,2	9
1,2	1,3	10
1,3	2,3	11
2,3	3,3	12
3,3	4,3	13
4,3	4,4	14
1,1	4,4	15

Setelah dilakukan analisis dengan menggunakan algoritma IDA* didapat hasil nilai evaluasi, adapun nilai evaluasi adalah sebagai berikut :

Tabel 3 Nilai evaluasi pola warna A

Pola warna A		
Koordinat Awal	Koordinat Akhir	Nilai evaluasi
(1,1)	(2,1)	2
(2,1)	(3,1)	2
(3,1)	(4,1)	2
(4,1)	(4,2)	2
(4,2)	(4,3)	2
(4,3)	(4,4)	2
(1,1)	(1,2)	1
(1,2)	(1,3)	2
(1,3)	(2,3)	2
(2,3)	(3,3)	2
(3,3)	(4,3)	2

Setelah koordinat Pola A dihitung selanjutnya akan dilakukan pencarian langkah-langkah algoritma IDA* pada pola A yang dapat menghasilkan solusi dan langkah-langkah yang tidak menghasilkan solusi dengan menggunakan metode pruning berikut proses dengan metode pruning :



Gambar 7 Himpunan kemungkinan jalur lintasan pola A

Metode pruning akan mendata dua nilai yaitu alpha dan beta, Keadaan awal nilai alpha adalah –infinity, dan beta bernilai awal +infinity. Ketika nilai beta menjadi lebih kecil dari alpha,berarti keadaan bukan merupakan hasil solusi dan tidak perlu untuk dilakukan penelusuran lebih jauh. Jadi kesimpulan pada Gambar 7 adalah jalur yang berwarna hijau merupakan jalur solusi dalam pembentukan lintasan untuk warna pola A karena sesuai dengan analisis metode pruning dan aturan permainan. Sedangkan jalur yang berwana merah merupakan jalur yang bukan menghasilkan solusi dikarenakan jalur ini di anggap memotong jalur pola warna lainnya dan juga dilakukan analisis metode pruning dengan menghasilkan nilai $\beta < \alpha$.

2.11 Pola Pencarian warna B

Tabel 4 Pola Pencarian warna B

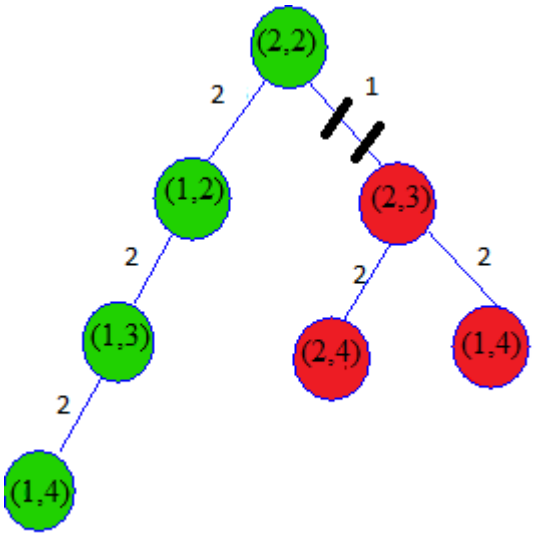
Pencarian Pola Warna B		
Simpul Awal	Perpindahan Akhir	Iterasi
2,2	1,4	1
2,2	1,2	2
1,2	1,3	3
1,3	1,4	4
2,2	1,4	5
2,2	2,3	6
2,3	2,4	7
2,4	1,4	8
2,2	1,4	9

Setelah dilakukan analisis dengan menggunakan algoritma IDA* didapat hasil nilai evaluasi, adapun nilai evaluasi adalah sebagai berikut :

Tabel 5 Nilai evaluasi pola warna B

Pola warna B		
Koordinat Awal	Koordinat Akhir	Nilai evaluasi
(2,2)	(1,2)	2
(1,2)	(1,3)	2
(1,3)	(1,4)	2
(2,2)	(2,3)	1
(2,3)	(2,4)	2
(2,4)	(1,4)	2

Setelah koordinat Pola B dihitung selanjutnya akan dilakukan pencarian langkah-langkah algoritma IDA* pada pola B yang dapat menghasilkan solusi dan langkah-langkah yang tidak menghasilkan solusi dengan menggunakan metode pruning berikut proses dengan metode pruning :



