

**PERBANDINGAN ANTARA BINARY TREE DENGAN
SINDERWINDER DALAM MAZE GENERATOR**

Skripsi



oleh

YOSHUA HENDRA M

22104969

PROGRAM STUDI INFORMATIKA FAKULTAS TEKNOLOGI
INFORMASI UNIVERSITAS KRISTEN DUTA WACANA

2018

PERBANDINGAN ANTARA BINARY TREE DENGAN SINDERWINDER DALAM MAZE GENERATOR

Skripsi



Diajukan kepada Program Studi Informatika Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana
Sebagai Salah Satu Syarat dalam Memperoleh Gelar
Sarjana Komputer

Disusun oleh

YOSHUA HENDRA M

22104969

PROGRAM STUDI INFORMATIKA FAKULTAS TEKNOLOGI
INFORMASI UNIVERSITAS KRISTEN DUTA WACANA 2018

2017

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

PERBANDINGAN ANTARA BINARY TREE DENGAN SIDEWINDER DALAM MAZE GENERATOR

yang saya kerjakan untuk melengkapi sebagian persyaratan menjadi Sarjana Komputer pada pendidikan Sarjana Program Studi Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana, bukan merupakan tiruan atau duplikasi dari skripsi keserjanaan di lingkungan Universitas Kristen Duta Wacana maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Jika dikemudian hari didapati bahwa hasil skripsi ini adalah hasil plagiasi atau tiruan dari skripsi lain, saya bersedia dikenai sanksi yakni pencabutan gelar keserjanaan saya.

Yogyakarta, 9 Januari 2018



YOSHUA HENDRA MANGLAPY
22104969

HALAMAN PERSETUJUAN

Judul Skripsi : PERBANDINGAN ANTARA BINARY TREE
DENGAN SIDEWINDER DALAM MAZE
GENERATOR

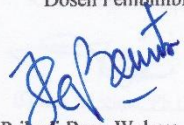
Nama Mahasiswa : YOSHUA HENDRA MANGLAPY
N I M : 22104969
Matakuliah : Skripsi (Tugas Akhir)
Kode : TIW276
Semester : Gasal
Tahun Akademik : 2017/2018

Telah diperiksa dan disetujui di
Yogyakarta,
Pada tanggal 9 Januari 2018

Dosen Pembimbing I


Antonius Rachmat C., S.Kom., M.Cs.

Dosen Pembimbing II


Prihadi Beny Waluyo, SSi., MT.

HALAMAN PENGESAHAN

PERBANDINGAN ANTARA BINARY TREE DENGAN SIDEWINDER
DALAM MAZE GENERATOR

Oleh: YOSHUA HENDRA MANGLAPY / 22104969

Dipertahankan di depan Dewan Penguji Skripsi
Program Studi Informatika Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana - Yogyakarta
Dan dinyatakan diterima untuk memenuhi salah satu syarat memperoleh gelar
Sarjana Komputer
pada tanggal 18 Desember 2017


Yogyakarta, 9 Januari 2018
Mengesahkan,

Dewan Penguji:

1. Antonius Rachmat C., S.Kom., M.Cs.
2. Prihadi Beny Waluyo, S.Si., MT.
3. Widi Hapsari, Dra. M.T.
4. Junius Karel, M.T.


Dekan

(Bardi Susanto, S.Kom., M.T.)


Ketua Program Studi

(Gloria Virginia, Ph.D.)

UCAPAN TERIMA KASIH

Pertama-tama penulis ucapkan syukur kepada Tuhan Yang Maha Esa karena dengan berkat dan karunia-Nya skripsi ini dapat diselesaikan. Skripsi yang berjudul “Implementasi Algoritma *Sidewinder* Sebagai *Generator Maze*” merupakan salah satu syarat untuk mendapatkan gelar sarjana komputer. Diselesaikannya skripsi ini tidak lepas dari partisipasi dan bantuan dari berbagai pihak. Oleh karena itu penulis ingin mengucapkan rasa terima kasih yang setulus-tulusnya kepada :

1. Bapak Antonius R. C., S.Kom., M.Cs dan Prihadi Beny Waluyo, SSi., MT., selaku dosen pembimbing yang selalu gigih dan sabar dalam memberikan arahan.
2. Kepada seluruh anggota keluarga yang saya cintai, Bapak, Ibu, Adik, dan saudara-saudara saya yang selalu memberikan dukungan baik berupa nasehat, motivasi dan doa dari kedua orang tua saya.
3. Seluruh staf pengajar dan pegawai Fakultas Teknologi Informasi UKDW atas keramahan dan ilmu yang telah diberikan kepada saya.
4. Kepada sahabat-sahabat saya di KELAS GOKIL UKDW : Max, Ian Jacob , Aan Setiawan, Bharep Pramono, Yoshua Hendra, Damasus Bagus, Roby Chandra, Stanley Dethan, Prima Noviarso, Rico, Mahendra Yadnya, Richi Costa dkk yang memberikan keceriaan dan kebersamaan.

