



BUKU TESIS - KI142502

**ANALISIS PERBANDINGAN KECERDASAN
BUATAN PADA *COMPUTER PLAYER* DALAM
MENGAMBIL KEPUTUSAN PADA
*GAME BATTLE RPG***

MUSTA'INUL ABDI
NRP. 5115201015

DOSEN PEMBIMBING
Dr. Darlis Herumurti, S. Kom., M. Kom.
Imam Kuswardayan, S. Kom., M.T.

PROGRAM MAGISTER
BIDANG KEAHLIAN INTERAKSI GRAFIKA DAN SENI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M. Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Musta'inul Abdi
NRP. 5115201015

Dengan judul:
Analisis Perbandingan Kecerdasan Buatan pada *Computer Player* dalam
Mengambil Keputusan pada *Game Battle RPG*

Tanggal Ujian : 22-6-2017
Periode Wisuda : 2016 Genap

Disetujui oleh:

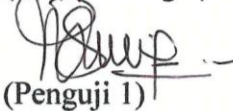
Dr. Darlis Heru Murti, S. Kom, M. Kom
NIP. 19771217 200312 1 001


(Pembimbing 1)

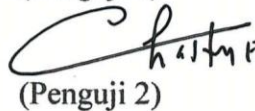
Imam Kuswardayan, S. Kom, MT
NIP. 19761215 200312 1 001


(Pembimbing 2)

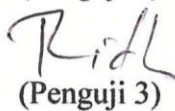
Dr. Eng. Nanik Suciati, S. Kom, M. Kom
NIP. 19710428 199412 2 001


(Penguji 1)

Dr. Eng. Chastine Fatchah, S. Kom, M. Kom
NIP. 19751220 200112 2 002


(Penguji 2)

Ridho Rahman Hariadi, S. Kom, M. Sc
NIP. 19870213 201404 1 001


(Penguji 3)

Dekan Fakultas Teknologi Informasi,


Dr. Agus Zamal Arifin, S. Kom, M. Kom.
NIP. 19720809 199512 1 001

[Halaman ini sengaja dikosongkan]

ANALISIS PERBANDINGAN KECERDASAN BUATAN PADA COMPUTER PLAYER DALAM MENGAMBIL KEPUTUSAN PADA GAME BATTLE RPG

Nama Mahasiswa : Musta'inul Abdi
NRP : 5115201015
Pembimbing : Dr. Darlis Herumurti, S. Kom., M. Kom.
Co-Pembimbing : Imam Kuswardayan, S. Kom., M.T.

ABSTRAK

Pemanfaatan kecerdasan buatan telah diimplementasikan kedalam banyak hal, salah satunya adalah *game*. Secara umum tujuan dibuatnya *game* adalah untuk membuat pengguna menjadi terhibur dan merasakan kesenangan ketika sedang atau telah bermain. Kecerdasan buatan di dalam *game* dibutuhkan untuk meningkatkan tantangan di dalam *game* dan membuat *game* menjadi lebih dinamis dan terarah. Sehingga akan menciptakan kesenangan bagi pengguna pada saat dan setelah memainkan *game*.

Beberapa penerapan kecerdasan buatan di dalam *game* diantaranya adalah dengan menggunakan metode *Support Vector Machine* (SVM). Dalam beberapa kasus *game* ada juga yang menggunakan metode *Decision Tree* yang akan mengatur perilaku *computer player* di dalam permainan. Metode yang lebih sederhana untuk mengatur perilaku *computer player* yaitu *Rulebase*. Pada penelitian ini akan dilakukan perbandingan kecerdasan buatan untuk mengatur perilaku *computer player* di dalam *game Role-Playing Game* (RPG). Yang dimaksud *computer player* pada penelitian ini adalah pemain atau karakter yang dijalankan oleh sistem di dalam *game*.

Tujuan dilakukannya perbandingan tersebut adalah untuk mengetahui metode kecerdasan buatan manakah yang paling baik diterapkan pada *game* berjenis *battle RPG*. Metode yang digunakan untuk menguji kecerdasan buatan yang diterapkan pada *game battle RPG* ini adalah dengan menggunakan skenario pertandingan.

Berdasarkan analisis yang telah dilakukan didapatkan hasil bahwa kecerdasan buatan dengan menggunakan metode SVM memiliki keunggulan dalam faktor jumlah kemenangan. Hal ini dibuktikan dengan persentase kemenangan metode SVM sebesar 72.5%, *Decision Tree* sebesar 50% dan *Rulebase* sebesar 25%. Berdasarkan data tersebut dapat disimpulkan bahwa pada penelitian ini metode SVM adalah metode pengambilan keputusan yang paling baik dibandingkan dengan metode *decision tree* dan *rulebase*.

Kata Kunci: *Decision Tree*, Kecerdasan Buatan, *Rulebase*, RPG, SVM.

[Halaman ini sengaja dikosongkan]

ANALYSIS OF ARTIFICIAL INTELLIGENCE COMPARISON FOR COMPUTER PLAYER IN DECISION MAKING ON RPG BATTLE GAME

Student's Name : Musta'inul Abdi
NRP : 5115201015
1st Advisor : Dr. Darlis Herumurti, S. Kom., M. Kom.
2nd Advisor : Imam Kuswardayan, S. Kom., M.T.

ABSTRACT

Artificial intelligence utilizing has been implemented into many things, one of them is a game. Generally, the purpose of games is to entertain users and make them feel excited while playing or even after it. Artificial intelligence in the game is needed to increase the challenge within the game and make it more dynamic and focused. Hence, it will create enjoyment for users at the time and after playing the game.

SVM, decision tree and rulebase are some of the artificial intelligences method that have been commonly used. Comparison of artificial intelligence will be implemented in this study in order to regulate the behavior of computer player in the Role-Playing Game (RPG). The computer player in this study is a player or character that is run by the system in the game.

The purpose of this study is to know which artificial intelligence technique is the best to be applied to battle RPG type games. Moreover, the method used to examine artificial intelligence which is applied to this RPG battle game is by utilizing a match scenario.

Regarding the analysis that has been done, the result shows that artificial intelligence using SVM method has an advantage in the number of winning factor. This is evidenced by the percentage of winning methods SVM, Decision tree and Rulebase, at about 72.5%, 50% and 25% respectively. Therefore, based on these data it can be concluded that in this study SVM method is the best decision-making method compared with decision tree and Rulebase method.

Keywords: *Decision Tree, Artificial Intelligent, Rulebase, RPG, SVM.*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Syukur Alhamdulillah kepada Allah SWT atas rahmat dan hidayahnya, sehingga tesis yang berjudul “Analisis Perbandingan Kecerdasan Buatan pada *Computer Player* dalam Mengambil Keputusan pada *Game Battle RPG*” dapat diselesaikan. Semoga tesis ini dapat memberikan manfaat pada perkembangan ilmu pengetahuan khususnya dalam bidang *game*. Selama proses penyusunan tesis ini. Secara khusus, ucapan terima kasih yang sebesar-besarnya disampaikan kepada:

1. Bapak Suprianto dan Ibu Imtihanah selaku orang tua yang selalu mendidik, membimbing dan memberikan motivasi sehingga tesis ini dapat diselesaikan.
2. Bapak Dr. Darlis Heru Murti, S. Kom, M. Kom dan Bapak Imam Kuswardayan, S. Kom, MT selaku dosen pembimbing yang telah memberikan ilmu, bimbingan, arahan, motivasi, serta saran selama perkuliahan dan pengerjaan tesis ini.
3. Ibu Dr. Eng. Nanik Suciati, S. Kom, M. Kom, Ibu Dr. Eng. Chastine Fatichah, S. Kom, M. Kom, serta Bapak Ridho Rahman Hariadi, S. Kom, M. Sc selaku dosen penguji yang telah banyak memberikan ilmu, arahan, perbaikan dan saran pada tesis ini.
4. Bapak Waskitho Wibisono, S. Kom., M.Eng., Ph.D. selaku Ketua Program Magister Teknik Informatika.
5. Teman-teman mahasiswa S2 Teknik Informatika angkatan 2015 yang telah membantu dan menjadi teman diskusi selama menyelesaikan penelitian ini maupun selama masa perkuliahan.
6. Keluarga, kerabat, rekan dan teman dekat yang tidak dapat disebutkan satu-persatu.

Laporan tesis ini masih jauh dari kesempurnaan, oleh karena itu kritik dan saran dari pembaca dibutuhkan untuk memperbaiki dan mengembangkan penelitian ini. Harapan yang diinginkan adalah, semoga penelitian ini dapat bermanfaat bagi pembaca maupun peneliti yang tertarik untuk mempelajari ataupun mengambil topik yang sama.

Surabaya, 26 Juni 2017

Musta'inul Abdi

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK.....	v
<i>ABSTRACT</i>	vii
KATA PENGANTAR	ix
DAFTAR ISI.....	xi
DAFTAR GAMBAR	xiii
DAFTAR TABEL.....	xv
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Perumusan Masalah	4
1.3 Tujuan dan Manfaat	4
1.3.1 Tujuan	4
1.3.2 Manfaat	4
1.4 Hipotesis Penelitian	5
1.5 Batasan Masalah	5
BAB 2 KAJIAN PUSTAKA.....	7
2.1 <i>Game</i>	7
2.2 <i>Role Playing Game (RPG)</i>	7
2.3 Kecerdasan Buatan (<i>Artificial Intelligent</i>).....	8
2.4 <i>Support Vector Machine (SVM)</i>	8
2.5 <i>Decision Tree</i>	9
2.6 <i>Rulebase dan Finite State Machine (FSM)</i>	10
BAB 3 METODE PENELITIAN	11
3.1 Studi Literatur	11
3.2 Desain dan Implementasi.....	12
3.2.1 Analisis Kebutuhan	12
3.2.2 Desain Sistem.....	13

3.2.3	Implementasi	19
3.3	Pengujian	19
3.3.1	Pembuatan Dataset	19
3.3.2	Skenario Pengujian	20
3.3.3	Analisis Hasil Pengujian.....	21
BAB 4	HASIL DAN PEMBAHASAN	23
4.1	Hasil Desain dan Implementasi	23
4.1.1	Hasil Analisis Kebutuhan	23
4.1.2	Hasil Desain Sistem.....	24
4.1.3	Hasil Implementasi	25
4.2	Hasil Pengujian.....	37
4.2.1	Hasil Pembuatan Dataset	37
4.2.2	Hasil Skenario Pengujian	39
4.2.3	Analisis Hasil Pengujian.....	40
BAB 5	KESIMPULAN DAN SARAN	45
5.1	Kesimpulan.....	45
5.2	Saran.....	45
DAFTAR PUSTAKA	47
LAMPIRAN	51
BIOGRAFI PENULIS	57

DAFTAR GAMBAR

Gambar 3. 1. Diagram alur penelitian.....	11
Gambar 3. 2 Diagram alir sistem	13
Gambar 3. 3 Contoh hasil pemisah dua class dengan menggunakan SVM.....	15
Gambar 3. 4 Ilustrasi metode SVM one agains all.	15
Gambar 3. 5 Struktur decision tree	16
Gambar 3. 6 Diagram FSM.....	18
Gambar 3. 7 Proses pembuatan dataset.....	20
Gambar 3. 8 Alur skenario percobaan kecerdasan buatan game battle RPG.....	20
Gambar 4. 1 Tampilan menu utama.....	27
Gambar 4. 2 Tampilan game battle RPG.	27
Gambar 4. 3 Tampilan hasil pertandingan	28
Gambar 4. 4 Pseudo-code fungsi SVM.....	29
Gambar 4. 5 Pseudo-code fungsi decision tree	31
Gambar 4. 6 Pseudo-code fungsi rulebase.....	33
Gambar 4. 7 Pseudo-code fungsi ultimate.	34
Gambar 4. 8 Pseudo-code fungsi attack.....	35
Gambar 4. 9 Pseudo-code fungsi defense.....	36
Gambar 4. 10 Pseudo-code fungsi heal.....	36
Gambar 4. 11 Perbandingan hasil metode yang berbeda.	41
Gambar 4. 12 Perbandingan SVM dan SVM.....	42

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 4. 1 Contoh sampel dataset.	38
Tabel 4. 2 Hasil pengujian skenario 1 hingga 3.....	39
Tabel 4. 3 Hasil pengujian skenario 4 hingga 6.....	40
Tabel 4. 4 Hasil analisis perbandingan antar metode.....	41
Tabel 4. 5 Hasil analisis perbandingan metode yang sama.....	42

[Halaman ini sengaja dikosongkan]

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkembangan kecerdasan buatan dari tahun ke tahun semakin meningkat. Penelitian yang dirilis hari ini dari Accenture (NYSE: ACN) pada 28 September 2016 memprediksi bahwa kecerdasan buatan (AI) dapat mencapai dua kali lipat tingkat pertumbuhannya hingga tahun 2035. Dampak teknologi AI pada bisnis diproyeksikan dapat meningkatkan produktivitas tenaga kerja hingga 40 persen sehingga akan mendorong pertumbuhan bisnis.

Pemanfaatan kecerdasan buatan diimplementasikan kedalam banyak hal. Salah satu bentuk pemanfaatan dan penerapan kecerdasan buatan adalah di dalam permainan komputer. Permainan komputer atau dalam masyarakat umum lebih dikenal sebagai game adalah aplikasi/perangkat lunak yang diciptakan bertujuan untuk memberikan hiburan dan kesenangan (fun) kepada penggunanya. Permainan komputer telah menjadi populer di kalangan masyarakat, sebagai suatu bentuk hiburan yang mendukung sebuah interaksi sosial. Dengan perkembangan teknologi dan perangkat keras komputer yang cukup pesat, memberikan potensi permainan komputer untuk terus berkembang (Chen & Kim 2012).