Gambar 8 Himpunan kemungkinan jalur lintasan pola B

Metode pruning akan mendata dua nilai yaitu alpha dan beta, Keadaan awal nilai alpha adalah –infinity, dan beta bernilai awal +infinity. Ketika nilai beta menjadi lebih kecil dari alpha,berarti keadaan bukan merupakan hasil solusi dan tidak perlu untuk dilakukan penelusuran lebih jauh. Jadi kesimpulan pada **Error! Reference source not found.**. adalah jalur yang berwarna hijau merupakan jalur solusi dalam pembentukan lintasan untuk warna pola B karena sesuai dengan analisis metode pruning dan aturan permainan. Sedangkan jalur yang berwana merah merupakan jalur yang bukan menghasilkan solusi dikarenakan jalur ini di anggap memotong jalur pola warna lainnya dan juga dilakukan analisis metode pruning dengan menghasilkan nilai $\beta < \alpha$.

2.12 Pola Pencarian warna C

Tabel 6 Pola Pencarian Pola C

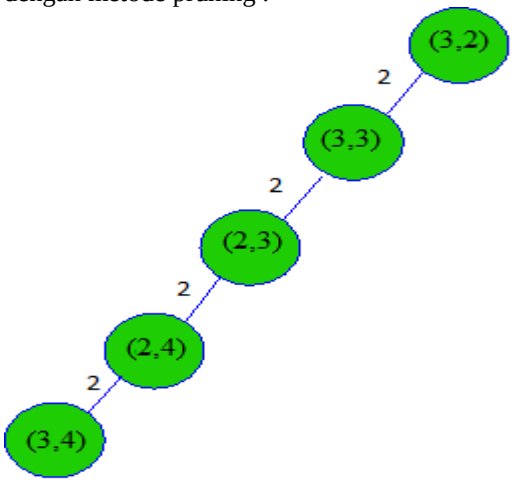
Pencarian Pola Warna C		
Simpul Awal	Perpindahan Akhir	Iterasi
3,2	3,4	1
3,2	3,3	2
3,3	2,3	3
2,3	2,4	4
2,4	3,4	5
3,2	3,4	6

Setelah dilakukan analisis dengan menggunakan algoritma IDA* didapat hasil nilai evaluasi, adapun nilai evaluasi sebagai berikut:

Tabel 7 Nilai evaluasi pola warna C

Pola warna C		
Koordinat Awal	Koordinat Akhir	Nilai evaluasi
(3,2)	(3,3)	2
(3,3)	(2,3)	2
(2,3)	(2,4)	2
(2,4)	(3,4)	2

Setelah koordinat Pola C dihitung selanjutnya akan dilakukan pencarian langkah-langkah algoritma IDA* pada pola C yang dapat menghasilkan solusi dan langkah-langkah yang tidak menghasilkan solusi dengan menggunakan metode pruning berikut proses dengan metode pruning :



Gambar 9 Himpunan kemungkinan jalur lintasan pola C

Jalur pola C merupakan jalur yang menghasilkan solusi dan sesuai dengan aturan permainan.

2.13 Analisis Kebutuhan Non Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga terdiri dari elemen atau komponen-komponen yang dibutuhkan oleh sistem yang akan dibangun sampai dengan sistem tersebut terimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan. Adapun kebutuhan yang digunakan dalam implementasi ini sebagai berikut :

1. Kebutuhan perangkat keras
2. Kebutuhan perangkat lunak

Analisis kebutuhan non fungsional bertujuan agar penerapan implementasi dapat digunakan sesuai dengan kebutuhan.

2.14 Analisis Kebutuhan Perangkat Keras

Untuk mengimplementasikan algoritma *iterative deepening* A*, diperlukan spesifikasi perangkat keras yang mendukung proses kerja dari suatu sistem. Adapun spesifikasi perangkat keras yang dibutuhkan dalam pembangunan aplikasi game *flow free color* sebagai berikut :

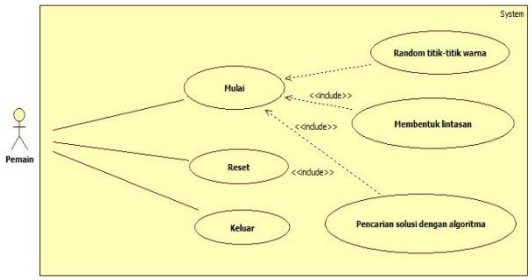
Tabel 8 Spesifikasi perangkat keras

No	Perangkat Keras	Spesifikasi
1	CPU	Intel Dual core 1.80 GHz
2	VGA	Intel(R) HD Graphics
3	Harddisk	80 GB
4	RAM	2 GB