Seluruh pihak yang ikut membantu namun tidak bisa dituliskan oleh penulis. Penulis mengucapkan terima kasih yang sebesar-besarnya.

DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN SAMPUL DALAM.....	ii
PERNYATAAN KEASLIAN SKRIPSI.....	iii
HALAMAN PERSETUJUAN.....	iv
HALAMAN PENGESAHAN.....	v
UCAPAN TERIMAKASIH.....	vi
DAFTAR GAMBAR.....	ix
DAFTAR TABEL.....	xi
DAFTAR GRAFIK.....	xii
INTISARI.....	xiii
BAB 1	
1.1 Latar Belakang Masalah.....	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan Penelitian.....	3S
1.5 Metode Penelitian.....	3
1.6 Sistematika Penulisan.....	4
BAB 2	
2.1 Tinjauan Pustaka.....	5
2.2 Landasan Teori.....	6
2.2.1 Konsep Sebuah Maze.....	6
2.2.1.1 Algoritma Sidewinder.....	8
2.2.1.2 Algoritma Binarytree.....	14
2.3 Maze Solving.....	16
2.3.1 Algoritma Recursive Backtracker.....	16
BAB 3	20
ANALISIS DAN PERANCANGAN SISTEM	20
3.1 Analisis Kebutuhan Sistem.....	20
3.1.1 Kebutuhan Fungsional.....	20
3.1.2 Kebutuhan Non Fungsional.....	20
3.2 Rancangan Kerja Sistem.....	21

3.2.1 Diagram Alir(Flowchart)	21
3.2.2 Diagram Alir Utama Sistem	21
3.2.3 Diagram Alir Algoritma Sidewinder	22
3.3 Rancangan Algoritma Sistem	25
3.3.1 Perancangan Input.....	25
3.3.2 Penerapan Algoritma	25
3.4 Rancangan Antar Muka Sistem	26
3.4.1 Tampilan Awal	26
3.4.2 Tampilan Generate Maze.....	27
3.4.3 Tampilan Acak Pintu keluar	28
3.4.4 Tampilan Solve Maze	28
3.5 Perancangan Pengujian sistem	29
BAB 4	31
IMPLEMENTASI DAN ANALISIS SISTEM.....	31
4.1 Implementasi Sistem	31
4.1.1 Antarmuka sistem	31
4.1.2 Tampilan Halaman Maze Generate.....	32
4.1.3 Tampilan Halaman Solver	33
4.1.4 Tampilan Halaman Tree	34
4.2 Analisis Sistem	35
4.2.1 Data Percobaan	35
4.2.2 Analisis Waktu.....	40
4.2.3 Analisis Traversed Nodes	40
4.2.4 Analisis Path Nodes	43
4.2.5 Analisis Perfect maze.....	44
BAB 5	45
KESIMPULAN	45
5.1 Kesimpulan.....	45
5.2 Saran	45
DAFTAR PUSTAKA	46
LAMPIRAN.....	47

