MEMBUAT PERGERAKAN NON-PLAYER CHARACTER (NPC) MENGGUNAKAN ALGORITMA DIJKSTRA

MAKING MOVEMENT OF NON-PLAYER CHARACTER (NPC) USING THE DIJKSTRA ALGORITHM

I Made Febi Nanda Mustika, Andrew Brian Osmond[2], Anton Siswo Raharjo ansori[3]
[1]Prodi S1 Teknik Komputer, Fakultas Teknik, Universitas Telkom
[2]Prodi S1 Teknik Komputer, Fakultas Teknik, Universitas Telkom
[3]Prodi S1 Teknik Komputer, Fakultas Teknik, Universitas Telkom
[1]**abosmond@student.telkomuniversity.ac.id**, [2]**raharjo@telkomuniversity.co.id**,
[3]**nandamust@student.telkomuniversity.ac.id**

Abstrak

Game* adalah sebuah sistem, dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan. *Game* bertujuan untuk menghibur, biasanya game banyak disukai oleh anak– anak hingga orang dewasa. Tugas Akhir ini berfokus pada pembuatan NPC dalam *game* yand dapat melakukan tugas-tugas tertentu seperti berjalan-jalan di dalam *game (wandering)* dan bergerak mengikuti perintah *(follow the leader)* dengan menggunakan metode Dijkstra *Shortest Path Finding.

Kata Kunci : Algoritma Dijkstra, *Game*. NPC.

Abstract

A game is a system, where players engage in artificial conflicts, here players interact with the system and conflicts in the game are engineered or artificial, in the game there are rules that aim to limit player behavior and determine the game. The game aims to entertain, usually games are favored by children to adults.This Final Project focuses on making the NPC in the game that is smart and able to perform certain tasks such as walking in the game (wandering) and moving to follow orders (follow the leader) using the Dijkstra Shortest Path Finding method.

Key Word* : *Dijkstra Algorithm, Game, NPC..

1. Pendahuluan

Seiring dengan perkembangan teknologi informasi perkembangan *game* begitu pesat mulai dengan jenis yang beragam, mulai dari yang hanya dapat dimainkan satu orang saja (*Single Player*) hingga *game* yang dapat dimainkan beberapa orang sekaligus (*Multi Player*). Klasifikasi *game* dilihat dari tema (*genre*) sangat banyak, diantaranya *Adventure*, *Arcade*, RPG (*Role Playing Game*), *Sports, Fighting plane*, dan masih banyak lainnya. *Game* saat ini sudah menjadi alternatif hiburan bagi tua, muda, pria, dan wanita.

Dalam *game*, kecerdasan buatan digunakan untuk membuat perilaku cerdas pada *non-player characters* (NPC), dan perilaku yang mensimulasikan kecerdasan manusia, seperti perilaku menyerang, menangkis, mendekati, atau menghindar. Banyak jenis metode yang bisa digunakan untuk membangun sistem kecerdasan buatan dalam sebuah *game*, salah satunya adalah algoritma Dijkstra.

Dijkstra merupakan salah satu algoritma yang popular dalam memecahkan persoalan terkait optimasi pencarian lintasan terpendek. Dijkstra juga dapat digunakan untuk memberi perilaku pada

NPC yaitu mencari jalur terpendek untuk mendekati karakter utama. Pada penelitian ini akan dibuat sistem pergerakan NPC menggunakan algoritma Dijkstra, harapannya algoritma dapat bekerja dengan optimal.

