Implementasi Kecerdasan Buatan Pada Permainan "Phantom Crown" Menggunakan Hierarchical Finite State Machine dan Decision Tree

Dimas Rahman Oetomo, Imam Kuswardayan, Nanik Suciati
Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: imam@its.ac.id

***Abstrak*—Phantom Crown merupakan permainan di mana dua pemain saling bertarung. Agar Phantom Crown lebih menyenangkan dibuatlah lawan bermain yang memiliki kecerdasan buatan. Implementasi kecerdasan buatan dilakukan menggunakan Decision Tree (DT). DT dipilih karena berguna untuk mengeksplorasi data dan menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target. Untuk menunjang kecerdasan buatan yang dibuat, diperlukan alur bermain yang dibuat menggunakan Hierarchical Finite State Machine (HFSM). HFSM digunakan karena memudahkan programmer dalam mengimplementasikan alur yang sudah didesain dan membuat sistem tidak terlalu kompleks. Eksperimen yang dilakukan kepada 6 orang menghasilkan persentase kepuasan pengguna sebesar 66%. Eksperimen tingkat kecerdasan buatan dilakukan dengan menandingkan musuh melawan musuh. Eksperimen menghasilkan 86% probabilitas menang untuk musuh dengan tingkat kecerdasan lebih tinggi.**

***Kata Kunci*— Decision Tree, Desain Permainan, Hierarchical Finite State Machine, Kecerdasan Buatan.**

I. PENDAHULUAN

PHANTOM Crown merupakan *fighting game* yang akan dirancang pada tugas akhir ini. *Fighting game* adalah s*ubgenre video game* dari *genre Action Game* dimana pemain mengkontrol sebuah karakter yang bertarung melawan musuh dipertarungan jarak dekat [1]. Pertarungan di dalam *fighting game* biasanya terdiri dari beberapa ronde. Agar permainan menjadi lebih menyenangkan musuh dalam permainan dapat dikendalikan oleh kecerdasan buatan.

Untuk membuat kecerdasan buatan yang dapat berkompetisi dengan manusia diperlukan alur pemodelan kebiasaan manusia ketika bermain permainan phantom crown. Untuk mengatasi itu penulis menggunakan HFSM seperti yang telah dilakukan oleh Peter M. Kielar, Oliver Handel, Daniel H. Biedermann, dan Andre Borrman pada penelitian mereka yaitu pemodelan kebiasaan pejalan kaki [2]. Pemodelan alur manusia belum cukup untuk membuat musuh permainan yang kompeten. Oleh karena itu penulis memilih DT sebagai kecerdasan buatan untuk memberikan prediksi pergerakan selanjutnya. Penggunaan DT untuk memprediksi telah dilakukan oleh Gerasimos Spanakis dkk dalam riset mereka [3].

Tujuan dari pengerjaan tugas akhir ini adalah mampu menghasilkan lawan bermain untuk *fighting game* yang memiliki kecerdasan buatan. Lawan bermain yang dihasilkan diharapkan untuk menjadi dasar untuk pengembangan selajutnya. Selain itu, tugas akhir ini diharapkan mampu memberikan solusi terhadap penerapan HFSM dan DT.

II. METODOLOGI

Dalam pembuatan kecerdasan buatan pada permainan phantom crown terdapat dua tahap, yaitu pembuatan alur kecerdasan buatan dan pembuatan *decision tree*.

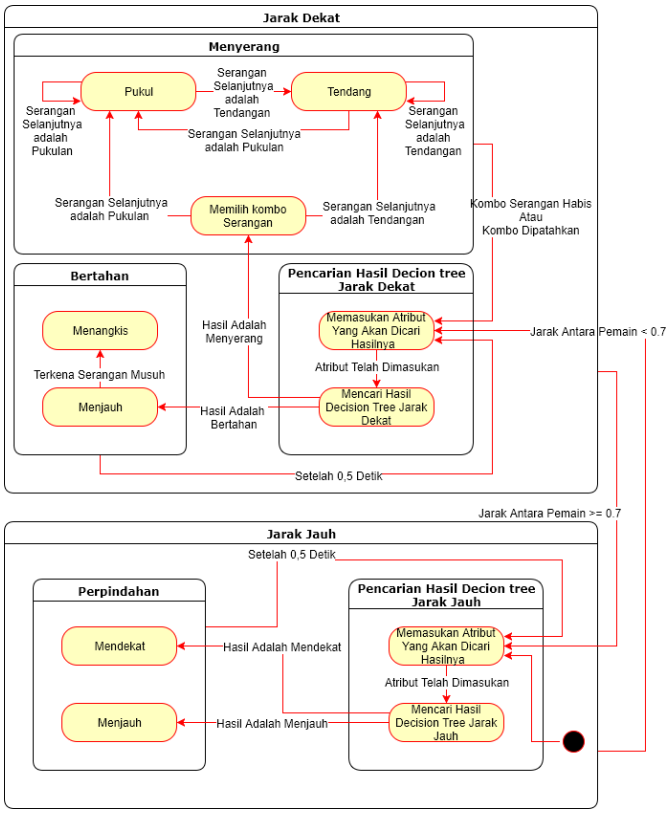
A. Tahap Pembuatan Alur Untuk Kecerdasan Buatan