DAFTAR GAMBAR

Gambar 2.1: Ilustrasi Struktur <i>Maze</i>	6
Gambar 2.2: <i>Maze</i> Tidak Sempurna	7
Gambar 2.3: <i>Maze</i> Sempurna	7
Gambar 2.4 : <i>Step 1 Sidewinder Algorithm</i>	8
Gambar 2.5 : <i>Step 2 Sidewinder Algorithm</i>	8
Gambar 2.6 : <i>Step 3 Sidewinder Algorithm</i>	9
Gambar 2.7 : <i>Step 4 Sidewinder Algorithm</i>	9
Gambar 2.8 : <i>Step 5 Sidewinder Algorithm</i>	9
Gambar 2.9 : <i>Step 6 Sidewinder Algorithm</i>	10
Gambar 2.10 : <i>Step 7 Sidewinder Algorithm</i>	10
Gambar 2.11 : <i>Step 8 Sidewinder Algorithm</i>	10
Gambar 2.12 : <i>Step 9 Sidewinder Algorithm</i>	11
Gambar 2.13 : <i>Step 10 Sidewinder Algorithm</i>	11
Gambar 2.14 : <i>Step 11 Sidewinder Algorithm</i>	11
Gambar 2.15 : <i>Step 12 Sidewinder Algorithm</i>	12
Gambar 2.16 : <i>Step 13 Sidewinder Algorithm</i>	12
Gambar 2.17 : <i>Step 14 Sidewinder Algorithm</i>	12
Gambar 2.18 : <i>Step 15 Sidewinder Algorithm</i>	13
Gambar 2.19 : <i>Step 16 Sidewinder Algorithm</i>	13
Gambar 2.20 : <i>Step 17 Sidewinder Algorithm</i>	13
Gambar 2.21 : <i>Pseudocode Recursive Backtacker</i>	14
Gambar 2.22 : Contoh Pohon Ruang Status	14
Gambar 2.23 : Contoh <i>Maze Solver</i>	15
Gambar 2.24 : Contoh <i>Maze Solver</i>	15
Gambar 2.25 : Contoh <i>Maze Solver</i>	15
Gambar 2.26 : Contoh <i>Maze Solver</i>	16
Gambar 2.27 : Contoh <i>Maze Solver</i>	17
Gambar 2.28 : Contoh <i>Maze Solver</i>	18

Gambar 2.29 : Contoh <i>Maze Solver</i>	19
Gambar 3.1 : Alir Utama Sistem.....	21
Gambar 3.2 : Diagram Alir Algoritma Sidewinder.....	22
Gambar 3.3 : Diagram Alir Algoritma <i>Binarytre</i>	23
Gambar 3.4 : Diagram Alir <i>Binarytree</i>	24
Gambar 3.5 : Tampilan Halaman Utama	26
Gambar 3.6 : Tampilan Halaman Pembuatan Maze	27
Gambar 3.7 : Tampilan Hasil <i>Generate maze</i>	27
Gambar 3.8 : Tampilan Acak Pintu Keluar.....	28
Gambar 3.9 : Tampilan Proses <i>solve</i>	28
Gambar 3.10 : Hasil Pencarian Jalan Keluar	29
Gambar 4.1 : Tampilan Sistem	31
Gambar 4.2 : Menu	32
Gambar 4.3 : Proses <i>Generate Maze</i>	33
Gambar 4.4 : Proses Tampilan <i>Solver</i>	33
Gambar 4.5 : Tampilan <i>Solver</i>	34
Gambar 4.6 : Hasil <i>Print Maze</i>	34
Gambar 4.7 : Contoh Ukuran <i>Maze</i> yang diukur	35

DAFTAR TABEL

Tabel 4.1 : Tabel Hasil Analisis Waktu <i>Generate Maze</i> Algoritma <i>Sidewinder</i> ..	35
Tabel 4.2 : Tabel Hasil Analisis Waktu <i>Generate Maze</i> Algoritma <i>BinaryTree</i> ..	36
Tabel 4.3 : Tabel <i>Maze</i> Ukuran Persegi <i>Sidewinder</i>	36
Tabel 4.4 : Tabel <i>Maze</i> Persegi <i>Binarytree</i>	37
Tabel 4.5 : Tabel <i>Maze</i> Persegi Panjang <i>Vertical Sidewinder</i>	37
Tabel 4.3 : Tabel <i>Maze</i> Persegi Panjang <i>Vertical Binarytree</i>	38
Tabel 4.4 : Tabel <i>Maze</i> Persegi Panjang <i>Horizontal Sidewinder</i>	38
Tabel 4.5 : Tabel <i>Maze</i> Persegi Panjang <i>Horizontal Binarytree</i>	39

©UKDW

DAFTAR GRAFIK

Grafik 4.1 : Hasil Analisis Perbandingan Waktu <i>Generate maze Sidewinder</i> dan <i>Binarytree</i>	40
Grafik 4.2 : Hasil Analisis Perbandingan Rata-Rata <i>Traversed Nodes Maze</i> Persegi.....	41
Grafik 4.3 : Hasil Analisis Perbandingan Rata-Rata <i>Traversed Node</i> Persegi Panjang Vertical.....	42
Grafik 4.4 : Hasil Analisis Perbandingan Rata-Rata <i>Traversed Nodes</i> Persegi Panjang Horizontal.....	42
Grafik 4.5 : Hasil Analisis Perbandingan Rata-Rata Path Node Persegi Panjang Vertical.....	43
Grafik 4.6 : Hasil Analisis Perbandingan Rata-Rata Path Node Persegi Panjang Horizontal.....	44

INTISARI

IMPLEMENTASI ALGORITMA SIDEWINDER SEBAGAI GENERATOR MAZE

Salah satu aspek dalam sebuah game yang mampu memberikan tingkat kesulitan berbeda-beda adalah desain *Map*. *Gameplay* sebuah *Game* yang bergantung pada desain *Map* adalah game bertema *Maze*. Untuk membuat *Maze* itu mudah jika dibuat secara manual, namun untuk membuat *Maze* secara cepat, acak dan otomatis dibutuhkan tambahan algoritma tertentu yang memang digunakan sebagai pembuat *Maze*.