2. Dasar Teori

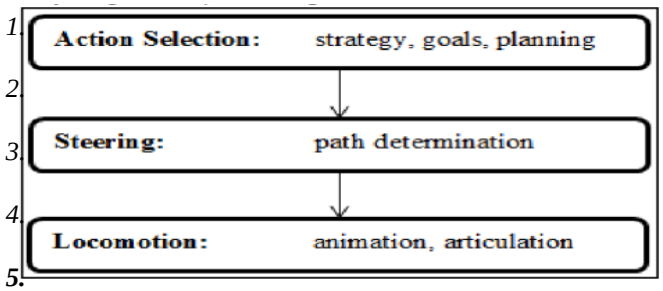
2.1 *Non Player Character (NPC)*

Non player character adalah karakter apa pun dalam game yang tidak dikendalikan oleh pemain, seperti monster, warga desa, dan hewan-hewan yang ada di hutan. NPC mewakili tokoh dalam cerita atau permainan dan memiliki kemampuan untuk mengimprovisasi tindakan mereka. Ini adalah sebuah kebalikan dari tokoh animasi dari sebuah film animasi yang tindakannnya di tulis dimuka, dan untuk "*avatar*" dalam sebuah permainan atau *virtual reality*, tindakan yang di arahkan pemain dilakukan secara *real time*. Dalam permainan, karakter otonom biasanya disebut NPC (*Non Player Character*)[5]. Perilaku *non player character* dibuat semirip mungkin dengan dunia nyata. Semakin banyak sifat yang dapat di lakukan karakter maka semakin banyak pula gerakan yang dapat dilakukan karakter[6].

Untuk memprediksi model dalam peta akan digunakan *self-organizing map*. Kombinasi algoritma Dijkstra akan mempengaruhi peta selanjutnya dan memungkinkan untuk menemukan jalur tependek antara dua posisi di peta berdasarkan nilai berat hambatan di area peta[7]. Raynold (1999) membagi perilaku NPC menjadi tiga lapisan yaitu :

- Seleksi Tindakan (*action selection*)
- Kendali (*steering*)
- Penggerak (*locomotion)*

Sesuai seperti yang ada di Gambar 2. 1 berikut:



Gambar 1. 1 Hirarki Gerak Perilaku

Keberadaan NPC sendiri dalam suatu *game* merupakan salah satu faktor dan komponen penting dalam permainan komputer modern yang dapat menentukan permainan tersebut menjadi menarik atau tidak.

Konsep agen cerdas merupakan salah satu model yang digunakan dalam membuat NPC. Sifat otonom dari agen cerdas merupakan keunggulan dalam memodelkan satu NPC *game.* Salah satu model atau kecerdasan buatan yang dapat digunakan dalam menentukan perilaku NPC yaitu *Fuzzy State Mechine. Non Player Character* mempunyai 4 jenis yang menjadi acuan, yaitu : NPC *Partner*, NPC *Enemy*, NPC *Quest*, NPC Pendukung Cerita[8]. Keempat NPC tersebut dapat berada dalam satu permainan tergantung jenis dan genre *game* yang dimainkan.

2.1 *Kecerdasan Buatan (Artificial Inteligence)*

Kecerdasan Buatan *(Artificial Intelligence)* merupakan salah satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia. Untuk mengetahui dan memodelkan proses–proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. Cerdas, berarti memiliki

pengetahuan ditambah pengalaman, penalaran (bagaimana membuat keputusan dan mengambil tindakan), moral yang baik[9].

Manusia cerdas (pandai) dalam menyelesaikan permasalahan karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki tentu akan lebih mampu menyelesaikan permasalahan. Tapi bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran,mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang dimiliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian juga dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik.

2.1 Algoritma *Dijkstra*

Algoritma yang ditemukan oleh Dijkstra untuk mencari *path* terpendek merupakan algoritma yang lebih efisien dibandingkan algoritma Warshall, meskipun implementasinya juga lebih sukar. Misalkan G adalah graf berarah berlabel dengan titik-titik V(G) = {v1, v2,..., vn} dan *path* terpendek yang dicari adalah dari v1 ke vn. Algoritma Dijkstra dimulai dari titik v1. dalam iterasinya, algoritma akan mencari satu titik yang jumlah bobotnya dari titik 1 terkecil. Titik-titik yang terpiih dipisahkan dan titik-titik tersebut tidak diperhatikan lagi dalam iterasi berikutnya.
Misalkan:
V (G) = {v1, v2,.., vn}
L = Himpunan titik-titik ε V (G) yang sudah terpilih dalam jalur *path* terpendek.
D(j) = Jumlah bobot *path* terkecil dari v1 ke vj.
w(i,j) = Bobot garis dari titik vi ke vj.
w*(1,j) = Jumlah bobot *path* terkecil dari v1 ke vj
Secara formal, algoritma Dijkstra untuk mencari path terpendek adalah sebagai berikut:

```
1. L = { };
V = {v2, v3,..,vn}.
2. Untuk i = 2, ..., n, lakukan D(i) – w(1, i)
3. Selama vn ∉ L lakukan:
a. Pilih titik vk ∈ V - L dengan D(k) terkecil.
L = L ∪ {vk}
b. Untuk setiap vj ∈ V - L lakukan:
Jika D(j) > D(k) + W(k,j) maka ganti D(j) dengan D(k) + W(k,j)
4. Untuk setiap vj ∈ V, w*(1, j) = D(j)
```

