Penerapan Algoritma A* dalam Pemilihan Rute Jalan Terpendek pada Permainan Stardew Valley

M. Rafli Al Khadafi - 13516056
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516056@std.stei.itb.ac.id

***Abstrak*—Permainan Stardew Valley memerlukan manajemen dan efisiensi waktu yang sangat baik agar dapat menjalankan pekerjaan perkebunan pada permainan dengan optimal. Salah satu cara memanajemen waktu adalah dengan memilih rute jalan yang paling pendek agar waktu yang digunakan untuk berjalan untuk sampai ke tujuan tidak terbuang sia-sia. Pencarian rute terpendek tersebut dalam hal ini akan menggunakan algoritma A*. Dengan menggunakan algoritma A* kita dapat melakukan *path planning* untuk dapat merencanakan jalur yang akan digunakan untuk sampai ke tujuan dengan efisien.**

Kata Kunci—Stardew Valley; A*; Rute; Terpendek;

I. PENDAHULUAN

Semakin berkembangnya ilmu pengetahuan dan teknologi terutama di bidang ilmu komputer/informatika, semakin berkembang pula industri-industri yang bergerak di bidang digital tersebut termasuk pula di dalamnya industri video game.

Teknologi Game pertama kali diciptakan oleh A.S. Douglas tahun 1952 di Universty of Cambridge yaitu OXO untuk mendemonstrasikan tesisnya mengenai interaksi antara komputer dan manusia. Kemudian Douglas berkreasi lagi dengan menciptakan game versi Tic-Tac-Toe yang diprogram pada komputer *EDSAC vaccum-tube* yang memiliki layar CRT (Cathode Ray Tube).

Gambar 1. Permainan Tic-Tac-Toe

Kemudian William Higinbotham menciptakan game Tennis for Two pada tahun 1958 yang dimainkan di osiloskop. Game sederhana yang menampilkan lapangan tennis dari samping ini memperlihatkan seolah bola dipengaruhi oleh gravitasi dan harus melewati net.

Pada tahun 1961 dimana komputer merupakan barang yang bisa dikatakan sangat mewah, Steve Russel membuat game bernama Spacewar karena ketertarikannya dengan kisah fiksi ilmiah karangan Edward E Smith yang berjudul Skylark. Memanfaatkan pekerjaannya yang menggunakan komputer mainframe MIT PDP-1 yang biasa dipakai untuk perhitungan statistik, Steve membuat Spacewar.

Video game pun seiring waktu mengalami perkembangan yang cukup pesat dimulai pada tahun 2000an ditandai dengan munculnya teknologi konsol 32 bit dan sampai sekarang video game pun dapat dinikmati di komputer pribadi sampai laptop.

Salah satu video game *indie* yang memikat hati para gamer ialah Stardew Valley. Terinspirasi dari permainan Harvest Moon yang rilis pada tahun 90an, Stardew Valley merupakan permainan simulasi berkebun yang dikembangkan oleh Eric Barone.

Tujuan dari game Stardew Valley ini adalah untuk mengatur perkebunan serta menjalin hubungan sosial dengan warga desa lainnya, menjalankan tugas yang diberikan oleh walikota ataupun oleh warga desa lainnya. Tantangannya adalah waktu dalam sehari permainan adalah 24 jam. Namun satu jam dalam permainan dapat dihabiskan dalam waktu 42 detik waktu sebenarnya [3]. Pemanfaatan waktu yang baik dapat dilakukan dengan mengatur rute perjalanan yang optimal sehingga waktu yang digunakan lebih efisien.

Masalah pemanfaatan mengatur rute ini dapat diselesaikan dengan algoritma A*

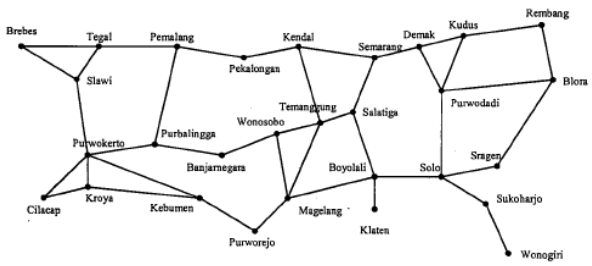
II. DASAR TEORI

A. *Graf*

1) *Pendahuluan*

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek dinyatakan sebagai noktah, bulatan, atau titik, sedangkan hubungan antara objek dinyatakan dengan garis.