Berdasarkan laporan survey yang dilakukan oleh Spil Games pada tahun 2013 menyebutkan bahwa lebih dari 1, 2 miliar orang di dunia bermain game. 46% diantaranya adalah perempuan dan 56% sisanya adalah laki-laki. Pada tahun 2016 ESA (Entertainment Software Association) membuat laporan bahwa lebih dari 150 juta orang di Amerika Serikat menjadi pengguna game. 63% dari jumlah tersebut memainkan game minimal 3 jam tiap pekan. 65% dari jumlah tersebut memiliki perangkat untuk bermain game di rumah mereka. Hal tersebut membuktikan bahwa perkembangan industri game dari tahun ke tahun semakin meningkat. Seiring dengan perkembangan game, harapan dan ekspektasi pengguna terhadap kualitas game juga terus berkembang (Aghlara & Hadidi 2011).

Secara umum tujuan dibuatnya *game* adalah untuk membuat pengguna menjadi terhibur dan merasakan kesenangan ketika sedang bermain *game*. Untuk itu dalam pembuatan sebuah *game* diperlukan perancangan agar tujuan tersebut dapat tercapai.

Salah satu jenis *game* yang populer saat ini adalah *game* berjenis RPG. *Role Playing Game* (RPG) adalah sebuah permainan yang memungkinkan pemain untuk terlibat secara langsung di dalam perencanaan strategi dan alur di dalam permainan untuk mencapai suatu tujuan tertentu (Bedoya-rodriguez et al. 2014). Kemajuan teknologi saat ini memungkinkan penggunaan komputer untuk memvisualisasikan permainan *RPG* sehingga memberikan pengalaman bermain dan realisme lebih kepada pemain (J. Patrick Williams, Sean Q. Hendricks 2006). Karakteristik dan fitur menarik dari *game* berjenis RPG adalah kemungkinan untuk dapat mengembangkan peran karakter yang dipilih yang perkembangan dan keterampilannya melibatkan pengguna secara langsung di dalam permainan, fitur inilah yang membuat *game* RPG menjadi lebih dramatis dan dinamis (M. Childress & Braswell 2006). Berdasarkan uraian diatas, dapat diambil kesimpulan bahwa *game* berjenis RPG adalah *game* yang memiliki banyak sekali kemungkinan. Untuk itu dalam pembuatan AI (*computer player*) diperlukan sebuah metode atau algoritma yang baik sehingga AI dapat berjalan dengan optimal.

Salah satu elemen yang penting dalam sebuah *game* ialah adanya kecerdasan buatan atau AI (*artificial intelligence*). AI merupakan salah satu unsur yang diperlukan dalam pembentukan *game* yakni untuk membuat permainan lebih dinamis dan terarah (Urh et al. 2015). *Game* AI merupakan *game* yang mengubah metode, proses, dan algoritma pada kecerdasan tersebut yang akan diaplikasikan ke pembuatan dan pengembangan *game* (Yannakakis & Togelius 2014). Yannakakis menyebutkan terdapat tiga panorama dalam *game* AI yakni perspektif metode (komputer), perspektif pengguna (manusia) dan perspektif interaksi pemain. Kecerdasan buatan sering digunakan pada kebanyakan *game* saling memiliki ketergantungan interaksi terhadap pemain. Sehingga AI memiliki peran yang penting untuk meningkatkan ketertarikan pengguna dalam bermain *game* (Mcgee & Abraham 2010). Dari uraian diatas menunjukkan bahwa penggunaan kecerdasan

buatan di dalam game dibutuhkan untuk meningkatkan tantangan di dalam game. membuat game menjadi lebih dinamis dan dapat menunjang realitas game, sehingga akan menciptakan kesenangan bagi pengguna pada saat memainkan game dan ketertarikan pengguna untuk memainkan game juga akan meningkat.

Dalam kasus game RPG yang karakteristiknya memiliki banyak kemungkinan, dibutuhkan metode yang mampu mengambil keputusan yang baik. Beberapa penerapan kecerdasan buatan di dalam game diantaranya adalah dengan menggunakan metode *Support Vector Machine* (SVM) (Ralaivola et al. 2005; Melendez 2009; Melendez 2010). Dalam penelitian tersebut dijelaskan bahwa metode SVM dapat meningkatkan kecepatan dalam mengambil keputusan. Tetapi kekurangan dari metode ini adalah metode SVM ini sangat bergantung pada banyaknya jumlah data *training set*. SVM merupakan sistem logika yang menggunakan data training untuk menentukan keputusan yang akan di ambil berikutnya. Data training tersebut digunakan sebagai data input untuk mencari vektor (hyperplane) terbaik sebagai pemisah dua buah class, dimana class dapat diartikan sebagai keputusan yang akan di ambil. Pada beberapa penelitian tentang game lainnya ada juga yang menggunakan metode *Decision Tree*, dimana metode tersebut digunakan untuk mengatur perilaku computer player di dalam permainan (Ouinlan 1990; Stone & Veloso 1997; Narayek 2004). Metode *decision tree* menggunakan data training untuk membangun pohon keputusan. Pohon keputusan tersebut digunakan sebagai dasar dalam pengambilan keputusan selanjutnya. Tetapi data yang digunakan untuk metode *decision tree* ini haruslah data yang berbentuk kategorikal. Metode lain untuk mengatur perilaku *computer player* yaitu menggunakan *rulebase* (Rostianingsih et al. 2013). Berbeda dengan SVM dan *decision tree*, *rulebase* merupakan sekumpulan *rule* yang digunakan untuk menentukan aksi. *Rulebase* terdiri dari dua bagian penting yaitu bagian kondisi dan bagian aksi.

Berdasarkan uraian diatas, disebutkan bahwa tujuan utama dari pengembangan game adalah sebagai sarana hiburan. Salah satu jenis game yang cukup populer adalah game berjenis RPG. Game RPG memiliki karakteristik banyaknya kemungkinan yang dapat terjadi dalam mencapai tujuan permainan.

Salah satu faktor yang menentukan tingkat *entertainment* dalam sebuah game adalah adanya AI. AI dalam sebuah game akan membuat permainan menjadi lebih dinamis, sehingga akan meningkatkan *user* dalam bermain. Dikarenakan karakteristik dalam RPG yang memiliki banyak kemungkinan, dibutuhkan algoritma AI yang dapat mengambil keputusan dengan baik. Oleh sebab itu pada penelitian ini akan dilakukan perbandingan kecerdasan buatan yang mengatur perilaku *computer player* dalam mengambil keputusan pada *game* RPG. Kecerdasan buatan yang akan dibandingkan adalah SVM, *Decision Tree*, dan *rulebase*. Diharapkan dengan penelitian ini dapat mengetahui metode manakah yang paling sesuai untuk mengatur perilaku *computer player* pada *game* RPG.

1.2 Perumusan Masalah

Dalam sub-bab ini menjelaskan rumusan masalah yang akan diangkat dalam penelitian ini, yaitu:

1. Bagaimana merancang strategi kecerdasan buatan untuk *computer player* pada *game* berjenis *battle* RPG?
2. Bagaimana pengujian dan analisis tingkat kecerdasan buatan *computer player* pada *game battle* RPG?

1.3 Tujuan dan Manfaat

1.3.1 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini diantaranya adalah:

1. Menganalisis metode perancangan strategi kecerdasan buatan untuk *computer player* pada *game* berjenis *battle* RPG.
2. Menganalisis tingkat kecerdasan buatan *computer player* menggunakan metode SVM, *decision tree* dan *rulebase* pada *game battle* RPG, dan mencari kecerdasan mana yang paling baik dalam pengambilan keputusan.

1.3.2 Manfaat

Beberapa manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Bagi pengembang *game*
Sebagai referensi bagi para pengembang *game* dalam menentukan metode kecerdasan buatan yang paling baik pada *game battle* RPG.

2. Bagi Peneliti Lain

Sebagai referensi bagi peneliti lain yang akan melakukan penelitian mengenai AI pada *game battle* RPG.

1.4 Hipotesis Penelitian

Ha : Ada perbedaan tingkat kecedasan dalam pengambilan keputusan pada metode SVM, *decision tree* dan *rulebase* yang diterapkan pada *game battle* RPG.

Ho : Tidak perbedaan tingkat kecedasan dalam pengambilan keputusan pada metode SVM, *decision tree* dan *rulebase* yang diterapkan pada *game battle* RPG.

1.5 Batasan Masalah

Berdasarkan masalah yang telah disebutkan sebelumnya, perlu diberikan pembatasan masalah agar penelitian lebih terarah dan menghindari meluasnya permasalahan. Berikut adalah pembatasan masalah pada penelitian ini:

1. *Game* diimplementasikan dan diujikan pada platform desktop.
2. Jenis *game* RPG yang dikembangkan berjenis *battle* RPG.
3. Kecerdasan buatan yang dibangun hanya mengatur perilaku *computer player* dalam mengambil keputusan untuk menyerang atau bertahan.
4. Metode kecerdasan buatan yang akan diimplementasikan dan dibandingkan di dalam *game* adalah SVM, *decision tree*, dan *rulebase*.
5. Aspek yang akan diukur dalam pengujian meliputi persentase kemenangan, sisa *health* dan waktu untuk mengalahkan musuh.

[Halaman ini sengaja dikosongkan]

BAB 2

KAJIAN PUSTAKA

2.1 *Game*

Setiap definisi *game* memiliki berbagai kesamaan dan perbedaan (Purkiss 2015). Beberapa definisi tentang *game* diantaranya: *Game* adalah sebuah kegiatan yang melibatkan pengguna kedalam sebuah tujuan yang terikat oleh aturan-aturan (Suits 2005) *Game* diciptakan melalui aturan permainan, yang bergantung pada tindakan pemain (Consalvo 2009). Pada dasarnya definisi dari *game* adalah sebuah sistem kontrol yang bebas di mana didalamnya terdapat suatu pertentangan, dan dibatasi oleh prosedur untuk menghasilkan sebuah tujuan (E.M. Avedon 1981). *Game* adalah sebuah sistem di mana pemain terlibat dalam pertempuran buatan, ditentukan oleh aturan, yang menghasilkan hasil yang terukur (Salen & Zimmerman 2004). *Game* adalah sebuah kegiatan dengan aturan. *Game* adalah bentuk permainan yang melibatkan sebuah interaksi, baik dengan pemain lain, dengan sistem permainan itu sendiri, ataupun dengan nasib dan keberuntungan (Brathwaite & Schreiber 2009). *Game* dapat didefinisikan sebagai suatu bentuk permainan yang memiliki tujuan dan struktur. *Game* menyediakan aktivitas yang menarik minat pemain sekaligus sebagai media hiburan (Maroney 2001). Layanan komputer yang didukung dengan *game* bertujuan untuk memotivasi dan mendukung aktivitas pelatihan pemain (Hamari & Koivisto 2013).

Dari pernyataan tersebut dapat diartikan bahwa *game* adalah kegiatan permainan yang memiliki aturan sebagai batasan dan memiliki tujuan yang harus dicapai (Purkiss & Khaliq 2016). Didalam *game* memiliki konflik atau pertempuran antara pemain dengan pemain lain maupun antara pemain dan sistem *game*, sehingga akan menimbulkan ketertarikan pemain terhadap *game*.

2.2 *Role Playing Game (RPG)*

RPG merupakan sebuah *game* yang memungkinkan pemain untuk dapat menentukan perencanaan strategi dan alur di dalam *game* untuk mencapai suatu tujuan tertentu (Bedoya-rodriguez et al. 2014). Karakteristik dari *game* berjenis

RPG adalah kemungkinan untuk dapat mengembangkan peran karakter yang dipilih yang perkembangannya dan keterampilannya melibatkan pengguna secara langsung di dalam permainan. Fitur inilah yang membuat *game* RPG menjadi lebih dinamis (M. D. Childress & Braswell 2006). Kemajuan teknologi saat ini memungkinkan penggunaan komputer untuk memvisualisasikan permainan *RPG* sehingga memberikan pengalaman bermain dan realisme lebih kepada pemain (J. Patrick Williams, Sean Q. Hendricks 2006).

2.3 Kecerdasan Buatan (*Artificial Intelligent*)

Kecerdasan buatan atau AI (*Artificial Intelligent*) merupakan ilmu tentang bagaimana membangun suatu sistem komputer yang menunjukkan kecerdasan seperti kecerdasan manusia (Millington & Funge 2009). AI merupakan area penelitian yang dinamis dalam topik riset ilmu komputer. Sampai saat ini, telah banyak penelitian mengenai perkembangan AI diantaranya *decision tree*, *neural network*, *evolutionary computing*, *machine learning*, *natural language processing*, dan *object oriented programming* (Millington & Funge 2009). Millington (Millington & Funge 2009) mengungkapkan bahwa terdapat enam model pada game AI yaitu pergerakan, pengambilan keputusan, strategi, infrastruktur, *agent-based AI*, dan *in the book*. Game AI memiliki tujuan untuk mendapatkan sebuah sistem yang dapat menunjukkan perilaku intelektual seperti perilaku manusia.

2.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah sistem klasifikasi yang menggunakan ruang hipotesis berupa fungsi-fungsi linier dalam sebuah ruang fitur (*feature space*), dilatih dengan algoritma pembelajaran yang didasarkan pada teori optimasi dengan mengimplementasikan *learning* yang berasal dari teori statistic (Cristianini & Shawe-Taylor 2000). Teori yang mendasari SVM sendiri sudah berkembang sejak 1960-an, tetapi baru diperkenalkan oleh Vapnik, Boser dan Guyon pada tahun 1992 dan sejak itu SVM berkembang dengan pesat (Boser et al. 1992). SVM merupakan salah satu teknik yang relatif baru dibandingkan dengan teknik lain, tetapi memiliki performansi yang lebih baik di berbagai bidang aplikasi seperti bioinformatics, pengenalan tulisan tangan, klasifikasi teks dan lain sebagainya (Zhang 2001).