Dalam penelitian ini, penulis membuat sebuah sistem *Generator Maze* dengan mengimplementasikan algoritma *Sidewinder* dan *Binarytree* sebagai pembuat *Maze*. Penulis menganalisis *Traversed Nodes* dan *Path Nodes* pada *Maze* dari hasil *Solver* menggunakan algoritma *Recursive Backtracker*. Dari *Nodes* tersebut *Maze* yang dibuat dengan algoritma *Sidewinder* dapat diuji kompleksitasnya dan kecepatan dari masing-masing algoritma.

Sistem yang dibangun 100% mampu menghasilkan *Maze* secara dinamis, algoritma *Sidwinder* mampu *generate maze* relatif lebih cepat dari algoritma *Binarytree*. Algoritma *Sidewinder* menghasilkan *maze* yang lebih rumit daripada Algoritma *Binarytree* dengan aturan potong *NW(North-West)*. Algoritma *Recursive Backtracker* mampu mencari jalan keluar dari *maze* yang telah dibuat.

Kata Kunci : *Maze*, *Solver*, Algoritma *Sidewinder*, Algoritma *Binarytree*, Algoritma *Recursive Backtracker*

INTISARI

IMPLEMENTASI ALGORITMA SIDEWINDER SEBAGAI GENERATOR MAZE

Salah satu aspek dalam sebuah game yang mampu memberikan tingkat kesulitan berbeda-beda adalah desain *Map*. *Gameplay* sebuah *Game* yang bergantung pada desain *Map* adalah game bertema *Maze*. Untuk membuat *Maze* itu mudah jika dibuat secara manual, namun untuk membuat *Maze* secara cepat, acak dan otomatis dibutuhkan tambahan algoritma tertentu yang memang digunakan sebagai pembuat *Maze*.

Dalam penelitian ini, penulis membuat sebuah sistem *Generator Maze* dengan mengimplementasikan algoritma *Sidewinder* dan *Binarytree* sebagai pembuat *Maze*. Penulis menganalisis *Traversed Nodes* dan *Path Nodes* pada *Maze* dari hasil *Solver* menggunakan algoritma *Recursive Backtracker*. Dari *Nodes* tersebut *Maze* yang dibuat dengan algoritma *Sidewinder* dapat diuji kompleksitasnya dan kecepatan dari masing-masing algoritma.

Sistem yang dibangun 100% mampu menghasilkan *Maze* secara dinamis, algoritma *Sidwinder* mampu *generate maze* relatif lebih cepat dari algoritma *Binarytree*. Algoritma *Sidewinder* menghasilkan *maze* yang lebih rumit daripada Algoritma *Binarytree* dengan aturan potong *NW(North-West)*. Algoritma *Recursive Backtracker* mampu mencari jalan keluar dari *maze* yang telah dibuat.

Kata Kunci : *Maze*, *Solver*, Algoritma *Sidewinder*, Algoritma *Binarytree*, Algoritma *Recursive Backtracker*

BAB 1

PENDAHULUAN

1.1 Latar Belakang Masalah

Perkembangan Game pada dunia IT semakin pesat hal ini dibuktikan dengan bermunculannya game-game baru. Banyak *genre* atau jenis game yang sudah beredar dipasaran seperti game aksi(*action*), teka-teki(*puzzle*), strategi(*strategy*), dll. Semua itu bisa didapatkan dengan membayar atau gratis. Banyak pula industry game baru yang bermunculan sehingga menghasilkan *game-game* yang baru dengan keunikan mereka sendiri. Dengan begitu peminat *game* juga semakin bertambah jumlahnya. Bahkan industry *game* dapat meraut keuntungan yang sangat besar melalui *game* yang mereka *release* apalagi perkembangan *gadget* jugasemakin maju. Semua industry *game* ini bersaing dalam membuat *game* yang lebih baik dibanding industry game lainnya, baik dari sisi *gameplay* sampai harga yang ditawarkan, semuanya demi menarik lebih banyak *gamer* untuk memainkan *game* buatan industry tersebut.