Gambar di bawah ini sebuah graf yang menyatakan peta jaringan jalan raya yang menghubungkan sejumlah kota di Provinsi Jawa Tengah.



Gambar 1. Jaringan jalan raya di provinsi Jawa Tengah

2) Definisi Graf

Graf G didefinisikan sebagai pasangan himpunan (V, E), ditulis dengan notasi G = (V, E) yang dalam hal ini V adalah himpunan tidak-kosong dari simpul-simpul (vertices atau node) dan E adalah himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul.

3) Jenis-jenis Graf

Graf dapat dikelompokkan menjadi beberapa kategori bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

a) Berdasarkan ada tidaknya gelang atau sisi ganda

1. Graf sederhana

   Graf sederhana adalah graf yang tidak memilikis sisi gelang atau sisi ganda

2. Graf tak-sederhana

   Graf tak-sederhana adalah graf yang memiliki gelang atau sisi ganda. Graf yang hanya memiliki sisi ganda disebut graf ganda dan graf yang memiliki sisi ganda maupun sisi gelang dinamakan graf semu.

b) Berdasarkan jumlah simpul

1. Graf berhingga

   Graf berhingga adalah graf yang memiliki simpul dengan jumlah berhingga.

2. Graf tak-berhingga

   Graf tak-berhingga adalah graf yang memiliki jumlah simpul yang tak berhingga banyaknya.

c) Berdasarkan orientasi arah

1. Graf berarah

   Graf berarah adalah graf yang sisinya diberikan orientasi arah menuju atau menjadi suatu simpul tertentu sehingga (u, v) $\neq$ (v, u).

2. Graf tak-berarah

   Graf tak-berarah adalah graf yang sisinya tidak memiliki orientasi arah tertentu sehingga (u, v) = (v, u).

4) Terminologi Dasar Graf

Di bawah ini didefinisikan beberapa terminologi yang sering dipakai.

1. Adjacent (bertetangga)

   Dua buah simpul pada graf tak-berarah dinyatakan bertetangga apabila kedua simpul tersebut dihubungkan oleh sisi.

2. Incident (Bersisian)

   Untuk sembarang sisi E = (u, v), sisi E dikatakan bersisian dengan simpul u dan v.

3. Isolated Vertex (Simpul Terpencil)

   Simpul terpencil ialah simpul yang tidak mempunyai sisi yang bersisian dengannya. Dapat juga dinyatakan bahwa simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul lainnya.

4. Graf Kosong (Null Graph atau Empty Graph)

   Graf kosong adalah graf yang himpunan sisinya merupakan himpunan kosong. Ditulis sebagai N_n dengan n adalah jumlah simpul.

5. Derajat (Degree)

   Derajat suatu simpul pada graf tak-berarah adalah jumlah sisi yang bersisian dengan simpul tersebut.

6. Lintasan (Path)

   Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G dengan melewati berbagai sisi dan simpul secara bergantian.

7. Siklus (Cycle) atau Sirkuit (Circuit)

   Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus.

8. Terhubung (Connected)

   Suatu graf tak-berarah merupakan graf terhubung jika untuk setiap simpul pada graf tersebut, terdapat lintasan yang menuju simpul tersebut.

9. Upagraf (Subgraph)

   Upagraf adalah bagian dari suatu graf G atau dapat disebut pula upagraf merupakan subset dari suatu graf.

10. Upagraf Merentang (Spanning Subgraph)

    Suatu upagraf disebut sebagai upagraf merentang apabila pada upagraf tersebut terdapat semua simpul graf utama.