Di dalam game, metode SVM digunakan untuk mengontrol *Non-Character Player* (NPC) (Melendez 2009; Melendez 2010). Dimana didalam penelitian tersebut dijelaskan bahwa metode SVM memiliki properti yang sangat cocok untuk mengontrol NPC di dalam game. Dalam penelitian lain yang telah dilakukan oleh (Ralaivola et al. 2005), mereka merancang sebuah game yang dinamakan dengan Game of Go. Di dalam game tersebut metode SVM digunakan untuk menentukan keputusan yang akan di ambil selanjutnya dalam menjalankan permainan. Pada keputusan yang lebih dari dua atau *multiclass* penggunaan SVM juga telah diterapkan pada penelitian yang di lakukan oleh (Melendez 2009; Melendez 2010), pengambilan keputusannya yaitu *explore*, *attack*, dan *run away*. Pada penelitian lain yang dilakukan oleh (Fei & Liu 2006) untuk mengatasi masalah multiclass dengan metode SVM, dimana metode SVM yang di bangun menggunakan SVM *one agains all*. Dalam metode tersebut peneliti membangun k buah model SVM (k adalah jumlah kelas). Setiap model klasifikasi ke-i dilatih dengan menggunakan keseluruhan data, untuk mencari solusi permasalahan.

2.5 Decision Tree

Metode *decision tree* merepresentasikan struktur percabangan pohon dalam membuat strategi keputusan (Ouinlan 1990; Narayek 2004). Deskripsi lain dari metode *decision tree* adalah sebuah metode yang merepresentasikan sebuah hirarki dari beberapa keputusan, pohonnya berawal dengan keputusan yang telah dibuat dan cabang berikutnya berasal dari keputusan yang telah ada (Sriram et al. 2009). Pada penelitian yang telah dilakukan oleh (Stone & Veloso 1997), peneliti menggunakan metode *decision tree* untuk membangun sebuah game robot sepakbola. Dalam penelitian tersebut setiap robot diberi kecerdasan buatan berbasis metode *decision tree* sehingga dapat mengatur perilaku setiap agen atau robot di dalam permainan. Penggunaan metode *decision tree* juga diterapkan pada penelitian yang telah dilakaukan oleh (Sriram et al. 2009), dalam penelitan tersebut metode *decision tree* digunakan untuk membangun sebuah strategi tak terkalahkan dalam permainan Tic-Tac-Toe.

2.6 *Rulebase dan Finite State Machine (FSM)*

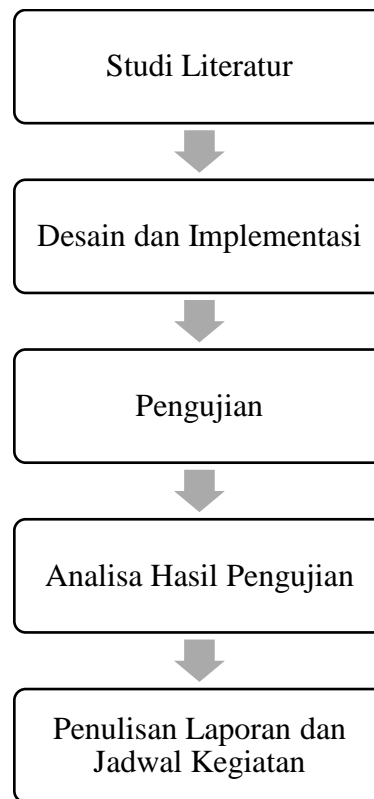
Ada banyak pengertian dari *rulebase* berdasarkan para peneliti, diantaranya yaitu *rulebase* merupakan bahasa script yang dirancang untuk memungkinkan ekspresi dari aturan yang terdiri dari pernyataan kondisional optional dan satu tindakan, dimana pernyataan kondisional tersebut terdiri dari satu atau lebih kondisi yang dikombinasikan dengan logika AND dan OR (Spronck et al. 2004; Spronck et al. 2003). Pengertian lainnya yaitu *rulebase* adalah metode yang terdiri dari sekumpulan *rule* yang digunakan untuk menentukan aksi. *Rulebase* terdiri dari dua bagian penting yaitu bagian kondisi dan bagian aksi (Rostianingsih et al. 2013). Pada penelitian tersebut, Rostianingsih menggunakan metode pemodelan *Finite State Machine (FSM)* untuk memodelkan *rulebase* dari AI gamenya.

Finite State Machine merupakan sebuah metode pemodelan yang mendeskripsikan aksi ke dalam state, dimana *mechine* dapat berpindah pada setiap aksi atau *state* berdasarkan kondisi yang ada (Cass 2002; Narayek 2004). Contohnya berpindah dari *state* menyerang ke dalam *state* menyelamatkan diri dikarenakan *health* akan habis. Pada penelitian yang telah dilakukan oleh (Bimantoro & Haryanto 2016) metode FSM digunakan untuk mengatur perilaku musuh pada game pengenalan unsur kimia. Didalam penelitian tersebut dijelaskan bahwa metode FSM dapat meningkatkan performa AI dengan baik sehingga menambah pengalaman user dalam bermain. Pada penelitian lain yang telah dilakukan oleh (Rostianingsih et al. 2013) menggunakan metode FSM untuk membuat game simulasi pertanian dan peternakan. Metode FSM didalam penelitian tersebut digunakan untuk mengatur NPC dalam melakukan aktifitas seperti menanam, mencangkul, memberi makan ternak dan lain-lain. Pada penelitian tersebut dijelaskan bahwa metode FSM dapat meningkatkan realitas didalam game.

BAB 3

METODE PENELITIAN

Pada bab ini akan dijelaskan mengenai desain, metode atau pendekatan yang digunakan untuk menjawab permasalahan penelitian / studi untuk mencapai tujuan penelitian, serta tahapan penelitian secara rinci. Alur penelitian yang dilakukan pada penelitian ini adalah sebagai berikut: (1) studi literatur, (2) desain dan implementasi, (3) evaluasi, (4) analisa hasil pengujian dan (5) penulisan laporan dan jadwal kegiatan, seperti terlihat pada Gambar 3.1.



Gambar 3. 1. Diagram alur penelitian

3.1 Studi Literatur

Pada tahap ini dilakukan studi literatur terhadap beberapa penelitian yang dianggap berhubungan dan relevan terhadap penelitian ini. Studi literatur dilakukan untuk mengetahui perkembangan penelitian lain dalam kurun beberapa tahun

terakhir pada pembahasan atau topik yang sama dengan topik penelitian yang akan dilakukan.

Pada penelitian ini dilakukan pencarian referensi atau literatur yang berkaitan dengan *game* RPG, *AI game*, SVM, *decision tree*, dan FSM. Dari hasil pencarian literatur yang telah dilakukan, didapatkan hasil sebagai berikut:

1. Tujuan dari pengembangan *game* adalah untuk membuat pengguna menjadi terhibur dan merasakan kesenangan ketika sedang atau telah bermain.
2. *Game* RPG adalah *game* yang memiliki banyak kemungkinan dalam menentukan alur/ jalannya permainan.
3. AI adalah salah satu elemen dalam *game* yang dapat menambah nilai entertainen dalam *game*, karena dengan penerapan *AI game* akan menjadi lebih dinamis.
4. Penerapan AI pada *game* RPG memerlukan algoritma yang sesuai, karena karakteristik dari *game* tersebut yang memiliki banyak kemungkinan.
5. SVM, *decision tree*, dan FSM adalah metode-metode yang dapat diterapkan untuk menentukan perilaku AI dalam mengambil keputusan.

Berdasarkan dari hasil studi literatur yang telah dilakukan, hal yang dapat diangkat menjadi sebuah penelitian adalah: analisis perbandingan kecerdasan buatan yang efektif untuk mengatur perilaku *computer player* dalam mengambil keputusan pada *game battle* RPG.

3.2 Desain dan Implementasi

Pada subbab ini akan dijelaskan mengenai desain dan implementasi sistem yang meliputi analisis kebutuhan, desain sistem, dan implementasi.

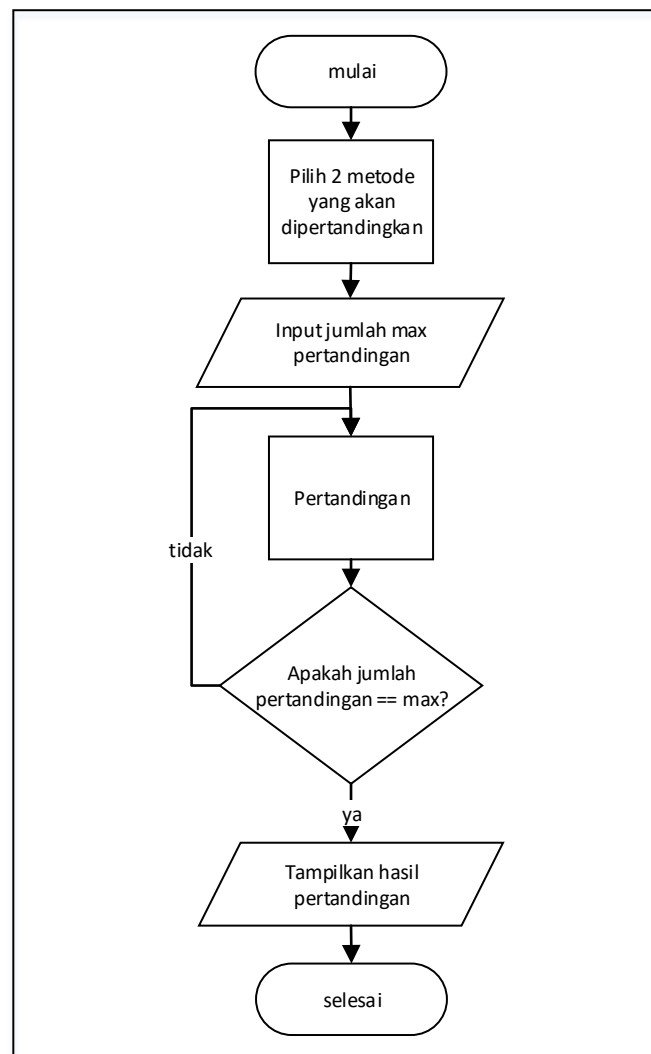
3.2.1 Analisis Kebutuhan

Analisis kebutuhan adalah suatu tahap pengumpulan informasi yang dapat dijadikan sebagai acuan dalam pengembangan *game battle* RPG. Pengumpulan informasi berupa kebutuhan perangkat keras dan perangkat lunak yang digunakan, serta informasi mengenai materi yang akan dijadikan sebagai konten dalam pembuatan *game battle* RPG. Oleh karena pengumpulan kebutuhan akan dilakukan

dengan cara observasi terhadap ahli dalam bidang yang bersangkutan maupun sumber-sumber relevan yang dapat dijadikan sebagai referensi dalam pengembangan *game battle* RPG. Tahap ini dilakukan agar nantinya *game battle* RPG yang dikembangkan sesuai dengan harapan dan kebutuhan penelitian.

3.2.2 Desain Sistem

Setelah melakukan tahap analisis kebutuhan, akan diketahui kebutuhan apa saja yang diperlukan untuk pembuatan dan pengembangan *game battle* RPG. Dari kebutuhan yang telah didapatkan selanjutnya akan direpresentasikan kedalam sebuah desain sistem.



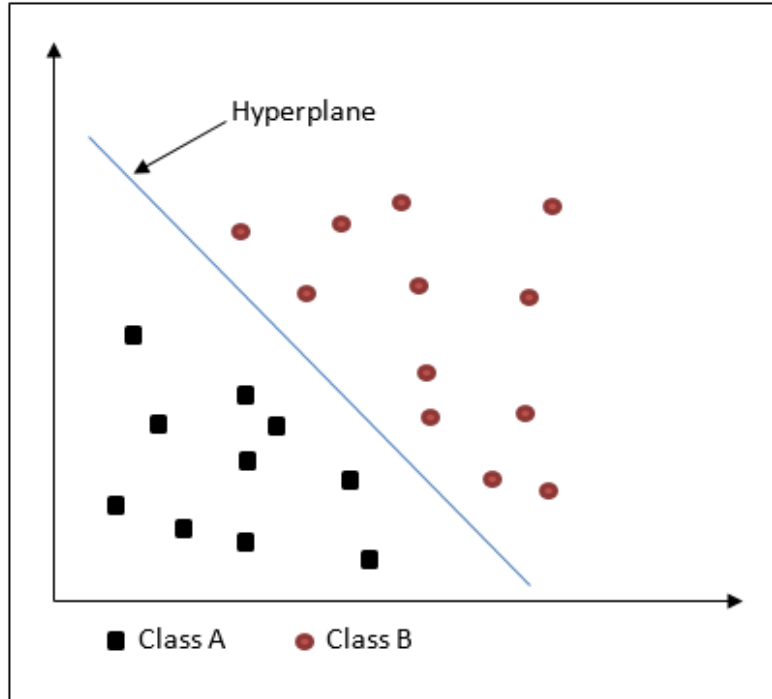
Gambar 3. 2 Diagram alir sistem

Secara garis besar *game battle* RPG dikembangkan dengan memiliki tiga buah kecerdasan buatan yaitu SVM, *decision tree* dan *rulebase*. Ketiga kecerdasan buatan tersebut akan dipertandingkan secara bergantian. Alur dari sistem yang akan dibangun dapat dilihat pada Gambar 3.2. Dimana tahap awal proses sistem adalah memilih dua metode kecerdasan yang akan dipertandingkan, kemudian menginputkan jumlah maksimal pertandingan yang akan dilakukan, selanjutnya sistem akan melakukan pertandingan sebanyak jumlah pertandingan yang telah ditentukan, setelah jumlah pertandingan mencapai jumlah maksimal pertandingan maka pertandingan akan dihentikan dan sistem akan menampilkan hasil dari pertandingan.