Game bergenre *puzzle* merupakan *game* yang mengasah otak pemainnya sehingga banyak *gamer* yang tertarik dengan *game* dengan *genre* ini. Salah satu jenis *game* bergenre *puzzle* adalah *game* labirin, tantangan pada *game* labirin adalah labirin itu sendiri. Semakin sulit sebuah *map* labirin maka semakin sulit mencari jalan keluarnya.

Dalam membuat *map* labirin bukanlah hal yang susah untuk dibuat tetapi kesulitan terjadi saat membuat beberapa *map* yang berbeda pada setiap *stage* pada *game* tersebut. *Programmer* sering merasa kesulitan ketika membuat *map* yang berbeda-beda untuk setiap *stage*, karena harus menata ulang *map* dan membuat secara manual.

Ada beberapa metode untuk pembuatan *maze* secara acak yang bisa *programmer* gunakan, tetapi setiap metode yang ada pasti memiliki beberapa kelemahan dan kelebihan.

Metode *Binary Tree* merupakan metode pembuatan labirin yang dapat membuat labirin sempurna hanya dengan melihat satu kolom pada satu waktu. *Binary Tree* akan memotong atau melewati kolom yang sedang aktif secara acak sesuai arah atau aturan yang sudah ditentukan diawal.

Metode *Sidewinder* merupakan metode yang lebih kompleks jika dibandingkan dengan metode *Sidewinder*. *Sidewinder* akan melewati atau memotong kolom sebelah kanan dari kolom yang sedang aktif secara acak. Dari permasalahan tersebut, penulis ingin membandingkan metode *Binary Tree* dan *sidewinder* dalam pembuatan labirin. Dengan membandingkan kedua metode tersebut penulis ingin mengetahui metode mana yang akan menyelesaikan labirin dengan waktu yang lebih singkat dan tepat.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah diatas, rumusan masalah dalam penelitian ini adalah:

Bagaimana hasil perbandingan algoritma *Sidewinder* dan *Binary Tree* diukur dengan kecepatan pembuatan dan kemampuan *generate maze* dari start sampai *finish*?

1.3 Batasan Masalah

Adapun batasan sistem yang akan dibuat dalam penelitian ini adalah sebagai berikut :

- a. Ukuran maksimal map yang diuji adalah 30 x30.
- b. Hanya *generate Maze* dengan 2 metode *Sidewinder* dan *Binary Tree*.
- c. Metode *Binary Tree* hanya menggunakan aturan NW(North-West).
- d. Hanya terdapat 1 titik awal dan 1 titik akhir.
- e. Titik mulai berada di pojok kiri atas.
- f. Titik akhir berada di pojok kanan bawah.

1.4 Tujuan Penelitian

Tujuan penelitian yang ingin dicapai oleh peneliti adalah sebagai berikut : Mengetahui metode mana yang paling cepat dan tepat menghasilkan *Maze* antara metode *sidewinder* dan *Binary Tree* dengan *Recursive Backtracker* sebagai *solver*nya.

1.5 Metode Penelitian

Metode penelitian yang akan digunakan adalah :

a. Studi Pustaka

Pada tahap ini dilakukan untuk memperoleh informasi dengan memanfaatkan berbagai macam sumber pustaka melalui buku, artikel, jurnal ilmiah, dan sumber lain yang berkaitan dengan *maze*, algoritma *Sidewinder* , algoritma *Binary Tree* dan algoritma *Recursive Backtracker*.

b. Pembuatan Sistem

Pada tahap ini berisi perancangan antarmuka sistem dan mengimplementasikan algoritma algoritma *Sidewinder*, algoritma *Binary Tree* dan algoritma *Recursive Backtracker* ke dalam bahasa program. Bahasa pemrograman yang digunakan adalah HTML5 dan javascript.

c. Pengujian Sistem

Pada tahap ini dilakukan pengujian kedua algoritma yang telah digunakan dalam penelitian ini. Algoritma tidak akan diuji sekaligus, melainkan diuji satu persatu.

Pengujian dilakukan dengan cara melakukan 10 kali percobaan dalam *generate maze* . Jadi tiap kali *generate maze* dan *solver maze* dijalankan mendapatkan 2 buah data yaitu *traveled nodes* dan *path nodes*.

d. Analisis Hasil Percobaan

Tahapan ini berisi tentang analisis pengujian sistem yang telah dilakukan sebelumnya. Hasil akhir analisis menampilkan presentase

keberhasilan *algoritma Sidewinder* dan *Binary tree* sebagai *generator maze* dan *algoritma Recursive Backtracker* sebagai *solver maze*.