11. Cut-Set

    Cut-set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung. Jadi, cut-set selalu menghasilkan dua buah komponen terhubung

12. Graf Berbobot (Weighted Graph)

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot)

B. *Algoritma A**

Algoritma A* merupakan sebuah algoritma komputer yang dapat digunakan untuk menentukan total minimum biaya lintasan dan juga saat kondisi yang tepat dapat memberikan solusi yang optimal. Dideskripsikan pertama kali oleh Peter Hart, Nils Nilsson, dan Bertram Raphael dari Stanford Research Institute [5]. Cara kerja dari algoritma ini hampir sama dengan algoritma Best First Search (BFS), akan tetapi dimodifikasi dengan fungsi heuristik [4].

1) *Fungsi Evaluasi Algoritma A**

Untuk fingsi evaluasinya dapat dituliskan sebagai berikut:

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

Keterangan:

f(n) : Perkiraan total biaya untuk sampai ke tujuan melalui n

g(n) : Total biaya untuk sampai n

h(n) : Perkiraan biaya untuk mencapai tujuan dari n

2) *Pseudocode Algoritma A**

Berikut merupakan pseudocode yang mendeskripsikan cara kerja algoritma A*.

```
function A*(start, goal)
    // The set of nodes already evaluated
    closedSet := {}

    // The set of currently discovered nodes that are not
evaluated yet.
    // Initially, only the start node is known.
    openSet := {start}

    // For each node, which node it can most efficiently be
reached from.
    // If a node can be reached from many nodes, cameFrom
will eventually contain the
    // most efficient previous step.
    cameFrom := an empty map

    // For each node, the cost of getting from the start
node to that node.
    gScore := map with default value of Infinity

    // The cost of going from start to start is zero.
    gScore[start] := 0

    // For each node, the total cost of getting from the
start node to the goal
    // by passing by that node. That value is partly known,
partly heuristic.
    fScore := map with default value of Infinity

    // For the first node, that value is completely
heuristic.
    fScore[start] := heuristic_cost_estimate(start, goal)

    while openSet is not empty
        current := the node in openSet having the lowest
fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        closedSet.Add(current)

        for each neighbor of current
            if neighbor in closedSet
                continue                    // Ignore the
neighbor which is already evaluated.

            if neighbor not in openSet      // Discover a
new node
                openSet.Add(neighbor)

            // The distance from start to a neighbor
            //the "dist_between" function may vary as per
the solution requirements.
            tentative_gScore := gScore[current] +
dist_between(current, neighbor)
            if tentative_gScore >= gScore[neighbor]
                continue                    // This is not a
better path.

            // This path is the best until now. Record it!
            cameFrom[neighbor] := current
            gScore[neighbor] := tentative_gScore
            fScore[neighbor] := gScore[neighbor] +
heuristic_cost_estimate(neighbor, goal)

    return failure

function reconstruct_path(cameFrom, current)
    total_path := [current]
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.append(current)
    return total_path
```

C. *Stardew Valley*

Stardew Valley merupakan permainan video simulasi berkebun bergenre indie yang dikembangkan oleh Eric Barone dan dikeluarkan oleh Chucklefish. Permainan ini dirilis untuk Windows pada Februari 2016, dan dirilis untuk OS X, Linux, PlayStation 4, dan Xbox One kemudian pada tahun yang sama.

Stardew Valley mendapatkan banyak kritik positif, dan merupakan salah satu permainan dengan penjualan terbaik di Steam pada beberapa bulan pertama setelah rilis. Pada akhir 2017 permainan ini terjual hingga 3.5 juta salinan lewat semua platform.

Gambar 2. Permainan Stardew Vallew

Pada permainan Stardew Valley sang tokoh utama diwariskan sebuah kebun oleh sang kakek. Sang tokoh utama pun harus pindah ke sebuah desa dan harus mengembangkan kebun sang kakek dan bersosialisasi dengan warga desa serta

melaksanakan tugas-tugas yang ada. Ada banyak rumah warga, toko, klinik, pertambangan, kebun, dan lain-lain yang bisa dikunjungi oleh tokoh utama dengan berbagai jalur yang dapat dilalui.