Pada bagian ini menjelaskan alur untuk pemodelan kecerdasan buatan untuk kecerdasan buatan. Perancangan pemodelan alur menggunakan HFSM. HFSM digunakan karena mengumpulkan *state* berdasarkan *behavior* sehingga memudahkan untuk melakukan klusterisasi [4].

Dalam alur HFSM yang telah dibuat, alur memiliki 2 *superstate* terbesar yaitu *superstate* jarak dekat dan *superstate* jarak jauh. *Superstate* jarak dekat memiliki fungsi untuk menggerakkan karakter ketika karakter berada di dalam jarak serang. *Superstate* jarak jauh memiliki fungsi untuk menggerakkan karakter ketika karakter berada di luar jarak serang. *Superstate* jarak dekat dan *superstate* jarak jauh memiliki *superstate* pencarian hasil decision tree yang berguna untuk memilih pergerakan yang akan dilakukan oleh karakter.

Superstate jarak dekat memiliki 2 *superstate* yang digunakan untuk menggerakkan karakter yaitu *superstate* menyerang dan *superstate* bertahan. *Superstate* menyerang berisi *state* untuk mengurangi kesehatan musuh. *Superstate* bertahan berisi *state* untuk menghalangi serangan musuh sehingga kesehatan karakter tidak berkurang.

Superstate jarak jauh memiliki *superstate* yang digunakan untuk melakukan pergerakan yaitu *superstate* perpindahan. *Superstate* perpindahan berisi state untuk berpindah menjauh



Gambar 1. Bentuk Alur HFSM Kecerdasan Buatan

atau mendekat. Gambar dari alur HFSM dapat dilihat pada Gambar 1.

a) Superstate Jarak Jauh

Superstate jarak jauh merupakan sebuah *superstate* yang bertujuan untuk mengatur pergerakan karakter ketika jarak antara karakter >= 0.7. *Superstate* jarak jauh memiliki *superstate* yang digunakan untuk melakukan perpindahan yaitu *superstate* perpindahan. Perpindahan karakter diatur oleh *superstate* pencarian hasil *decision tree* jarak jauh.

b) Superstate Jarak Dekat

Superstate jarak dekat merupakan sebuah *superstate* yang bertujuan untuk mengatur pergerakan karakter ketika jarak antara karakter < 0.7. *Superstate* jarak dekat memiliki fungsi untuk mengurangi kesehatan musuh dan menjaga kesehatan karakter itu sendiri. Ketika karakter ingin menyerang *state* harus berada dalam *superstate* menyerang. Ketika karakter ingin bertahan *state* harus berada dalam *superstate* bertahan. Pemilihan *superstate* menyerang atau *superstate* bertahan diatur oleh *superstate* pencarian hasil *decision tree* jarak dekat.

c) Superstate Pencarian Hasil Decision Tree Jarak Jauh

Superstate pencarian hasil decision tree jarak jauh digunakan untuk memilih perpindahan. *Superstate* ini berjalan ketika karakter baru saja masuk ke dalam superstate jarak jauh atau ketika perpindahan yang dilakukan telah selesai. Ketika masuk ke dalam *superstate*, *state* memasukan nilai atribut yang akan dicari hasilnya aktif. Atribut dimasukan oleh *state* adalah atribut perpindahan musuh, atribut pemain terkena serangan, atribut kesehatan pemain dan atribut kesehatan musuh.

Setelah atribut dimasukan, hasilnya akan dicari di dalam *decision tree* jarak jauh. Hasil output dari *superstate* adalah mendekat atau menjauh.

d) Superstate Pencarian Hasil Decision Tree Jarak Dekat

Superstate pencarian hasil *decision tree* jarak dekat digunakan untuk memilih *superstate* menyerang atau *superstate* bertahan. Superstate ini berjalan ketika karakter baru saja masuk ke dalam *superstate* jarak dekat atau ketika aksi yang dilakukan dalam *superstate* menyerang atau *superstate* bertahan telah selesai. Ketika masuk ke dalam *superstate*, *state* memasukan nilai atribut yang akan dicari hasilnya aktif. Atribut dimasukan oleh *state* adalah atribut musuh terkena serangan, atribut pemain terkena serangan, atribut kesehatan pemain dan atribut kesehatan musuh.

Setelah atribut dimasukan hasilnya akan dicari di dalam *decision tree* jarak dekat. Hasil *output* dari *superstate* adalah menyerang atau bertahan.

e) Superstate Perpindahan

Superstate perpindahan adalah sebuah *superstate* yang menampung *state* mendekat dan menjauh. Jika *superstate* mendapat masukan mendekat, maka akan masuk *state* mendekat. Jika *superstate* mendapat masukan menjauh, maka akan masuk *state* jauh. Alur akan keluar dari *superstate* jika telah melewati 0,5 detik.

f) Superstate Bertahan

Superstate bertahan adalah sebuah *superstate* yang menampung *state* menangkis dan menjauh. Jika *superstate* mendapat masukan menjauh, maka akan masuk ke *state* menjauh. Jika mendapat masukan terkena serangan ketika dalam *state* menjauh, maka akan masuk ke *state* menangkis. Alur akan keluar dari *superstate* jika telah melewati 0,5 detik.

g) Superstate Menyerang

Superstate menyerang adalah sebuah *superstate* yang menampung *state* memilih kombo serangan, pukul dan tendang. Ketika mendapat masukan menyerang, maka akan masuk ke *state* memilih kombo serangan. Di dalam *state* kombo serangan, program memilih kombo serangan secara acak.

Kombo serangan berisi daftar serangan-serangan yang akan dilakukan. Jika serangan selanjutnya adalah pukul akan masuk ke *state* pukul. Jika serangan selanjutnya adalah tendang akan masuk ke *state* tendang. Alur akan dilakukan hingga kombo serangan habis atau kombo serangan dipatahkan.

B. Tahap Pembuatan Decision Tree

Perancangan DT dimulai dengan menyusun nilai dari setiap atribut. Setelah nilai atribut disusun nilai atribut akan dihitung berdasarkan algoritma ID3 [5]. Hasil dari perhitungan adalah sebuah *tree* yang menentukan pergerakan dari karakter.

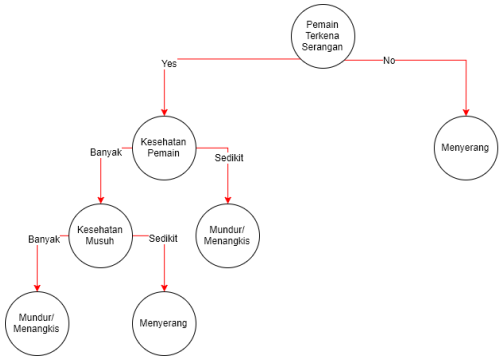
a) Penghitungan Decision Tree Jarak Dekat

Pada tahap ini penulis melakukan pemasukan data ke dalam Tabel 1. yang akan dihitung dengan algoritma ID3. Hasil penghitungan adalah sebuah *tree* yang digunakan untuk pemilihan pergerakan karakter. Data yang dimasukan ke dalam tabel dikumpulkan dari pemain yang telah mencoba permainan.

Tabel 1.
Nilai Atribut Decision Tree Jarak Dekat

No	Pemain Terkena Serangan	Musuh Terkena Serangan	Kesehatan Pemain	Kesehatan Musuh	Hasil
1	Ya	Tidak	Banyak	Banyak	Bertahan
2	Tidak	Ya	Banyak	Banyak	Menyerang
3	Tidak	Tidak	Sedikit	Banyak	Menyerang
4	Tidak	Tidak	Banyak	Sedikit	Menyerang
5	Ya	Ya	Banyak	Banyak	Bertahan
6	Ya	Tidak	Sedikit	Banyak	Bertahan
7	Ya	Tidak	Banyak	Sedikit	Menyerang
8	Tidak	Ya	Sedikit	Banyak	Menyerang
9	Tidak	Ya	Banyak	Sedikit	Menyerang
10	Tidak	Tidak	Banyak	Banyak	Menyerang
11	Tidak	Tidak	Sedikit	Sedikit	Menyerang
12	Ya	Ya	Sedikit	Banyak	Bertahan
13	Ya	Ya	Banyak	Sedikit	Menyerang
14	Ya	Ya	Sedikit	Sedikit	Bertahan
15	Ya	Tidak	Sedikit	Sedikit	Bertahan
16	Tidak	Ya	Sedikit	Sedikit	Menyerang

Dari Tabel 1. dicari *information gain* (IG) terbaik dari setiap atribut. Buat simpul dari atribut tersebut, ulangi perhitungan IG yang dilakukan hingga mendapatkan *leaf*. Jika atribut yang dipilih ada pada percabangan yang sama maka tidak diikutkan lagi dalam perhitungan IG. Bentuk *tree* dari penghitungan algoritma ID3 berdasar Tabel 1. adalah sebagai berikut.



Gambar. 2. Bentuk Decision Tree Jarak Dekat

b) Penghitungan Decision Tree Jarak Jauh

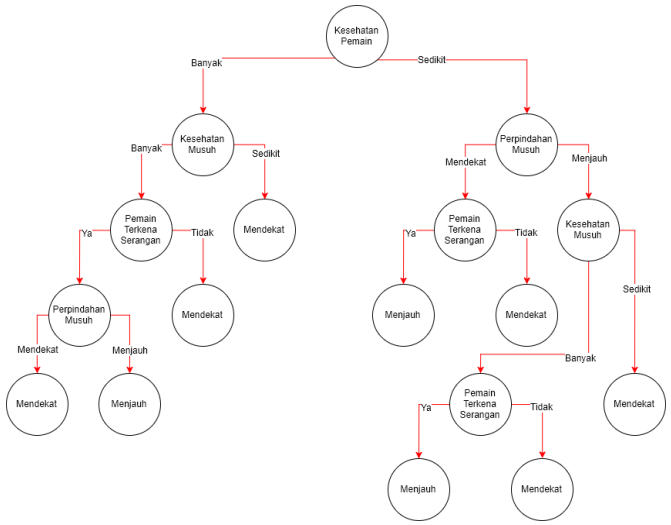
Pada tahap ini penulis melakukan pemasukan data ke dalam Tabel 2. yang akan dihitung dengan algoritma ID3. Hasil penghitungan adalah sebuah *tree* yang digunakan untuk pemilihan pergerakan karakter. Data yang dimasukan ke dalam tabel dikumpulkan dari pemain yang telah mencoba permainan.

Dari Tabel 2. dicari IG terbaik dari setiap atribut. Buat simpul dari atribut tersebut, ulangi perhitungan IG yang dilakukan hingga mendapatkan *leaf*. Jika atribut yang dipilih ada pada percabangan yang sama maka tidak diikutkan lagi dalam perhitungan IG.

Tabel 2.
Nilai Atribut Decision Tree Jarak Jauh

No	Pemain Terkena Serangan	Musuh Terkena Serangan	Kesehatan Pemain	Kesehatan Musuh	Hasil
1	Ya	Tidak	Banyak	Banyak	Bertahan
2	Tidak	Ya	Banyak	Banyak	Menyerang
3	Tidak	Tidak	Sedikit	Banyak	Menyerang
4	Tidak	Tidak	Banyak	Sedikit	Menyerang
5	Ya	Ya	Banyak	Banyak	Bertahan
6	Ya	Tidak	Sedikit	Banyak	Bertahan
7	Ya	Tidak	Banyak	Sedikit	Menyerang
8	Tidak	Ya	Sedikit	Banyak	Menyerang
9	Tidak	Ya	Banyak	Sedikit	Menyerang
10	Tidak	Tidak	Banyak	Banyak	Menyerang
11	Tidak	Tidak	Sedikit	Sedikit	Menyerang
12	Ya	Ya	Sedikit	Banyak	Bertahan
13	Ya	Ya	Banyak	Sedikit	Menyerang
14	Ya	Ya	Sedikit	Sedikit	Bertahan
15	Ya	Tidak	Sedikit	Sedikit	Bertahan
16	Tidak	Ya	Sedikit	Sedikit	Menyerang

Bentuk *tree* dari penghitungan algoritma ID3 berdasar Tabel 2 adalah sebagai berikut.



Gambar. 2. Bentuk Decision Tree Jarak Jauh

III. UJI COBA DAN PEMBAHASAN

A. *Uji Coba Tingkat Kesulitan*

Berdasarkan Tabel 3. karakter dengan tingkat kesulitan lebih tinggi dapat kalah dari karakter dengan tingkat kesulitan lebih rendah, jika jarak tingkat kesulitan tidak jauh misal *hard* dan normal. Berdasarkan 22 percobaan, terdapat 3 percobaan dimana karakter dengan tingkat kesulitan lebih rendah mengalahkan karakter dengan tingkat kesulitan lebih tinggi, sehingga persentase ketepatan tingkat kesulitan adalah 86%.

Tabel 3.
Hasil Uji Coba Tingkat Kesulitan Kecerdasan Buatan

No	Ronde	Tingkat Kesulitan Pemain 1	Tingkat Kesulitan Pemain 2	Pemenang	Pemenang Seharusnya
1	Ronde 1	Very Hard	Hard	Pemain 1	Pemain 1
	Ronde 2	Very Hard	Hard	Pemain 1	Pemain 1
2	Ronde 1	Very Hard	Normal	Pemain 1	Pemain 1
	Ronde 2	Very Hard	Normal	Pemain 1	Pemain 1
3	Ronde 1	Very Hard	Easy	Pemain 1	Pemain 1
	Ronde 2	Very Hard	Easy	Pemain 1	Pemain 1
4	Ronde 1	Very Hard	Very Easy	Pemain 1	Pemain 1
	Ronde 2	Very Hard	Very Easy	Pemain 1	Pemain 1

5	Ronde 1	Hard	Normal	Pemain 1	Pemain 1
	Ronde 2	Hard	Normal	Pemain 2	Pemain 1
	Ronde 3	Hard	Normal	Pemain 2	Pemain 1
6	Ronde 1	Hard	Easy	Pemain 1	Pemain 1
	Ronde 2	Hard	Easy	Pemain 1	Pemain 1
7	Ronde 1	Hard	Very Easy	Pemain 1	Pemain 1
	Ronde 2	Hard	Very Easy	Pemain 1	Pemain 1
8	Ronde 1	Normal	Easy	Pemain 1	Pemain 1
	Ronde 2	Normal	Easy	Pemain 2	Pemain 1
	Ronde 3	Normal	Easy	Pemain 1	Pemain 1
9	Ronde 1	Normal	Very Easy	Pemain 1	Pemain 1
	Ronde 2	Normal	Very Easy	Pemain 1	Pemain 1
10	Ronde 1	Easy	Very Easy	Pemain 1	Pemain 1
	Ronde 2	Easy	Very Easy	Pemain 1	Pemain 1

B. Uji Coba Pengguna

Berdasarkan hasil uji coba pengguna melawan kecerdasan buatan dapat dilihat Tabel 4. dapat disimpulkan sebagai berikut:

- Musuh dengan tingkat kesulitan *easy* dan *very easy* dinilai terlalu mudah untuk pemain karena hanya dapat memberikan rata-rata kerusakan pemain dibawah 40%.
- Musuh dinilai kurang sulit untuk pemain karena hanya dapat mengalahkan pemain sekali.

Tabel 4.
Hasil Uji Coba Karakter Dengan Kecerdasan Buatan Sebagai Lawan Bermain

Musuh	Pemain	Kerusakan Musuh	Rata-rata Kerusakan Musuh	Kerusakan Pemain	Rata-rata Kerusakan Pemain	Pemenang
Very Hard	RD	100%	90%	60%	76,7%	Pemain
Very Hard	IGA	100%		80%		Pemain
Very Hard	A	100%		60%		Pemain
Very Hard	MRP	100%		80%		Pemain
Very Hard	RBP	60%		100%		Musuh
Very Hard	GR	80%		80%		Draw
Hard	RD	100%	97%	40%	58%	Pemain
Hard	IGA	100%		60%		Pemain
Hard	A	100%		50%		Pemain
Hard	MRP	100%		60%		Pemain
Hard	RBP	80%		60%		Pemain
Hard	GR	100%		80%		Pemain
Normal	RD	100%	100%	40%	45%	Pemain
Normal	IGA	100%		50%		Pemain
Normal	A	100%		20%		Pemain
Normal	MRP	100%		50%		Pemain
Normal	RBP	100%		60%		Pemain
Normal	GR	100%		50%		Pemain
Easy	RD	100%	100%	20%	23%	Pemain
Easy	IGA	100%		20%		Pemain

- Musuh kurang dapat mempertahankan diri dari pemain karena selalu mendapat rata-rata kerusakan musuh lebih dari 80%.
- Musuh dengan tingkat kesulitan very hard dapat memberikan rata-rata kerusakan pemain cukup besar yaitu 76,7%.

Easy	A	100%		20%		Pemain
Easy	MRP	100%		20%		Pemain
Easy	RBP	100%		40%		Pemain
Easy	GR	100%		20%		Pemain
Very Easy	RD	100%	100%	0%	10%	Pemain
Very Easy	IGA	100%		20%		Pemain
Very Easy	A	100%		0%		Pemain
Very Easy	MRP	100%		0%		Pemain
Very Easy	RBP	100%		20%		Pemain
Very Easy	GR	100%		20%		Pemain

Berdasarkan hasil kuesioner dari 6 responden yang

Tabel 5.
Hasil Kuesioner Pengguna

No	Parameter Tingkat Kesulitan	Penilaian 1	2	3	4	Rata-Rata Persentase Penilaian	Rata-Rata Total Persentase Penilaian Parameter
1	Aplikasi sudah memiliki tingkatan kesulitan yang tepat		**3**	**2**	**1**	**67%**	**66%**
2	Tingkat kesulitan very easy dapat membantu pemain memahami permainan		**3**	**3**		**63%**	
3	Tingkat kesulitan very hard dapat teman bermain yang menantang		**3**	**2**	**1**	**67%**	
	Parameter Imersif						
4	Saya merasa lebih untuk memainkan permainan phantom karena memiliki kecerdasan buatan		**4**	**2**		**58%**	**61%**
5	Saya merasakan sensasi seperti melawan manusia	**1**	**2**	**2**	**1**	**63%**	

dirangkum dalam Tabel 5., aplikasi belum sesuai target diatas 70% karena hanya memiliki 66% untuk nilai persentase kepuasan. Aplikasi mendapat nilai di bawah target karena parameter tingkat kesulitan dan parameter imersif dinilai responden belum sesuai karena hanya memiliki nilai 66% dan 61%.

	Parameter Performa Permainan						
6	Nilai dinamis pada permainan		**2**	**3**	**1**	**71%**	**70%**
7	Kesesuaian interaksi dengan hasil yang seharusnya		**1**	**3**	**2**	**79%**	
8	Performa atau kinerja pada permainan		**3**	**2**	**1**	**67%**	
9	Saya merasa tertarik untuk menggunakan aplikasi ini	**1**	**3**	**1**	**1**	**58%**	
10	Saya terhibur dengan aplikasi ini	**1**	**1**	**2**	**2**	**71%**	
Persentase kepuasan							**66%**

IV. KESIMPULAN/RINGKASAN

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Musuh dengan tingkat kesulitan *very hard* dapat memberikan rata-rata kerusakan kepada pemain manusia sebanyak 76,7%.
2. Rancangan alur kecerdasan buatan dengan HFSM dapat mengendalikan karakter dengan baik. Karakter dapat mengakses seluruh fitur karakter yang diberikan.
3. Berdasarkan hasil uji coba tingkat kesulitan, tingkat kesulitan memiliki akurasi ketepatan sebanyak 86%.
4. Berdasarkan kuesioner pengguna, aplikasi belum sesuai target diatas 70% karena hanya memiliki 66% untuk nilai persentase kepuasan. Menurut pengguna musuh dengan kecerdasan buatan terlalu mudah.

DAFTAR PUSTAKA

[1] "The Next Generation 1996 Lexicon A to Z: Fighting Game". Next Generation. No. 15. Imagine Media. March 1996. p. 33.

[2] P. M. Kielar, O. Handel, D. H. Biedermann, and A. Borrmann, "Concurrent Hierarchical Finite State Machines for Modeling Pedestrian Behavioral Tendencies," Transp. Res. Procedia, vol. 2, pp. 576–584, Jan. 2014.E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, akan dipublikasikan.

[3] G. Spanakis, G. Weiss, B. Boh, V. Kerkhofs, and A. Roefs, "Utilizing Longitudinal Data to Build Decision Trees for Profile Building and Predicting Eating Behavior," *Procedia Comput. Sci.*, vol. 100, pp. 782–789, 2016.

[4] "The Gist of Hierarchical FSM | AiGameDev.com." [Online]. Available: http://aigamedev.com/open/article/hfsm-gist/. [Accessed: 21-Jan-2018].

[1] Utgoff, P. E. (1989). Incremental induction of decision trees. Machine learning, 4(2), 161-186. doi:10.1023/A:1022699900025