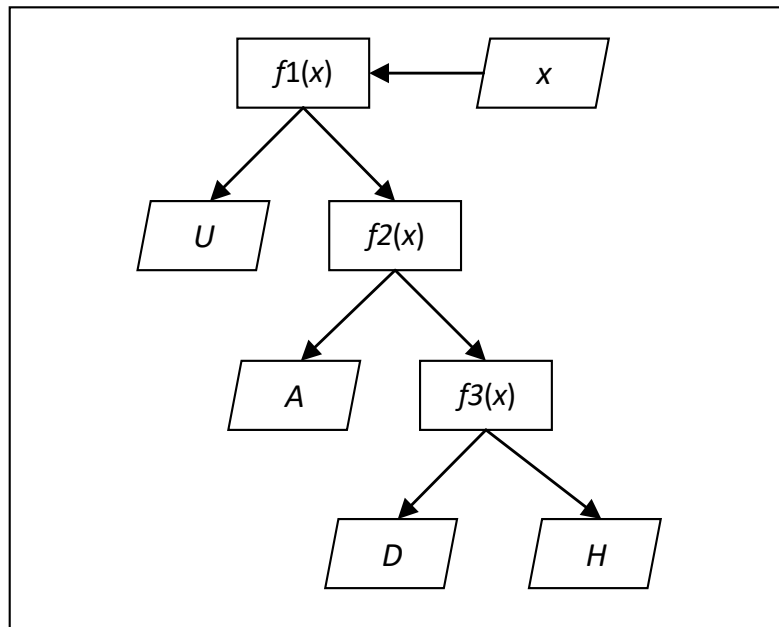
1. SVM

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas. Cara menemukan *hyperplane* tersebut yaitu melalui proses pembelajaran. Proses pembelajarannya menggunakan data training yang telah ada sebelumnya. *Hyperplane* pemisah terbaik antara kedua class dapat ditemukan dengan mengukur margin *hyperplane* tersebut dan mencari titik maksimalnya. Margin adalah jarak antara *hyperplane* dengan data terdekat dari masing-masing kelas. Pada Gambar 3.3 menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua *class*.

Pada penelitian ini terdapat empat keputusan *computer player* yaitu *attack*, *ultimate attack*, *defense*, dan *heal*, maka akan digunakan SVM *one against all*. Dengan menggunakan metode ini, akan dibangun empat buah model SVM. Setiap model akan dilatih dengan keseluruhan dataset untuk mencari solusi permasalahan. Atribut dataset input untuk metode SVM ini meliputi *health*, *energi*, *enemy health*, *damage*, dan *armor*. Model SVM 1 adalah metode untuk mengklasifikasi data apakah masuk kedalam kelas *ultimate attack* atau bukan kelas *ultimate attack*. Model SVM 2 adalah metode untuk mengklasifikasi data apakah masuk kedalam kelas *attack* atau bukan kelas *attack*. Model SVM 3 adalah metode untuk mengklasifikasi data apakah masuk kedalam kelas *defense* atau bukan kelas *heal*. Ilustrasi metode SVM *one against all* ini dapat dilihat pada Gambar 3.4.



Gambar 3. 3 Contoh hasil pemisah dua *class* dengan menggunakan SVM.



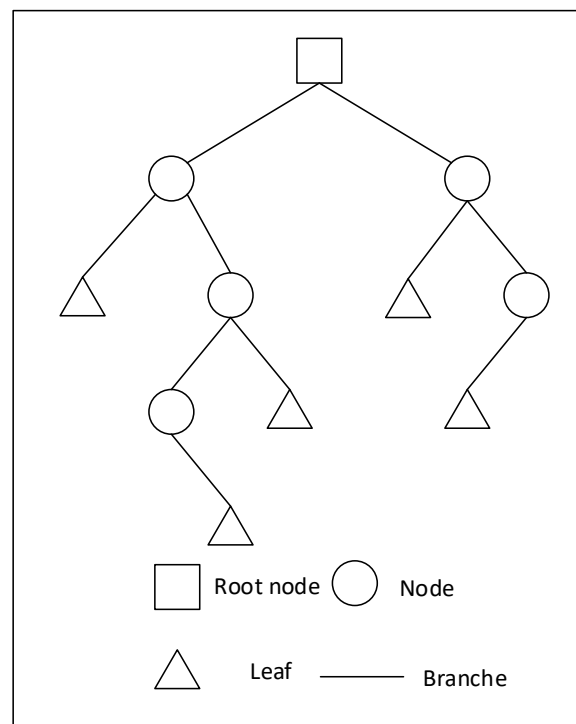
Gambar 3. 4 Ilustrasi metode SVM *one against all*.

Pada Gambar 3.4, x merupakan data input, $f_1(x)$ adalah model SVM 1, U merupakan kelas *ultimate attack*, $f_2(x)$ adalah model SVM 2, A merupakan kelas

attack, $f_3(x)$ adalah model SVM 3, D merupakan kelas *defence*, dan H merupakan kelas *heal*. Fungsi kernel yang digunakan didalam metode SVM ini adalah fungsi kernel polynomial.

2. *Decision Tree*

Decision tree adalah algoritma yang merepresentasikan struktur percabangan pohon dalam membuat strategi keputusan, yaitu terdapat *root* (akar) dan *nodes* (simpul), *branches* (cabang) dan *leaf* (daun). Simpul merupakan representasi dari atribut dan cabang merupakan penghubung antar simpul. Simpul awal dari *decision tree* disebut dengan akar (*root node*) dan simpul akhir dari percabangan *decision tree* adalah daun/ keputusan. Terdapat dua atau lebih cabang dari setiap simpul untuk menghubungkan satu simpul dengan simpul yang lain. Setiap simpul sesuai dengan karakteristik atribut tertentu dan cabangnya sesuai dengan kisaran nilai. Kisaran nilai ini harus memberikan partisi dari karakteristik nilai himpunan yang diberikan. Dalam membangun pohon keputusan digunakan dataset yang telah ada. Gambar 3.5 mendeskripsikan tentang struktur dari metode *decision tree*.



Gambar 3. 5 Struktur *decision tree*

Dalam penelitian ini atribut data input untuk metode *decision tree* ini meliputi *health*, *energi*, *enemy health*, *damage*, dan *armor*. Kemudian akan dikelompokkan menjadi empat kelas yaitu *attack*, *ultimate attack*, *defense*, dan *heal*.

3. Rulebase

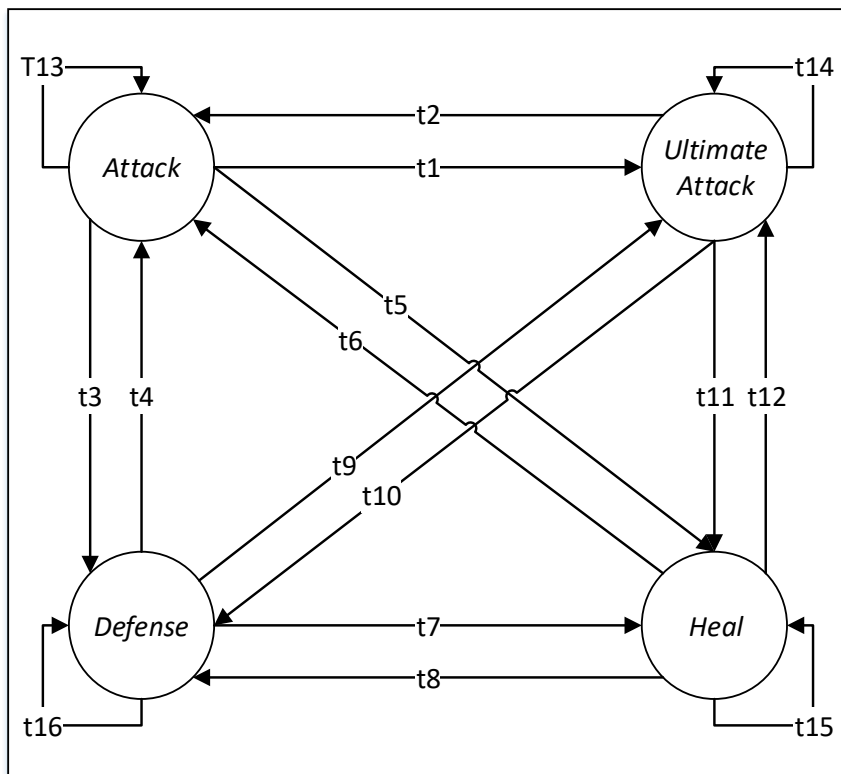
Rulebase adalah sekumpulan *rule* atau kondisi untuk menentukan aksi, didalam game ini ada empat aksi yaitu yaitu *attack*, *ultimate attack*, *defense*, dan *heal*. Aksi *attack* dilakukan jika energi cukup untuk *attack*, energi tidak mencukupi untuk *ultimate attack*, *health* musuh dalam kondisi *low* dan atau *health* dalam kondisi *medium* atau *high*. Aksi *ultimate attack* dilakukan jika energi cukup untuk *ultimate attack*, energi dalam keadaan *high*, dan atau *health* musuh dalam kondisi *low* atau *medium*. Aksi *defense* dilakukan jika *health* dalam kondisi *low*, energi cukup untuk *defense*, dan atau energi tidak cukup untuk *attack*. Aksi *heal* akan dilakukan jika energi tidak mencukupi untuk *attack*, *ultimate attack*, atau *defense*, dan atau *health* dalam kondisi *low*, *medium* atau *high*. Kemudian *rulebase* tersebut akan di modelkan menggunakan diagram FSM.

FSM adalah model yang umum digunakan untuk merancang kecerdasan buatan untuk *computer player* di game. Kelebihan metode ini adalah pada kesederhanaan komputasinya dan kemudahan dalam pemahaman dan implementasinya. Penentuan diagram *state* sesuai dengan jenis perilaku yang sudah ditentukan, dengan kondisi-kondisi tertentu yang dapat memperlancar alur permainan. Setiap *state* dapat berpindah ke state lainnya jika memenuhi kondisi yang telah ditentukan sebelumnya.

Pada Gambar 3.6 ada empat *state* yaitu *attack*, *ultimate attack*, *defense*, dan *heal* yang mungkin terjadi dengan enam belas kondisi (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, dan t16). Pada tiap *state*-nya dapat berpindah dari satu *state* ke *state* yang lain jika kondisi terpenuhi. Kondisi t1 adalah apakah energi cukup untuk *ultimate attack*? Jika ya maka berpindah ke *state ultimate attack*. Kondisi t2 adalah apakah energi tidak mencukupi untuk *ultimate attack*? Jika ya maka berpindah ke *state attack*. Kondisi t3 adalah apakah *health* dalam kondisi *low*? Jika ya maka berpindah ke *state defense*. Kondisi t4 adalah apakah *health*

musuh dalam kondisi *low*? Jika ya maka berpindah ke *state attack*. Kondisi t5 adalah apakah energi tidak mencukupi untuk *attack*? Jika ya maka pindah ke *state heal*.

Kondisi t6 adalah apakah energi cukup untuk *attack*? Jika ya maka pindah ke *state attack*. Kondisi t7 adalah apakah energi tidak mencukupi untuk *defense*? Jika ya maka pindah ke *state heal*. Kondisi t8 adalah apakah energi mencukupi untuk *defense* dan *health* dalam kondisi *low*? Jika ya maka pindah ke *state defense*. Kondisi t9 adalah apakah energi cukup untuk *ultimate attack*? Jika ya maka berpindah ke *state ultimate attack*. Kondisi t10 adalah apakah *health* dalam kondisi *low* dan energi cukup untuk *defense*? Jika ya maka berpindah ke *state defense*. Kondisi t11 adalah apakah energi tidak cukup untuk *ultimate attack* dan energi dalam kondisi *medium*? Jika ya maka berpindah ke *state heal*. Kondisi t12 adalah apakah energi cukup untuk *ultimate attack* dan energi dalam kondisi *high*? Jika ya maka berpindah ke *state ultimate attack*.



Gambar 3. 6 Diagram FSM.

Kondisi t13 adalah apakah energi cukup untuk attack dan energi tidak cukup untuk *ultimate attack* dan health dalam kondisi high atau medium? Jika ya maka tetap di *state attack*. Kondisi t14 adalah apakah energi cukup untuk *ultimate attack* dan health musuh dalam keadaan *low* atau *medium*? Jika ya maka tetap di *state ultimate attack*. Kondisi t15 adalah apakah energi tidak cukup untuk *attack* atau *ultimate attack* atau *defense*? Jika ya maka tetap di *state heal*. Kondisi t16 adalah apakah energi cukup untuk *defense* dan keadaan *health low* dan energi tidak cukup untuk *attack*? Jika ya maka tetap di *state defense*.

3.2.3 Implementasi

Implementasi yang dimaksud pada tahap ini adalah implementasi setelah dilakukan analisis kebutuhan dan desain. Rancangan *game* yang telah dipersiapkan kemudian diimplementasikan dalam bahasa pemrograman sehingga semua fungsi dapat dijalankan dengan baik oleh pengguna. Dalam tahap ini kebutuhan yang diperlukan adalah perangkat lunak/*software* sebagai alat bantu pembuatan desain antar muka, pengolah suara dan penulisan kode program. Selain itu juga diperlukan perangkat keras sebagai lingkungan pengembangan dan pengujian fungsi pada *game battle* RPG. Produk dalam penelitian ini adalah berupa *game battle* RPG.

3.3 Pengujian

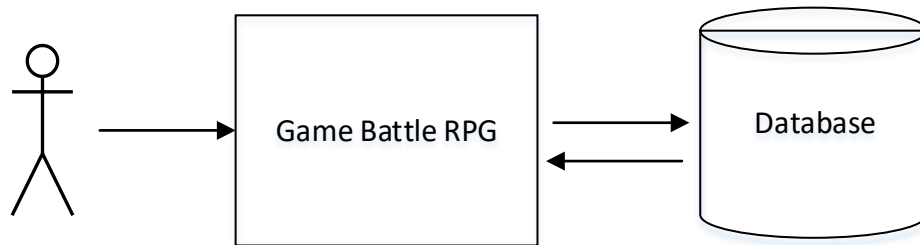
Pada subbab ini akan dijelaskan tentang metode pengujian sistem dan lingkungan uji coba terhadap sistem yang akan dibuat.

3.3.1 Pembuatan Dataset

Diperlukan adanya dataset untuk dipergunakan untuk klasifikasi sebagai data *training*. Pada penelitian ini dataset digunakan untuk *training* metode SVM dan *decision tree*. Dataset tersebut berisi nilai-nilai dari atribut dataset yaitu *health*, energi, *enemy health*, *damage*, dan *armor*.