2.15 Analisis Kebutuhan Fungsional

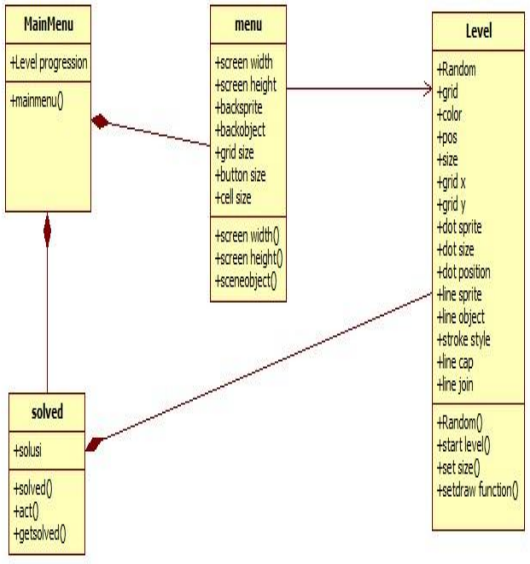
Pemodelan yang digunakan pada *game* yang akan dibuat adalah analisis perancangan dan pengembangan perangkat lunak berbasis objek, dimana alat yang akan digunakan untuk pengembangan perangkat lunak yaitu pemodelan *Unified Modeling Language* (UML). Pemodelan dalam analisis tersebut antara lain pembuatan *use case diagram*, definisi aktor, definisi *use case*, skenario *use case*, *activity diagram*, *sequence diagram, dan class diagram.*

A. *Use Case* Diagram



Gambar 10 *Use Case* Diagram

B. *Class* Diagram



Gambar 11 *Class* Diagram

2.16 Implementasi dan Pengujian

Tahap implementasi merupakan tahap yang dilakukan setelah analisis dan perancangan sistem selesai dilakukan, dan selanjutnya akan diimplementasikan kedalam bahasa pemrograman dan diuji kedalam aplikasi. Aplikasi yang telah dibangun akan diimplementasikan untuk mengetahui sistem dapat berjalan sesuai tujuan atau tidak. Tujuan dari implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem, yang terdiri dari penjelasan mengenai lingkungan implementasi, baik itu lingkungan perangkat keras, maupun lingkungan perangkat lunak, serta implementasi program.

2.17 Implementasi Perangkat Keras

Spesifikasi perangkat keras (*hardware*) yang digunakan dalam implementasi algoritma iterative deepening A* untuk menentukan jalur solusi dan metode pruning pada *game flow free color* adalah sebagai berikut :

Tabel 9 Spesifikasi Perangkat Keras

Perangkat Keras	Spesifikasi Perangkat Keras
Processor	Intel(R) Celeron (R) @ 2.16 GHz
Resolusi Layar	1366 x 768 pixel
Memori	Memori 2 GB
Hardisk	*Hardisk* 320 GB
Keyboard dan *touchpad*	Standar

2.18 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak (*software*) yang digunakan dalam implementasi algoritma iterative deepening A* dan metode pada *game flow free color* sebagai berikut:

Tabel 10 Spesifikasi Perangkat Lunak

Perangkat Lunak	Spesifikasi Perangkat Lunak
Sistem Operasi	Windows 7
Tools Program	JetBrains Webstrom

2.19 Implementasi *Class*

Berikut merupakan implementasi *class entity* yang terdapat pada flow.js sebagai berikut:

Tabel 11 Implementasi *Class Entity*

Class	Stereo Type
Menu.js	*Entity*
Level.js	*Entity*

2.20 Implementasi pengujian

Pada tahap ini pengujian terhadap *game flow free color* dilakukan dengan dua macam teknik, yaitu pengujian Black-box dan pengujian metode. Pengujian black box berfokus pada kesalahan yang terdapat dalam program, sedangkan pengujian metode berfokus pada pengujian implementasi metode pada simulasi .



Gambar 12 Antarmuka Menu Utama



Gambar 13 Antarmuka pengujian

Berdasarkan solusi yang dihasilkan pada pola A berada pada koordinat (1,1), (2,1), (3,1), (4,1), (4,2), (4,3), (4,4), pada pola B berada pada koordinat (1,2), (2,2), (3,2), (3,3), (3,4), sedngkan untuk pola C berada pada koordinat (1,3), (1,4), (2,4), (2,3).

3. PENUTUP

Berdasarkan hasil yang didapat dalam penelitian dan penyusunan skripsi ini serta disesuaikan dengan tujuan, maka dapat diperoleh kesimpulan bahwa :

1. Untuk implementasi algoritma iterative deepening A* pada game flow free color ini dapat digunakan dengan fungsi rumus heuristic sebagai pencarian nilai lintasan, namun pencarian jalur solusi yang menggunakan pohon pencarian depth first search, hal itu dapat memperbanyak pencarian simpul-simpul yang terdapat di setiap pola jalur pencarian, namun dalam hal ini ditambahkan sebuah metode yaitu pruning, metode ini digunakan untuk mengabaikan jalur yang tidak menghasilkan solusi.

2. Algoritma iterative deepening A* dapat melakukan pencarian titik-titik warna sesuai dengan jenis warna dengan penyebaran titik-titik warna secara random, dan tingkat keakurasian yang tepat pada titik-titik warna yang disebarkan secara random sehingga titik-titik warna dapat terhubung sesuai dengan jenis warna.

Berdasarkan hasil dari penelitian ini, maka dapat diberikan saran-saran untuk diberikan dalam pengembangan selanjutnya yaitu

1. Dapat ditambahkan sebuah algoritma untuk melakukan random titik-titik warna yang memiliki jumlah grid yang banyak agar pencarian algoritma IDA* metode pruning dapat berjalan dengan baik, tetapi untuk penambahan algoritma harus diperhatikan terlebih dahulu karena algoritma IDA* pada kasus ini adalah untuk mencari jalur solusi.
2. Dalam hal pengujian harus ditambahkan fitur pencarian titik-titik warna pertahap, karena pada sistem ini dilakukan semua proses pencarian titik-titik warna secara bersamaan. Dan juga tampilan perhitungan proses pertahap.
3. Untuk lebih akurat dalam perhitungan titik-titik warna pengembang dapat menambahkan metode untuk menghitung keakuratan titik-titik warna.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[1] L. Hermawan dan R. Kristoforus Jawa Bendi, "Penerapan Algoritma A* pada Aplikasi Puzzle," dalam Seminar Nasional Teknologi Informasi dan Komunikasi, Palembang, 2013.
[2] Mike, "Flow Free Color," Big Duck Games, 13 07 2012. [Online]. Available: http://blog.bigduckgames.com. [Diakses 12 02 2015].
[3] S. Kusumadewi, Artificial Intelligence, Yogyakarta: Graha Ilmu, 2003.
[4] Suyanto, Artificial Intelligence, Bandung: Informatika, 2011.
[5] J. S. Russell dan P. Norvig, Artificial Intelligence – A Modern Approach 3rd Edition., New Jersey: Prentice hall, 2010.
[6] R. S. Pressman, Rekayasa Perangkat Lunak Pendekatan Praktis, Yogyakarta: Andi, 2012.
[7] A. Ismail, Education Game, Yogyakarta: Pro-u Media, 2009.
[8] A. Rollings dan D. Morris, Game Architecture and Design, New Rider: A New Edition, 2004.
[9] A. Nilwan, Pemrograman Animasi dan Game Profesional, Jakarta: Elex Media Komputindo, 2010.
[10] W. Budiharto dan D. Suhartono, Artificial Intelligence Konsep dan Penerapannya, Yogyakarta: Andi, 2014.
[11] S. J. Russell dan P. Norvig, Artificial Intelligence – A Modern Approach 2rd Edition., New Jersey: Prentice hall, 2010.
[12] B. Hariyanto, Struktur Data : Pondasi Membuat Program Elegan dan Efisien, Bandung: Informatika, 2008.
[13] S. Rosa dan M. Shalahuddin, Rekayasa Perangkat Lunak, Bandung: Informatika, 2013.
[14] F. S. J. Hill, Computer Graphics Using OpenGL (2nd ed.), New Jersey: Prentice hall, 2001.
[15] M. Rahmat, "Pencarian Solusi Permainan Flow Free Menggunakan Brute Force dan Pruning," Jurnal Teknik Informatika, 2013.