1.6 Sistematika Penulisan

Sistematika penulisan laporan Tugas Akhir ini dikelompokkan menjadi 5 bab, yaitu :

Bab 1, Pendahuluan. Bab ini berisi gambaran umum tentang penelitian yang dilakukan, dimana mencakup antara lain latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, metode penelitian, dan sistematika penulisan penelitian.

Bab 2, Landasan Teori. Bab ini berisi tentang tinjauan pustaka dan landasan teori yang digunakan dalam penelitian ini. Tinjauan pustaka menguraikan berbagai teori yang didapat dari berbagai sumber terkait dengan penelitian ini.

Bab 3, Perancangan Sistem. Bab ini berisi tentang rancangan sistem yang dibangun dalam penelitian ini. Rancangan sistem yang dibuat berupa spesifikasi dari sistem, rancangan antarmuka sistem berupa input dan output.

Bab 4, Implementasi Sistem. Bab ini berisi tentang implementasi sistem yang telah dirancang sebagaimana sudah dirancang pada bab III.

Bab 5, Kesimpulan dan Saran. Bab ini berisi tentang kesimpulan hasil penelitian yang telah dilakukan dan saran yang mungkin dapat dilakukan untuk pengembangan penelitian selanjutnya.

BAB 5 KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan analisis sistem, maka dapat disimpulkan sebagai berikut :

- a. Algoritma *Sidewinder* dan *Binarytree* terbukti 100% mampu menghasilkan sebuah *perfect maze*.
- b. Algoritma *Recursive Backtracker* 100% akurat mencari jalan keluar dari sebuah maze yang digenerate menggunakan Algoritma *Sidewinder* dan Algoritma *Binarytree*.
- c. Algoritma *Sidewinder* memiliki catatan waktu yang lebih cepat dibandingkan dengan Algoritma *Binarytree* dalam mengenerate *maze*. Algoritma *Binarytree* unggul dalam jumlah *Traversed node* dan *path note* yang lebih sedikit dibanding Algoritma *Sidewinder* sehingga membentuk *maze* yang tidak rumit.

5.2 Saran

Saran yang diberikan penulis untuk pengembangan sistem selanjutnya adalah sebagai berikut :

- a. Kedua algoritma ini (*Sidewinder* dan *Binarytree*) dapat diimplementasikan secara langsung dalam sebuah *game* sebagai pembuat *map* secara dinamis.
- b. Dapat dilakukan penelitian lebih lanjut menggunakan algoritma *Recursive Backtracker* untuk mencari jalan keluar pada kategori *imperfect maze*.

DAFTAR PUSTAKA

- Buck, J. (2011). *Maze Generation: Sidewinder algorithm*. From The Buckblog assorted ramblings by Jamis Buck: <http://weblog.jamisbuck.org/2011/2/3/maze-generation-sidewinder-algorithm#>
- Buck, J. (2015). *Mazes for Programmers*. The Pragmatic Programmers.
- Kristiadi, V. N. (2015). PERBANDINGAN ALGORITMA "GROWING TREE" DAN "HUNT AND KILL" PADA PEMBUATAN RANDOM MAP MAZE.
- Lee , H. L., Lee , C. F., & Chen , L. H. (2010, August 3). A perfect maze based steganographic method. *The Journal of Systems and Software*, 2528-2535.
- Prayoga, H. (2014). Implementasi Algoritma Backtracking Pada Game Puzzle Kakuro.
- Pribadi , O. (2015). Maze Generator Dengan Menggunakan Algoritma Depth-First-Search. *Jurnal TIMES*, IV(1).
- Sasongko, A. C. (2008). Penerapan Metode Recursive Backtracker Sebagai Creator Dan Solver Dalam Game Maze.
- Sebastio A. (2016). *Implementasi Algoritma Sidewinder Sebagai Generator Maze*. Tugas Akhir. Fakultas Teknik Informatika UKDW: Yogyakarta.
- Sukumar , T., & Santha , D. (2012, July). Maze Based Data Hiding Using Back Tracker Algorithm. *International Journal of Engineering Research and Applications*, II(4), 499-504.
- Tando , A. (2012). Perbandingan Algoritma Depth-First Search dan Algoritma Hunt-and-Kill dalam Pembuatan Labirin.