Dataset dibangun dari percobaan *user* melawan metode kecerdasan buatan *rulebase*. Dimana *user* yang dipilih merupakan pakar dalam *game*. Pada setiap proses pengumpulan data, akan disimpan perlakuan user dalam menghadapi musuh didalam *game battle* RPG. Kemudian akan dipilih data-data yang memiliki skenario

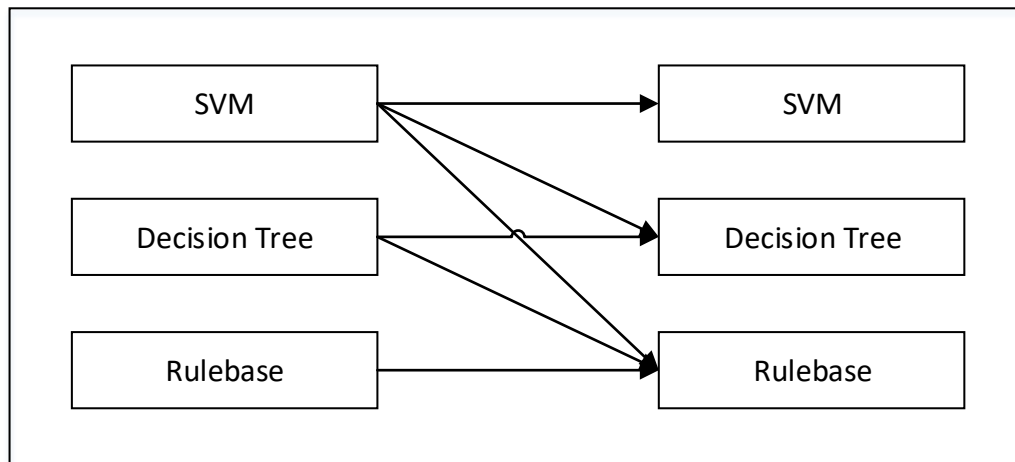
kemenangan didalam setiap pertandingan. Sehingga akan menghindari skenario kekalahan dalam pertandingan. Gambaran proses pembuatan dataset dapat dilihat pada Gambar 3.7.



Gambar 3. 7 Proses pembuatan dataset

3.3.2 Skenario Pengujian

Pengujian dilakukan dengan cara mengimplementasikan dan membandingkan secara langsung ketiga buah metode kecerdasan tersebut didalam permainan. Skenario pengujiannya adalah sebagai berikut:



Gambar 3. 8 Alur skenario percobaan kecerdasan buatan *game battle* RPG.

Skenario 1: membandingkan antara metode kecerdasan buatan SVM dengan *decision tree*. Skenario 2: membandingkan antara metode kecerdasan buatan SVM dengan *rulebase*. Skenario 3: membandingkan antara metode kecerdasan buatan *decision tree* dengan *rulebase*. Skenario 4: membandingkan antara metode kecerdasan buatan SVM dengan SVM. Skenario 5: membandingkan antara metode kecerdasan buatan *decision tree* dengan *decision tree*. Skenario 6:

membandingkan antara metode kecerdasan buatan *rulebase* dengan *rulebase*. Gambaran alur percobaannya dapat dilihat pada Gambar 3.9.

Setiap skenario akan di ujikan sebanyak 100 kali. Dalam setiap pengujian skenario akan dihitung presentase kemenangan *computer player*. Untuk menentukan giliran awal digunakan metode random.

3.3.3 Analisis Hasil Pengujian

Langkah yang dilakukan dalam melakukan analisis hasil pengujian antar metode adalah mencari persentasi kemenangan setiap metode kecerdasan buatan dengan menggunakan persamaan 1,

$$\text{persentase kemenangan} = \frac{\text{jumlah kemenangan}}{\text{jumlah semua pertandingan}} \times 100\% \quad (1)$$

sehingga persentasi kemenangan dapat di skalakan dari 0 hingga 100, dimana jika *computer player* mendapat persentasi mendekati 0 maka dapat diartikan bahwa *computer player* tersebut banyak mengalami kekalahan dalam setiap pertandingan. Sedangkan jika *computer player* mendapat persentasi mendekati 100 maka dapat diartikan bahwa *computer player* tersebut mendapat banyak kemenangan di setiap pertandingannya. Diasumsikan bahwa metode kecerdasan buatan yang memiliki persentase kemenangan tertinggi adalah metode kecerdasan yang paling baik.

[Halaman ini sengaja dikosongkan]

BAB 4

HASIL DAN PEMBAHASAN

Pada bab ini dikemukakan hasil pembuatan dan pengembangan, pengujian, analisis data serta pembahasan. Setiap hasil yang diperoleh akan dilakukan analisa dan pembahasan untuk mendapatkan suatu informasi. Informasi yang diperoleh akan mempengaruhi hasil kesimpulan.

4.1 Hasil Desain dan Implementasi

Sub-bab yang pertama akan menjelaskan mengenai hasil pembuatan *game battle* RPG yang terdiri dari hasil analisis kebutuhan, hasil desain sistem dan hasil implementasi.

4.1.1 Hasil Analisis Kebutuhan

Analisi kebutuhan dilakukan dengan dua cara yaitu melakukan observasi dan mencari referensi yang relevan dengan *game battle* RPG yang akan dibuat. Observasi dilakukan terhadap pihak yang berhubungan dengan penelitian yaitu developer, desainer *game* dan pakar kecerdasan buatan. Observasi terhadap developer desainer *game* dan pakar kecerdasan buatan juga dilakukan dengan cara wawancara. Hal ini dilakukan untuk mengetahui gambaran mengenai fungsi, aturan main, skenario dan desain *game* yang akan dibuat.

Cara selanjutnya dalam tahapan pengumpulan kebutuhan yaitu dengan mencari referensi yang relevan terhadap *game battle* RPG yang akan dibuat. Dalam hal ini sumber referensi yang akan digunakan yaitu referensi mengenai fungsionalitas dan desain *game*. Referensi mengenai desain pada *game* yang akan dikembangkan berasal dari *game-game* dengan jenis yang sama atau hampir sama yang sudah ada sebelumnya dipasaran. Desain *game* yang diamati dari *game-game* tersebut meliputi: fungsionalitas, aturan main, skenario dan tampilan antar muka. Pada penelitian ini referensi *game* diambil dari sejumlah *game* yang sudah ada di Google Playstore. Tujuan dari melihat dan mengamati desain pada *game-game* tersebut adalah untuk mempermudah dalam pembuatan desain dari *game battle* RPG. Referensi mengenai kecerdasan buatan pada *game* yang akan dikembangkan

berasal dari penelitian-penelitian serupa yang membahas tentang kecerdasan buatan *game*.

Dari proses analisis kebutuhan yang sudah dilakukan didapatkan kebutuhan fungsionalitas dalam pembuatan *game battle* RPG sebagai berikut:

1. Peneliti dapat melihat *gameplay* yang disajikan dalam *game battle* RPG.
2. Peneliti dapat memilih *computer player* dengan kecerdasan buatan mana yang akan dipertandingkan didalam *game battle* RPG.
3. Peneliti dapat menentukan jumlah pertandingan dalam setiap percobaan.
4. Desain dan tampilan antar muka pada *game battle* RPG disesuaikan dengan jenis *game battle* RPG.
5. Pengambilan keputusan pada *computer player* pada *game battle* RPG harus dapat dilakukan dengan cepat dan akurat.
6. Kecerdasan buatan pengambil keputusan pada *game battle* RPG adalah SVM, *decision tree* dan *rulebase*.

4.1.2 Hasil Desain Sistem

Desain sistem adalah tahap dimana hasil dari analisis kebutuhan yang telah diperoleh kemudian dijadikan sebagai representasi dari desain *game battle* RPG yang akan dikembangkan. Berdasarkan hasil dari kebutuhan yang diperoleh maka didapatkan hasil dari pembuatan desain sistem pada *game battle* RPG sebagai berikut:

1. Konten utama pada *game* adalah berupa permainan RPG. Desain permainan RPG yang digunakan adalah tipe *battle* dimana terdapat dua buah *computer player* yang saling menyerang.
2. Setiap *computer player* memiliki kecerdasan buatan yang berbeda untuk mengambil keputusan. Dalam hal ini kecerdasan buatan SVM, *decision tree* dan *rulebase*.
3. Dataset digunakan sebagai data training kecerdasan buatan yang akan dipakai oleh kecerdasan buatan untuk mengambil keputusan.
4. Terdapat *reward* berupa penambahan nilai serangan dan poin.
5. Terdapat *punishment* berupa pengurangan nilai serangan dan poin.

6. Terdapat indicator untuk mengetahui status *computer player* didalam permainan.
7. Konten RPG didalam *game* diterapkan dengan adanya pengembangan status atribut pada *computer player*.

4.1.3 Hasil Implementasi

Setelah desain dari *game battle* RPG telah dibuat, tahap selanjutnya adalah mengimplementasikan desain tersebut menjadi sebuah aplikasi perangkat lunak. Implementasi dilakukan dengan bantuan sejumlah perangkat lunak diantaranya:

1. *Adobe Flash CS6* sebagai alat bantu penulisan kode program, pembuatan animasi dan pembuatan antar muka. Dalam hal ini *game battle* RPG yang dikembangkan dibuat dengan menggunakan bahasa pemrograman *Action Script 2*.
2. *Netbeans 8* sebagai alat bantu penulisan kode program, pembuatan kecerdasan buatan.
3. *XAMPP* sebagai alat bantu pembuatan database.
4. *MDM Zinc* sebagai alat bantu koneksi aplikasi dengan database.
5. *Audacity* versi 2.1.3 sebagai alat bantu dalam pengolahan audio.
6. *PhotoScape* versi 3.7 sebagai alat bantu dalam pengolahan gambar bertipe bitmap.
7. *Adobe Illustrator CS6* sebagai alat bantu dalam pengolahan gambar bertipe vector.

Selain menggunakan bantuan perangkat lunak, pengembangan *game battle* RPG juga dibuat dengan bantuan perangkat keras yaitu seperangkat komputer dengan spesifikasi: Sistem Operasi Microsoft Windows 10 64bit, Processor Core i3-6100 3.7Ghz, RAM 2666Mhz 8GB, Hardisk 1TB, dan VGA NVIDIA GTX 1060 6GB. Perangkat ini digunakan sebagai lingkungan pengembangan dan pengujian *game battle* RPG. Sejumlah konten yang ada pada *game battle* RPG juga berasal dari beberapa sumber diantaranya: sejumlah konten vector dan gambar diambil dari situs www.freepik.com, sejumlah konten audio diambil dari situs www.freesound.org.

Game battle RPG dibuat dalam bentuk dua dimensi dan diimplementasikan pada *platform desktop* yaitu windows. Hasil implementasi *game* yang telah dikembangkan akan dijelaskan dalam dua bagian yaitu: hasil pembuatan antar muka dan hasil pembuatan sistem.

4.1.1.1 Hasil Pembuatan Antarmuka

Berdasarkan tampilan hasil pembuatan antarmuka yang terlihat pada Gambar 4.1, Gambar 4.2 dan Gambar 4.3, terdapat komponen-komponen pada *game battle* RPG ini. Berikut ini adalah penjelasan lebih detail mengenai komponen-komponen tersebut:

1. Tombol yang berisi nama-nama metode sehingga dapat dipilih metode kecerdasan untuk *computer player 1* dan *computer player 2* yang akan di pertandingan.
2. Textbox yang berfungsi sebagai inputan nilai jumlah pertandingan yang akan dilakukan.
3. Tombol mulai berfungsi untuk memulai pertandingan.
4. 2 buah karakter yang memiliki kemampuan (*skill*) untuk menyerang super (*ultimate attack*), menyerang (*attack*), bertahan (*defense*), dan menambah energi (*heal*). Nama karakter sesuai dengan metode kecerdasan buatan yang digunakan oleh karakter tersebut.
5. Timer yang divisualisasikan dalam bentuk angka, untuk memberikan informasi berapa lama waktu yang telah berjalan dalam *battle*.
6. Indikator sisa *health* dan mana yang divisualisasikan dengan angka dan bar, dimana bar hijau untuk indikator sisa *health* dan bar biru untuk indikator sisa mana.
7. Indikator *skill* yang sedang digunakan, divisualisasikan dalam bentuk huruf, U untuk *skill ultimate attack*, A untuk *skill attack*, D untuk *skill defense*, dan H untuk *skill heal*.
8. Indikator *armor* yang divisualisasikan dengan bentuk tameng (*shield*) dan angka.
9. Indikator *damage* yang divisualisasikan dengan bentuk dua pedang dan angka.

10. Halaman status hasil pertandingan yang menunjukkan jumlah kemenangan dari *computer player 1* dan *computer player 2*.
11. Tombol selesai berfungsi untuk kembali ke menu utama.



Gambar 4. 1 Tampilan menu utama.



Gambar 4. 2 Tampilan *game battle* RPG.



Gambar 4. 3 Tampilan hasil pertandingan

4.1.1.2 Hasil Pembuatan Sistem

Desain sistem adalah tahap dimana hasil dari analisis kebutuhan yang telah diperoleh kemudian dijadikan sebagai representasi dari desain *game battle* RPG yang akan dikembangkan. Berdasarkan hasil dari kebutuhan yang diperoleh maka didapatkan hasil dari pembuatan desain sistem sebagai berikut:

1. SVM

Berikut ini adalah hasil rancangan sistem berupa *pseudo-code* dari metode SVM yang dapat dilihat pada Gambar 4.4. Pada proses awal metode SVM data inputan yang digunakan adalah nilai dari *health*, energi, *enemy health*, *damage* dan *armor*. Selanjutnya akan nilai tersebut akan di proses oleh metode SVM sehingga akan menghasilkan keputusan. Keputusan yang di dapatkan selanjutnya akan dicek apakah keputusan tersebut adalah A dan energi cukup untuk memenuhi *cost attack*, jika memenuhi maka akan di cek apakah *enemy flag defense* bernilai *true*, jika memenuhi maka akan mengurangi nilai *enemy health* sebesar *damage attack* di tambah *damage* dan dikurangi *enemy armor*. Jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost attack*, dan mengubah *enemy flag defense* menjadi *false*. Jika tidak memenuhi maka nilai *enemy health* hanya akan dikurangi

sebesar *damage attack* di tambah *damage* saja. Dan jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost attack*.

```

function SVM(){
    keputusan = SVM(Health, Energi, EnemyHealth, Damage, Armor);
    if(keputusan == "A" && Energi >= CostAttack) {
        if (EnemyFlagDefense == true){
            EnemyHealth = EnemyHealth - ((DamageAttack + Damage) -
EnemyArmor);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostAttack;
            EnemyFlagDefense = false;
        }
        else {
            EnemyHealth = EnemyHealth - (DamageAttack + Damage);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostAttack;
        }
    }
    else if(keputusan == "U" && Energi >= CostUltimate){
        if (EnemyFlagDefense == true){
            EnemyHealth = EnemyHealth - ((DamageUltimate + Damage)
- EnemyArmor);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostUltimate;
            EnemyFlagDefense = false;
        }
        else {
            EnemyHealth = EnemyHealth - (DamageUltimate + Damage);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostUltimate;
        }
    }
    else if(keputusan == "D" && Energi >= CostDefense) {
        flagDefPlayer1 = true;
        Energi = Energi - CostDefense;
    }
    else{
        Energi = Energi + 20;
        if (Energi >= 100) Energi = 100;
    }
}

```

Gambar 4. 4 *Pseudo-code* fungsi SVM.

Kemudian keputusan yang di dapatkan akan dicek kembali apakah keputusan tersebut adalah U dan dan energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka *state* akan di beri status *ultimate*, kemudian akan di cek apakah *enemy flag defense* bernilai *true*, jika memenuhi maka akan mengurangi nilai *enemy health* sebesar *damage ultimate* di tambah *damage* dan dikurangi *enemy armor*. Jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost ultimate*, dan mengubah *enemy flag defense* menjadi *false*. Jika tidak memenuhi maka nilai *enemy health* hanya akan dikurangi sebesar *damage ultimate* di tambah *damage* saja. Dan jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost ultimate*. Kondisi selanjutnya yaitu apakah keputusan tersebut adalah D dan dan energi cukup untuk memenuhi *cost defense*, jika memenuhi maka *flag defense* akan dirubah menjadi *true* dan energi akan dikurangi sebesar *cost defense*. Jika semua kondisi diatas tidak memenuhi maka sistem akan langsung memutuskan *heal* dengan menambah energi sebesar 20 dan jika nilai energi melebihi 100 maka akan dinormalisasi menjadi 100.

2. *Decision tree*

Berikut ini adalah hasil rancangan sistem berupa *pseudo-code* dari metode *decision tree* yang dapat dilihat pada Gambar 4.4. Pada proses awal metode *decision tree* data inputan yang digunakan adalah nilai dari *health*, energi, *enemy health*, *damage* dan *armor*. Selanjutnya akan nilai tersebut akan di proses oleh metode *decision tree* (J48) sehingga akan menghasilkan keputusan. Keputusan yang di dapatkan selanjutnya akan dicek apakah keputusan tersebut adalah A dan energi cukup untuk memenuhi *cost attack*, jika memenuhi maka akan di cek apakah *enemy flag defense* bernilai *true*, jika memenuhi maka akan mengurangi nilai *enemy health* sebesar *damage attack* di tambah *damage* dan dikurangi *enemy armor*. Jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost attack*, dan mengubah *enemy flag defense* menjadi *false*. Jika tidak memenuhi maka nilai *enemy health* hanya akan dikurangi sebesar *damage attack* di tambah *damage*

saja. Dan jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost attack*.

```

function DecisionTree(){
    keputusan = J48(Health, Energi, EnemyHealth, Damage, Armor);
    if(keputusan == "A" && Energi >= CostAttack) {
        if (EnemyFlagDefense == true){
            EnemyHealth = EnemyHealth - ((DamageAttack + Damage) -
EnemyArmor);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostAttack;
            EnemyFlagDefense = false;
        }
        else {
            EnemyHealth = EnemyHealth - (DamageAttack + Damage);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostAttack;
        }
    }
    else if(keputusan == "U" && Energi >= CostUltimate){
        if (EnemyFlagDefense == true){
            EnemyHealth = EnemyHealth - ((DamageUltimate + Damage)
- EnemyArmor);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostUltimate;
            EnemyFlagDefense = false;
        }
        else {
            EnemyHealth = EnemyHealth - (DamageUltimate + Damage);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostUltimate;
        }
    }
    else if(keputusan == "D" && Energi >= 10) {
        flagDefPlayer1 = true;
        Energi = Energi - costDef;
    }
    else{
        Energi = Energi + 20;
        if (Energi >= 100) Energi = 100;
    }
}

```

Gambar 4. 5 Pseudo-code fungsi *decision tree*

Kemudian keputusan yang di dapatkan akan dicek kembali apakah keputusan tersebut adalah U dan dan energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka *state* akan di beri status *ultimate*, kemudian akan di cek apakah *enemy flag defense* bernilai *true*, jika memenuhi maka akan mengurangi nilai *enemy health* sebesar *damage ultimate* di tambah *damage* dan dikurangi *enemy armor*. Jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost ultimate*, dan mengubah *enemy flag defense* menjadi *false*. Jika tidak memenuhi maka nilai *enemy health* hanya akan dikurangi sebesar *damage ultimate* di tambah *damage* saja. Dan jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost ultimate*. Kondisi selanjutnya yaitu apakah keputusan tersebut adalah D dan dan energi cukup untuk memenuhi *cost defense*, jika memenuhi maka *flag defense* akan dirubah menjadi *true* dan energi akan dikurangi sebesar *cost defense*. Jika semua kondisi diatas tidak memenuhi maka sistem akan langsung memutuskan *heal* dengan menambah energi sebesar 20 dan jika nilai energi melebihi 100 maka akan dinormalisasi menjadi 100.

3. Rulebase

Berikut ini adalah hasil rancangan sistem berupa *pseudo-code* dari metode *rulebase* dapat dilihat pada Gambar 4.6. *Pseudo-code* dari keputusan *ultimate* dapat dilihat pada Gambar 4.7. *Pseudo-code* dari keputusan *attack* dapat dilihat pada Gambar 4.8. *Pseudo-code* dari keputusan *defense* dapat dilihat pada Gambar 4.9. *Pseudo-code* dari keputusan *heal* dapat dilihat pada Gambar 4.10.


```

function Rulebase()
{
  if (Health <= 20){
    if (Energi >= CostUltimate) Ultimate();
    else if (Energi >= CostAttack && EnemyHealth <= 20) Attack();
    else if (Energi >= CostDefense) Defense();
    else Heal();
  }
  else if (State == "Ultimate" || Energi >= CostUltimate) Ultimate();
  else if (State == "Attack" || Energi >= CostAttack) Attack();
  else if (State == "Defense" || Energi >= CostDefense) Defense();
  else Heal();
}

```

Gambar 4. 6 *Pseudo-code* fungsi *rulebase*.

Pada proses awal metode *rulebase* data inputan yang digunakan adalah nilai dari *health*, energi dan *enemy health*. Selanjutnya akan di cek apakah *health* lebih kecil atau sama dengan 20, apabila memenuhi maka akan di cek kembali apakah energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka akan menuju ke *state ultimate*. Jika tidak memenuhi maka akan dicek kembali apakah energi cukup untuk memenuhi *cost attack* dan *enemy health* lebih kecil atau sama dengan 20, jika memenuhi maka akan menuju ke *state attack*. Jika tidak memenuhi maka akan di cek kembali apakah energi cukup untuk memenuhi *cost defense*, jika memenuhi maka akan menuju ke *state defense*. Jika tidak memenuhi maka akan langsung menuju ke *state heal*. Kondisi selanjutnya yang akan di cek adalah apakah *state* sebelumnya adalah *state ultimate* atau energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka akan menuju ke *state ultimate*. Jika tidak memenuhi maka akan dicek kembali apakah *state* sebelumnya adalah *state attack* atau energi cukup untuk memenuhi *cost attack*, jika memenuhi maka akan menuju ke *state attack*. Jika tidak memenuhi maka akan di cek kembali apakah *state* sebelumnya adalah *state defense* atau energi cukup untuk memenuhi *cost defense*, jika memenuhi maka akan menuju ke *state defense*. Jika tidak memenuhi maka akan langsung menuju ke *state heal*.

```

function Ultimate(){
    if (Energi >= CostUltimate){
        State = "Ultimate";
        if (EnemyFlagDefense == true){
            EnemyHealth = EnemyHealth - ((DamageUltimate + Damage) - EnemyArmor);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostUltimate;
            EnemyFlagDefense = false;
        }
        else {
            EnemyHealth = EnemyHealth - (DamageUltimate + Damage);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostUltimate;
        }
    }
    else if (Energi >= CostAttack) Attack();
    else if (Energi >= CostDefense) Defense();
    else Heal();
}

```

Gambar 4. 7 Pseudo-code fungsi *ultimate*.

Didalam fungsi *state ultimate* akan ada beberapa kondisi lagi yang harus di penuhi yaitu, sistem akan mengecek apakah energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka *state* akan di beri status *ultimate*, kemudian akan di cek apakah *enemy flag defense* bernilai *true*, jika memenuhi maka akan mengurangi nilai *enemy health* sebesar *damage ultimate* di tambah *damage* dan dikurangi *enemy armor*. Jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost ultimate*, dan mengubah *enemy flag defense* menjadi *false*. Jika tidak memenuhi maka nilai *enemy health* hanya akan dikurangi sebesar *damage ultimate* di tambah *damage* saja. Dan jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost ultimate*. Kondisi selanjutnya yaitu apakah energi cukup untuk memenuhi *cost attack*, jika memenuhi maka akan menuju ke *state attack*. Jika tidak memenuhi maka akan di cek kembali apakah energi cukup untuk memenuhi *cost defense*, jika memenuhi maka akan menuju ke *state defense*. Jika tidak memenuhi maka akan langsung menuju ke *state heal*.

```

function Attack()
{
    if (Energi >= CostUltimate) Ultimate();
    else if (Energi >= CostAttack)
    {
        State = "Attack";
        if (EnemyFlagDefense == true){
            EnemyHealth = EnemyHealth - ((DamageAttack + Damage) - EnemyArmor);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostAttack;
            EnemyFlagDefense = false;
        }
        else {
            EnemyHealth = EnemyHealth - (damageAttackPlayer1 +
Damage);
            if (EnemyHealth <= 0) EnemyHealth = 0;
            Energi = Energi - CostAttack;}
        }
    else if (Energi >= CostDefense) Defense();
    else Heal();
}

```

Gambar 4. 8 *Pseudo-code* fungsi *attack*.

Didalam fungsi *state attack* juga akan ada beberapa kondisi lagi yang harus di penuhi yaitu, sistem akan mengecek apakah energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka akan menuju ke *state ultimate*. Kondisi selanjutnya yaitu apakah energi cukup untuk memenuhi *cost attack*, jika memenuhi maka *state* akan di beri status *attack*, kemudian akan di cek apakah *enemy flag defense* bernilai *true*, jika memenuhi maka akan mengurangi nilai *enemy health* sebesar *damage attack* di tambah *damage* dan dikurangi *enemy armor*. Jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost attack*, dan mengubah *enemy flag defense* menjadi *false*. Jika tidak memenuhi maka nilai *enemy health* hanya akan dikurangi sebesar *damage attack* di tambah *damage* saja. Dan jika nilai *enemy health* hasil operasi tersebut lebih kecil dari nol maka akan di normalisasi menjadi nol. Kemudian nilai energi juga akan dikurangi sebesar *cost attack*. Kondisi selanjutnya adalah apakah energi cukup untuk memenuhi *cost defense*, jika memenuhi maka akan menuju ke *state defense*. Jika tidak memenuhi maka akan langsung menuju ke *state heal*.

```

function Defense()
{
    if (Energi >= CostUltimate) ulti();
    else if ((Energi >= CostAttack && Health >= 20) || (Energi >= CostAttack &&
EnemyHealth <= 20))
    {
        Attack();
    }
    else if (Health >= 50) Heal();
    else if (Energi >= CostDefenseense)
    {
        State = "Defense";
        FlagDefense = true;
        Energi = Energi - CostDefense;
    }
    else Heal();
};

```

Gambar 4. 9 *Pseudo-code* fungsi *defense*.

Didalam fungsi *state defense* juga akan ada beberapa kondisi lagi yang harus di penuhi yaitu, sistem akan mengecek apakah energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka akan menuju ke *state ultimate*. Kondisi selanjutnya yaitu apakah energi cukup untuk memenuhi *cost attack*, jika memenuhi maka akan menuju ke *state attack*. Jika tidak memenuhi maka akan di cek kembali apakah energi cukup untuk memenuhi *cost defense*, jika memenuhi maka *state* akan di beri status *defense*, *flag defense* akan dirubah menjadi *true* dan energi akan dikurangi sebesar *cost defense*. Jika tidak memenuhi maka akan langsung menuju ke *state heal*.

```

function Heal(){
    if (Energi >= CostUltimate) Ultimate();
    else if (Energi >= CostAttack && Health >= 20) Attack();
    else if (Energi >= CostDefense && Health <= 20) def();
    else
    {
        State = "Heal";
        Energi = Energi + 20;
        if (Energi >= 100) Energi = 100;
    }
}

```

Gambar 4. 10 *Pseudo-code* fungsi *heal*.

Didalam fungsi *state heal* juga akan ada beberapa kondisi lagi yang harus di penuhi yaitu, sistem akan mengecek apakah energi cukup untuk memenuhi *cost ultimate*, jika memenuhi maka akan menuju ke *state ultimate*. Kondisi selanjutnya yaitu apakah energi cukup untuk memenuhi *cost attack*, jika memenuhi maka akan menuju ke *state attack*. Jika tidak memenuhi maka akan di cek kembali apakah energi cukup untuk memenuhi *cost defense*, jika memenuhi maka akan menuju ke *state defense*. Jika tidak memenuhi maka *state* akan di beri status *heal*, energi akan ditambahkan nilai sebesar 20 dan jika nilai energi melebihi 100 maka akan dinormalisasi menjadi 100.