Menurut algoritma di atas, path terpendek dari titik v1 ke vn adalah melalui titik-titik dalam L secara berurutan, dan jumlah bobot path terkecilnya adalah D(n).

3. Implementasi Sistem

Dalam bab ini membahas mengenai implementasi metode terhadap aplikasi yang sudah dibuat dan juga pengujian metode yang di terapkan. Serta melakukan uji coba pada aplikasi yang telah dibangun, apakah telah sesuai dengan perancangan dan hasil yang diharapkan.

4. Implementasi Algoritma Dijkstra pada Perilaku NPC

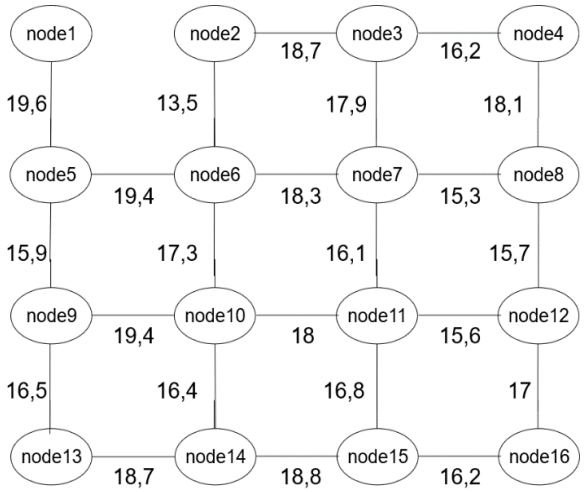
Proses implementasi adalah proses penerapan komponen system utama yang dibangun berdasarkan rancangan yang telah dibuat dan diajukan sebelumnya. Implementasi kecerdasan buatan pada penelitian ini diterapkan pada perilaku pencarian pada NPC dengan memanfaatkan metode Dijkstra. Metode Dijkstra digunakan sebagai pembangkit perilaku pencarian NPC. Posisi NPC dijadikan sebagai starting point dan karakter prajurit diinisiasi sebagai tujuan.

5. Implementasi Node

Simulasi dilakukan terhadap enam belas (16) node. Koordinat node-node ini di *input* didalam Unity begitu begitu juga dengan jaraknya. Berikut Tabel 4. 1 yang menunjukkan data 16 node yang disimulasikan pada sistem beserta jarak, dan koneksi antar masing-masing node.

Tabel 1. 1 Koneksi Dan Jarak Antar Node

Node	Koneksi dan Jarak			
Node1	Node5 (19.6)	-	-	-
Node2	Node3 (18.7)	Node6 (13.5)	-	-
Node3	Node2 (18.7)	Node4 (16.2)	Node7 (17.9)	-
Node4	Node3 (16.2	Node8 (18.1)	-	-
Node5	Node1 (19.6)	Node6 (19.4)	Node9 (15.9)	-
Node6	Node2 (13.5)	Node5 (19.4)	Node7 (18.3)	Node10 (17.3)
Node7	Node3 (17.9)	Node6 (18.3)	Node8 (15.3)	Node11 (16.1)
Node8	Node4 (18.1)	Node7 (15.3)	Node12 (15.7)	-
Node9	Node5 (15.9)	Node10 (19.4)	Node13 (16.5)	-
Node10	Node6 (17.3)	Node9 (19.4)	Node11 (18)	Node14 (16.4)
Node11	Node7 (16.1)	Node10 (18)	Node12 (15.6)	Node15 (16.8)
Node12	Node8 (15.7)	Node11 (15.6)	Node16 (17)	-
Node13	Node9 (16.5)	Node14 (18.7)	-	-
Node14	Node10 (16.4)	Node13 (18.7)	Node15 (18.8)	-
Node15	Node11 (16.8)	Node14 (18.8)	Node16 ()	-
Node16	Node12 (17)	Node15 (16.2)	-	-