Gambar 3. Peta Permainan Stardew Valley

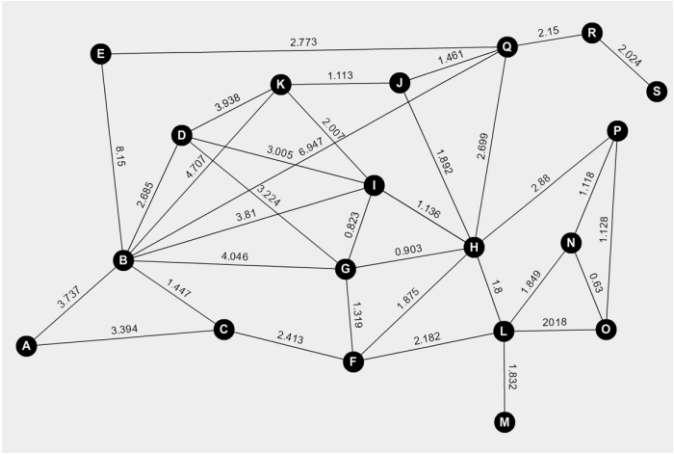
III. Penerapan Algoritma A* dalam Penentuan Rute Terpendek pada Permainan Stardew Valley

Seperti yang sudah dijelaskan, manajemen waktu merupakan salah satu aspek terpenting dalam memainkan permainan Stardew Valley dengan optimal. Oleh karena itu, pemain harus mengatur waktu dengan sangat efisien yaitu salah satunya dengan mengatur agar setiap perjalanan yang ditempuh ke suatu tempat merupakan rute yang paling cepat agar waktu tidak terbuang sia-sia untuk menempuh jalan yang cukup jauh.

Persoalan ini disebut sebagai shortest path problem yang dapat diselesaikan salah satunya dengan algoritma A*. Pertama-tama kita akan membentuk peta permainan di atas menjadi bentuk graf.

Gambar 4. Peta Permainan Stardew Valley dengan Rute berbentuk Graf

Jika kita tulis ulang graf pada peta tersebut sambal dihitung jarak untuk masing-masing sisi dapat menjadi seperti ini.



Gambar 5. Peta Permainan Stardew Valley yang Berbentuk Graf

Dengan keterangan pada graf sebagai berikut:

Sym	Keterangan	Sym	Keterangan
A	Menara Penyihir	K	Tempat bermain anak
B	Kebun	L	Rumah Walikota
C	Peternakan Marnie	M	Pantai
D	Pemberhentian Bus	N	Rumah Pandai Besi
E	Pemandian air panas	O	Museum
F	Rumah Jodie, Kent, dan Sam	P	JojaMart
G	Alun-alun	Q	Rumah Tukang Kayu
H	Saloon	R	Gua
I	Toserba	S	Penggalian
J	Pusat Komunitas		

Heuristik yang akan digunakan adalah jarak setiap simpul ditarik garis lurus ke tujuan pada peta asli permainan. Contohnya heuristik dari Tempat bermain anak ke Menara Penyihir setelah ditarik garis lurus adalah 6,255.

Gambar 6. Contoh Heuristik berupa garis lurus yang ditarik dari suatu simpul ke tujuan

Sehingga kita mendapatkan fungsi evaluasi

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

Keterangan:

f(n) : Perkiraan total jarak untuk sampai ke tujuan melalui n

g(n) : Total jarak yang ditempuh untuk sampai n

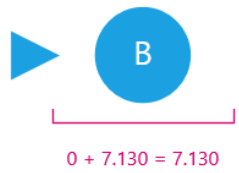
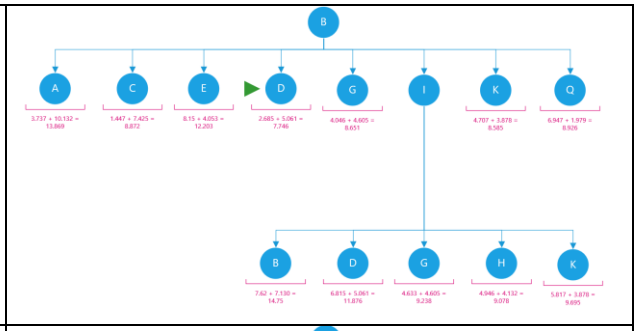
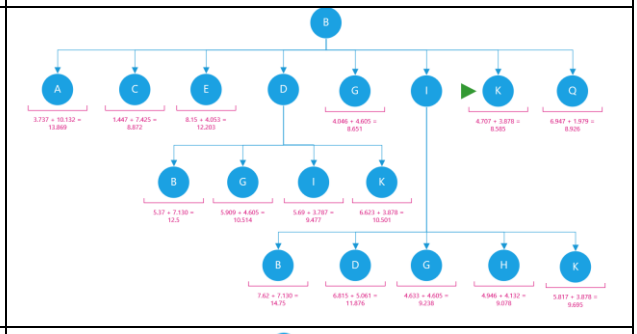
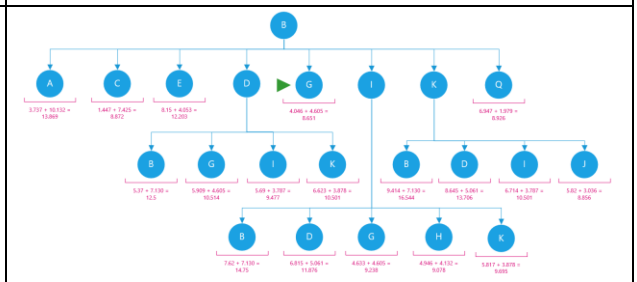
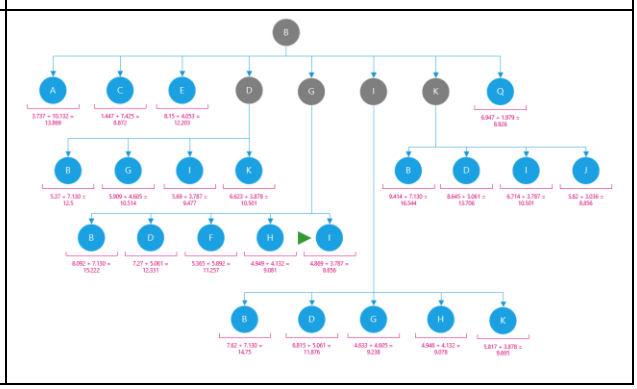
h(n) : Perkiraan jarak untuk mencapai tujuan dari n (heur)

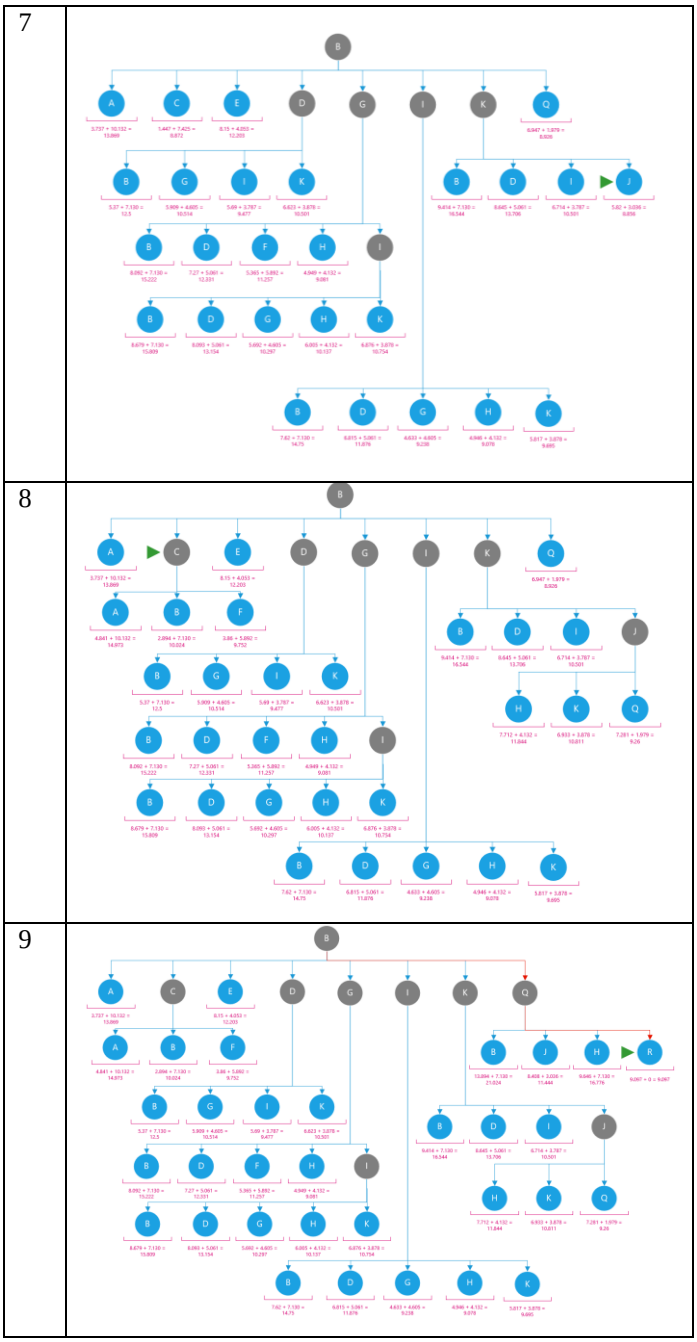
*1) Contoh pencarian A**

Jika kita ingin mencari rute tercepat untuk menuju Gua dari Kebun, dengan diketahui heuristik sebagai jarak garis lurus setiap lokasi menuju gua adalah sebagai berikut:

Lokasi	Jarak	Lokasi	Jarak
Menara Penyihir	10.132	Tempat bermain anak	3.878
Kebun	7.130	Rumah Walikota	4.624
Peternakan Marnie	7.425	Pantai	6.202
Pemberhentian Bus	5.061	Rumah Pandai Besi	4.107
Pemandian air panas	4.053	Museum	4.524
Rumah Jodie, Kent, dan Sam	5.892	JojaMart	3.096
Alun-alun	4.605	Rumah Tukang Kayu	1.979
Saloon	4.132	Gua	0
Toserba	3.787	Penggalian	1.877
Pusat Komunitas	3.036		

Pohon pencarian yang akan dihasilkan dengan algoritma A* adalah sebagai berikut.

No	Pohon pencarian yang dihasilkan
1	
2	
3	
4	
5	
6	



Dari pohon pencarian yang telah dibangkitkan didapatkan rute tercepat untuk mencapai Gua dari Kebun menggunakan algoritma A* adalah B – Q – R atau Kebun – Rumah Tukang Kayu – Gua. Dalam peta permainan rute yang didapatkan seperti ini:

Gambar 7. Rute Terpendek dari Kebun ke Gua ditandai dengan warna merah

IV. KESIMPULAN

Algoritma A* dapat digunakan untuk menyelesaikan berbagai persoalan terutama dalam mencari jalur terpendek (*shortest path*). Penerapan algoritma A* ini dapat kita gunakan pada permainan video, salah satunya adalah pada permainan Stardew Valley. Dengan menggunakan A* kita dapat mengetahui jarak terpendek dari suatu lokasi ke lokasi lain. Dengan memanfaatkan fungsi heuristik, dapat dipastikan bahwa solusi yang didapatkan merupakan solusi optimal dengan kasus uji yang dicantumkan berupa rute dari Kebun ke Gua.

REFERENCES

[1] Arifa Nida, Menelusuri Sejarah dan Perkembangan Teknologi Game di Dunia. [Online] tersedia dalam https://www.klikmania.net/sejarah-dan-perkembangan-teknologi-game/ Diakses pada tanggal 12 Mei 2018

[2] Rinaldi Munir, Route/ Path Planning Using A Star and UCS. 2016. [Online] Tersedia dalam http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf. Diakses pada tanggal 12 Mei 2018.

[3] Stardew Valley Community, Time. [Online] tersedia dalam http://stardewvalley.wikia.com/wiki/Time. Diakses pada tanggal 12 Mei 2018.

[4] Cia, Penjelasan Algoritma A* Beserta Contoh Kasus Algoritma A*. 2017. [Online] tersedia dalam http://www.clocariusedu.tk/2017/11/penjelasan-algoritma-A-star-beserta-contoh.html. diakses pada tanggal 12 Mei 2018

[5] Hart, P. E.; Nilsson, N. J.; Raphael, B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths. 1968. IEEE Transactions on Systems Science and Cybernetics SSC4.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Mei 2018

M Rafli Al Khadafi - 13516056