4.2 Hasil Pengujian

Sub-bab ini akan menjelaskan tentang hasil dari pengujian dari game yang telah dibuat. Hasil dari pengujian ini mencakup hasil pembuatan dataset yang digunakan dalam pengujian, hasil dari skenario pengujian yang telah dilakukan dan hasil dari analisis hasil uji coba yang telah didapatkan.

4.2.1 Hasil Pembuatan Dataset

Dataset yang digunakan di dalam *game* ini ada 1, dataset tersebut digunakan untuk metode kecerdasan buatan SVM dan metode *decision tree*. Atribut-atribut dari dataset ini adalah *health*, energi, *enemy health*, *damage*, dan *armor*. Dataset yang digunakan untuk setiap metode SVM dan *decision tree* berisi data sebanyak 156 data. Nilai maksimum dari atribut *health* adalah sebesar 300, untuk atribut energi memiliki nilai maksimum sebesar 100 dan nilai minimum sebesar 0, untuk atribut *enemy health* memiliki nilai maksimum sebesar 300, untuk atribut *damage* memiliki nilai minimum sebesar 0, dan atribut *armor* memiliki nilai minimum sebesar 0. Contoh dataset dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Contoh sampel dataset.

No.	<i>Health</i>	<i>Energi</i>	<i>Enemy_Health</i>	<i>Damage</i>	<i>Armor</i>	<i>Decision</i>
1	300	100	300	0	0	U
2	260	60	260	0	2	U
3	220	20	220	0	4	H
4	200	40	220	0	4	U
5	200	0	180	0	6	H
6	200	20	180	0	6	H
7	180	40	180	0	6	A
8	160	20	160	2	6	A
9	160	0	138	2	8	H
10	140	20	138	2	8	A
11	140	0	116	2	10	H
12	118	20	116	2	10	H
13	118	40	116	2	10	U
14	118	0	74	4	10	H
15	118	20	74	4	10	H
16	96	40	74	4	10	U
17	96	0	30	6	10	H
18	74	20	30	6	10	A
19	74	20	30	6	10	A
20	50	0	4	8	10	H
21	50	20	4	8	10	A
22	300	100	300	0	0	U
23	260	60	260	0	2	U
24	218	20	220	2	2	A
25	194	0	198	2	4	H
26	194	20	198	2	4	H
27	168	40	198	2	4	U
28	168	0	156	4	4	H
29	142	20	156	4	4	D
30	142	10	156	4	4	H
31	120	30	156	4	4	D
32	120	20	156	4	4	H
33	98	40	156	4	4	U
34	98	0	112	4	6	H
35	72	20	112	4	6	A
36	72	0	88	6	6	H
37	44	20	88	6	6	A
38	44	0	62	8	6	H
39	14	20	62	8	6	D
40	14	10	62	8	6	D

Dataset yang telah dibuat tersebut selanjutnya di uji menggunakan metode *10-cross validation* dimana pengujiannya dilakukan sebanyak dua kali yaitu pertama menggunakan metode SVM dan yang kedua menggunakan metode *decision tree*. Untuk metode SVM menggunakan *10-cross validation* dan fungsi kernel *polynomial* menghasilkan akurasi sebesar 60.2564%. Selanjutnya untuk metode *decision tree* menggunakan *10-cross validation* menghasilkan akurasi sebesar 71.1538 %.

4.2.2 Hasil Skenario Pengujian

Pada penelitian ini pengujian dilakukan untuk mengetahui perbedaan tingkat kecerdasan buatan pada game battle RPG antara metode SVM, *decision tree*, dan *rulebase*. Ada enam skenario percobaan yang akan dilakukan dalam pengujian ini. Setiap skenario di ujikan sebanyak 100 kali pertandingan.

Pengujian pertama yaitu metode SVM melawan metode *decision tree*. Dalam 100 kali pertandingan, metode SVM mendapatkan jumlah kemenangan sebanyak 58 kali. Sedangkan untuk metode *decision tree* mendapatkan 42 kali kemenangan.

Selanjutnya pengujian kedua yaitu metode SVM melawan metode *rulebase*. Dalam 100 kali pertandingan, metode SVM mendapatkan jumlah kemenangan sebanyak 87. Sedangkan untuk metode *rulebase* mendapatkan 13 kali kemenangan.

Selanjutnya pengujian ketiga yaitu metode *decision tree* melawan metode *rulebase*. Dalam 100 kali pertandingan, metode *decision tree* mendapatkan jumlah kemenangan sebanyak 58 kali. Sedangkan untuk metode *rulebase* mendapatkan 42 kali kemenangan. Hasil pengujian skenario 1 hingga 3 dapat dilihat pada Tabel 4.3.

Tabel 4. 2 Hasil pengujian skenario 1 hingga 3

Metode	Jumlah Kemenangan		
	SVM	<i>Decision tree</i>	Rulebase
SVM	-	42	13
<i>Decision tree</i>	58	-	42
Rulebase	87	58	-

Pengujian keempat yaitu metode SVM melawan metode SVM. Dalam 100 kali pertandingan, metode SVM *computer player 1* mendapatkan jumlah kemenangan sebanyak 48 kali. Sedangkan untuk SVM *computer player 2* mendapatkan 52 kali kemenangan. Pengujian kelima yaitu metode *decision tree* melawan metode *decision tree*. Dalam 100 kali pertandingan, metode *decision tree computer player 1* mendapatkan jumlah kemenangan sebanyak 50 kali, sedangkan untuk *decision tree computer player 2* mendapatkan 50 kali kemenangan. Pengujian keenam yaitu metode *rulebase* melawan metode *rulebase*. Dalam 100 kali pertandingan, metode *rulebase computer player 1* mendapatkan jumlah kemenangan sebanyak 49 kali, sedangkan untuk *rulebase computer player 2* mendapatkan 51 kali kemenangan. Hasil pengujian skenario 4 hingga 6 dapat dilihat pada Tabel 4.3.

Tabel 4. 3 Hasil pengujian skenario 4 hingga 6

Metode	Kemenangan	
	<i>Computer player 1</i>	<i>Computer player 2</i>
SVM	48	52
<i>Decision tree</i>	50	50
<i>Rulebase</i>	49	51

Selanjutnya, data yang telah didapatkan akan diproses ke tahap analisis. Sehingga nantinya akan didapatkan perbedaan kecerdasan pada setiap metode.

4.2.3 Analisis Hasil Pengujian

Hasil yang telah didapatkan dari proses uji coba selanjutnya akan dianalisis.

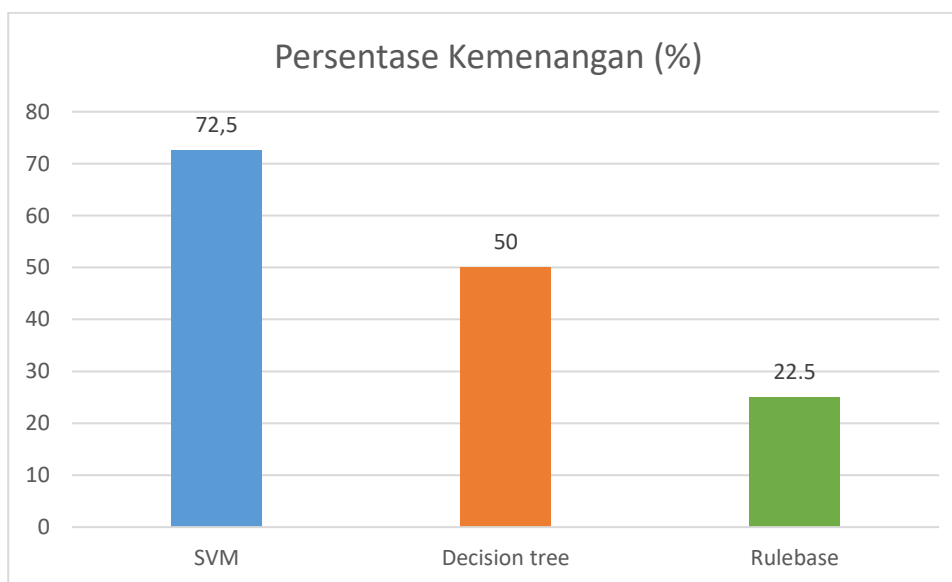
4.2.3.1 Analisis Perbandingan Antar Metode

Proses analisis dari hasil pengujian yaitu dengan mencari persentase rata-rata kemenangan setiap metode kecerdasan buatan. Berdasarkan hasil pengujian yang telah dilakukan, dapat diketahui bahwa persentase kemenangan kecerdasan buatan dengan menggunakan metode SVM memiliki persentase kemenangan paling tinggi yaitu sebesar 72.5% dibandingkan dengan metode *decision tree* yaitu sebesar 50% dan metode *rulebase* yaitu sebesar 22.5%.

Berdasarkan faktor tingkat kemenangan, dapat disimpulkan bahwa kecerdasan buatan dengan menggunakan metode SVM memiliki keunggulan dalam faktor jumlah kemenangan. Sehingga dapat disimpulkan bahwa metode SVM adalah metode pengambilan keputusan yang paling baik dari pada metode *decision tree* dan *rulebase* pada *game* berjenis *battle* RPG. Hal ini akan menjawab hipotesis penelitian yaitu; ada perbedaan antara penggunaan satu metode kecerdasan buatan untuk *computer player* dengan metode kecerdasan buatan lain pada *game battle* RPG. Hasil analisis perbandingan antar metode selengkapnya dapat dilihat pada Tabel 4.4 dan Gambar 4.11.

Tabel 4. 4 Hasil analisis perbandingan antar metode

Metode	SVM	<i>Decision tree</i>	<i>Rulebase</i>
SVM	-	42	13
<i>Decision tree</i>	58	-	42
<i>Rulebase</i>	87	58	-
Total	145	100	55
Rata-rata	72.5	50	22.5



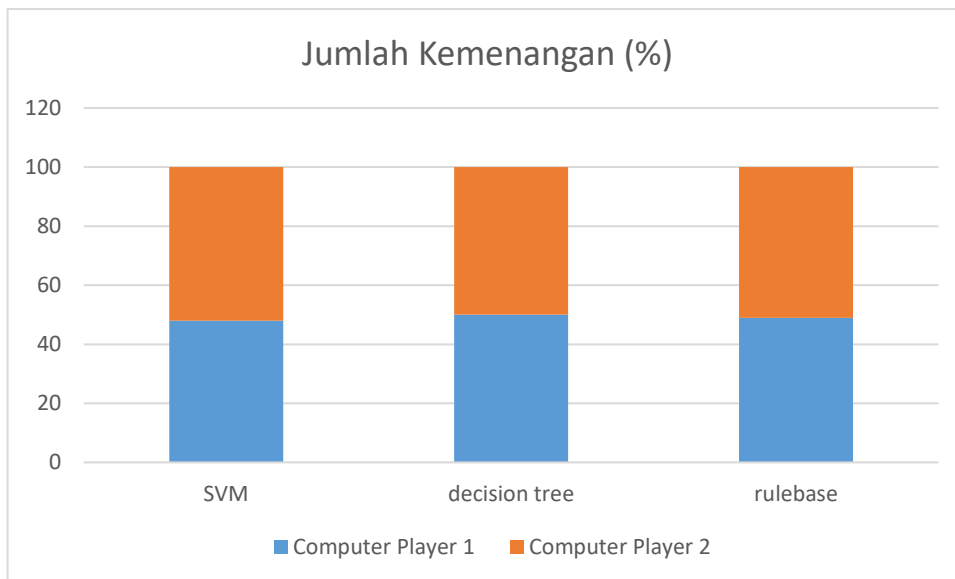
Gambar 4. 11 Perbandingan hasil metode yang berbeda.

4.2.3.2 Analisis Perbandingan Metode yang Sama

Langkah yang dilakukan dalam melakukan analisis hasil pengujian dengan metode kecerdasan yang sama, akan di lihat pada ketiga faktor yaitu selisih persentase kemenangan, selisih persentase sisa health dan selisih rata-rata waktu pada *computer player 1* dan *computer player 2*. Diasumsikan bahwa semakin kecil selisih persentase faktor-faktor antara kedua computer player tersebut maka semakin seimbang kedua metode kecerdasan buatan tersebut.

Tabel 4. 5 Hasil analisis perbandingan metode yang sama

Metode	Kemenangan (%)		Selisih
	<i>Computer player 1</i>	<i>Computer player 2</i>	
SVM	48	52	4
<i>Decision tree</i>	50	50	0
<i>Rulebase</i>	49	51	2



Gambar 4. 12 Perbandingan SVM dan SVM.

Berdasarkan hasil pengujian yang telah dilakukan, dapat diketahui bahwa selisih persentase kemenangan kecerdasan buatan dengan menggunakan metode SVM dan SVM hanya sebesar 4%, metode decision tree dan decision tree sebesar 0%, metode rulebase dan rulebase hanya sebesar 2%. Dapat disimpulkan bahwa untuk kedua computer player yang memiliki metode kecerdasan yang sama, bila dipertandingkan maka akan mendapatkan kemiripan hasil jumlah kemenangan. Hal

tersebut dikarenakan didalam setiap kondisi pada saat pertandingan akan memiliki kecenderungan kesamaan keputusan yang dihasilkan. Hasil analisis perbandingan metode yang sama selengkapnya dapat dilihat pada Tabel 4.3 dan Gambar 10.

[Halaman ini sengaja dikosongkan]

BAB 5

KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan yang dapat diambil berdasarkan pada penelitian yang telah dilakukan. Dalam Bab 5 ini diuraikan juga tentang hal-hal yang perlu dipertimbangkan untuk pengembangan penelitian lebih lanjut. Penjelasan yang lebih terperinci tentang hal-hal tersebut diuraikan pada sub-bab berikut.

5.1 Kesimpulan

Dari hasil penelitian dan analisis yang dilakukan, dapat diambil beberapa kesimpulan diantaranya adalah sebagai berikut:

1. Perancangan dan pembuatan aplikasi *game battle* RPG sebagai media perbandingan kecerdasan buatan dibuat dalam beberapa tahapan yaitu analisis kebutuhan, desain dan implementasi, dan pengujian.
2. Pengujian metode kecerdasan buatan pada *computer player* dilakukan dengan cara mempertandingan sebanyak 100 kali terhadap dua buah metode kecerdasan buatan. Berdasarkan data hasil pengujian yang telah dilakukan, didapatkan hasil dari analisis bahwa kecerdasan buatan dengan menggunakan metode SVM memiliki keunggulan dalam faktor jumlah kemenangan. Sehingga dapat disimpulkan bahwa metode SVM adalah metode pengambilan keputusan yang paling baik dari pada metode *decision tree* dan *rulebase* pada *game* berjenis *battle* RPG. Hal ini akan menjawab hipotesis penelitian yaitu; ada perbedaan antara penggunaan satu metode kecerdasan buatan untuk *computer player* dengan metode kecerdasan buatan lain pada *game battle* RPG.

5.2 Saran

Berikut merupakan beberapa saran yang dapat digunakan sebagai acuan untuk pengembangan penelitian lebih lanjut di masa yang akan datang.

1. Variasi *game* yang diteliti, baik yang berkaitan dengan teknologi yang digunakan maupun jenis *game* dapat ditingkatkan seiring dengan perkembangan jaman dan teknologi.
2. Metode kecerdasan buatan yang diteliti dapat menggunakan metode klasifikasi yang lain. Sehingga harapannya dapat meningkatkan rasio kemenangan untuk *computer player* pada *game battle RPG*.

DAFTAR PUSTAKA

- Aghlara, L. & Hadidi, N., 2011. The effect of digital games on Iranian children ' s vocabulary retention in foreign language acquisition. *International Conference on Education and Educational Psychology (ICEEPSY 2011)*, 29, pp.552–560.
- Bedoya-rodriguez, S. et al., 2014. Augmented Reality RPG Card-based Game. *Games Media Entertainment (GEM), 2014 IEEE*, pp.3–6.
- Bimantoro, T. & Haryanto, H., 2016. Pemodelan Perilaku Musuh Menggunakan Finite State Machine (FSM) Pada Game Pengenalan Unsur Kimia. *Journal of Applied Intelligent Sistem*, 1(3), pp.210–219.
- Boser, B.E., Guyon, I.M. & Vapnik, V.N., 1992. A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, pp.144–152.
- Brathwaite, B. & Schreiber, I., 2009. *Challenges for game designers*,
- Cass, S., 2002. Mind Games. *IEEE Spectrum*, 39(12), pp.40–45.
- Chen, P. & Kim, T., 2012. *Serious Games : The Challenge*,
- Childress, M. & Braswell, R., 2006. Using Massively Multiplayer Online Role-Playing Games for Online Learning. *Distance Education*, 27(2), pp.187–196.
- Childress, M.D. & Braswell, R., 2006. Using Massively Multiplayer Online Role-Playing Games for Online Learning. , 27(2), pp.187–196.
- Consalvo, M., 2009. There is No Magic Circle. *Games and Culture 2009*, 4(4), pp.408–417.
- Cristianini, N. & Shawe-Taylor, J., 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press New York, NY, USA ©2000.
- E.M. Avedon, B.S.-S., 1981. *The Study of Games*, New York: John Wiley & Sons, Inc.
- Fei, B. & Liu, J., 2006. Binary Tree of SVM : A New Fast Multiclass Training and Classification Algorithm. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 17(3), pp.696–704.

- Hamari, J. & Koivisto, J., 2013. Social motivations to use gamification: an empirical study of gamifying exercise. *Proceedings of the 21st European Conference on Information Systems SOCIAL*, (JUNE), pp.1–12.
- J. Patrick Williams, Sean Q. Hendricks, W.K.W., 2006. *Gaming As Culture: Essays on Reality, Identity And Experience in Fantasy Games*, McFarland & Company, Inc. Publishers.
- Maroney, K., 2001. My Entire Walking Life. *The Games Journal | A Magazine About Boardgames*.
- Mcgee, K. & Abraham, A.T., 2010. Real-time team-mate AI in Games Categories and Subject Descriptors. *ACM*, pp.124–131.
- Melendez, P., 2009. Controlling non-player characters using Support Vector Machines. *Proceeding Future Play '09 Proceedings of the 2009 Conference on Future Play on @ GDC Canada*, pp.33–34.
- Melendez, P., 2010. *Decision-Making in game characters using Support Vector Machines*, University of Abertay Dundee.
- Millington, I. & Funge, J., 2009. *Artificial Intelligence for Games*, Elsevier.
- Narayek, A., 2004. AI in Computer Games. *Queue - Game Development*, 1(10), p.58.
- Quinlan, J.R., 1990. *Decision Trees and Decisionmaking*. *IEEE Transaction on Sistem Man and Cybernetics*, 20, pp.339–346.
- Purkiss, B., 2015. A Study of Interaction in Idle Games & Perceptions on the Definition of a Game. *Games Entertainment Media Conference (GEM), 2015 IEEE*.
- Purkiss, B. & Khaliq, I., 2016. A study of interaction in idle games & perceptions on the definition of a game. *2015 IEEE Games Entertainment Media Conference, GEM 2015*.
- Ralaivola, L., Wu, L. & Baldi, P., 2005. SVM and Pattern-Enriched Common Fate Graphs for the Game of Go. *European Symposium on Artificial Neural Networks Bruges (Belgium)*, 19, pp.27–29.
- Rostianingsih, S., Budhi, G.S. & Wijaya, H.K., 2013. Game Simulasi Finite State Machine untuk Pertanian dan Peternakan. *Konferensi Nasional Sistem Informasi, Universitas Kristen Petra*, pp.2–7.

- Salen, K. & Zimmerman, E., 2004. *Rules of Play: Game Design Fundamentals*,
- Spronck, P. et al., 2003. Online Adaptation of Game Opponent AI in Simulation and in Practice. *Proceedings of the 4th International Conference on Intelligent Games and Simulation*, pp.93–100.
- Spronck, P., Sprinkhuizen-kuyper, I. & Postma, E., 2004. Online Adaptation of Game Opponent AI with Dynamic Scripting. *International Journal of Intelligent Games and Simulation*, 3(1), pp.45–53.
- Sriram, S. et al., 2009. Implementing a No-Loss State in the Game of Tic-Tac-Toe using a Customized *Decision Tree* Algorithm. *IEEE International Conference on Information and Automation*, pp.1211–1216.
- Stone, P. & Veloso, M., 1997. Using *Decision Tree* Confidence Factors for Multiagent Control. *Proceedings of the second international conference on Autonomous agents*, pp.86–91.
- Suits, B., 2005. *The Grasshopper: Games, Life and Utopia*,
- Urh, M. et al., 2015. The Model for Introduction of Gamification into E-learning in Higher Education. *Procedia - Social and Behavioral Sciences*, 197(February), pp.388–397. Available at: <http://www.sciencedirect.com/science/article/pii/S1877042815041555>.
- Yannakakis, G.N. & Togelius, J., 2014. A Panorama of Artificial and Computational Intelligence in Games. , (c).
- Zhang, T., 2001. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. , 22(2), pp.103–104.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

No.	Health	Energy	Enemy_Health	Damage	Armor	Decision
1	300	100	300	0	0	U
2	260	60	260	0	2	U
3	220	20	220	0	4	H
4	200	40	220	0	4	U
5	200	0	180	0	6	H
6	200	20	180	0	6	H
7	180	40	180	0	6	A
8	160	20	160	2	6	A
9	160	0	138	2	8	H
10	140	20	138	2	8	A
11	140	0	116	2	10	H
12	118	20	116	2	10	H
13	118	40	116	2	10	U
14	118	0	74	4	10	H
15	118	20	74	4	10	H
16	96	40	74	4	10	U
17	96	0	30	6	10	H
18	74	20	30	6	10	A
19	74	20	30	6	10	A
20	50	0	4	8	10	H
21	50	20	4	8	10	A
22	300	100	300	0	0	U
23	260	60	260	0	2	U
24	218	20	220	2	2	A
25	194	0	198	2	4	H
26	194	20	198	2	4	H
27	168	40	198	2	4	U

28	168	0	156	4	4	H
29	142	20	156	4	4	D
30	142	10	156	4	4	H
31	120	30	156	4	4	D
32	120	20	156	4	4	H
33	98	40	156	4	4	U
34	98	0	112	4	6	H
35	72	20	112	4	6	A
36	72	0	88	6	6	H
37	44	20	88	6	6	A
38	44	0	62	8	6	H
39	14	20	62	8	6	D
40	14	10	62	8	6	D
41	300	100	300	0	0	A
42	260	80	280	2	0	D
43	220	70	280	2	0	U
44	198	30	238	2	2	H
45	198	50	238	2	2	U
46	174	10	196	4	2	H
47	174	30	196	4	2	H
48	148	50	196	4	2	U
49	148	10	152	6	2	H
50	122	30	152	6	2	A
51	122	10	126	8	2	H
52	96	30	126	8	2	A
53	96	10	98	8	4	H
54	68	30	98	8	4	A
55	68	10	70	8	6	H
56	38	30	70	8	6	A
57	38	10	42	10	6	H

58	8	30	42	10	6	H
59	8	50	42	10	6	U
60	300	100	300	0	0	U
61	280	60	260	2	0	U
62	240	20	218	2	2	A
63	198	0	196	2	4	H
64	198	20	196	2	4	A
65	198	0	174	4	4	H
66	174	20	174	4	4	A
67	148	0	150	6	4	H
68	148	20	150	6	4	A
69	148	0	124	6	6	H
70	122	20	124	6	6	A
71	96	0	98	6	8	H
72	96	20	98	6	8	A
73	96	0	72	8	8	H
74	70	20	72	8	8	H
75	42	40	72	8	8	U
76	42	0	24	10	8	H
77	42	20	24	10	8	H
78	14	40	24	10	8	U
79	300	100	300	0	0	D
80	280	90	300	0	0	U
81	258	50	260	0	2	U
82	214	10	220	0	4	H
83	214	30	220	0	4	D
84	172	20	220	0	4	H
85	172	40	220	0	4	U
86	144	0	180	0	6	H
87	144	20	180	0	6	H

88	116	40	180	0	6	U
89	116	0	140	2	6	H
90	88	20	140	2	6	A
91	88	0	118	2	8	H
92	58	20	118	2	8	H
93	58	40	118	2	8	U
94	28	0	76	2	10	H
95	28	20	76	2	10	D
96	6	10	76	2	10	H
97	6	30	76	2	10	A
98	300	100	300	0	5	U
99	260	60	260	0	9	A
100	218	40	240	0	11	U
101	196	0	200	2	11	H
102	196	20	200	2	11	H
103	196	40	200	2	11	D
104	184	30	200	2	11	A
105	160	10	178	3	11	H
106	160	30	178	3	11	A
107	135	10	155	3	13	H
108	135	30	155	3	13	D
109	123	20	155	3	13	A
110	123	0	132	4	13	H
111	97	20	132	4	13	A
112	97	0	108	5	13	H
113	70	20	108	5	13	A
114	70	0	83	6	13	H
115	42	20	83	6	13	H
116	42	40	83	6	13	U
117	13	0	37	6	17	H

118	13	20	37	6	17	H
119	300	100	300	0	5	U
120	260	60	260	2	5	H
121	220	80	260	2	5	U
122	198	40	218	4	5	U
123	198	0	174	4	9	H
124	175	20	174	4	9	H
125	175	40	174	4	9	U
126	152	0	130	6	9	H
127	152	20	130	6	9	A
128	129	0	104	7	9	H
129	129	20	104	7	9	A
130	105	0	77	8	9	H
131	105	20	77	8	9	A
132	80	0	49	9	9	H
133	80	20	49	9	9	A
134	54	0	20	9	11	H
135	54	20	20	9	11	H
136	28	40	20	9	11	U
137	300	100	300	0	5	U
138	260	60	260	2	5	D
139	223	50	260	2	5	U
140	199	10	218	4	5	H
141	199	30	218	4	5	A
142	174	10	194	5	5	H
143	174	30	194	5	5	H
144	149	50	194	5	5	A
145	149	30	169	5	7	H
146	123	50	169	5	7	H
147	123	70	169	5	7	U

148	96	30	124	7	7	A
149	96	10	97	8	7	D
150	76	0	97	8	7	H
151	76	20	97	8	7	H
152	48	40	97	8	7	A
153	48	20	69	8	9	D
154	28	10	69	8	9	H
155	28	30	69	8	9	D
156	8	20	69	8	9	A

BIOGRAFI PENULIS



Musta'inul Abdi dilahirkan di Pematang Tebih, Riau pada tanggal 30 Oktober 1991 dan merupakan anak pertama dari tiga bersaudara dari pasangan Suprianto dan Imtihanah. Penulis menempuh pendidikan dasar di SD Negeri 005 Tandun pada tahun 1997 hingga tahun 2003, lalu melanjutkan pendidikan menengah pertama di MTs Negeri Tandun pada tahun 2003 hingga tahun 2006, dan pendidikan menengah atas di SMA Negeri 1 Ujung Batu pada tahun 2006 hingga tahun 2009. Penulis melanjutkan jenjang perguruan tinggi pada Jurusan Teknik Informatika di Politeknik Negeri Lhokseumawe pada tahun 2010 hingga tahun 2014. Setelah itu penulis melanjutkan jenjang magister Teknik Informatika di Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2015 hingga tahun 2017.

Selain Pendidikan, penulis juga meluangkan waktu sebagai *software developer* dan *machine engineer* dalam skala kecil. Penulis mempunyai ketertarikan dalam bidang otomotif. Penulis juga mempunyai hobi lain yaitu bermain *game* (desktop dan mobile). Penulis dapat dihubungi melalui email yaitu: m.abdi4444@gmail.com.

[Halaman ini sengaja dikosongkan]