Gambar 1. 2 Susunan Node Awal

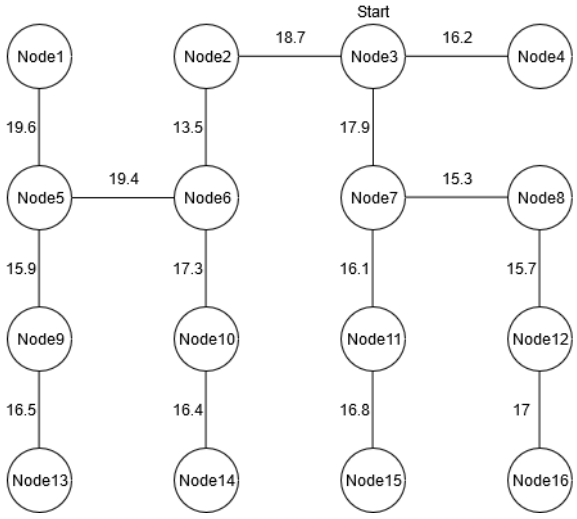
Dari Gambar 1. 2 dapat terlihat node-node yang ada beserta koneksi dan jarak antara satu node dengan lainnya, untuk uji cobanya, titik awal node ditentukan pada Node3 dan titik akhirnya tergantung pada posisi karakter utama (antara Node 13, Node14, Node15, Node16, Node9, atau Node12). Pertama akan dilakukan proses penghitungan manual, kemudian hasilnya akan dibandingkan dengan hasil yang didapatkan oleh program.

6. Perhitungan Manual

Tabel 1.2 Hasil Perhitungan Manual

Node	Total Cost	Previous Node
Node1	71.2	Node5
Node2	18.7	Node3
Node3	0	Node3
Node4	16.2	Node3
Node5	51.6	Node6
Node6	32	Node2
Node7	17.9	Node3
Node8	33.2	Node7
Node9	67.5	Node5
Node10	49.5	Node6
Node11	34	Node7
Node12	48.9	Node8
Node13	83.8	Node9
Node14	65.9	Node10
Node15	50.8	Node11
Node16	65.9	Node12

Berdasarkan tabel diatas, kita bisa membuat susunan node baru dengan memperhatikan Node, Total Cost, dan Previous Node. Pertama kita pilih node pertama yaitu node3, lalu lihat Node mana saja yang memiliki Previous Node berupa Node3, node yang memiliki Previous Code Node3 adalah Node2, Node4, dan Node7, lalu gambarkan dan hubungkan ketiga node tersebut dengan Node3 dan tulis jaraknya, lalu lihat node mana saja yang memiliki Previous Code Node2, Node4, atau Node7 lalu hubungkan juga. Lakukan sampai semua node habis.



Gambar 1. 2 Susunan Node Awal

7. Kesimpulan

Proses implementasi algoritma Dijkstra pada NPC telah berhasil, NPC mampu mencari jalurtercepat untuk sampai ke karakter utama, walaupun terhalang jalur, atau posisi nodenya diubah.

Daftar Pusaka :

[1] Craig, W., Reynolds. *"Stering Behavior For Autonomus Characters"* Boullevard, California, n.d.

[2] U. A. Khan., Y. Okada *EMOTIONAL DECISION MAKING RESPONSE OF NON-PLAYABLE CHARACTERS IN A ROLE-PLAYING GAME,* Japan, 2016.

[3] Prihatmanto, A.S *"Strategy and behavior models of non-playable character using computational intelligance approach",* Malaysia, 2013.

[4] Matahari, R., Samuel "*Pemetaan Perilaku Non-Playable Character Pada Permainan Berbasis Role Playing Game Menggunakan Metode Finite State Machine",* Yogyakarta, 2015.

[5] John McCarthy, 1956, AI: