IMPLEMENTASI *FIGHTING SYSTEM* MENGGUNAKANAN *DYNAMIC SCRIPTING* DAN *MACRO ACTION* DALAM *COMPUTER ROLE-PLAYING GAME*

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun Oleh :

FEBRI ABDULLAH

NIM. 0810680036

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

LEMBAR PERSETUJUAN

IMPLEMENTASI *FIGHTING SYSTEM* MENGGUNAKAN *DYNAMIC SCRIPTING* DAN *MACRO ACTION* DALAM *COMPUTER ROLE-PLAYING GAME*

Disusun Oleh :

FEBRI ABDULLAH

NIM. 0810680036

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 17 Juni 2013

Dosen Pembimbing I	**Dosen Pembimbing II**
Himawat Aryadita, S.T, M.Sc	**Eriq M. Adams J., S.T., M.Kom**
NIP. 19801018 200801 1 003	**NIK. 850410 06 1 1 0027**

LEMBAR PENGESAHAN

IMPLEMENTASI *FIGHTING SYSTEM* MENGGUNAKAN *DYNAMIC SCRIPTING* DAN *MACRO ACTION* DALAM *COMPUTER ROLE-PLAYING GAME*

Disusun Oleh :

FEBRI ABDULLAH

NIM. 0810680036

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 10 Juli 2013

Penguji I	Penguji II
Denny Sagita R, S.Kom., M.Kom	**Aryo Pinandito, S.T., M.MT**
NIP. 851124 06 1 1 0250	**NIK. 870724 06 1 1 0374**

Penguji III

Fajar Pradana, S.ST., M.Eng

-

Mengetahui

Ketua Program Studi Teknik Informatika

Drs. Marji, MT.

NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 Juli 2013
Mahasiswa

Febri Abdullah
0810680036

KATA PENGANTAR

Dengan nama Allah Yang Maha Pengasih dan Penyayang. Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Implementasi *Fighting System* Menggunakan *Dynamic Scripting* dan *Macro Action* dalam *Computer Role-Playing Game*". Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Penyusunan Tugas Akhir ini dapat terlaksana karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu. Oleh karena itu, melalui kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Orang Tua penulis, Zaenal Arifin dan Nur Faizah, dan seluruh keluarga yang senantiasa tiada henti hentinya memberikan doa demi terselesainya tugas akhir ini.
2. Almarhum Ayahanda Suwito.
3. Bapak Himawat Aryadita, S.T, M.Sc selaku dosen penasehat akademik dan dosen pembimbing I yang telah memberikan ilmu dan saran untuk proposal skripsi ini.
4. Bapak Eriq M. Adams J., ST., Mkom selaku dosen pembimbing II yang juga memberikan ilmu dan saran untuk proposal skripsi ini.
5. Seluruh dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
6. Teman-teman yang telah membantu memberi kritik dan saran atas proposal ini.

Penulis menyadari bahwa Tugas Akhir ini masih terdapat banyak kekurangan, oleh karena itu kritik dan saran yang bersifat membangun sangat

diharapkan untuk menyempurnakan Tugas Akhir ini. Penulis berharap Tugas Akhir ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, 27 Mei 2013

Penulis

ABSTRAK

Febri Abdullah. 2013: Implementasi *Fighting System* Menggunakan *Dynamic Scripting* dan *Macro Action* dalam *Computer Role-Playing Game*. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Himawat Aryadita, S.T, M.Sc dan Eriq M Adams J, ST., M.Kom.

Secara umum, game AI adaptif mengacu ke AI berbasis *agent* yang memiliki kemampuan untuk melakukan *learning* dari tiap kesalahannya dan kemampuan untuk meng*generate* taktik baru yang lebih baik di setiap pertempurannya. *Dynamic scripting* adalah sebuah metode untuk melakukan hal tersebut. Metode ini akan melakukan proses *learning* dalam mengatur sebuah *agent* dalam melakukan proses *generate script* dan melakukan pemberian *reward* atau *penalty* pada tiap *rule* yang telah digunakan di dalam pertempuran sebelumnya. *Dynamic scripting* mempunyai kecenderungan dalam meng*generate* taktik yang terlampau kuat dengan tingkat keberagaman yang rendah dan mengarah ke taktik yang sama. Hal tersebut bisa menjadi poin yang kurang menarik bagi pemain game. Dengan menggabungkan *Dynamic Scripting* dan *macro action* yang mempunyai *building block* yang lebih besar, seharusnya membuat metode ini mampu memperluas keberagaman taktik yang disajikan. Penelitian ini bertujuan untuk menunjukkan bagaimana metode *dynamic scripting* dan *macro action* diimplementasikan ke dalam sebuah *fighting system*. Penulis menyimpulkan bahwa penggunaan *macro action* mampu meningkatkan kecepatan adaptasi dan keberagaman dari *script* yang di*generate* oleh metode *dynamic scripting* tanpa mengesampingkan tingkat efektifitas taktik tersebut.

Kata Kunci: *Game, Artificial Intelligence, reinforcement learning, dynamic scripting, macro actions.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang Permasalahan

Kualitas dari sebuah game sangat ditentukan oleh tingkat kepuasan pemain dalam memainkannya. Ketidakpuasan dari seorang pemain pada umumnya disebabkan karena game tersebut terlalu mudah untuk ditaklukkan. Suatu game mempunyai tujuan utama untuk menghibur pemainnya. Hal tersebut bisa dilakukan dengan membuat game tersebut lebih menantang untuk ditaklukkan oleh seorang pemain. Tingkat kesulitan dari taktik yang digunakan komputer di dalam suatu game sangat ditentukan oleh kecerdasan buatan, atau yang lebih sering disebut *Artificial Intelegent* (AI), yang dimiliki game tersebut. AI dari suatu game bisa dikatakan berhasil jika bisa membuat pemain menjadi tertantang untuk terus memainkan game tersebut [SZI-09:1].

AI dalam game bisa dikatakan menarik jika mempunyai dua unsur yaitu efektifitas dan keberagaman. Semakin efektif suatu AI akan menjadikan game tersebut semakin menantang bagi pemain. Di samping itu, keberagaman dalam AI juga diperlukan karena akan menyuguhkan berbagai macam taktik yang menjadikan game tersebut tidak membosankan bagi pemain. Suatu AI yang adaptif seharusnya mampu meningkatkan efektifitasnya secara otomatis sekaligus mempertahankan keberagamannya dengan tetap menyuguhkan taktik yang berbeda kepada pemain [SZI-09:1].

Untuk game yang kompleks, seperti *Computer Role-Playing Game* (*CRPG*) dan game strategi, di mana terdapat ratusan, bahkan ribuan pilihan *action* yang bisa di ambil di setiap turn-nya, penggabungan AI dalam game tersebut sulit dilakukan. Dalam implementasi AI untuk game yang lebih kompleks, kebanyakan pengembang game menggunakan *script*, yaitu kumpulan dari beberapa *rule* yang dieksekusi secara berurutan. *Script* adalah sebuah teknik pilihan di sebuah industri game untuk mengimplementasikan AI game karena *script* bisa dimengerti, bisa diprediksikan, mampu beradaptasi sesuai dengan keadaan tertentu, mudah

diimplementasikan, bisa dijabarkan dengan mudah, dan bisa digunakan oleh orang-orang yang notabene bukan seorang programmer.

Script pada umumnya bersifat statik dan cenderung panjang dan kompleks, yang mana menimbulkan 2 permasalahan. Yang pertama, karena tingkat kompleksitas yang dimiliki, *script* memiliki kemungkinan besar mengandung beberapa kelemahan yang bisa digunakan oleh seorang pemain untuk mengalahkan musuh yang kuat dengan mudah. Permasalahan yang ke dua adalah, karena sifatnya yang statis, *script* tidak bisa menangani taktik pemain yang tidak terduga dan tidak mampu menimbang tingkat kesulitan untuk bisa menyesuaikan dengan pemain pemula maupun yang sudah ahli [SPR-05:2]. Dengan kata lain, *script* bisa dikatakan kurang efektif dan tidak memiliki keberagaman karena sifat yang dimilikinya adalah statis.

Dynamic scripting adalah suatu teknik *reinforcement learning* yang mampu mempelajari *script-script* dalam game AI secara otomatis. Pada dasarnya, teknik *dynamic scripting* bisa digunakan untuk mencapai efektifitas dan keberagaman AI yang lebih tinggi dibandingkan AI game yang bersifat statis. Akan tetapi, *dynamic scripting* sendiri mempunyai beberapa kelemahan yaitu *script* yang dihasilkan cenderung mengarah ke taktik yang efektif dengan tingkat kemiripan antar *script* yang cukup tinggi, sehingga bisa dikatakan kurang beragam. Hal tersebut sangat memungkinkan bagi pemain untuk merasa jenuh dalam memainkan game tersebut [SZI-09-1].

Secara sederhana, *macro action* adalah urutan-urutan dari beberapa *action* yang ditangani sebagai unit tunggal. Dengan mengelompokkan beberapa *action* dasar ke dalam sebuah *macro action* akan diperoleh keberagaman cara untuk mencapai suatu tujuan yang sama. Dengan *building block* yang lebih besar yang dimiliki *macro action*, seharusnya *dynamic scripting* bisa meningkatkan keberagaman taktik yang disajikan tanpa mengesampingkan efektifitas dari taktik tersebut [SZI-09:3].

Atas dasar tersebut, penulis ingin mengimplementasikan *dynamic scripting* menggunakan *macro action* ke dalam *agent* AI dinamis di dalam sebuah *fighting system* berjenis *turn-based strategy* dengan harapan bisa meningkatkan

keberagaman taktik tanpa mengesampingkan keefektifan dari taktik yang disajikan oleh *agent* AI tersebut.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang diangkat pada bagian latar belakang, maka rumusan masalah dikhususkan pada :

a. Bagaimana implementasi metode *dynamic scripting* menggunakan *macro action* dalam *agent* AI untuk memperluas keberagaman dan mempermudah pergantian taktik tanpa mengesampingkan keefektifan dari taktik yang disajikan oleh *agent* AI tersebut ?
b. Bagaimana pengujian metode *dynamic scripting* menggunakan *macro action* dalam game AI ?

1.3 Tujuan

Tujuan dari penulisan skripsi ini adalah membangun *agent* AI yang adaptif dan mampu menyajikan taktik yang beragam tanpa mengesampingkan keefektifannya dari taktik tersebut.

1.4 Batasan Masalah

Lingkup aspek kajian dalam penelitian ini sebagai berikut :

a. Metode yang digunakan dalam penelitian ini adalah *dynamic scripting* menggunakan *macro action*
b. Metode *learning macro action* yang digunakan adalah *cross entrophy learning*
c. Microsoft Visual C# 2010 Express dan Unity MonoDevelop sebagai lingkungan pengembangan
d. Unity 3D 3.4.0 sebagai game engine
e. Spesifikasi komputer yang digunakan dalam penelitian ini adalah:
   - Prosesor : Intel® Core(TM) i3, @ 2.13GHz
   - Memory : 4096MB RAM
   - VGA : ATI Mobility Radeon HD 5470

- Sistem Operasi : Windows 7 Home Basic 64-bit

1.5 Manfaat

Manfaat yang bisa didapatkan dari penelitian ini adalah:

a. Bagi penulis
   1. Menerapkan ilmu yang telah diperoleh dari Teknik Informatika Universitas Brawijaya.
   2. Mendapatkan pemahaman tentang bagaimana membangun sebuah *agent* AI untuk game dengan metode *dynamic scripting* menggunakan *macro action*.

b. Bagi pengembang
   1. Bisa digunakan sebagai *agent* AI untuk mengembangkan game sejenis.
   2. Bisa dijadikan sebagai referensi dalam mengembangkan *agent* AI.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam skripsi ini adalah:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, dan sistematika penulisan.

BAB II Dasar Teori

Menguraikan teori dasar dan teori penunjang yang berkaitan dengan *dynamic scripting* dan *macro action*.

BAB III Metode Penelitian

Membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, pengambilan data, perancangan perangkat lunak, implementasi algoritma, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan

Membahas analisis algoritma yang dibutuhkan dalam *dynamic scripting* menggunakan *macro action*.

BAB V Implementasi

Membahas tentang implementasi algoritma ke dalam suatu game.

BAB VI Pengujian

Memuat proses dan hasil pengujian terhadap metode yang telah direalisasikan.

BAB VII Penutup

Memuat kesimpulan serta saran yang diperoleh dari implementasi dan pengujian metode yang sudah diterapkan untuk pengembangan lebih lanjut.

BAB II
DASAR TEORI

Pada bab ini berisi pembahasan tentang teori dasar yang berhubungan dengan pengembangan *agent* AI yang dilakukan. Teori dasar yang akan dibahas pada bab ini yaitu penjelasan tentang game AI, *Computer Role-Playing Game* (*CRPG*), *dynamic scripting*, dan *macro action*.

2.1 Game AI

Kecerdasan buatan atau yang biasa disebut *Artificial Intelligence* (AI) memiliki tujuan untuk membuat komputer mampu berpikir untuk melakukan sesuatu layaknya manusia atau hewan [MIL-06:4].

Menggunakan AI game yang lebih kompleks bukan berarti bahwa tingkah laku dari karakter game tersebut akan terlihat lebih bagus dibanding dengan menggunakan AI yang sederhana. AI yang baik adalah AI yang mampu mecocokkan tingkah laku yang sesuai dengan algoritma yang sudah dirancang [MIL-06:21].

Kualitas dari sebuah game pada umumnya sangat penting bagi kesuksesan penjualannya. Kualitas dari sebuah game bisa ditinjau dari kualitas grafis dan kualitas AI yang dimiliki game tersebut. Pada saat grafis terus mengalami peningkatan setiap tahunnya, AI masih berada di level kualitas yang sama seperti 10 tahun yang lalu.

AI dari game komputer yang ada di pasaran pada umumnya berdasarkan pada *script*. Terdapat sejumlah *rule* yang dieksekusi dengan urutan yang sudah ditentukan. *Script* yang paling kompleks bisa ditemukan di dalam sebuah *Computer Role-Playing Game* (*CRPG*). AI yang berdasar pada *script* mempunyai beberapa kelemahan. Kelemahan yang paling menonjol adalah *script* tidak mampu beradaptasi sesuai dengan strategi yang digunakan pemain. Hal ini memungkinkan pemain menggunakan beberapa trik dan strategi sederhana untuk menyerang titik lemah dari AI tersebut untuk mengalahkan musuh yang paling sulit sekalipun dengan mudah [TIM-06:9].

2.2 *Computer Role-Playing Games* (*CRPG*)

Computer Role-Playing Game (*CRPG*) terkenal dengan struktur ceritanya yang sangat kuat dan seringkali melibatkan pengalaman bermain pemainnya dengan cerita yang sudah dibuat dengan sangat hati-hati. Game ini juga seringkali mempunyai sistem pembuatan karakter yang kompleks yang menyuguhkan ratusan pilihan kepada pemain untuk membuat tipe dan penampilan dari karakter yang ingin dimainkan.

Personalisasi yang dimiliki *CRPG* mempunyai sedikit efek, atau bahkan tidak, terhadap cerita dan penempatan kata-kata dalam dialog yang disuguhkan. Cerita dari *CRPG* seringkali berdasar pada jalur yang sudah ditentukan; pemain bermain untuk mendapatkan experience, menyelesaikan suatu checkpoint untuk melanjutkan progres cerita. Pemain dimungkinkan untuk memiliki beberapa pilihan di sepanjang jalan selama berada di branch point yang sudah ditentukan di dalam cerita, meskipun pilihan-pilihan tersebut secara umum mempunyai dampak kecil kepada pemain dan jalan cerita secara keseluruhan tetap sama, atau paling tidak menuju ke ending yang berbeda yang sudah ditentukan sebelumnya.

Sistem pertempuran dari *CRPG* seringkali bersifat kompleks dan kuat yang memungkinkan bagi banyak pilihan yang penting dan penuh arti melalui strategi pemainnya bisa terjadi. Personalisasi dari karakter-karakternya mempunyai hubungan yang kuat di dalam *action* yang mungkin dilakukan karakter tersebut di dalam suatu pertempuran. Sebagai contoh, seorang *rogue* bisa melakukan *stealth* dan *backstab*, sedangkan *mage* bisa melepaskan mantra sihir [SUL-11:1].

2.3 *Dynamic Scripting*

2.3.1 Pengertian *Dynamic Scripting*

Dynamic scripting adalah suatu teknik reinforcement learning yang mampu mempelajari *script-script* dalam game AI secara otomatis. Secara umum, kinerja dari teknik ini terhitung cepat, efektif, tangguh, dan efisien. *Dynamic scripting* menangani beberapa *rulebase* untuk tiap tipe musuh dalam suatu game. *Rule* dari *rulebase* tersebut secara manual didesain menggunakan *domain-spesific*

knowledge. Ketika sebuah musuh baru muncul, *rule-rule* yang ada di dalam *script* yang digunakan untuk mengontrol musuh tersebut diekstrak dari *rulebase* sesuai dengan tipe musuh tersebut. *Rulebase* yang menjadi sumber untuk mengekstrak *script* beradapatasi dengan cara terus merubah bobot tiap *rule* yang bersangkutan untuk merepresentasikan kesuksesan atau kegagalan dari *rule-rule* tersebut di dalam *script* [SZI-09:3].

Game AI adaptif adalah game AI yang mampu berganti selama game dimainkan dengan tujuan beradaptasi seiring berubahnya keadaan di dalam game. *Dynamic scripting* adalah sebuah teknik terkini yang dikembangkan untuk mewujudkan game AI adaptif. Dengan *dynamic scripting* game AI bisa berkembang dengan sendirinya selama permainan berlangsung. Dengan kata lain, jika terdapat suatu kelemahan di dalam AI tersebut yang ditemukan oleh pemain, AI secara cepat dapat beradaptasi dengan sendirinya selama game berlangsung sehingga membuatnya semakin sulit, atau bahkan tidak mungkin bagi pemain untuk menyalahgunakan kelemahan AI tersebut. Suatu game AI yang menggunakan teknik *dynamic scripting* mempunyai kualitas untuk beradaptasi terhadap taktik dan strategi yang digunakan oleh pemain selama game dimainkan. Hal in bisa memaksa pemain untuk mengganti strateginya selama game berlangsung secara berulang-ulang sehingga membuat game lebih menantang dan memuaskan pemainnya. Sebuah penelitian tentang teknik *dynamic scripting* telah menunjukkan bahwa teknik tersebut berhasil menyuguhkan komputer dengan kemampuan yang adaptif [TIM-06:9].

Sebuah game AI yang adaptif harus memiliki empat persyaratan komputasional dan empat persyaratan fungsional.

Persyaratan komputasional adalah syarat yang wajib dimiliki suatu game AI yang adaptif. Jika game AI yang adaptif tidak memenuhi syarat komputasional menjadikan game AI tersebut tidak berguna dalam penerapannya. Persyaratan fungsional kurang begitu penting jika dibandingkan dengan persyaratan komputasional. Kegagalan suatu game AI yang adaptif dalam memenuhi persyaratan fungsionalnya berarti bahwa pengembang game yang memakainya kurang bersedia menyertakan teknik ini di dalam game yang sedang

dikembangkan walaupun teknik ini memberikan hasil yang baik dan memenuhi semua persyaratan komputasionalnya.

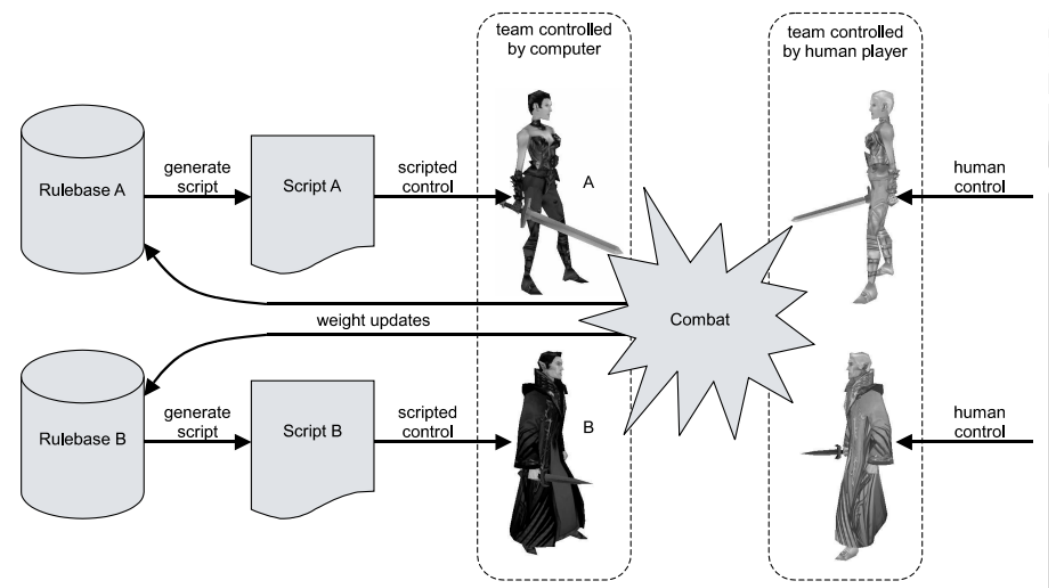
Keempat persyaratan komputasional tersebut antara lain:

- Kecepatan: game AI yang adaptif harus mampu cepat dalam hal komputasi karena learning akan dilakukan selama game dimainkan.
- Keefektifan: game AI yang adaptif harus efektif selama proses learningnya untuk menghindari hasil yang lebih buruk dibanding AI yang dirancang secara manual. Game AI yang adaptif dikatakan efektif jika mampu menghasilkan *action* atau tingkah laku yang baik dan masuk akal. Dengan tercapainya syarat ini maka ketakukan dari pengembang game yang selama ini terjadi, di mana game AI ditakutkan akan menghasilkan *action* atau tingkah laku yang tidak masuk akal, sudah terpecahkan.
- Ketangguhan: game AI yang adaptif harus tangguh sesuai dengan ketidakteraturan keadaan yang dimiliki kebanyakan game.
- Efisien: game AI yang adaptif harus efisien sesuai dengan banyaknya kemungkinan learning yang harus dilakukan, karena di dalam sebuah game, pengalaman seorang pemain hanya berkisar pada banyaknya pertempuran sejenis yang dijumpai.

Keempat persyaratan fungsional adalah:

- Kejelasan: game AI yang adaptif harus memberikan hasil yang mudah ditafsirkan karena kebanyakan pengembang game jarang menggunakan teknik learning yang memiliki hasil yang sulit dimengerti.
- Keanekaragaman: game AI yang adaptif harus menghasilkan beranekaragam tingkah laku karena *agent* yang memiliki perilaku yang mudah diprediksi bisa dikatakan kurang menghibur dibanding dengan *agent* yang memiliki perilaku yang sulit diprediksi.
- Konsistensi: rata-rata jumlah dari kemungkinan learning yang dibutuhkan sebuah game AI yang adaptif untuk memberikan hasil yang baik seharusnya memiliki tingkat konsistensi yang tinggi.
- Skalabilitas: tingkat kesulitan yang dihasilkan game AI yang adaptif harus mempu menyesuaikan dengan kemampuan pemain.

Dalam *dynamic scripting*, probabilitas dari sebuah *rule* untuk terpilih untuk sebuah *script* ditentukan oleh weight yang terikat pada tiap *rule*. Tujuan utama dari *dynamic scripting* adalah untuk mengadaptasikan weight di dalam *rulebase* sehingga fitness dari *action* yang sudah ditentukan di dalam *rulebase* diharapkan bisa berubah dengan cepat, walaupun terdapat perubahan *environment*.



Gambar 2.1. Gambaran Umum *Dynamic Scripting*
Sumber: [SPR-05:6]

Teknik *dynamic scripting* di dalam suatu konteks game komersial diilustrasikan di dalam gambar 2.1. Dari gambar 2.1, tim yang mengenakan kostum berwarna abu-abu dikontrol oleh pemain, sedang yang berwarna hitam dikontrol oleh komputer. *Rulebase* yang terkait dengan dengan tiap *agent* yang dikontrol komputer (“A” dan “B”) mengandung *rule-rule* yang didesain secara manual yang berasal dari sebuah domain-spesific knowledge. Sudah seharusnya bahwa sebagian besar dari *rule* di dalam *rulebase* ditetapkan sebagai *rule* yang efektif, atau setidaknya masuk akal, untuk dijadikan suatu *action* atau tingkah laku yang mewakili *agent*nya [SPR-05:3-6].

Urutan dari *rule* di dalam *script* ketika *dynamic scripting* menghasilkan *script* baru ditentukan dengan cara menentukan nilai prioritas yang dimiliki *script*

tersebut secara manual. *Rule* dengan nilai prioritas lebih besar berada di atas *rule* dengan nilai prioritas yang lebih kecil. Dalam kasus prioritas dengan nilai yang sama, *rule* dengan weight yang lebih besar berada di atas *rule* dengan bobot yang lebih kecil. Untuk *rule* dengan nilai prioritas dan bobot yang sama, urutan dari kedua *rule* tersebut ditentukan secara acak [SPR-05:14].

2.3.2 Prosedur *Dynamic Scripting*

Dua prosedur utama dari teknik *dynamic scripting* adalah *Script Generation* dan *Weight Adjustment*. *Rulebase* direpresentasikan dengan sebuah array dari beberapa objek *rule*. Tiap objek *rule* mempunyai tiga atribut, yaitu:

- Bobot. Mengandung informasi mengenai besarnya bobot dari sebuah *rule* sebagai nilai integer.
- *Line*. Mengandung teks sebenarnya dari *rule* untuk ditambahkan ke dalam *script* ketika *rule* tersebut terpilih.
- *Activated*. Merupakan variabel bertipe boolean yang mengindikasikan apakah *rule* tersebut diaktifkan ketika *script* yang mengandung *rule* tersebut diaktifkan.

Gambar 2.2 menjelaskan tentang prosedur dari *Script Generation*. Di dalam algoritma tersebut, fungsi '*InsertInScript*' berfungsi untuk menambahkan *line* ke dalam *script*. Jika *line* sudah ada di dalam *script*, fungsi tersebut tidak memiliki efek dan mengembalikan nilai '*false*'. Sebalikya, jika sebuah *line* dimasukkan, maka fungsi tersebut mengembalikan nilai '*true*'. Algoritma tersebut bertujuan untuk meletakkan *line* sejumlah '*scriptsize*' ke dalam *script*, tetapi mungkin berakhir dengan *line* yang lebih sedikit jika memerlukan lebih dari sejumlah '*maxtries*' percobaan untuk menemukan *line* baru. Fungsi '*FinishScript*' menambahkan satu atau beberapa *line* yang umumnya bisa diterapkan ke dalam *script* untuk memastikan bahwa *script* akan selalu bisa menemukan sebuah *action* untuk dieksekusi.

Algorithm 1 Script Generation

```
1: ClearScript()
2: sumweights = 0
3: for i = 0 to rulecount − 1 do
4:    sumweights = sumweights + rule[i].weight
5: end for
6: {Repeated roulette wheel selection}
7: for i = 0 to scriptsize − 1 do
8:    try = 0; lineadded = false
9:    while try < maxtries and not lineadded do
10:      j = 0; sum = 0; selected = −1
11:      fraction = random(sumweights)
12:      while selected < 0 do
13:         sum = sum + rule[j].weight
14:         if sum > fraction then
15:            selected = j
16:         else
17:            j = j + 1
18:         end if
19:      end while
20:      lineadded = InsertInScript(rule[selected].line)
21:      try = try + 1
22:   end while
23: end for
24: FinishScript()
```

Gambar 2.2. Algoritma *Script Generation*

Sumber: [SPR-05:10]

Gambar 2.3 menjelaskan tentang algoritma *Weight Adjustment*. Fungsi '*Calculate Adjustment*' menghitung *reward* atau *penalty* dari tiap *rule* yang sudah diaktifkan. Parameter *Fitness* adalah sebuah ukuran yang merepresentasikan performa dari *script* di dalam suatu pertempuran. Fungsi '*Distribute Remainder*' mendistribusikan selisih dari total weight dari *script* yang dihitung dengan total bobot original. Secara umum hal ini akan diimplementasikan sebagai perulangan dari semua bobot dari *rule*, dengan cara menambahkan *remainder* ke tiap bobot dari *rule* jika hal tersebut tidak menyebabkan bobot dari *rule* yang diproses melampaui batas yang ditentukan. Ketika keseluruhan bobot di dalam *rulebase* mendekati batas yang ditentukan, hal tersebut bisa menjadi proses yang memakan waktu yang cukup lama yang bisa mengganggu jalannya permainan. Sebagai

pemecahannya, *remainder* dapat dibawa ke pemanggilan *Weight Adjustment* berikutnya.

Algorithm 2 Weight Adjustment

```
1:  active = 0
2:  for i = 0 to rulecount − 1 do
3:     if rule[i].activated then
4:        active = active + 1
5:     end if
6:  end for
7:  if active <= 0 or active >= rulecount then
8:     return {no updates are needed.}
9:  end if
10: nonactive = rulecount − active
11: adjustment = CalculateAdjustment(Fitness)
12: compensation = −active ∗ adjustment/nonactive
13: remainder = 0
14: {Credit assignment}
15: for i = 0 to rulecount − 1 do
16:    if rule[i].activated then
17:       rule[i].weight = rule[i].weight + adjustment
18:    else
19:       rule[i].weight = rule[i].weight + compensation
20:    end if
21:    if rule[i].weight < minweight then
22:       remainder = remainder + (rule[i].weight − minweight)
23:       rule[i].weight = minweight
24:    else if rule[i].weight > maxweight then
25:       remainder = remainder + (rule[i].weight − maxweight)
26:       rule[i].weight = maxweight
27:    end if
28: end for
29: DistributeRemainder();
```

Gambar 2.3. Algoritma *Weight Adjustment*

Sumber: [SPR-05:11]

Perlu diperhatikan bahwa di dalam algoritma *Script Generation*, perhitungan dari '*sumweight*' pada baris 3 sampai 5 seharusnya menunjukkan hasil yang sama, yaitu jumlah dari semua bobot dari semua *rule* yang diinisialisasi. Akan tetapi, perhitungan tersebut dilakukan untuk memastikan bahwa algoritma *Script Generation* akan berhasil meskipun algoritma *Weight Adjustment* tidak mendistribusikan nilai dari *remainder* secara keseluruhan.

2.3.3 Fungsi *Weight Update*

Fungsi *Weight Update* didasarkan pada dua fungsi *fitness*, yaitu:

- Fungsi *Team-Fitness* $F(g)$, di mana g mengacu pada sebuah tim.
- Fungsi *Agent-Fitness* $F(a,g)$, di mana a mengacu pada *agent* dan g mengacu pada tim di mana *agent* a berasal.

Kedua fungsi *fitness* tersebut menghasilkan nilai dengan rentang [0,1]. Nilai dari *fitness* dihitung dihitung pada saat $t=T$, di mana T adalah *time step* di mana semua *agent* dari sebuah tim dinyatakan 'kalah', misalnya semua nyawa dari semua *agent* bernilai nol atau lebih kecil. Sebuah tim di mana semua *agent*nya mengalami kekalahan dinyatakan sebagai tim yang kalah dalam pertempuran. Sebuah tim yang setidaknya memiliki satu *agent* yang masih 'bertahan' diartikan sebagai tim pemenang. Dalam suatu kondisi yang sangat jarang sekali terjadi, kedua tim dimungkinkan mengalami kekalahan pada waktu yang bersamaan.

Fungsi *Team-Fitness* didefinisikan di dalam Persamaan (2-1).

$$F(g) = \sum_{c \in g} \begin{cases} 0 & \{g \text{ lost}\} \\ \frac{1}{2N_g}\left(1 + \frac{h_T(c)}{h_0(c)}\right) & \{g \text{ won}\} \end{cases} \quad (2\text{-}1)$$

g mengacu pada sebuah tim, c mengacu pada sebuah *agent*, $Ng \in N$ adalah jumlah dari semua *agent* yang ada di dalam tim g, dan $h_T(c) \in N$ adalah nyawa dari *agent* c pada saat t. Berdasarkan persamaan tersebut, tim yang 'kalah' mempunyai nilai *fitness* sebesar 0, sedangkan tim yang 'menang' mempunyai nilai *fitness*>0.5.

Fungsi perhitungan *fitness* sebuah *agent* dijelaskan di dalam Persamaan (2-2).

$$F(a,g) = \frac{1}{10}\left(3F(g) + 3A(a) + 2B(g) + 2C(g)\right) \quad (2\text{-}2)$$

Dari Persamaan (2-2) di atas, a mengacu pada *agent* yang *fitness*nya akan dihitung, dan g mengacu pada tim di mana *agent* a berasal. Persamaan tersebut mengandung 4 komponen, yaitu:

- $F(g)$. *Fitness* dari tim g. Berasal dari Persamaan (2-1).
- $A(a) \in [0,1]$. *Rating* dari kemampuan bertahan *agent* a.
- $B(g) \in [0,1]$. Ukuran dari nyawa semua *agent* yang berada di tim g.
- $C(g) \in [0,1]$. Ukuran dari besarnya *damage* dari semua *agent* yang menjadi lawan dari tim g.

Besarnya kontribusi masing-masing komponen tersebut terhadap hasil akhir ($F(a,g)$) ditentukan secara manual dengan memperhatikan pertimbangan bahwa *agent* seharusnya memberikan *reward* yang besar kepada kemenangan tim dan tingkat kelangsungan hidup masing-masing *agent* tersebut (dinyatakan dalam komponen $F(g)$ dan $A(a)$). Fungsi tersebut memberikan sedikit *reward* kepada tingkat kelangsungan hidup dari rekan satu tim dari *agent* yang bersangkutan dan kepada besarnya *damage* dari semua *agent* yang menjadi lawan dari tim *agent* yang bersangkutan (dinyatakan dalam komponen $B(g)$ dan $C(g)$). Dengan demikian, fungsi perhitungan *agent fitness* yang di dalam Persamaan (2-2) adalah ukuran yang baik terhadap tingkat kesuksesan dari *script* yang mengontrol *agent* yang bersangkutan.

Komponen $A(a)$, $B(g)$, dan $C(g)$ dinyatakan di dalam Persamaan (2-3), Persamaan (2-4), dan Persamaan (2-5).

$$A(a) = \frac{1}{3} \begin{cases} \min\left(\frac{D(a)}{D_{max}}, 1\right) & \{h_T(a) \leq 0\} \\ 2 + \frac{h_T(a)}{h_0(a)} & \{h_T(a) > 0\} \end{cases} \tag{2-3}$$

$$B(g) = \frac{1}{2N_g} \sum_{c \in g} \begin{cases} 0 & \{h_T(c) \leq 0\} \\ 1 + \frac{h_T(c)}{h_0(c)} & \{h_T(c) > 0\} \end{cases} \tag{2-4}$$

$$C(g) = \frac{1}{2N_{\neg g}} \sum_{c \notin g} \begin{cases} 1 & \{h_T(c) \leq 0\} \\ 1 - \frac{h_T(c)}{h_0(c)} & \{h_T(c) > 0\} \end{cases} \tag{2-5}$$

Di dalam persamaan di atas, *a* dan *g* adalah sama seperti yang disebutkan dalam Persamaan (2-2). *c*, *Ng*, dan $h_T(c)$ adalah sama seperti yang disebutkan dalam Persamaan (2-1). $N\neg g \in N$ adalah jumlah dari banyaknya *agent* di dalam tim yang menjadi lawan *g*. $D(a) \in N$ adalah waktu kematian dari *agent a*, dan *Dmax* adalah sebuah nilai konstan (dalam suatu experimen, *Dmax* diberi nilai sebesar 100, yang berarti sebanding dengan 10 giliran pemain, yang berarti lebih lama dari lama pertempuran pada umumnya). Nilai dari bobot dibatasi oleh rentang [*Wmin*, *Wmax*]. Pemberian *reward* dan *penalty* hanya dilakukan kepada *rule-rule* di dalam *script* yang telah dieksekusi. Nilai baru dari bobot dihitung dengan sebagai W+∆W, di mana W adalah nilai bobot yang dimiliki sebuah *rule* dan ∆W dinyatakan dalam formula di bawah ini (yang merupakan implementasi dari fungsi '*Calculate Adjustment*' dari prosedur *Weight Adjustment*):

$$\triangle W = \begin{cases} -\lfloor P_{max}\dfrac{b-F}{b}\rfloor & \{F < b\} \\ \lfloor R_{max}\dfrac{F-b}{1-b}\rfloor & \{F \geq b\} \end{cases} \qquad (2\text{-}6)$$

Dalam fungsi penghitungan nilai ∆W dalam Persamaan (2-6), $Rmax \in N$ dan $Pmax \in N$ adalah nilai *reward* maksimum dan nilai *penalty* maksimum. *F* adalah *agent fitness*. $b \in [0,1]$ adalah nilai *break-even*. Pada titik *break-even*, bobot dari sebuah *rule* tidak berubah sama sekali. Untuk menjaga agar jumlah dari semua bobot di dalam *rulebase* tetap konstan, perubahan bobot dieksekusi melalui distribusi kembali dari semua bobot dari tiap *rule* yang ada di dalam *rulebase* [SPR-05:15-17].

2.4 *Macro Action*

Secara sederhana, *macro action* adalah urutan-urutan dari beberapa *action* yang ditangani sebagai unit tunggal. Dengan mengelompokkan beberapa *action* dasar ke dalam sebuah *macro action* akan diperoleh keberagaman cara untuk mencapai suatu tujuan yang sama. Akan tetapi, dengan teknik pengelompokan

yang kurang baik *macro action* juga bisa mengurangi performa learning. Dengan *building block* yang lebih besar yang dimiliki *macro action*, seharusnya *dynamic script*ing bisa mengganti style bermain dengan lebih cepat.

Untuk mencapai tujuannya, *macro action* paling tidak harus memenuhi tiga syarat di bawah ini:

- Efektifitas. *Macro* harus efektif karena taktik yang efektif akan dibentuk dari *macro* yang efektif pula.
- Keberagaman. *Macro* harus berbeda satu sama lain dan harus merepresentasikan style permainan yang berbeda satu sama lain.
- Ukuran yang sesuai. Ukuran dari besarnya *macro* harus diseimbangkan di antara dua nilai yaitu; *rule* tunggal dan *script* yang lengkap. Di dalam kasus yang pertama, *macro* bisa dikatakan tidak berguna. Sedangkan di kasus yang ke dua tidak ada kemungkinan untuk mengkombinasikan beberapa *macro* yang merupaka cara yang ampuh untuk meningkatkan tingkat keberagaman.

Gambar 2.4 merupakan contoh *script* yang diawali oleh sebuah *macro action*:

```
[ opening macro #7 ]
 cast( "Monster Summoning I", closestenemy );
 cast( "Magic Missile", closestenemy );
 cast( "Chromatic Orb", closestenemy );
[ simple rules ]
 if healthpercentage < 50 then
    drink( "Potion of Healing" );
 cast( "Fireball", closestenemy );
 cast( "Luck" );
 cast( "Stinking Cloud", closestenemy );
 cast( "Grease", closestenemy );
 cast( "Flame Arrow", closestenemy );
 cast( "Magic Missile", closestenemy );
```

Gambar 2.4. Contoh *Script*

Sumber: [SZI-09:27]

2.5 *Difficulty Scaling*

Banyak peneliti dan game developer beranggapan bahwa game AI bisa dikatakan menghibur ketika game AI tersebut sulit dikalahkan. Walaupun anggapan tersebut dibenarkan oleh beberapa pemain yang tangguh, namun

beberapa pemain pemula beranggapan bahwa sebuah game menjadi sangat menarik jika cukup menantang dan mudah dikalahkan. Untuk memastikan bahwa game tetap menarik, permasalahannya bukan mengenai bagaimana sebuah computer bisa memproduksi taktik yang lemah sehingga pemain bisa menang, akan tetapi tentang bagaimana memproduksi taktik yang tidak terlalu kuat dengan ketentuan bahwa, dalam kemungkinan yang seimbang, pemain tidak menyadarinya.

Difficulty scaling adalah sebuah teknik adaptasi otomatis di dalam sebuah game yang mengatur tingkat kesulitan di dalam game sehingga sesuai dengan pemain. Ketika diaplikasikan ke dalam sebuah game AI, *difficulty scaling* berusaha mencapai sebuah permainan yang seimbang, misalnya sebuah game di mana kekuatan permainan sebuah *computer* dan pemain game tersebut seimbang.

Banyak game menyajikan sebuah pengaturan tingkat kesulitan, misalnya sebuah nilai yang menunjukkan seberapa sulit game tersebut. Pengaturan tingkat kesulitan tersebut bertujuan agar pemain pemula dan pemain yang sudah berpengalaman bisa menikmati tingkat tantangan yang sesuai yang disajikan oleh game tersebut. Pada umumnya, *difficulty scaling* memunyai tiga permasalahan utama. Yang pertama, pengaturan kesulitan yang disajikan sangat terbatas, yaitu hanya beberapa tingkat kesulitan (umumnya tiga atau empat). Ke dua, pengaturannya ditentukan oleh pemain, yang pada umumnya pemain tidak dapat mengukur tingkat kesulitan mana yang sesuai dengan kemampuannya. Ke tiga, pengaturan yang disajikan memiliki cakupan yang terbatas, (umumnya) hanya berpengaruh kepada tingkat kekuatan yang dimiliki *agent* yang dikontrol oleh computer, bukan taktik dari *agent* tersebut. Oleh karena itu, walaupun mempunyai tingkat kesulitan yang tinggi, musuh masih bisa menunjukkan *behaviour* yang serupa dengan musuh yang mempunyai tingkat kesulitan yang rendah dengan tingkat kekuatan yang lebih tinggi.

Ketiga kekurangan yang disebutkan di atas mungkin bisa tertutupi dengan cara mengaplikasikan *dynamic scripting* dengan sebuah mekanisme *difficulty scaling* yang memadai. *Dynamic scripting* mengganti taktik computer sesuai dengan bagaimana game tersebut dimainkan. Seperti, (i) *dynamic scripting*

berubah di dalam range yang kecil (tidak secara kasar/*coarse*), (ii) *dynamic scripting* berubah secara otomatis (pemain tidak dapat memilih sesuai keinginan), dan (iii) *dynamic scripting* berpengaruh pada taktik yang digunakan *computer* (cakupan dari *dynamic scripting* lebih luas).

Bagian ini menjelaskan bagaimana *dynamic scripting* bisa digunakan untuk membuat taktik baru yang digunakan musuh dan melakukan pengaturan tingkat kesulitan dari game AI tersebut agar sesuai dengan level yang dimiliki player. Lebih khususnya, di dalam bagian ini *dynamic scripting* mempunyai tujuan untuk meng*generate script* dengan jumlah kemenangan yang hampir sama dengan jumlah kekalahannya, walaupun dengan keadaan yang berubah-ubah. Di dalam bagian ini akan dijelaskan mengenai tiga teknik yang berbeda yang ditambahkan ke dalam *dynamic scripting* sehingga *computer* mampu melakukan learning mengenai bagaimana melakukan pertempuran yang seimbang dengan player di dalam sebua game. Teknik tersebut adalah (i) *high-fitness penalising*, (ii) *weight clipping*, dan (iii) *top culling*.

2.5.1 *High-Fitness Penalising*

Di dalam *dynamic scripting*, algoritma *Weight Adjustment* memberikan *reward* yang berbanding lurus dengan nilai *fitness* yang didapatkan: semakin besar nilai *fitness*nya, semakin besar pula nilai *reward*nya. Untuk menyajikan *behaviour* dengan tingkat kesulitan menengah, algoritma *Weight Adjustment* dapat dirubah sehingga memberikan *reward* yang tinggi kepada nilai *fitness* yang bernilai sedang, dan nilai *reward* yang kecil atau bahkan sebuah *penalty* kepada nilai *fitness* yang terlalu tinggi. Di dalam teknik ini, nilai *F* di dalam algoritma *Weight Adjustment* akan digantikan oleh *F'* yang didapatkan dari fungsi *high-fitness penalising* di dalam gambar 2.11.

Di dalam fungsi *high-fitness penalising*, nilai *F* adalah nilai dari *fitness* yang sebelumnya sudah dihitung, dan $p \in [0.5, 1]$, $p>b$, adalah nilai puncak *reward*. Semakin besar nilai *p*, *behaviour* musuh akan menjadi semakin efektif.

Dikarenakan ketergantungan nilai *p* pada taktik yang digunakan pemain, maka akan ditetapkan bahwa besarnya nilai tersebut akan beradaptasi sesuai dengan tingkat kesulitan di dalam game.

$$F' = \begin{cases} \frac{F}{p} & \{F \leq p\} \\ \frac{1-F}{p} & \{F > p\} \end{cases} \quad (2\text{-}7)$$

Pada awalnya, nilai *p* diinisialisasi sebesar *pinit*. Untuk setiap pertempuran yang dimenangkan oleh *computer*, nilai *p* berkurang sebesar *pdec* dengan batas minimum *pmin*. Dan untuk setiap pertempuran yang dimenangkan oleh tim yang menjadi lawan *computer* (pemain), nilai *p* bertambah sebesar *pinc* dengan batas maksimum *pmax*.

2.5.2 *Weight Clipping*

Ketika algoritma *Weight Update* dieksekusi, nilai maksimum *Wmax* menunjukkan tingkat optimasi maksimum yang dapat dicapai sebuah taktik. Nilai *Wmax* yang tinggi mengindikasikan tingkat pertumbuhan nilai weight yang tinggi, sehingga dapat dikatakan bahwa *rule-rule* yang paling efektif akan terpilih dalam selang waktu yang kecil. Hal ini akan berakibat *script* akan mendekati optimal. Sedangkan nilai *Wmax* yang kecil akan membatasi perkembangan nilai bobot dari sebuah *rule*. Hal ini akan menyebabkan taktik yang di*generate* menjadi lebih beragam dengan tingkat kesulitan yang cenderung mudah.

Teknik *weight clipping* akan merubah nilai *Wmax* secara otomatis dengan tujuan menyajikan game yang memiliki tingkat kesulitan yang seimbang dengan pemain. *Weight clipping* bertujuan memberikan nilai *Wmax* yang kecil ketika *computer* mempunyai frekuensi kemenangan yang lebih tinggi jika dibandingkan pemain, dan nilai *Wmax* yang besar ketika *computer* memiliki frekuensi kemenangan yang lebih rendah dibanding pemain.

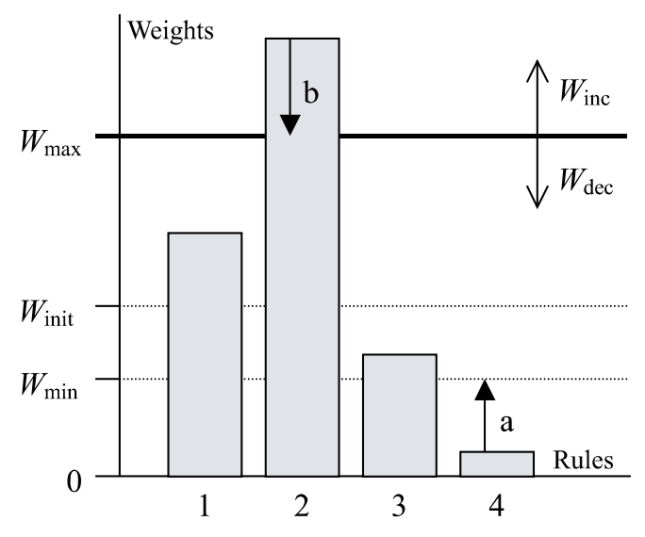
Setelah *computer* berhasil memenangkan pertempuran, nilai *Wmax* akan berkurang sebesar *Wdec* persen (akan tetapi tidak lebih kecil dibanding *Winit*

yang diinisialisasi). Setelah *computer* kalah dalam sebuah pertempuran, nilai *Wmax* akan turun sebesar *Winc* persen.

Gambar 2.5 menjelaskan proses *weight clipping* dengan parameter-parameter yang digunakan. Sebelum pemanggilan algoritma *Weight Adjustment*, nilai *Wmax* berganti sebesar *Winc* atau *Wdec* persen, tergantung dari hasil pertempuran yang dilakukan. Setelah pemanggilan algoritma *Weight Adjustment*, bobot dari *rule* 4 terlalu kecil, sehingga bobot dari *rule* tersebut akan naik menjadi sebesar nilai *Wmin* (ditunjukkan oleh anak panah '*a*'), sedangkan bobot dari *rule* 2 terlalu tinggi, dan akan berkurang menjadi sebesar nilai *Wmax* (ditunjukkan oleh anak panah '*b*').

2.5.3 *Top Culling*

Teknik *top culling* bisa dikatakan hampir sama dengan teknik *weight clipping*. Teknik ini memakai mekanisme adaptasi untuk nilai *Wmax* yang sama dengan teknik *weight clipping*. Perbedaannya adalah, di dalam *top culling* tidak ada pembatasan besarnya nilai weight tiap *rule* sebesar *Wmax*. Akan tetapi, *rule* dengan nilai bobot yang lebih besar dari *Wmax* tidak dapat dipilih untuk di*generate* ke dalam *script*. Akibatnya, ketika *computer* mempunyai frekuensi kemenangan yang lebih tinggi dibanding pemain, *rule-rule* yang paling efektif akan memiliki nilai bobot yang melampaui *Wmax*. Hal tersebut menyebabkan *rule* tersebut tidak dapat dipilih untuk di*generate* ke dalam *script*, sehingga musuh akan menggunakan taktik yang cenderung lemah. Kebalikannya, ketika *computer* mempunyai frekuensi kemenangan yang lebih rendah dibanding pemain, *rule-rule* dengan nilai bobot yang tinggi akan dapat dipilih untuk di*generate* ke dalam *script*. Jadi, ketika teknik *weight clipping* mampu memperoleh taktik yang lemah dengan cara memperbesar kemungkinan taktik tersebut untuk dapat di*generate* ke dalam *script*, teknik *top culling* memperoleh taktik yang lemah dengan cara membatasi akses terhadap taktik yang cenderung paling efektif. [SPR-05: 28-31]



Gambar 2.5. Proses *Weight Clipping* dan *Top Culling*

Sumber: [SPR-05:30]

BAB III
METODE PENELITIAN

Bab ini menjelaskan mengenai langkah-langkah yang akan ditempuh dalam penyusunan skripsi. Langkah-langkah tersebut meliputi studi literatur, perancangan, implementasi dan pengujian dari dari metode yang akan diterapkan. Kesimpulan dan saran disertakan sebagai catatan atas metode yang diterapkan dan kemungkinan arah pengembangan selanjutnya. Gambar 3.1 bersisi penjelasan mengenai diagram alir metode yang digunakan di dalam penelitian ini.

3.1 Studi Literatur

Tahap ini dilakukan untuk mendapatkan dasar-dasar teori dan sumber acuan untuk mengembangkan *agent* yang mengimplementasikan metode *dynamic scripting* menggunakan *macro action*. Informasi dan pustaka yang berkaitan dengan tugas akhir ini didapat dari buku, situs internet, penjelasan yang diberikan dosen pembimbing, dan rekan-rekan mahasiswa. Adapun teori-teori yang dipelajari tentang:

1. Game AI
2. *Computer Role-Playing Game* (*CRPG*)
3. *Dynamic Scripting*
4. *Macro Action*

3.2 Perancangan

Tahap ini berkisar tentang analisis dan perancangan algoritma yang dibutuhkan dalam teknik *dynamic scripting* menggunakan *macro action* dan gambaran umum mengenai bagaimana *agent* yang akan dibuat akan diimplementasikan ke dalam suatu *fighting system* di dalam game.

Tahap ini diawali dengan melakukan perancangan data untuk melakukan analisis mengenai data-data apa saja yang diperlukan dalam membangun sebuah *agent* AI yang menerapkan metode *dynamic scripting* menggunakan *macro action* dan dilanjutkan dengan melakukan perancangan algoritma untuk membangun

sebuah *agent* AI yang menerapkan metode *dynamic scripting* menggunakan *macro action*.

Gambar 3.2 menjelaskan tentang diagram alir tahapan yang akan dilakukan dalam proses perancangan. Gambar 3.3 menjelaskan diagram blok dari metode *dynamic scripting*.

3.3 Implementasi

Proses yang dilakukan dalam tahap ini antara lain; implementasi algoritma yang sudah dirancang, implementasi *skill*, implementasi karakter, dan implementasi *script*. Gambar 3.4 menjelaskan tentang diagram alir tahapan yang akan dilakukan dalam proses implementasi.

Implementasi algoritma utama dilakukan dengan mengacu kepada perancangan algoritma yang sudah dibuat dalam mengembangkan *agent* AI yang diterapkan ke dalam suatu battle scene di dalam game.

Data-data yang diperlukan untuk membangun *agent* akan dibuat dalam format XML. Pembuatan *agent* AI menggunakan teknik *dynamic scripting* dengan *macro action* dan implementasi *agent* ke dalam suatu *fighting system* di dalam game 3D akan dilakukan menggunakan game engine Unity3D dan Microsoft Visual C# 2010.

3.4 Pengujian

Proses pengujian akan dilakukan melalui tiga tahap yaitu pengujian metode *dynamic scripting*, pengujian *dynamic scripting* dengan *macro action*, dan pengujian metode *dificulty scaling*. Gambar 3.5 menjelaskan tentang diagram alir tahapan yang akan dilakukan dalam proses pengujian.

Pengujian akan dilakukan dengan mengacu pada metode pengujian yang dilakukan pada penelitian sebelumnya. Pengujian dilakukan dengan cara mempertemukan satu tim yang terdiri dari beberapa *agent* yang menggunakan metode *dynamic scripting* (*agent* dinamis) dengan tim yang terdiri dari beberapa *agent* yang menggunakan *rule* statis (*agent* statis) ke dalam sejumlah pertempuran. Berikut adalah parameter-parameter yang akan diujikan:

1. *Turning Point*

Turning point dapat digunakan untuk mendapatkan ukuran mengenai berapa banyak pertempuran yang dibutuhkan *agent* adaptif dalam memproduksi *script* yang baik secara konsisten. *Turning point* didapatkan dengan cara melakukan *recording* terhadap 10 pertempuran terakhir dari jumlah total pertempuran yang ada. Jika *score* dari *agent* adaptif lebih besar dari pada *agent* statis di 10 pertempuran terakhir, maka dapat dikatakan bahwa *turning point* sudah tercapai.

2. Efektifitas

Ukuran mengenai ketangguhan *agent* adaptif dapat dilakukan dengan cara menghitung banyaknya pertempuran yang berhasil dimenangkan oleh *agent* adaptif di 100 pertempuran terakhir dari sejumlah pertempuran. Tingkat kemenangan yang besar mengindikasikan bahwa algoritma tersebut mempunyai tingkat efektifitas yang tinggi.

3. Keberagaman

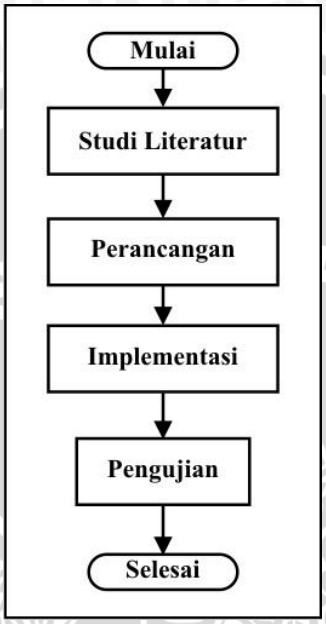
Untuk menghitung tingkat keberagaman digunakan metode berikut. Jumlah dari pertempuran yang diukur dinyatakan dengan *N*. Untuk setiap pertempuran $i \in \{1, \ldots, N\}$, ditinjau dari 100 pertempuran terakhir (dengan total pertempuran sebanyak 500). Untuk tiap *rule* $k \in \{1, \ldots, M\}$, frekuensi dari *rule k* di dalam pertempuran *i* dinyatakan sebagai p_{ik}, dengan kata lain:

$$\mathbf{p}_i := (p_{i,1}, \ldots, p_{i,M}) \tag{3-1}$$

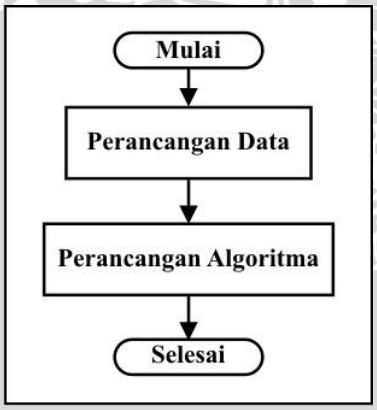
Tingkat keberagaman (dinyatakan dalam *D*) dapat diperoleh dari rata-rata jarak dari sepasang vektor p_i:

$$D := \frac{\sum_{1 \le i < i' \le N} \|\mathbf{p}_i - \mathbf{p}_{i'}\|}{N(N-1)/2} \tag{3-2}$$

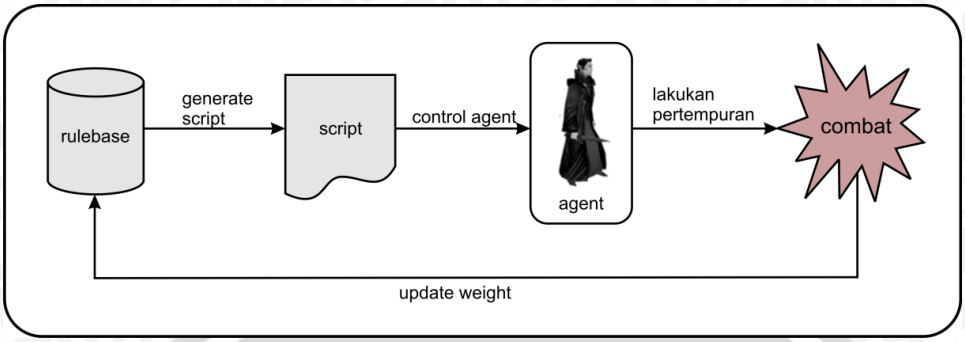
Nilai *D* yang besar mengindikasikan keberagaman yang tinggi [SZI-09:15].



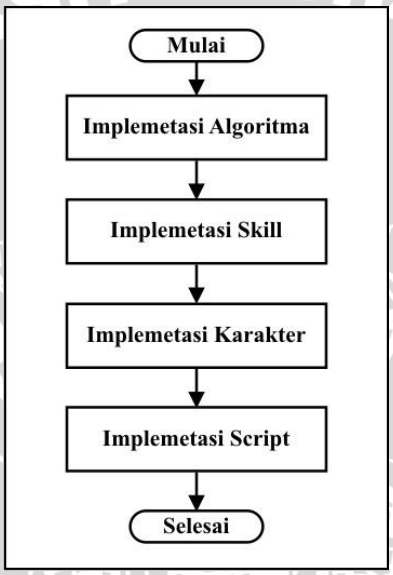
Gambar 3.1 Diagram Alir Metode Penelitian



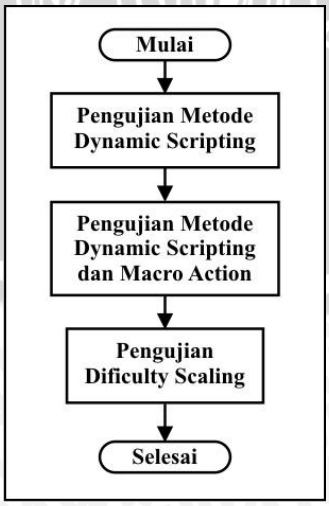
Gambar 3.2 Diagram Alir Perancangan



Gambar 3.3 Diagram Blok Metode *Dynamic Scripting*



Gambar 3.4 Diagram Alir Implementasi



Gambar 3.5 Diagram Alir Pengujian

BAB IV
PERANCANGAN

Bab ini membahas perancangan yang diperlukan untuk implementasi *fighting system* menggunakan *dynamic scripting* dan *macro action* dalam *Computer Role-Playing Game*. Perancangan dibagi menjadi tiga bagian yaitu penjelasan umum mengenai *agent* AI yang akan dibuat, perancangan data dan perancangan algoritma yang akan digunakan untuk membangun *agent* AI menggunakan *dynamic scripting* dan *macro action*.

4.1 Penjelasan Umum *Agent* AI Menggunakan Metode *Dynamic Scripting* dan *Macro Action*

Tingkat kesulitan dari taktik yang digunakan *computer* di dalam suatu game sangat ditentukan oleh kecerdasan buatan, atau yang lebih sering disebut *Artificial Intelegent* (AI), yang dimiliki game tersebut.

Agent AI merupakan sebuah entitas yang mampu memperoleh informasi dari data-data yang ada di dalam game yang digunakan untuk menentukan *action* apakah yang akan dilakukan berdasarkan dari informasi tersebut.

Agent AI yang akan dirancang dalam skripsi ini adalah *agent* AI dinamis yang diharapkan mampu beradaptasi dengan permainan pemain dan mampu menyajikan taktik yang beragam tanpa mengesampingkan efektifitas dari taktik yang disajikan.

Teknik *dynamic scripting* diharapkan mampu membuat *agent* AI menjadi dinamis dan beradaptasi dengan kemampuan pemain. Secara sederhana, *macro action* adalah urutan-urutan dari beberapa *action* yang ditangani sebagai unit tunggal. Dengan *building block* yang lebih besar yang dimiliki *macro action*, seharusnya teknik *dynamic scripting* mampu meningkatkan keberagaman taktik yang disajikan tanpa mengesampingkan efektifitas dari taktik tersebut.

Bisa dikatakan bahwa teknik *dynamic scripting* memiliki dua algoritma utama yaitu *Script Generation* dan *Weight Adjustment*.

Script Generation berfungsi membangun *script* dari *rulebase* dan *Macro Action Base* berdasarkan besarnya weight yang dimiliki oleh tiap *rule* dan *macro action* yang ada di dalam *rulebase* dan *Macro Action Base*. *Script* yang berhasil di-*generate* nantinya akan digunakan untuk mengontrol behaviour *agent* AI di dalam suatu pertempuran.

Setelah pertempuran yang dilakukan oleh *agent* AI, *Weight Adjustment* akan mengupdate *weight* dari *rule* dan *macro action* berdasarkan data dari pertempuran yang sudah dilakukan.

Agent AI yang dibangun nantinya akan diimplementasikan ke dalam suatu fighting system dalam game. *Fighting system* yang akan dirancang berupa *turn-based strategy* yang di dalamnya terdapat 2 tim yang bertempur. Setiap tim terdiri beberapa karakter dengan jumlah maksimal 3 karakter.

Setiap karakter dalam *fighting system* ini bergerak dan melakukan *action* sesuai dengan *action* atau *skill* yang dipilih. Suatu karakter bisa memilih *action*-nya ketika *Action Time* dari karakter tersebut penuh. Tiap *action* atau *skill* juga memiliki syarat minimum untuk dapat dipilih suatu karakter. Suatu karakter dianggap menyelesaikan gilirannya ketika selesai mengeksekusi sebuah *action* atau *skill*.

Suatu tim dinyatakan kalah jika seluruh karakter dari tim tersebut KO atau HP (*Health Point*) dari seluruh karakter di dalam tim tersebut mencapai angka 0. Tim yang bisa bertahan sampai akhir pertempuran dinyatakan menang.

4.2 Perancangan Data

Tahap ini berkisar pada penjabaran mengenai data-data yang diperlukan untuk membangun sebuah *agent* AI menggunakan teknik *dynamic scripting* dan *macro action*.

4.2.1 Perancangan *Skill*

Skill merepresentasikan *action* yang dimiliki sebuah karakter. Sebuah *skill* bisa dieksekusi oleh sebuah karakter dalam suatu turn di dalam pertempuran. Pengeksekusian sebuah *skill* membutuhkan sejumlah *Magic Point* (MP) dari

karakter. Besarnya *Magic Point* (MP) yang dibutuhkan untuk mengeksusi sebuah *skill* berbeda-beda bergantung pada tingkat kekuatan *skill* tersebut. Tabel 4.1 menjelaskan tentang kebutuhan yang dibutuhkan untuk membangun sebuah *skill*.

4.2.2 Perancangan *Rule*

Sebuah *rule* menggambarkan bagaimana suatu karakter melakukan *action*-nya. Behaviour dari *agent* AI sendiri sangat ditentukan oleh *rule-rule* yang membangun *script* yang dimiliki *agent* AI tersebut. Sebuah *rule* yang tidak valid bisa menyebabkan *agent* AI mengeksekusi *action* yang tidak masuk akal seperti melakukan penyerangan kepada *agent* AI lain dalam tim yang sama. Kebutuhan untuk membangun sebuah *rule* dijelaskan pada Tabel 4.3.

4.2.3 Perancangan *Macro Action*

Macro action adalah urutan-urutan dari beberapa *rule* yang ditangani sebagai unit tunggal. Pengelompokan beberapa *rule* bisa merepresentasikan sebuah taktik. Dengan *building block* yang lebih besar yang dimiliki *macro action*, seharusnya teknik *dynamic scripting* mampu meningkatkan keberagaman taktik yang disajikan. Tabel 4.5 menjelaskan tentang kebutuhan yang dibutuhkan untuk membangun sebuah *macro action*.

Di dalam sebuah pertempuran, sebuah karakter umumnya hanya mampu mengeksekusi maksimal 6 sampai 8 buah *rule*. Dengan alasan tersebut, penelitian ini akan menggunakan *macro action* yang terdiri dari tiga buah *rule*, sehingga sebuah karakter diharapkan mampu mengeksekusi 2 buah *macro action* (*Opening Macro* dan *Midgame Macro*) di dalam sebuah pertempuran.

4.2.4 Perancangan *Script*

Sebuah *script* yang merepresentasikan behaviour dari sebuah *agent* AI terdiri dari beberapa *rule*. Sebuah *script* yang valid selalu mengkondisikan *agent* AI untuk dapat mengeksekusi sebuah *rule* di tiap gilirannya. Sebuah *script* juga bisa diawali oleh beberapa *macro action*.

Di dalam penelitian ini, sebuah *script* dari *agent* AI yang menggunakan *dynamic scripting* dan *macro action* diawali oleh dua buah *macro action*. *Macro action* pertama disebut *Opening Macro*, dan *macro action* yang berada di bawah *Opening Macro* disebut *Midgame Macro*. Masing-masing *macro action* tersebut terdiri dari tiga *rule*. Sejumlah *rule* juga akan ditambahkan di akhir *script* untuk memastikan kevalidan *script* yang di*generate*. Kebutuhan untuk membangun sebuah *script* dijelaskan pada Tabel 4.6.

4.2.5 Perancangan Karakter

Merepresentasikan sekumpulan variabel yang dibutuhkan untuk membangun sebuah karakter. Parameter-parameter yang ada di dalam karakter bisa menentukan *rule* mana yang akan dieksekusi oleh sebuah *agent* AI. Tabel 4.7 menjelaskan tentang kebutuhan yang dibutuhkan untuk membangun sebuah karakter.

Tabel 4.1 Perancangan *Skill*

Data	**Keterangan**
Name	Nama dari *skill* yang akan dibuat.
ID	ID dari *skill* yang akan dibuat.
MP Consumed	Besarnya *Magic Point* (MP) yang dibutuhkan sebuah karakter untuk mengeksekusi *Skill* tersebut.
Hit Rate	Besarnya tingkat kesuksesan eksekusi *skill*. Semakin besar *Hit Rate* yang dimiliki sebuah *skill*, semakin besar pula kemungkinan suatu karakter berhasil mengeksekusi *skill* tersebut.
Skill Type	Tipe dari *skill*. Terdapat 2 jenis tipe, yaitu; *Magical* dan *Physical*. Besarnya *damage* yang dihasilkan *skill* dengan tipe *Magical* bergantung pada besarnya nilai *Magic* (MAG) dari karakter yang mengeksekusi *skill* tersebut dan besarnya *Wisdom* (WIS) dari karakter yang menjadi target dari *skill* tersebut,

	sedangkan besarnya *damage* yang dihasilkan *skill* dengan tipe *Physical* bergantung pada besarnya *Strength* (STR) dari karakter yang mengeksekusi *skill* tersebut dan besarnya *Defend* (DEF) dari karakter yang menjadi target dari *skill* tersebut.
Skill Target	Target dari *skill* yang akan dibuat. Ada 3 jenis target yaitu; *Self*, *Ally*, *Enemy*. Sebuah *skill* yang mempunyai *Skill Target* bertipe *Self* berarti *skill* tersebut diperuntukkan untuk karakter yang mengeksekusi *skill* yang dimaksud. Dan sebuah *skill* yang mempunya *Skill Target* bertipe *Ally* berarti *skill* tersebut diperuntukkan untuk karakter yang ada di dalam kelompok yang sama dari karakter yang mengeksekusi *skill* yang dimaksud. Sedangkan sebuah *skill* yang mempunya *Skill Target* bertipe *Enemy* berarti *skill* tersebut diperuntukkan untuk karakter yang ada di dalam kelompok yang menjadi lawan dari karakter yang mengeksekusi *skill* yang dimaksud.
Skill Target Count	Jumlah target dari *skill* yang akan dibuat. *Skill Target Count* diklasifikasikan menjadi 2 yaitu; *Single* dan *Party*. *Skill Target Count* dengan nilai *Single* mempunyai arti bahwa *skill* tersebut akan memberikan dampak atau efek hanya ke target karakter yang sebelumnya dipilih. Sedangkan sebuah *skill* yang mempunyai *Skill Target Count* dengan nilai *Party* akan memberikan dampak ke seluruh karakter dalam kelompok yang sama dengan target karakter yang sebelumnya dipilih.
Skill Effect	Merepresentasikan efek apa saja yang diberikan oleh *skill* yang dimaksud. Beberapa *stat* dari karakter yang mungkin terpengaruh yaitu; Level, HP, MP, STR, DEF, MAG, WIS, DEX, AGI.
Status Effect	Merepresentasikan *Status Effect* apa saja yang mungkin ditimbulkan oleh *skill* yang dimaksud jika berhasil mengenai suatu atau beberapa karakter. (Beberapa jenis *Status Effect* yang

	akan dirancang bisa dilihat di Tabel 4.2)

Tabel 4.2 Perancangan *Status Effect*

Status Effect	**Keterangan**
KO	HP dari suatu karakter yang memiliki *Status Effect KO* akan bernilai 0 atau dinyatakan mati dan tidak bisa melakukan pertempuran. Begitu juga sebaliknya, suatu karakter yang memiliki HP < = 0 dinyatakan mimiliki *Status Effect KO*.
Poison	HP suatu karakter yang terjangkit *Poison* akan berkurang pada saat karakter tersebut menyelesaikan tiap gilirannya.
Silent	Suatu karakter yang terjangkit *Silent* tidak bisa mengeksekusi *skill* bertipe *Magic*.
Paralyze	Suatu karakter yang terjangkit *Paralyze* tidak bisa mengeksekusi *skill* bertipe Physical.
Protect	Nilai DEF dari suatu karakter yang memiliki *Status Effect Protect* akan naik beberapa persen dari nilai DEF sebelumnya. Hal ini menyebabkan karakter akan memiliki daya tahan terhadap serangan bertipe *Physical* lebih besar dari sebelumnya.
Shell	Nilai WIS dari suatu karakter yang memiliki *Status Effect Shell* akan naik beberapa persen dari nilai WIS sebelumnya. Hal ini menyebabkan karakter akan memiliki daya tahan terhadap serangan bertipe *Magical* lebih besar dari sebelumnya.
Weak	Suatu karakter yang terjangkit *Weak* akan memiliki nilai daya serang dan bertahan yang lebih lemah dari sebelumnya.
Haste	Nilai AGI dari suatu karakter yang memiliki *Status Effect Haste* akan naik beberapa persen dari nilai sebelumnya. Hal ini berarti bahwa karakter dengan *Status Effect Haste* akan membutuhkan waktu lebih sedikit untuk mencapai gilirannya.
Slow	Nilai AGI dari suatu karakter yang memiliki *Status Effect Slow* akan turun beberapa persen dari nilai sebelumnya. Hal ini

	berarti bahwa karakter dengan *Status Effect Slow* akan membutuhkan waktu lebih banyak untuk mencapai gilirannya.
Faith	Nilai MAG dari suatu karakter yang memiliki *Status Effect Faith* akan naik beberapa persen dari nilai sebelumnya. Hal ini menyebabkan karakter akan memberikan *damage* bertipe *Magical* yang lebih besar.
Brave	Nilai STR dari suatu karakter yang memiliki *Status Effect Brave* akan naik beberapa persen dari nilai sebelumnya. Hal ini menyebabkan karakter akan memberikan *damage* bertipe *Physical* yang lebih besar.

Tabel 4.3 Perancangan *Rule*

Data	**Keterangan**
ID	ID dari *rule* yang dimaksud. ID pada *rule* dimaksudkan untuk membuat satu *rule* bersifat unique dari *rule* yang lain.
Rule Condition Target	Target dari *rule* yang dimaksud. *Rule Condition Target* bisa bernilai *Self* yang berarti bahwa target adalah karakter yang memiliki *rule* tersebut, *Ally* yang berarti bahwa target adalah semua karakter yang berada dalam tim yang sama dengan pemilik *rule* tersebut, dan *Enemy* yang berarti bahwa target adalah semua karakter yang berada di dalam tim yang menjadi lawan dari karakter yang memiliki *rule* tersebut.
Stat	*Stat* yang menjadi acuan atau tujuan dari syarat *rule* yang dimaksud. *Stat* yang dimaksud adalah stat dari *rule* condition target yang dimiliki *rule* tersebut. *Stat* bisa bernilai Level, HP, MP, STR, DEF, MAG, WIS, AGI, atau DEX.
Rule Condition	Kondisi yang menjadi syarat *rule* tersebut bisa dieksekusi. (Jenis-jenis kondisi yang akan dirancang bisa dilihat di Tabel 4.4)
Condition	Presentase besarnya nilai *Stat* yang dimiliki *Rule Condition*

Modifier	*Target* yang menjadi syarat bahwa *rule* tersebut bisa dieksekusi.
Skill	*Skill* yang akan dieksekusi oleh karakter yang memiliki *rule* tersebut jika syarat-syarat dan kondisi sudah terpenuhi.
Executed	Bernilai awal '*False*', dan akan menjadi '*True*' jika *rule* berhasil dieksekusi.
Weight	Nilai bobot dari *rule* yang dimaksud. Nilai ini mempengaruhi besarnya kemungkinan *rule* tersebut di*generate* ke dalam *script*.

Tabel 4.4 Perancangan *Rule Condition*

Rule Condition	**Keterangan**
Any	Jika sebuah *rule* memiliki *Rule Condition* bernilai *Any*, maka *rule* tersebut mengabaikan syarat *Stat* dan *Modifier*-nya.
More Than	Kondisi yang berarti bahwa *rule* bisa dieksekusi jika nilai *Stat* dari *Rule Condition Target* yang dimaksud bernilai lebih besar dari *Condition Modifier* yang dimiliki *rule* yang dimaksud.
Less Than	Kondisi yang berarti bahwa *rule* bisa dieksekusi jika nilai *Stat* dari *Rule Condition Target* yang dimaksud bernilai lebih kecil dari *Condition Modifier* yang dimiliki *rule* yang dimaksud.
Most	Kondisi yang berarti bahwa *rule* yang dimaksud bisa dieksekusi jika nilai *Stat* yang dimaksud memiliki nilai terbesar di antara semua *Rule Condition Target* yang dimiliki *rule* yang dimaksud. *Rule Condition* ini mengabaikan nilai *Condition Modifier* yang dimiliki.
Least	Kondisi yang berarti bahwa *rule* yang dimaksud bisa dieksekusi jika nilai *Stat* yang dimaksud memiliki nilai terkecil di antara semua *Rule Condition Target* yang dimiliki *rule* yang dimaksud. *Rule Condition* ini mengabaikan nilai *Condition Modifier* yang dimiliki.
Is Inflicted (X)	*Rule* yang dimaksud bisa dieksekusi jika salah satu target karakter di dalam *Rule Condition Target* terjangkit *Status*

	Effect '*X*'. '*X*' bisa bernilai *KO, Poison, Silent, Paralyze, Protect, Shell, Weak, Haste, Slow, Brave,* dan *Faith*.
Is Not Inflicted (X)	*Rule* yang dimaksud bisa dieksekusi jika salah satu target karakter di dalam *Rule Condition Target* tidak terjangkit *Status Effect* '*X*'. '*X*' bisa bernilai *KO, Poison, Silent, Paralyze, Protect, Shell, Weak, Haste, Slow, Brave,* dan *Faith*.

Tabel 4.5 Perancangan *Macro Action*

Data	**Keterangan**
ID	ID dari *macro action* yang dimaksud. Layaknya sebuah *rule*, ID di dalam *macro action* bertujuan untuk membuat sebuah *macro action* bersifat *unique* satu sama lain.
Weight	Nilai bobot dari *macro action* yang dimaksud. Layaknya sebuah *rule*, besarnya *Weight* mempengaruhi besarnya kemungkinan sebuah *macro action* digenerate ke dalam sebuah *script*.
Rules	Sejumlah *rule* yang membangun *macro action*. Di dalam penelitian ini, jumlah dari *rule* yang membangun sebuah *macro action* ditentukan sebesar 3 *rule*.
Executed	Bernilai '*False'* pada saat inisialisasi awal, dan akan bernilai '*True*' jika berhasil dieksekusi.

Tabel 4.6 Perancangan *Script*

Data	**Keterangan**
Opening Macro	Sebuah *macro action* yang mengawali *script* dari *agent* AI menggunakan *dynamic scripting* dan *macro action*.
Midgame Macro	Sebuah *macro action* yang digenerate setelah *Opening Macro* di dalam *script* dari *agent* AI menggunakan *dynamic scripting* dan *macro action*
Basic Rules	Sejumlah *rule* yang digenerate untuk membangun sebuah *script*.

Tabel 4.7 Perancangan Karakter

Data	**Keterangan**
Name	Nama dari karakter yang akan dibuat.
ID	ID dari karakter yang akan dibuat.
Level	Level dari karakter yang akan dibuat.
Health Point (HP)	Merepresentasikan besarnya nyawa dari karakter yang akan dibuat. Semakin besar HP dari suatu karakter, semakin lama karakter tersebut bertahan menerira serangan dari karakter lain. Suatu karakter dikatakan *KO* atau tidak bisa melanjutkan pertarungan jika HP dari karakter tersebut mencapai nilai 0.
Magic Point (MP)	Merepresentasikan besarnya kapasitas magic yang dimiliki suatu karakter. MP diperlukan oleh sebuah karakter untuk mengeksekusi sebuah *skill*. Besarnya MP yang dibutuhkan untuk mengeksusi sebuah *skill* berbeda-beda bergantung pada tingkat kekuatan *skill* tersebut. Dengan kata lain, suatu karakter yang memiliki MP yang tinggi akan lebih bisa mengeksekusi *skill* lebih banyak dibanding karakter yang mempunyai sedikit MP.
Strength (STR)	Merepresentasikan kekuatan fisik dari sebuah karakter. Semakin besar STR suatu karakter, semakin besar pula damage yang dihasilkan ketika karakter tersebut mengeksekusi *skill* bertipe *Physical* terhadap karakter lain.
Defense (DEF)	Merepresentasikan pertahanan fisik dari sebuah karakter. Semakin besar DEF suatu karakter, semakin kecil *damage* yang diterima karakter tersebut ketika ada karakter lain mengeksekusi *skill* bertipe *Physical* terhadap karakter tersebut.
Magic (MAG)	Merepresentasikan kekuatan *magic* dari sebuah karakter. Semakin besar MAG suatu karakter, semakin besar pula *damage* yang dihasilkan ketika karakter tersebut mengeksekusi *skill* bertipe *Magical* terhadap karakter lain.

Wisdom (WIS)	Merepresentasikan pertahanan *magic* dari sebuah karakter. Semakin besar WIS suatu karakter, semakin kecil *damage* yang diterima karakter tersebut ketika ada karakter lain mengeksekusi *skill* bertipe *Magical* terhadap karakter tersebut.
Agility (AGI)	Merepresentasikan kelincahan pertahanan dari sebuah karakter. Semakin besar AGI suatu karakter, semakin besar pula peluang karakter tersebut untuk bisa menghindari suatu serangan.
Dexternity (DEX)	Merepresentasikan kecepatan serangan dari sebuah karakter. Semakin besar DEX suatu karakter, semakin besar pula peluang tingkat kesuksesan karakter tersebut dalam mengeksekusi *skill*.
Skills	Daftar dari *skill* yang dimiliki sebuah karakter.
Rule Base	Sejumlah *rule* dengan bobot tertentu yang digunakan sebagai acuan sebuah karakter dalam menggenerate sebuah *script*.
Opening Macro Action Base	Sejumlah *macro action* dengan bobot tertentu yang digunakan sebagai acuan sebuah karakter dalam menggenerate sebuah *Opening Macro* ke dalam *script*.
Midgame Macro Action Base	Sejumlah *macro action* dengan bobot tertentu yang digunakan sebagai acuan sebuah karakter dalam menggenerate sebuah *Midgame Macro* ke dalam *script*.

4.3 Perancangan Algoritma

Teknik *dynamic scripting* pada umumnya terdiri dari 2 algoritma utama yaitu *Script Generation* dan *Weight Update*. *Script Generation* digunakan untuk menggenerate sebuah *script* yang valid untuk dijadikan sebagai kontrol *behaviour* oleh *agent* AI yang akan melakukan sebuah pertempuran. Sedangkan *Weight Update* digunakan untuk mengupdate nilai bobot dari *rule* dan *macro action* yang sebelumnya di*generate* untuk pertempuran yang sebelumnya dilakukan oleh *agent* AI.

Secara umum, diagram alir mengenai jalannya *agent* AI menggunakan teknik *dynamic scripting* dan *macro action* ditunjukkan dalam Gambar 4.1.

4.3.1 Algoritma *Script Generation*

Algoritma ini berfungsi untuk menggenerate *script* yang akan digunakan untuk mengontrol *agent* di dalam pertempuran. Algoritma ini dijalankan setiap terjadi instansiasi *agent* AI.

Algoritma ini akan melakukan seleksi sejumlah *rule* dari *rulebase*, sebuah *Opening Macro* dari *Opening Macro Base*, dan sebuah *Midgame Macro* dari sebuah *Midgame Macro Base* untuk di*generate* menjadi sebuah *script*. Sejumlah *rule* juga akan ditambahkan di akhir *script* untuk memastikan kevalidan *script* yang di*generate*.

Diagram alir yang menjelaskan mengenai proses algoritma *Script Generation agent* AI menggunakan teknik *dynamic scripting* dan *macro action* ditunjukkan dalam Gambar 4.2.

4.3.2 Algoritma *Do Battle*

Algoritma ini mengatur bagaimana sebuah pertempuran berlangsung. Pada tiap gilirannya, *agent* AI akan melakukan seleksi *rule* yang memenuhi syarat untuk dieksekusi sesuai dengan urutan dari *script* yang dimiliki *agent* AI tersebut.

Prioritas seleksi *rule* dimulai dari *Opening Macro*, *Midgame Macro*, kemudian sejumlah *rule* yang ditambahkan setelah *Midgame Macro* untuk mengantisipasi jika ternyata tidak ada *rule* di dalam *Opening Macro* dan *Midgame Macro* yang memenuhi syarat untuk dieksekusi. Jika dari keseluruhan proses seleksi tidak ditemukan *rule* yang memenuhi syarat, maka *agent* AI akan mengeksekusi *rule default* yang sebelumnya ditambahkan secara otomatis di dalam algoritma *Script Generation*.

Diagram alir yang menjelaskan mengenai proses algoritma *Do Battle* yang dimiliki *agent* AI menggunakan teknik *dynamic scripting* dan *macro action* ditunjukkan dalam Gambar 4.3.

4.3.3 Algoritma *Get Battle Record*

Di dalam algoritma ini akan didapatkan parameter-parameter yang nantinya akan digunakan untuk meng*update* nilai bobot dari *rule* dan *macro action* yang digenerate oleh *agent* AI dalam pertempuran sebelumnya.

Diagram alir yang menjelaskan mengenai proses algoritma *Get Battle Record agent* AI menggunakan teknik *dynamic scripting* dan *macro action* ditunjukkan dalam Gambar 4.4.

4.3.4 Algoritma *Weight Update*

Algoritma ini berfungsi untuk mengupdate nilai bobot di dalam *rulebase*, *Opening Macro Base*, dan *Midgame Macro Base* dari sebuah karakter yang mengimplementasikan *agent* AI menggunakan *dynamic scripting* dan *macro action*. Algoritma ini dieksekusi di akhir pertempuran yang melibatkan *agent* AI. Algoritma ini akan melakukan update nilai bobot dari *rule* dan *macro action* yang sebelumnya digunakan di dalam pertempuran *agent* dinamis

Diagram alir yang menjelaskan mengenai proses algoritma *Weight Update agent* AI menggunakan teknik *dynamic scripting* dan *macro action* ditunjukkan dalam Gambar 4.5.

4.3.5 Algoritma *Difficulty Scaling*

Algoritma *difficulty scaling* diperlukan untuk mengatur tingkat kesulitan *agent*. Pengaturan tingkat kesulitan *agent* dimaksudkan untuk mengendalikan *agent* dalam mengenrate sebuah *scipt*. Dengan pengaturan tersebut, kemungkinan sebuah *agent* untuk menggenerate sebuah *script* yang mendekati sempurna (tingkat keefektifan yang terlalu tinggi) bisa diperkecil. Pengaturan tingkat kesulitan juga bisa dimaksudkan sebagai langkah untuk mendukung fitur dari sebuah game yang menyediakan beberapa pilihan tingkat kesulitan dalam memainkan game tersebut.

Di dalam penelitian ini akan dibandingkan tiga metode pengaturan tingkat kesulitan untuk mendukung teknik *dynamic scripting*. Adapun metode yang akan dibandingkan adalah *high-fitness penalising*, *weight clipping*, dan *top culling*.

4.3.5.1 Algoritma *High-Fitess Penalising*

Teknik ini dimaksudkan untuk memberikan *penalty* ke setiap *rule* atau *macro action* yang memiliki nilai *fitness* terlalu tinggi, sehingga diharapkan kemungkinan dipilihnya *rule* atau *macro action* yang memiliki bobot tidak terlalu tinggi menjadi lebih besar. Alur mengenai pemanggilan algoritma ini secara umum ditunjukkan di dalam Gambar 4.6.

4.3.5.2 Algoritma *Weight Clipping*

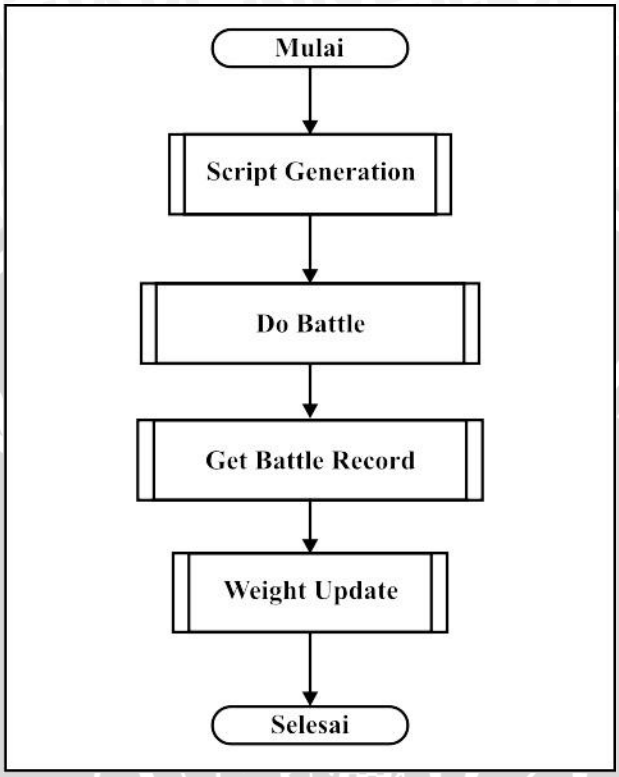
Algoritma di dalam *weight clipping* akan membatasi nilai maksimum dan minimum weight sebagai parameter yang dibutuhkan di dalam algoritma *Weight Update*. Algoritma ini dijalankan sebelum pemanggilan algoritma *Weight Update*. Dengan meningkatkan nilai bobot minimum (*Wmin*) yang terlalu rendah dan memangkas nilai bobot maksimum (*Wmax*) yang terlalu tinggi, diharapkan selisih bobot antar *rule* dan *macro action* menjadi lebih kecil.

Nilai yang dihasilkan dari algoritma ini akan digunakan sebagai parameter di dalam algoritma *Weight Update*. Alur mengenai pemanggilan algoritma ini secara umum ditunjukkan di dalam Gambar 4.7.

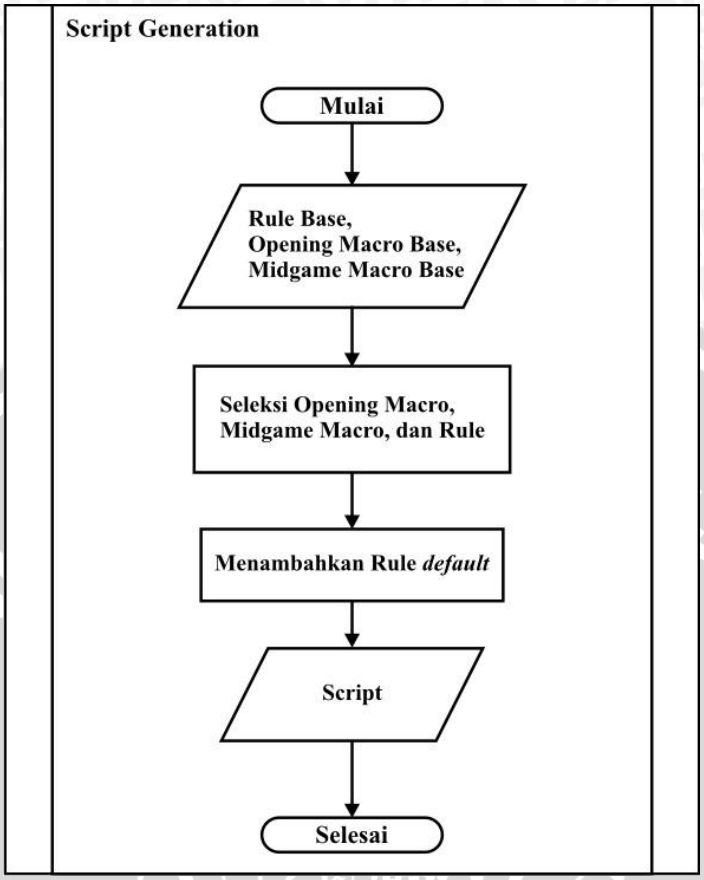
4.3.5.3 Algoritma *Top Culling*

Algoritma dan fungsi *update* parameter yang dipakai di dalam *top culling* sama dengan *weight clipping*. Berbeda sedikit dari *weight clipping*, di dalam *top culling* tidak ada pembatasan besarnya nilai bobot tiap *rule* atau *macro action* sebesar '*wmax*'.

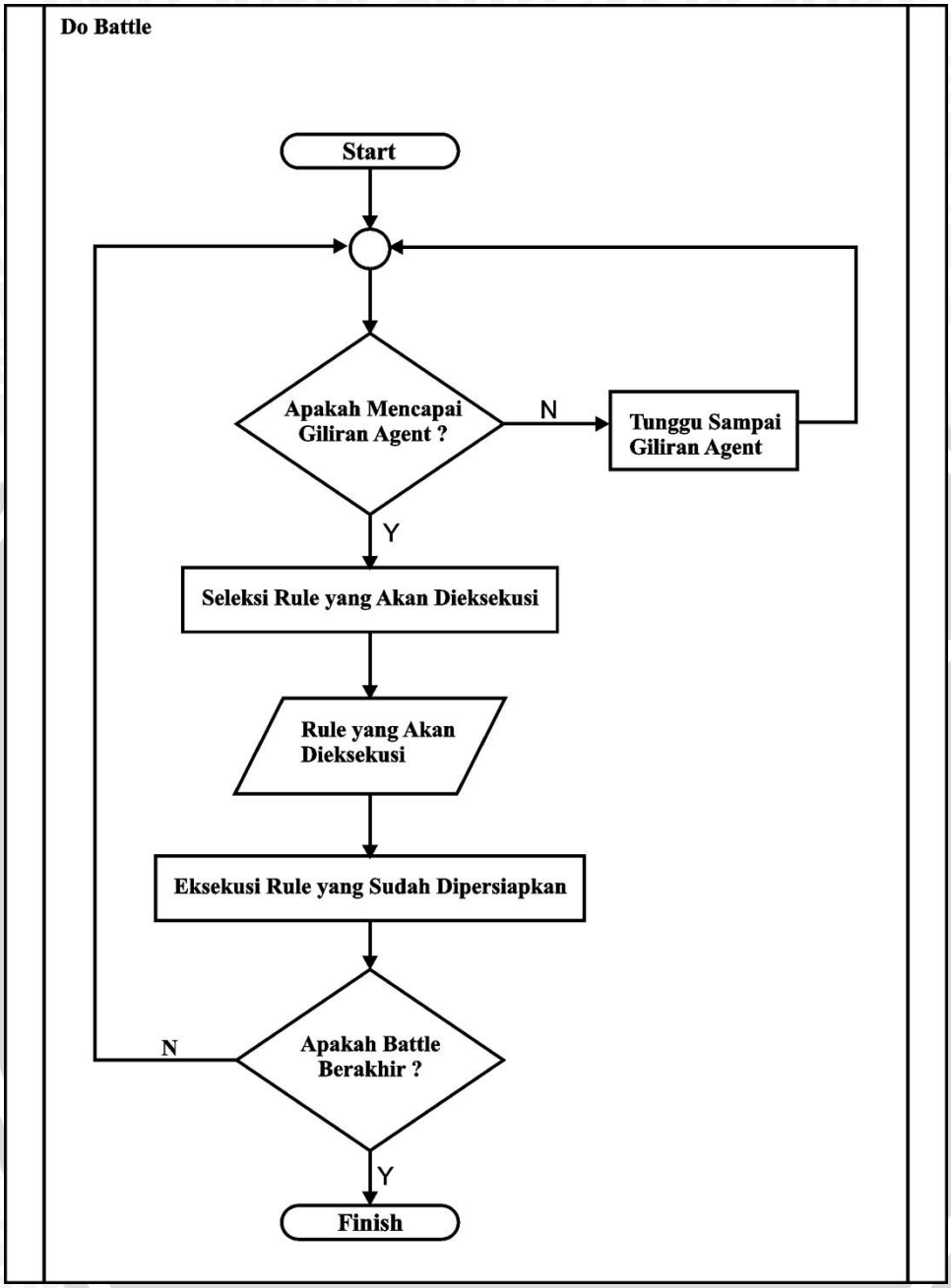
Nilai yang dihasilkan dari proses *update* di dalam algoritma *weight clipping* akan digunakan sebagai parameter di dalam proses seleksi *rule* dan *macro action* sebelum pemanggilan algoritma *Script Generation*. *Rule* atau *macro action* yang memiliki bobot lebih besar dari '*wmax*' tidak dapat terpilih untuk di*generate* ke dalam *script*. Alur mengenai pemanggilan algoritma ini secara umum ditunjukkan di dalam Gambar 4.8.



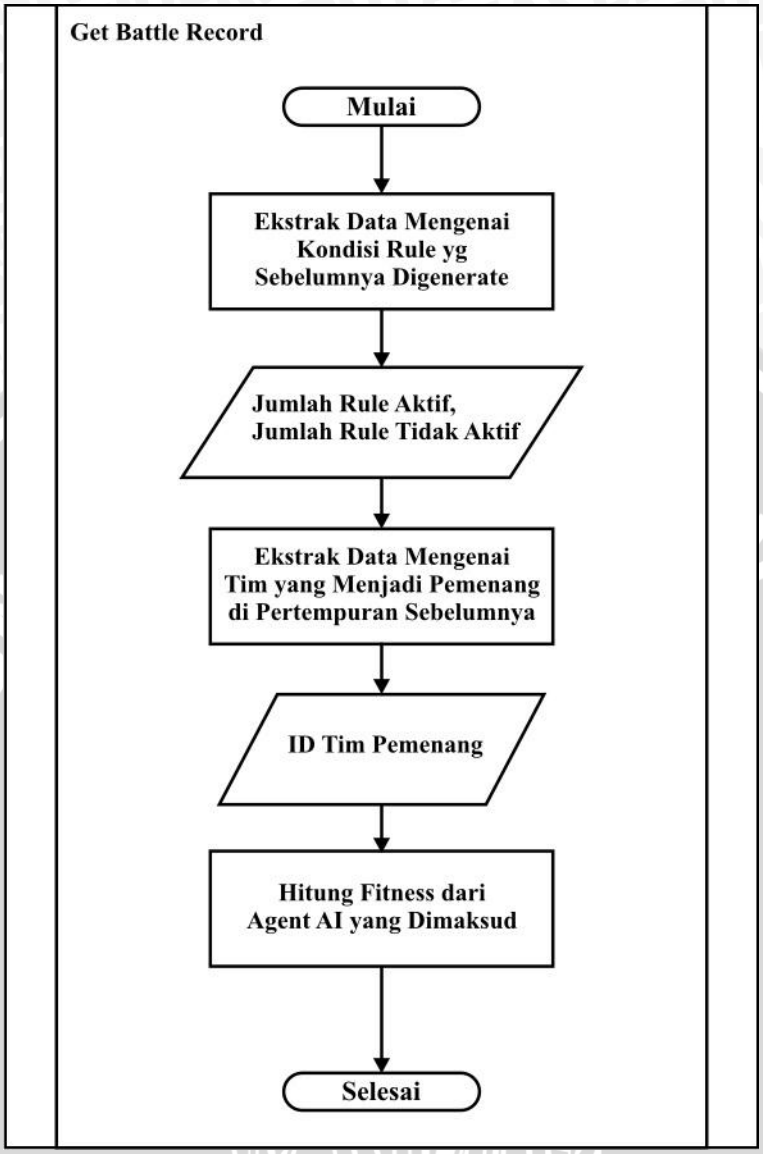
Gambar 4.1 Diagram Alir Kinerja *Dynamic Scripting*



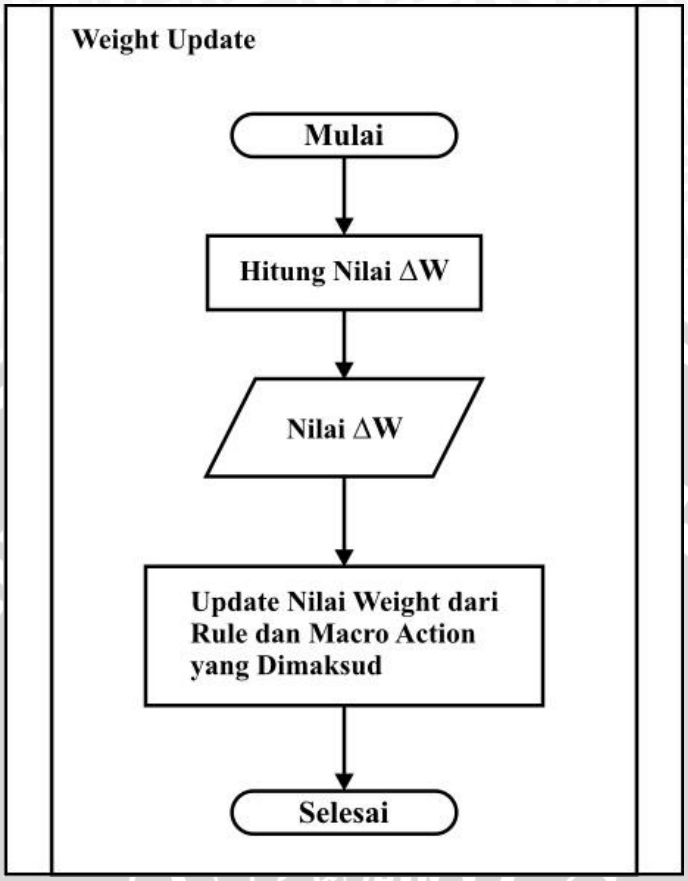
Gambar 4.2 Diagram Alir Algoritma *Script Generation*



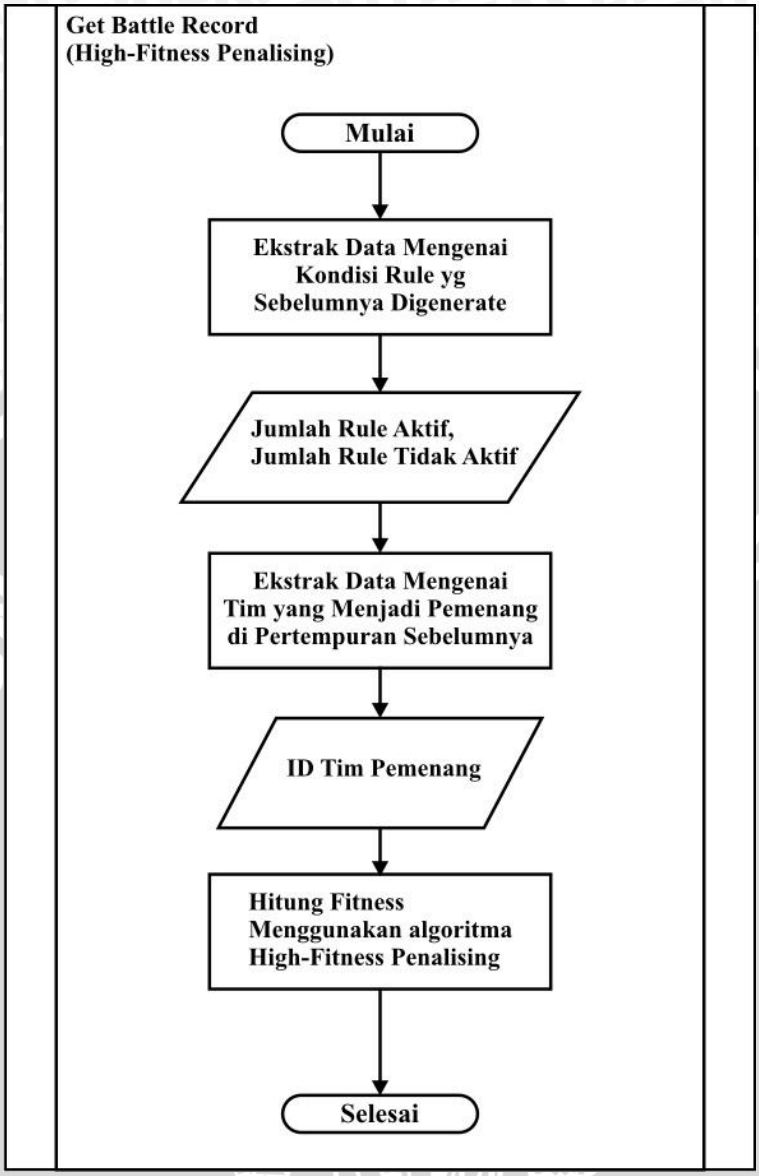
Gambar 4.3 Diagram Alir Algoritma *Do Battle*



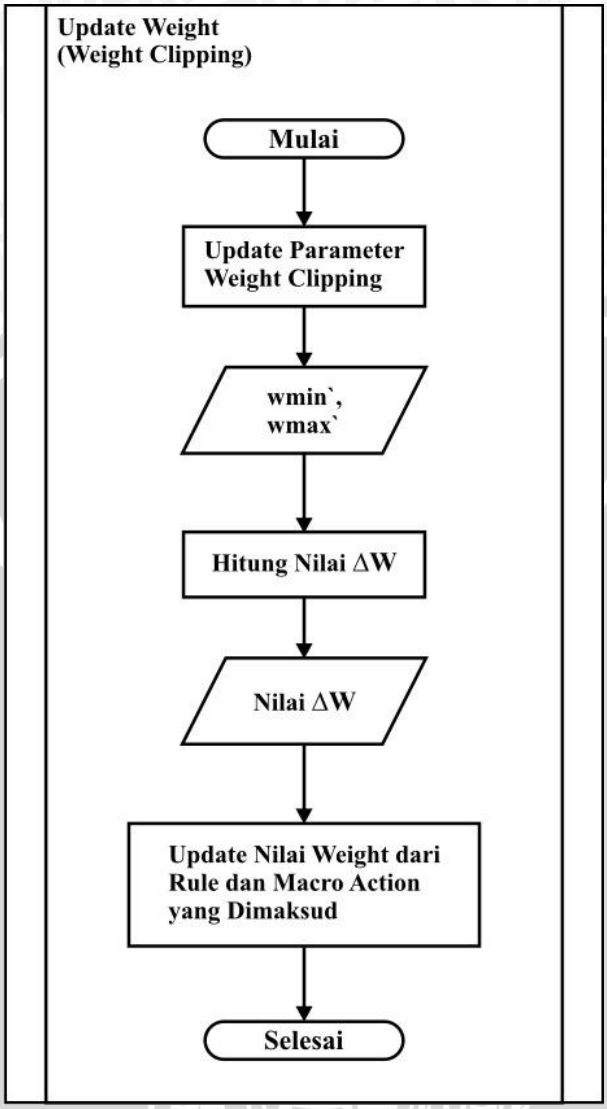
Gambar 4.4 Diagram Alir Algoritma *Get Battle Record*



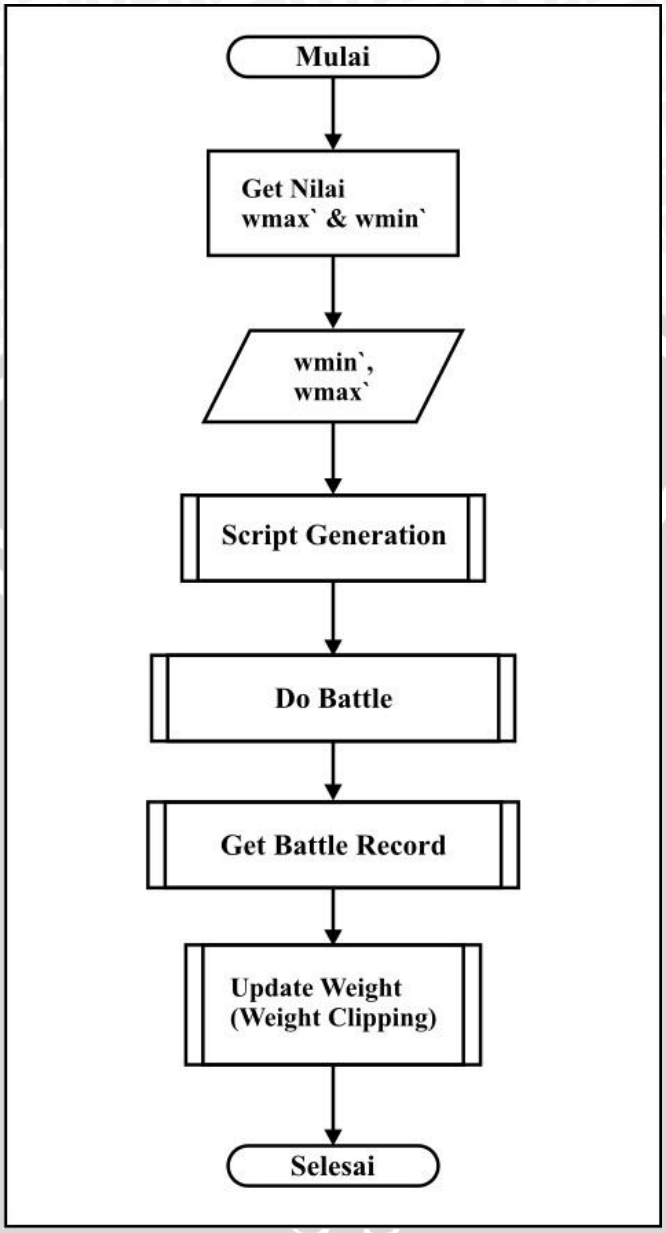
Gambar 4.5 Diagram Alir Algoritma *Weight Update*



Gambar 4.6 Diagram Alir Algoritma *High-Fitness Penalising*



Gambar 4.7 Diagram Alir Algoritma *Weight Clipping*



Gambar 4.8 Diagram Alir Algoritma *Top Culling*

BAB V
IMPLEMENTASI

Bab ini membahas seputar implementasi *agent* AI menggunakan *dynamic scripting* dan *macro action* berdasarkan data dan kebutuhan yang didapatkan dari proses perancangan. Pada bab ini akan dijelaskan mengenai spesifikasi sistem, batasan-batasan dalam implementasi, implementasi algoritma, implementasi *skill*, implementasi karakter, dan implementasi *agent* AI ke dalam *fighting system* game.

5.1 Spesifikasi Sistem

Pada tahap ini akan dijelaskan tentang spesifikasi sistem, baik spesifikasi perangkat keras maupun spesifikasi perangkat lunak, yang digunakan untuk mengembangkan *agent* AI menggunakan *dynamic scripting* dan *macro action*.

Spesifikasi perangkat keras yang digunakan untuk mengembangkan *agent* AI menggunakan *dynamic scripting* dan *macro action* dijelaskan di dalam Tabel 5.1.

Spesifikasi perangkat lunak yang digunakan untuk mengembangkan *agent* AI menggunakan *dynamic scripting* dan *macro action* dijelaskan di dalam Tabel 5.2.

Tabel 5.1 Tabel Spesifikasi Perangkat Keras

Processor	Intel(R) Core(TM) i3 CPU M330 @ 2.13GHz
Memory (RAM)	4096MB
Graphic Card	ATI Mobility Radeon HD 5470 2746MB

Tabel 5.2 Tabel Spesifikasi Perangkat Lunak

Operating System	Windows 7 Home Basic 64-bit (6.1, Build 7600)
DirectX Version	DirectX 11
Direct3D Version	8.14.10.0723

OpenGL Version	6.14.10.9262
Programming Language	C#
Framework	.NET Framework 4
Game Engine	Unity3 Version 3.4.0f5
Integrated Development Environtment	Microsoft Visual C# 2010 Express, Unity MonoDevelop 2.4

5.2 Batasan-batasan Implementasi

Berikut adalah batasan-batasan dalam implementasi *agent* AI menggunakan *dynamic scripting* dan *macro action*:

1. *Agent* AI yang dirancang berbentuk *Dynamic-Link Library (*DLL).
2. *Agent* AI akan diimplementasikan ke dalam suatu *battle system* menggunakan Unity3 Game Engine.
3. Metode learning *macro action* yang digunakan adalah *cross entrophy learning*.

5.3 Implementasi Algoritma

Implementasi algoritma di dalam proses perancangan *agent* AI menggunakan *dynamic scripting* dan *macro action* mengacu pada perancangan algoritma yang sudah dibuat.

5.3.1 Implementasi Algoritma *Script Generation*

Algoritma ini berfungsi untuk meng-*generate* sebuah *script* yang akan digunakan untuk mengontrol behaviour *agent* AI di dalam suatu pertempuran. Algoritma utama di dalam *Script Generation* dibagi menjadi dua yaitu algoritma *Select Macro* dan *Select Rules*.

Sebuah *script* akan diawali oleh sebuah *macro action* yang disebut *Opening Macro*, dilanjutkan dengan sebuah *Midgame Macro*. Sejumlah *rule* juga akan ditambahkan setelah *Midgame Macro* untuk memastikan bahwa *script* bisa berjalan dengan baik. Sejumlah *default rule* juga akan ditambahkan di akhir *script*

untuk memastikan bahwa *agent* AI selalu mempunyai sebuah *rule* untuk dieksekusi di tiap gilirannya.

5.3.1.1 Algoritma *Select Macro*

Algoritma ini melakukan seleksi sebuah *macro action* dari sejumlah *macro* yang ada di dalam sebuah *macro base*. Sebuah *Opening Macro* akan diseleksi dari sejumlah *macro action* yang terdapat di dalam *Opening Macro Base*. Sedangkan sebuah *Midgame Macro* akan diseleksi dari sejumlah *macro action* yang terdapat di dalam *Midgame Macro Base*.

Algoritma *Select Macro*

```
1   sumweight = 0
2   for i=0 to macrocount – 1 do
3       sumweight = sumweight  + macro[i].weight
4   end for
5   j = 0; sum = 0; selected = -1
6   fraction = random(sumweight)
7   while selected < 0 do
8       sum = sum + macro[j].weight
9       if sum > = fraction then
10          selected = j
11      else
12          j = j + 1
13      end if
14  end while
15  return macro[selected]
```

Gambar 5.1 Algortima *Select Macro*

5.3.1.2 Algoritma *Select Rule*

Algoritma ini melakukan seleksi sejumlah *rule* dari sejumlah *rule* yang terdapat di dalam *rulebase*.

	Algoritma *Select Rules*
1	*sumweights* = 0
2	**for** *i*=0 **to** *rulecount* – 1 **do**
3	*sumweights* = *sumweights* + *rule*[*i*].*weight*
4	**end for**
5	**for** *i*=0 **to** *size* – 1 **do**
6	*try*=0; *lineadded*=false
7	**while** *try* < *maxtries* **and not** *lineadded* **do**
8	*j*=0; *sum*=0; *selected*= -1
9	*fraction* = random(*sumweights*)
10	**while** *selected* < 0 **do**
11	*sum* = *sum* + *rule*[*j*].*weight*
12	**if** *sum* > *fraction* **then**
13	*selected* = *j*
14	**else**
15	*j* = *j* + 1
16	**end if**
17	**end while**
18	**for** *s*=0 **to** *rulecount* – 1 **do**
19	**if** *rule*[*selected*] = *rule*[*s*] **then**
20	*lineadded* = false
21	**break**
22	**else**
23	*lineadded* = true
24	**end if**
25	**end for**
26	**if** *lineadded* = true **then**
27	*selectedrules*.Add(*rule*[*selected*])
28	**end if**

```
29          try = try+1
30      end while
31  end for
32  AddDefaultRules()
33
34  return selectedrules
```

Gambar 5.2 Algortima *Select Rules*

Algoritma ini bertujuan untuk meng*generate* sejumlah '*size' rule* ke dalam *script*. Akan tetapi, algoritma ini memungkinkan terjadi proses generate *script* dengan jumlah yang lebih kecil dari '*size'* apabila di dalam sebuah proses seleksi dibutuhkan lebih dari '*maxtries'* percobaan untuk mendapatkan sebuah *rule*. Di dalam penelitian ini digunakan nilai '*maxtries'* sebesar 3.

5.3.2 Implementasi Algoritma *Do Battle*

Di dalam penelitian ini, *agent* AI diimplementasikan ke dalam sebuah pertempuran berjenis *turn-based strategy*. Dalam sebuah pertempuran, setiap karakter akan menunggu gilirannya untuk mengeksekusi sebuah *rule*. Sebuah *agent* AI akan menseleksi *rule* dari dalam *script* untuk kemudian dieksekusi.

Pada setiap proses seleksi *rule* di dalam sebuah pertempuran, *agent* AI akan mendapatkan informasi mengenai *skill* yang akan dieksekusi dan sejumlah target dari *skill* tersebut.

Algoritma *Do Battle*

```
1  if agent is ready then
2      agent.DoGambit()
3      agent.CastSkill(agent.PreparedSkill, agent.PreparedTarget)
4      agent.CheckStatusEffect()
5  else
6      agent.Wait()
```

7	**end if**

Gambar 5.3 Algortima *Do Battle*

'*DoGambit'* melakukan proses seleksi *rule* dari dalam *script* yang dimiliki oleh *agent* AI. Proses ini akan menghasilkan informasi mengenai *skill* yang akan dieksekusi oleh *agent* AI dan target dari *skill* tersebut.

'*CastSkill'* digunakan untuk mengeksekusi *skill* yang sebelumnya telah dipersiapkan di dalam proses '*DoGambit'*.

'*CheckStatusEffect'* digunakan untuk mengurangi nilai '*TurnCount'* dari semua *Status Effect* yang ada di dalam *agent* AI. Sebuah *Status Effect* akan dihilangkan dari sebuah karakter apabila '*TurnCount'* dari sebuah *Status Effect* tersebut bernilai 0.

5.3.3 Implementasi Algoritma *Get Battle Record*

Algoritma ini berfungsi untuk memperoleh data-data yang diperlukan dari sebuah pertempuran yang sebelumnya dilakukan oleh *agent* AI. Data-data yang diperoleh nantinya akan digunakan sebagai parameter di dalam algortima *Weight Update*.

Algoritma *Get Battle Record* memiliki tiga algoritma utama yaitu algoritma *Get Used Gambit*, *Get Win Party*, dan *Get Agent Fitness*.

5.3.3.1 Algoritma *Get Used Gambit*

Algoritma ini berfungsi untuk mendapatkan *script* yang dibangun dari sejumlah *macro action* dan *rule* yang tereksekusi di pertempuran sebelumnya dari sebuah *agent* AI. Algoritma ini juga berfungsi untuk memperoleh informasi mengenai jumlah *rule* dan *macro action* yang berhasil diaktifkan/dieksekusi dan yang tidak diaktifkan/dieksekusi di dalam pertempuran sebelumnya dari sebuah *agent* AI.

Algoritma *Get Used Gambit*

```
1   for i=0 to agent.Gambit.BasicRules.Count – 1 do
2       if agent.Gambit.BasicRule[i].executed then
3           executedGambit.BasicRule.Add(agent.Gambit.BasicRule[i])
4           activated += 1
5       else
6           nonActivated += 1
7       end if
8   end for
9   if agent.Gambit.Openingmacro is not null then
10      if agent.Gambit.Openingmacro. executed then
11          executedGambit.Openingmacro = agent.Gambit.Openingmacro
12          activated += 1
13      else
14          nonActivated += 1
15      end if
16  end if
17
18  if agent.Gambit.Migamemacro is not null then
19      if agent.Gambit. Migamemacro.executed then
20          executedGambit. Migamemacro = c.Gambit. Migamemacro
21          activated += 1
22      else
23          nonActivated += 1
24      end if
25  end if
26
27  return executedGambit
```

Gambar 5.4 Algortima *Get Used Gambit*

5.3.3.2 Algoritma *Get Win Party*

Algoritma ini digunakan untuk memperoleh ID dari kelompok/*party* yang menjadi pemenang di dalam suatu pertempuran.

Algoritma *Get Win Party*

```
1   winID = -1
2   HPSum = 0
3
4   for i = 0 to allyParty.agents.Count  – 1 do
5       HPSum += allyParty.agents[i].LastHP
6   end for
7
8   if HPSum > 0 then
9       winID = allyParty.ID
10  else
11      winID = enemyParty.ID
12  end if
```

Gambar 5.5 Algortima *Get Win Party*

5.3.3.3 Algoritma *Get Agent Fitness*

Algoritma ini berfungsi untuk mendapatkan nilai *fitness* dari sebuah *agent* AI. Formula yang untuk mendapatkan nilai *fitness* dari sebuah *agent* AI mengacu pada persamaan 2.

Algoritma *Get Agent Fitness* dibagi menjadi empat algoritma utama yaitu algoritma *Calculate Team Fitness, Calculate Agent Survival, Calculate Team Survival*, dan *Calculate Damage Inflicted.*

5.3.3.3.1 Algoritma *Calculate Team Fitness*

Algoritma *Calculate Team Fitness* berfungsi menghitung *fitness* tim dari *agent* AI.

Algoritma *Calculate Team Fitness*	
1	*teamfitness* = 0
2	**if** *allyParty*.*isWin* **then**
3	**for** *i*=0 **to** *agentcount* – 1 **do**
4	*teamfitness* = *teamfitness* + ((0.5 * *agentcount*) * (1 + (*agent*[*i*].*lasthealth*/*agent*[*i*].*firsthealth*)))
5	**end for**
6	**end if**
7	**return** *teamfitness*

Gambar 5.6 Algortima *Calculate Team Fitness*

Jika sebuah tim mengalami kekalahan (semua *agent* di dalam tim tersebut memiliki HP <= 0), maka nilai *fitness* dari tim tersebut bernilai 0.

5.3.3.3.2 Algoritma *Calculate Agent Survival* (A)

Algoritma ini berfungsi menghitung tingkat kemampuan bertahan dari sebuah *agent*.

Algoritma *Calculate Agent Survival*	
1	*a*=0
2	**if** *agent*.*lasthealth* < = 0 **then**
3	*a* = (1/3) * (Min((*agent*.*lassturn*/*dmax*) , 1)
4	**else**
5	*a* = (1/3) * (2 + (*agent*.*lasthealth* / *agent*.*firsthealth*))
6	**end if**
7	**return** *a*

Gambar 5.7 Algortima *Calculate Agent Survival*

'*dmax'* adalah sebuah nilai konstan. Dalam suatu experimen, Nilai '*dmax'* diinisialisasi sebesar 100, yang berarti sebanding dengan 10 babak pertempuran, yang berarti lebih lama dari lama pertempuran pada umumnya. [SPR-05: 16]

5.3.3.3.3 Algoritma *Calculate Team Survival* (B)

Algoritma ini berfungsi untuk menghitung tingkat kemampuan bertahan sebuah tim setelah melakukan sebuah pertempuran.

Algoritma *Calculate Team Survival*	
1	*b*=0
2	**for** *i*=0 **to** *agentcount* – 1 **do**
3	**if** *agent*[*i*].*lasthealth* > 0 **then**
4	*b* = *b* + (0.5 * *agentcount* * (1 + (*agent*[*i*].*lasthealth* / *agent*[*i*].*firsthealth*)))
5	**end if**
6	**end for**
7	**return** *b*

Gambar 5.8 Algortima *Calculate Team Survival*

5.3.3.3.4 Algoritma *Calculate Damage Inflicted* (C)

Algoritma ini berfungsi menghitung seberapa besar *damage* yang sudah diberikan kepada tim lawan selama pertempuran.

Algoritma *Calculate Damage Inflicted*	
1	*c*=0
2	**for** *i*=0 **to** *agentcount* – 1 **do**
3	**if** *agent*[*i*].*lasthealth* < = 0 **then**
4	*c* = *c* + 1
5	**else**
6	*c* = *c* + (0.5 * *agentcount* *

```
        ( 1 – (agent[i].lasthealth / agent[i].firsthealth )))
7       end if
8   end for
9   return c
```

Gambar 5.9 Algortima *Calculate Damage Inflicted*

'*agent'* yang dimaksud adalah *agent* yang berasal dari tim lawan (tim yang berlawanan dengan tim dari algoritma *Calculate Team Survival*).

5.3.4 Implementasi Algoritma *Weight Update*

Algoritma ini berfungsi memberikan update nilai bobot (*Weight*) di tiap *rule* dan *macro action* yang digunakan di dalam pertempuran yang sebelumnya dilakukan oleh *agent* AI.

Algoritma *Weight Update*

```
    {R bisa berarti rulebase, Opening Macro Base, maupun Midgame Macro Base }

1   adjustment = CalculateAdjustment(Fitness)
2   compensation = – active * adjustment / nonactive
3   remainder = 0
4   for i=0 to R .Count– 1 do
5       if R[i].executed then
6           R[i].weight = R[i].weight + adjustment
7       else
8           R[i].weight = R[i].weight + compensation
9       end if
10      if R[i].weight < minweight then
11          remainder = remainder + (R[i].weight – minweight)
12          R[i].weight = minweight
13      else if R [i].weight > maxweight then
14          remainder = remainder + (R[i].weight – maxweight)
```

15	*R*[*i*].*weight* = *maxweight*
16	**end if**
17	**end for**
18	DistributeRemainder(*R*)

Gambar 5.10 Algortima *Weight Update*

5.3.4.1 Implementasi Algoritma *Calculate Adjustment*

Algoritma *Calculate Adjustment* ini berfungsi untuk memperoleh nilai *Adjustment* (ΔW) yang digunakan untuk mengupdate bobot dari *rule-rule* dan *macro action* yang telah di*generate* di dalam *script* dan digunakan untuk mengontrol *agent* AI di dalam pertempuran sebelumnya.

Algoritma *Calculate Adjustment*	
1	Δ*W*=0
2	
3	**if** *agentfitness* < *breakevenpoint* **then**
4	Δ*W* = – (*maxpenalty* * (*breakevenpoint-agentfitness*) / *breakevenpoint*)
5	**else**
6	Δ*W* = *maxreward* * (*agentfitness* – *breakeventpoint*) / (1 – *breakevenpoint*)
7	**end if**

Gambar 5.11 Algortima *Calculate Adjustment*

Pada saat *fitness* dari *agent* berada pada *break even point*, bobot dari sebuah *rule* atau *macro action* tidak berubah sama sekali.

5.3.4.2 Implementasi Algoritma *Distribute Remainder*

Algoritma ini berfungsi untuk menambahkan selisih bobot dari sebuah *rule* atau *macro action* yang melampaui batas ke semua *rule* atau *macro action*.

Algoritma *Distribute Remainder*	
	{*R* bisa berarti *Rulebase*, *Opening Macro Base*, maupun *Midgame Macro Base*}

```
1  fraction = remainder / R.Count
2
3  for i=0 to R.count – 1 do
4      sum = rule[i].weight + fraction
5      if sum > minweight and sum < maxweight then
6          R[i].weight = sum
7          remainder = remainder – fraction
8      end if
9  end for
```

Gambar 5.12 Algortima *Distribute Remainder*

5.3.5 Implementasi Algoritma *Difficulty Scaling*

Di dalam penelitian ini akan dibandingkan tiga metode pengaturan tingkat kesulitan yaitu *high-fitness penalising*, *weight clipping*, dan *top culling*.

5.3.5.1 Implementasi Algoritma *High-Fitness Penalising*

Teknik ini dimaksudkan untuk memberikan *penalty* ke setiap *rule* atau *macro action* yang memiliki nilai *fitness* terlalu tinggi.

Algoritma *High-Fitness Penalising*

```
1  InitP(pinit, pmin, pmax, pdec, pinc)
2  for each battle do
3      if allyParty.isWin then
4          p = Max((p – pdec), pmin)
5      else
6          p = Min((p + pinc), pmax)
7      end if
8      if fitness < = p then
9          fitness = fitness / p
10     else
```

```
11          fitness = (1-fitness) / p
12      end if
13  end for
```

Gambar 5.13 Algortima *High-Fitness Penalising*

Pada awalnya, nilai '*p*' diinisialisasi sebesar '*pinit*'. Untuk setiap pertempuran yang dimenangkan oleh tim *agent*, nilai '*p*'berkurang sebesar '*pdec*' dengan batas minimum '*pmin*'. Dan untuk setiap pertempuran yang dimenangkan oleh tim yang menjadi lawan *agent*, nilai '*p*' bertambah sebesar '*pinc*' dengan batas maksimum '*pmax*'.

5.3.5.2 Implementasi Algoritma *Weight Clipping*

Algoritma di dalam metode *weight clipping* akan membatasi nilai bobot maksimum dan nilai bobot minimum sebagai parameter yang dibutuhkan di dalam algoritma *Weight Update*.

Algoritma *Weight Clipping*

```
1   InitW(winit, wmin, wmax, wdec, winc)

2   for each battle do
3       if allyParty.isWin then
4           wmax = wmax –  (wmax * wdec / 100)
5           if wmax < winit then
6               wmax = winit
7           end if
8       else
9           wmax = wmax + (wmax * winc / 100)
10      end if
11  end for
```

Gambar 5.14 Algortima *Weight Clipping*

Algoritma ini dieksekusi sebelum pemanggilan algoritma *Weight Update*. Untuk setiap pertempuran yang dimenangkan oleh tim *agent*, nilai '*wmax*' berkurang sebesar '*wdec*' dengan batas minimum '*winit*. Dan untuk setiap pertempuran yang dimenangkan oleh tim yang menjadi lawan *agent*, nilai '*wmax*' bertambah sebesar '*winc*'.

Nilai '*wmin*' dan '*wmax*' di dalam algoritma *weight clipping* ini nantinya akan dijadikan sebagai parameter di dalam algoritma *Weight Update*.

5.3.5.3 Implementasi Algoritma *Top Culling*

Algoritma *Top Culling* menjelaskan proses seleksi *rule* atau *macro action* yang dieksekusi sebelum pemanggilan algoritma *Script Generation*. Sedangkan proses *update* parameter yang digunakan sama dengan algoritma *Weight Clipping*.

	Algoritma *Top Culling*
	{*R* bisa berarti *rulebase, Opening Macro Base,* maupun *Midgame Macro Base* }
	{*result* bisa berarti *rulebase, Opening Macro Base,* maupun *Midgame Macro Base* }
1	**for** *i*=0 **to** *R.count* – 1 **do**
2	**if** *R*[*i*].*Weight* < *wmax* **then**
3	*result.*Add(*R*[*i*])
4	**end if**
5	**end for**
6	
7	**return** *result*

Gambar 5.15 Algortima *Top Culling*

5.4 Implementasi *Skill*

Skill merepresentasikan *action* dari sebuah karakter. Sebuah karakter hanya bisa mengeksekusi satu buah *skill* pada tiap gilirannya. Penjelasan mengenai *skill* yang dibuat bisa dilihat di Tabel 5.3. Tabel 5.4 menjelaskan *Skill*

Effect yang dimiliki oleh beberapa *skill*. Tabel 5.5 menjelaskan *Status Effect* yang dimiliki oleh beberapa *skill*.

Tabel 5.3 Implementasi *Skill*

ID	***Name***	***MP Cons.***	***Hit Rate***	***Type***	***Target***	***Target Count***
0	Attack	0	90	PHYSICAL	ENEMY	SINGLE
1	Defend	0	100	PHYSICAL	SELF	SINGLE
2	Cure	20	100	MAGICAL	ALLY	SINGLE
3	Cura	50	100	MAGICAL	ALLY	PARTY
4	Curaga	50	100	MAGICAL	ALLY	SINGLE
5	Fire	20	90	MAGICAL	ENEMY	SINGLE
6	Fira	50	80	MAGICAL	ENEMY	PARTY
7	Firaga	50	70	MAGICAL	ENEMY	SINGLE
8	Silence	35	60	MAGICAL	ENEMY	SINGLE
9	Poison	35	70	MAGICAL	ENEMY	SINGLE
10	Drain	35	80	MAGICAL	ENEMY	SINGLE
11	Protect	30	100	MAGICAL	ENEMY	SINGLE
12	Shell	30	100	MAGICAL	ENEMY	SINGLE
13	Weakness	35	70	MAGICAL	ENEMY	SINGLE
14	Poisona	30	100	MAGICAL	ALLY	SINGLE
15	Silenca	30	100	MAGICAL	ALLY	SINGLE
16	Dispel	50	100	MAGICAL	ALLY	SINGLE
17	Haste	35	100	MAGICAL	ALLY	SINGLE
18	Slow	40	80	MAGICAL	ALLY	SINGLE
19	Faith	50	100	MAGICAL	ALLY	SINGLE
20	Brave	50	100	MAGICAL	ALLY	SINGLE

Tabel 5.4 Implementasi *Skill Effect*

Skill ID	***Skill Name***	***Skill Effects***			
		Target Stat	***Target Cast***	***Main Mod.***	***Secondary Mod.***
0	Attack	HP	TARGET	-1	0
1	Defend	DEF	CASTER	0	40
		MP	CASTER	0	25
2	Cure	HP	TARGET	2	0
3	Cura	HP	TARGET	2	0
4	Curaga	HP	TARGET	4	0
5	Fire	HP	TARGET	-2	0
6	Fira	HP	TARGET	-2	0
7	Firaga	HP	TARGET	-4	0
10	Drain	HP	TARGET	-2	0
		HP	CASTER	1	0

Tabel 5.5 Implementasi *Status Effect*

Skill ID	***Skill Name***	***Status Effects***		
		Effect	***Effect Type***	***Turn Count***
8	Silence	SILENT	CURSER	5
9	Poison	POISON	CURSER	5
11	Protect	PROTECT	CURSER	5
12	Shell	SHELL	CURSER	5
13	Weakness	WEAK	CURSER	5
14	Poisona	POISON	CLEANSER	1
15	Silenca	SILENCE	CLEANSER	1
16	Dispel	PROTECT	CLEANSER	1
		SHELL	CLEANSER	1
		HASTE	CLEANSER	1
		SLOW	CLEANSER	1

		WEAK	CLEANSER	1
17	Haste	HASTE	CURSER	5
18	Slow	SLOW	CURSER	5
19	Faith	FAITH	CURSER	5
20	Brave	BRAVE	CURSER	5

5.5 Implementasi Karakter

Sebuah karakter merupakan sebuah objek yang mempunyai *agent* AI, baik *agent* AI statis maupun dinamis. Karakter di dalam sebuah game mempunyai tipe yang bermacam-macam, seperti; *Knight*, *Mage*, *Hunter*, dan lain sebagainya.

Di dalam penelitian ini sendiri akan mengimplementasikan karakter yang bertipe *Mage*. Alasan dipilihnya karakter ini adalah karena karakter dengan tipe *Mage* cenderung memiliki banyak sekali pilihan *action* atau *skill* dibandingkan dengan karakter tipe lain. Tabel 5.6 menjelaskan stat dan parameter yang akan digunakan untuk membangun sebuah karakter dengan tipe *Mage*.

Tabel 5.6 Implementasi Karakter

Level	1
Name	Mage
Health Point (HP)	100
Magic Poinr (MP)	150
Strength (STR)	10
Defense (DEF)	25
Magic (MAG)	40
Wisdom (WIS)	40
Dexternity (DEX)	25
Agility (AGI)	25
Skills	Attack
	Defend
	Cure

	Cura
	Curaga
	Fire
	Fira
	Firaga
	Silence
	Poison
	Drain
	Protect
	Shell
	Weakness
	Poisona
	Silenca
	Dispel
	Haste
	Slow
	Faith
	Brave

5.6 Implementasi *Script*

Beberapa *script* yang merepresentasikan sebuah taktik telah dirancang untuk mengontrol *agent* AI statis. Di dalam penelitian ini telah dirancang tiga jenis *script* yaitu; '*Assault'*, '*Defensive'*, dan '*Cunning'*. *Scipt* '*Assault'* cenderung melakukan serangan-serangan frontal dengan hanya sedikit memperhatikan masalah pertahanan tim, sedangkan *script* '*Defensive'* cenderung mementingkan pertahanan tim dan lebih berhati-hati dalam masalah penyerangan, dan *script* '*Cunning'* cenderung memakai taktik yang licik untuk melumpuhkan lawan dengan serangan-serangan yang kebanyakan akan menyebabkan lawan terjangkit *Status Effect* yang berdampak negatif.

Tabel 5.7 menjelaskan implementasi *script* '*Assault'*. Tabel 5.8 menjelaskan implementasi *script* '*Defensive'*. Tabel 5.9 menjelaskan implementasi *script* '*Cunning'*.

Rulebase yang dipakai di dalam *agent* AI dinamis ditunjukkan dalam Tabel 5.10. Sejumlah *macro action* yang membangun *Opening Macro Base* untuk *agent* AI dinamis dijelaskan di dalam tabel 5.11. Tabel 5.12 menjelaskan tentang sejumlah *macro action* yang membangun *Midgame Macro Base* untuk *agent* AI dinamis.

Tabel 5.7 *Assault Script*

Assault Script	
ID	**Rule**
0	`if Ally HP < 15% then Cast(Curaga)`
1	`if Enemy has smallest HP then Cast(Firaga)`
2	`if Enemy has smallest HP then Cast(Fira)`
3	`if Enemy has smallest HP then Cast(Fire)`
4	`Cast(Defend)`

Tabel 5.8 *Defensive Script*

Defensive Script	
ID	**Rule**
0	`if Ally HP < 20% then Cast(Curaga)`
1	`if Ally HP < 60% then Cast(Cure)`
2	`if Ally Status is not Shell then Cast(Shell)`
3	`if Enemy has smallest HP then Cast(Fire)`
4	`Cast(Defend)`

Tabel 5.9 *Cunning Script*

Cunning Script	
ID	**Rule**
0	`if Enemy Status is not Silent then Cast(Silence)`
1	`if Enemy Status is not Slow then Cast(Slow)`
2	`if Enemy Status is not Poison then Cast(Poison)`
3	`if Enemy has smallest HP then Cast(Drain)`
4	`Cast(Defend)`

Tabel 5.10 *Rulebase*

Rulebase	
ID	**Rule**
0	if Ally HP < 20% then Cast(Curaga)
1	if Ally HP < 60% then Cast(Cure)
2	if Ally Status is Silent then Cast(Silenca)
3	if Ally Status is Poison then Cast(Poisona)
4	if Ally Status is Slow then Cast(Dispel)
5	if Ally Status is Weak then Cast(Dispel)
6	if Enemy Status is not Silent then Cast(Silence)
7	if Enemy Status is not Poison then Cast(Poison)
8	if Enemy Status is not Slow then Cast(Slow)
9	if Ally Status is not Faith then Cast(Faith)
10	if Ally Status is not Haste then Cast(Haste)
11	if Ally Status is not Shell then Cast(Shell)
12	if Enemy Status is not Weak then Cast(Weakness)
13	if Enemy has smallest HP then Cast(Firaga)
14	if Enemy has smallest HP then Cast(Fira)
15	if Enemy has smallest HP then Cast(Fire)
16	if Enemy has smallest HP then Cast(Drain)
17	Cast(Defend)

Tabel 5.11 *Opening Macro Base*

Opening Macro Base	
ID	***Rule***
0	if Ally HP < 20% then Cast(Curaga) if Ally Status is not Faith then Cast(Faith) if Enemy has smallest HP then Cast(Firaga)
1	if Ally HP < 20% then Cast(Curaga) if Enemy has smallest HP then Cast(Firaga) if Ally Status is Silent then Cast(Silenca)
2	if Ally HP < 60% then Cast(Cure) if Enemy Status is not Silent then Cast(Silence) if Ally Status is Silent then Cast(Silenca)
3	if Ally Status is not Haste then Cast(Haste) if Ally Status is not Faith then Cast(Faith) if Enemy has smallest HP then Cast(Firaga)
4	if Enemy has smallest HP then Cast(Fira) if Ally Status is Silent then Cast(Silenca) if Ally Status is Weak then Cast(Dispel)

5	if Enemy Status is not Silent then Cast(Silence) if Ally HP < 20% then Cast(Curaga) if Enemy has smallest HP then Cast(Firaga)
6	if Enemy Status is not Silent then Cast(Silence) if Ally HP < 60% then Cast(Cure) if Ally Status is Silent then Cast(Silenca)
7	if Enemy Status is not Silent then Cast(Silence) if Ally Status is Silent then Cast(Silenca) if Enemy has smallest HP then Cast(Drain)
8	if Enemy Status is not Silent then Cast(Silence) if Enemy has smallest HP then Cast(Fire) if Ally Status is not Haste then Cast(Haste)
9	if Ally Status is Silent then Cast(Silenca) if Enemy Status is not Silent then Cast(Silence) if Ally HP < 20% then Cast(Curaga)
10	if Ally Status is Silent then Cast(Silenca) if Ally HP < 20% then Cast(Curaga) if Enemy has smallest HP then Cast(Firaga)
11	if Ally Status is Silent then Cast(Silenca) if Ally HP < 20% then Cast(Curaga) if Enemy Status is not Silent then Cast(Silence)
12	if Enemy has smallest HP then Cast(Firaga) if Enemy Status is not Poison then Cast(Poison) if Enemy has smallest HP then Cast(Drain)
13	if Ally Status is not Faith then Cast(Faith) if Ally Status is Silent then Cast(Silenca) if Ally Status is not Haste then Cast(Haste)
14	if Ally Status is not Faith then Cast(Faith) if Ally HP < 20% then Cast(Curaga) if Ally Status is Silent then Cast(Silenca)
15	if Enemy Status is not Silent then Cast(Silence) if Enemy Status is not Poison then Cast(Poison) if Enemy Status is not Slow then Cast(Slow)
16	if Enemy has smallest HP then Cast(Fira) if Enemy Status is not Poison then Cast(Poison) if Ally Status is not Faith then Cast(Faith)
17	if Ally Status is Silent then Cast(Silenca) if Enemy Status is not Poison then Cast(Poison) if Enemy has smallest HP then Cast(Fira)
18	if Enemy has smallest HP then Cast(Fira) if Ally Status is Silent then Cast(Silenca) if Enemy has smallest HP then Cast(Fire)
19	if Enemy has smallest HP then Cast(Firaga) if Ally HP < 20% then Cast(Curaga) if Ally Status is Poison then Cast(Poisona)

Tabel 5.12 *Midgame Macro Base*

Midgame Macro Base	
ID	***Rule***
0	if Enemy has smallest HP then Cast(Firaga) if Enemy Status is not Poison then Cast(Poison) if Enemy has smallest HP then Cast(Fire)
1	if Enemy has smallest HP then Cast(Firaga) if Enemy Status is not Weak then Cast(Weakness) if Enemy has smallest HP then Cast(Drain)
2	if Enemy has smallest HP then Cast(Firaga) if Enemy has smallest HP then Cast(Fire) if Ally HP < 60% then Cast(Cure)
3	if Enemy has smallest HP then Cast(Fira) if Enemy Status is not Silent then Cast(Silence) if Enemy Status is not Poison then Cast(Poison)
4	if Enemy has smallest HP then Cast(Drain) if Enemy Status is not Poison then Cast(Poison) if Enemy Status is not Weak then Cast(Weakness)
5	if Enemy Status is not Silent then Cast(Silence) if Enemy has smallest HP then Cast(Drain) if Enemy has smallest HP then Cast(Fire)
6	if Enemy Status is not Poison then Cast(Poison) if Enemy has smallest HP then Cast(Fire) if Ally HP < 60% then Cast(Cure)
7	if Ally Status is not Shell then Cast(Shell) if Enemy has smallest HP then Cast(Drain) if Enemy has smallest HP then Cast(Fire)
8	if Ally Status is Silent then Cast(Silenca) if Ally Status is not Haste then Cast(Haste) if Enemy has smallest HP then Cast(Fire)
9	if Ally Status is Silent then Cast(Silenca) if Enemy has smallest HP then Cast(Fire) if Enemy Status is not Weak then Cast(Weakness)
10	if Ally Status is not Haste then Cast(Haste) if Enemy Status is not Slow then Cast(Slow) if Enemy has smallest HP then Cast(Fire)
11	if Ally HP < 60% then Cast(Cure) if Enemy has smallest HP then Cast(Firaga) if Enemy Status is not Weak then Cast(Weakness)
12	if Ally HP < 60% then Cast(Cure) if Ally Status is Silent then Cast(Silenca) if Enemy has smallest HP then Cast(Fire)
13	if Ally Status is not Faith then Cast(Faith) if Enemy Status is not Slow then Cast(Slow) if Enemy has smallest HP then Cast(Fire)
14	if Enemy Status is not Weak then Cast(Weakness) if Ally Status is Silent then Cast(Silenca) if Enemy Status is not Slow then Cast(Slow)

15	if Ally HP < 60% then Cast(Cure) if Ally Status is not Faith then Cast(Faith) if Enemy Status is not Silent then Cast(Silence)
16	if Enemy has smallest HP then Cast(Fira) if Ally Status is not Faith then Cast(Faith) if Enemy Status is not Poison then Cast(Poison)
17	if Ally HP < 60% then Cast(Cure) if Enemy has smallest HP then Cast(Fira) if Ally Status is not Faith then Cast(Faith)
18	if Enemy has smallest HP then Cast(Fire) if Ally Status is Poison then Cast(Poisona) if Ally Status is Silent then Cast(Silenca)
19	if Enemy has smallest HP then Cast(Firaga) if Ally HP < 20% then Cast(Curaga) if Ally Status is Poison then Cast(Poisona)

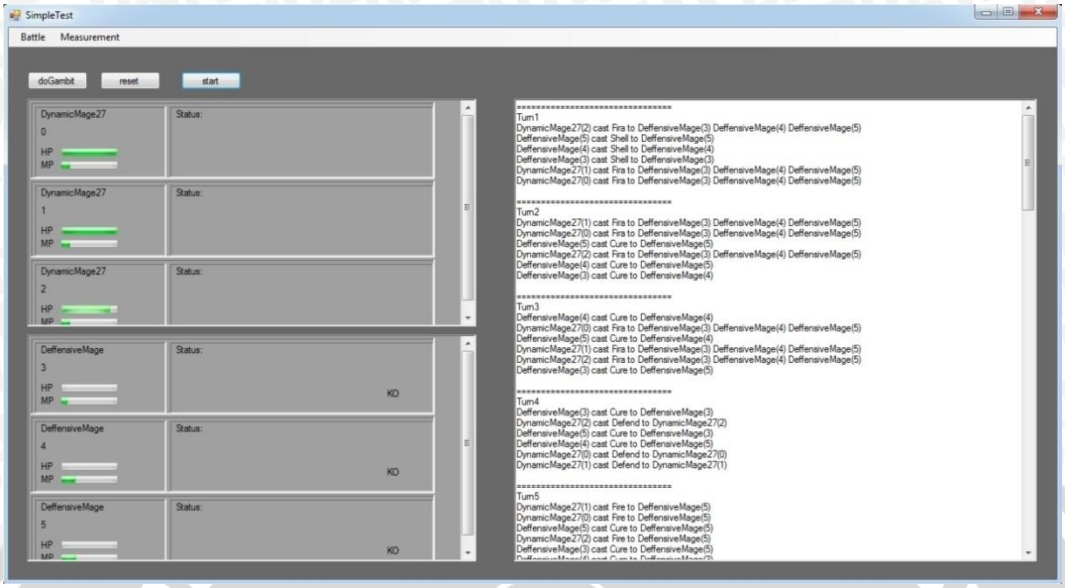
5.7 Implementasi *Agent* AI ke Dalam *Fighting System* Game

Fighting system yang akan dirancang berupa *turn-based strategy* yang di dalamnya terdapat 2 tim yang bertempur. Setiap tim terdiri beberapa karakter dengan jumlah maksimal 3 karakter.

Setiap karakter dalam *fighting system* ini bergerak dan melakukan *action* sesuai dengan *action* atau *skill* yang dipilih. Suatu karakter bisa memilih *action*nya ketika *action time* dari karakter tersebut penuh. Tiap *action* atau *skill* juga memiliki syarat minimum untuk dapat dieksekusi suatu karakter. Suatu karakter dianggap menyelesaikan gilirannya ketika selesai mengeksekusi sebuah *action* atau *skill*.

Suatu tim dinyatakan kalah jika seluruh karakter dari tim tersebut *KO* atau HP (*Health Point*) dari seluruh karakter di dalam tim tersebut mencapai angka 0. Tim yang bisa bertahan sampai akhir pertempuran dinyatakan menang.

Gambar 5.16 menunjukkan implementasi *agent* AI ke dalam sebuah *fighting system* untuk keperluan simulasi sederhana. Gambar 5.17 menunjukkan implementasi *agent* AI ke dalam *fighting system* di sebuah game yang dirancang menggunakan Unity 3D Game Engine untuk simulasi yang lebih kompleks.



Gambar 5.16 *Simpletest*

Gambar 5.17 *Gametest*

BAB VI
PENGUJIAN

Bab ini membahas pengujian dan analisis *agent* AI menggunakan *dynamic scripting* dan *macro action*. Teknik dan parameter pengujian mengacu pada pengujian yang dilakukan pada penelitian sebelumnya yaitu *turning point*, efektifitas, dan keberagaman.

6.1 Pengujian

Pengujian dilakukan dengan mempertemukan satu tim yang terdiri dari tiga *agent* AI dinamis dengan satu tim yang terdiri dari tiga *agent* AI statis ke dalam sejumlah pertempuran. *Agent* yang merepresentasikan *script* '*Assault*' adalah *agent Assault*. *Agent Defensive* merepresentasikan *script* '*Defensive*'. *Agent Cunning* merepresentasikan *script* '*Cunning*'. Pengujian juga dilakukan dengan mempertemukan *agent* AI dinamis melawan *Random Team*, yaitu sebuah tim yang dipilih secara acak yang kemungkinan terdiri dari tiga *agent Assault*, atau tiga *agent Defensive*, atau tiga *agent Cunning*. Selain itu, pengujian juga dilakukan dengan mempertemukan *agent* AI dinamis melawan *Random Agent*, yaitu sebuah tim yang mungkin terdiri dari *agent Assault*, *agent Defensive*, *agent Cunning* yang dipilih secara acak.

Pegujian algoritma di dalam penelitian ini didapatkan dengan cara melakukan pertempuran sebanyak 250 X 30 pertempuran untuk tiap jenis *script* yang menjadi lawan *agent* AI dinamis. Untuk tiap 250 pertempuran, bobot dari *rule* dan *macro action agent* AI dinamis akan di*reset* ulang ke nilai awalnya sebesar 100. Nilai minimum bobot yang dipakai adalah 0, dan nilai maksimum bobot yang dipakai sebesar 200.

6.1.1 Parameter

Parameter-parameter yang diujikan di dalam penilitan ini mengacu kepada parameter-parameter yang diujikan di dalam penelitian yang sebelumnya. Adapun parameter-parameter tersebut adalah *turning point*, efektifitas, dan keberagaman.

6.1.1.1 *Turning Point*

Turning point dapat digunakan untuk mendapatkan ukuran mengenai berapa banyak pertempuran yang dibutuhkan *agent* adaptif dalam memproduksi *script* yang baik secara konsisten. *Turning point* didapatkan dengan cara melakukan *recording* terhadap 10 pertempuran terakhir dari jumlah total 250 pertempuran. Jika fitness dari *agent* adaptif lebih besar dari pada *agent* statis di 10 pertempuran terakhir, maka dapat dikatakan bahwa *turning point* sudah tercapai. Semakin kecil nilai *turning point* mengindikasikan bahwa *agent* dinamis memerlukan waktu yang lebih cepat dalam beradaptasi dan memproduksi *script* yang baik.

Nilai *turning point = '>241'* mengindikasikan bahwa *agent* dinamis memerlukan jumlah pertempuran di atas 241 pertempuran untuk mencapai *turning point*.

6.1.1.2 Efektifitas

Ukuran mengenai ketangguhan *agent* adaptif dapat dilakukan dengan cara menghitung banyaknya pertempuran yang berhasil dimenangkan oleh *agent* adaptif di 100 pertempuran terakhir dari total 250 pertempuran.

6.1.1.3 Keberagaman

Untuk menghitung tingkat keberagaman digunakan formula di dalam Persamaan (3-2). Semakin tinggi tingkat keberagaman berarti *agent* AI adaptif lebih menjanjikan taktik yang beragam di tiap pertempuran.

6.1.2 Hasil Pengujian

Di dalam penelitian ini dilakukan pengujian kepada beberapa jenis *agent* AI dinamis. Pengujian dilakukan melalui beberapa tahap/skenario pengujian. Tahap pertama dalam proses ini adalah pengujian metode *dynamic scripting* yang digunakan sebagai pembanding di dalam tahap selanjutnya. Tahap ke dua adalah pengujian metode *dynamic scripting* menggunakan *macro action*. Dan tahap yang ke tiga adalah pengujian metode *difficulty scaling* sebagai pengatur tingkat

kesulitan *agent*. Pengujian *difficulty scaling* sendiri dibagi menjadi tiga bagian yaitu pengujian metode *high-fitness penalising*, pengujian metode *weight clipping*, dan pengujian metode *top culling*.

Skenario I menunjukkan pengujian tahap pertama yaitu metode *dynamic scripting*. Pengujian tahap ke dua, yaitu metode *dynamic scripting* menggunakan *macro action*, ditunjukkan dalam Skenario II.

Pengujian metode *difficulty scaling* ditunjukkan di dalam Skenario III, Skenario IV, dan Skenario V. Skenario III menjelaskan pengujian metode *high-fitness penalising*. Pengujian metode *weight clipping* dijelaskan di dalam Skenario IV. Skenario V menjelaskan pengujian metode *top culling*.

6.1.2.1 Skenario I

Agent yang merepresentasikan metode ini tidak memiliki *macro action* base, baik *Opening Macro Base* maupun *Midgame Macro Base*. Jumlah *rule* yang digenerate adalah 9 *rule*.

Pada skenario ini akan dilakukan penambahan nilai parameter '*MaxPenalty*' untuk mendapatkan hasil yang terbaik dari metode *dynamic scipting*. Tabel 6.1 menjelaskan nilai dari parameter-parameter yang akan diuji di dalam Skenario I.

Tabel 6.1 Parameter Pengujian Metode *Dynamic Scripting*

Nama *Agent*	Parameter		
	break-event point	*MaxReward*	*MaxPenalty*
DS1	0.2	10	5
DS2	0.2	10	8

Hasil pengujian *agent* DS1 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.2, Tabel 6.3, dan Tabel 6.4. Hasil pengujian *agent* DS2 melawan sejumlah *agent* statis ditunjukkan oleh Tabel 6.5, Tabel 6.6, dan Tabel 6.7.

Tabel 6.2 Hasil Pengujian *Turning Point Agent* DS1

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	43.333 %	167	69.092	133
Defensive	6.667 %	64.210	38.73	67
Cunning	0 %	14	5.01	11
Random Team	0 %	33.1	19.18	35
Random Agent	0 %	41.967	24.958	40

Tabel 6.3 Hasil Pengujian Keberagaman *Agent* DS1

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.771	0.020	0.781
Defensive	0.771	0.003	0.771
Cunning	0.766	0.003	0.766
Random Team	0.77	0.003	0.769
Random Agent	0.773	0.004	0.773

Tabel 6.4 Hasil Pengujian Efektifitas *Agent* DS1

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	27.833	24.421	43
Defensive	35.533	3.421	34
Cunning	49.867	3.919	50
Random Team	38.267	4.494	39
Random Agent	38.433	5.042	39

Tabel 6.5 Hasil Pengujian *Turning Point Agent* DS2

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	76.667 %	197.367	80.689	>241
Defensive	0 %	39.5	14.654	38
Cunning	0 %	12.467	4.199	10
Random Team	0 %	35.867	15.865	36
Random Agent	0 %	33.133	13.053	37

Tabel 6.6 Hasil Pengujian Keberagaman *Agent* DS2

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.658	0.041	0.650
Defensive	0.773	0.002	0.774
Cunning	0.768	0.003	0.767
Random Team	0.775	0.003	0.776
Random Agent	0.776	0.003	0.776

Tabel 6.7 Hasil Pengujian Efektifitas *Agent* DS2

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	3.333	12.568	0
Defensive	44.133	4.232	44
Cunning	52.367	4.222	52
Random Team	44.767	4.224	46
Random Agent	44.733	4.898	46

6.1.2.2 Skenario II

Agent yang merepresentasikan metode ini mempunyai sejumlah *macro action* untuk digenerate ke dalam *script* di tiap pertempuran yang dilakukan. Sebuah *macro action* terdiri dari tiga *rule*. *Agent* ini hanya mengenerate satu buah *rule* setelah *Midgame Macro* berhasil digenerate.

Pada skenario ini akan dilakukan penambahan nilai parameter '*MaxPenalty*' untuk mendapatkan hasil yang terbaik dari metode *dynamic scipting* dan *macro action*. Tabel 6.8 menjelaskan nilai dari parameter-parameter yang akan diuji di dalam Skenario II.

Tabel 6.8 Parameter Pengujian Metode *Dynamic Scripting* dan *Macro Action*

Nama *Agent*	Parameter		
	break-event point	*MaxReward*	*MaxPenalty*
DSMA1	0.2	10	5
DSMA2	0.2	10	8

Hasil pengujian parameter *turning point agent* DSMA1 melawan sejumlah *agent* statis ditunjukkan oleh Tabel 6.9. Hasil pengujian parameter keberagaman *agent* DSMA1 melawan sejumlah *agent* statis ditunjukkan oleh Tabel 6.10. Tabel 6.11 menunjukkan hasil pengujian parameter efektifitas *agent* DSMA1 melawan sejumlah *agent* statis. Hasil pengujian *agent* DSMA2 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.12, Tabel 6.13, dan Tabel 6.14.

Tabel 6.9 Hasil Pengujian *Turning Point Agent* DSMA1

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	23.333 %	201.633	30.848	183
Defensive	0 %	49.333	26.418	56
Cunning	0 %	10.433	1.695	10
Random Team	0 %	32.333	19.023	28
Random Agent	0 %	55	35.372	48

Tabel 6.10 Hasil Pengujian Keberagaman *Agent* DSMA1

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.823	0.013	0.829
Defensive	0.864	0.044	0.836
Cunning	0.825	0.022	0.828
Random Team	0.861	0.044	0.833
Random Agent	0.837	0.021	0.833

Tabel 6.11 Hasil Pengujian Efektifitas *Agent* DSMA1

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	32.733	11.209	35
Defensive	50.167	5.96	51
Cunning	71.667	4.795	73
Random Team	45.4	3.838	46
Random Agent	42.5	4.804	43

Tabel 6.12 Hasil Pengujian *Turning Point Agent* DSMA2

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	149	44.091	161
Defensive	0 %	14.933	7.129	11
Cunning	0 %	10	0	10
Random Team	0 %	14.333	7.612	12
Random Agent	0 %	12.333	3.477	10

Tabel 6.13 Hasil Pengujian Keberagaman *Agent* DSMA2

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.859	0.005	0.859
Defensive	0.999	0.017	1.000
Cunning	0.941	0.012	0.940
Random Team	0.978	0.045	1.000
Random Agent	0.955	0.048	0.965

Tabel 6.14 Hasil Pengujian Efektifitas *Agent* DSMA2

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	49.033	6.682	49
Defensive	62.533	3.971	64
Cunning	84	3.342	84
Random Team	58.167	4.152	58
Random Agent	59.333	5.967	59

6.1.3 Pengujian *Difficulty Scaling*

Metode ini diperlukan untuk mengatur tingkat kesulitan *agent*. Pengaturan tersebut diharapkan bisa memperkecil kemungkinan sebuah *agent* untuk meng*generate* sebuah *script* yang mendekati sempurna (tingkat keefektifan yang terlalu tinggi).

Pengujian yang dilakukan menggunakan *agent* AI yang merepresentasikan metode *dynamic scripting* menggunakan *macro action* dengan nliai parameter *break-even point* sebesar 0.2, *MaxReward* sebesar 10, dan *MaxPenalty* sebesar 8.

6.1.3.1 Skenario III

Skenario ini melakukan pengujian metode *high-fitness penalising*. Metode ini melakukan *penalty* terhadap nilai *fitness* yang terlalu besar. Pada skenario pengujian ini akan dilakukan penambahan nilai parameter '*pdec*' untuk mendapatkan nilai yang optimal untuk pengaturan tingkat kesulitan *agent*. Tabel 6.15 mejelaskan nilai dari parameter-parameter yang akan diuji di dalam skenario III.

Tabel 6.15 Parameter Pengujian Metode *High-Fitness Penalising*

Nama *Agent*	Parameter				
	pinit	*pmin*	*pmax*	*pinc*	*pdec*
HFP1	0.45	0.4	0.5	0.01	0.01
HFP2	0.45	0.4	0.5	0.01	0.05
HFP3	0.45	0.4	0.5	0.01	0.1

Hasil pengujian *agent* HFP1 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.16, Tabel 6.17, dan Tabel 6.18. Hasil pengujian *agent* HFP2 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.19, Tabel 6.20, dan Tabel 6.21. Hasil pengujian *agent* HFP3 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.22, Tabel 6.23, dan Tabel 6.24.

Tabel 6.16 Hasil Pengujian *Turning Point Agent* HFP1

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	126.5	64.146	146
Defensive	0 %	16.433	9.324	13
Cunning	0 %	10.1	0.547	10
Random Team	0 %	15.933	11.785	10

Random Agent	0 %	15.467	5.643	14

Tabel 6.17 Hasil Pengujian Keberagaman *Agent* HFP1

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.859	0.004	0.859
Defensive	0.866	0.002	0.866
Cunning	0.979	0.015	0.979
Random Team	0.869	0.010	0.868
Random Agent	0.885	0.041	0.869

Tabel 6.18 Hasil Pengujian Efektifitas *Agent* HFP1

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	45.467	9.957	48
Defensive	66.433	4.688	66
Cunning	81.733	3.423	82
Random Team	59.367	4.499	60
Random Agent	59.4	5.021	60

Tabel 6.19 Hasil Pengujian *Turning Point Agent* HFP2

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	137.2	38.888	147
Defensive	0 %	19.5	11.069	15
Cunning	0 %	10	0	10
Random Team	0 %	17.267	7.569	17
Random Agent	0 %	15.367	7.103	12

Tabel 6.20 Hasil Pengujian Keberagaman *Agent* HFP2

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.907	0.039	0.903
Defensive	0.872	0.019	0.871
Cunning	0.976	0.011	0.977

Random Team	0.951	0.056	0.969
Random Agent	0.975	0.041	0.991

Tabel 6.21 Hasil Pengujian Efektifitas *Agent* HFP2

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	45.533	6.213	46
Defensive	66.033	5.455	66
Cunning	80.733	4.545	81
Random Team	57.167	4.086	58
Random Agent	56	4.503	56

Tabel 6.22 Hasil Pengujian *Turngin Point Agent* HFP3

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	138.933	47.871	139
Defensive	0 %	17.767	9.525	14
Cunning	0 %	10	0	10
Random Team	0 %	14.9	6.723	12
Random Agent	0 %	11.967	2.709	11

Tabel 6.23 Hasil Pengujian Keberagaman *Agent* HFP3

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.976	0.014	0.983
Defensive	0.883	0.041	0.868
Cunning	0.971	0.012	0.974
Random Team	0.973	0.046	0.988
Random Agent	0.993	0.027	1.006

Tabel 6.24 Hasil Pengujian Efektifitas *Agent* HFP3

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	44.433	5.899	45
Defensive	64.333	5.245	64

Cunning	80.833	4.927	80
Random Team	57.867	4.305	58
Random Agent	54.433	4.125	56

6.1.3.2 Skenario IV

Skenario ini melakukan pengujian metode *weight clipping*. Metode ini akan membatasi nilai maksimum dan minimum weight sebagai parameter yang dibutuhkan di dalam algoritma *Weight Update*. Pada skenario pengujian ini akan dilakukan penambahan nilai parameter '*wdec*' untuk mendapatkan nilai yang optimal untuk pengaturan tingkat kesulitan *agent*. Tabel 6.25 mejelaskan nilai dari parameter-parameter yang akan diuji di dalam Skenario IV yang diimplementasikan ke dalam sebuah *agent.*

Tabel 6.25 Parameter Pengujian Metode *Weight Clipping*

Nama *Agent*	Parameter				
	winit	*wmin*	*wmax*	*winc*	*wdec*
WClip1	100	0	200	5	5
WClip2	100	0	200	5	15
WClip3	100	0	200	5	30

Hasil pengujian *agent* WClip1 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.26, Tabel 6.27, dan Tabel 6.28. Hasil pengujian *agent* WClip2 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.29, Tabel 6.30, dan Tabel 6.31. Hasil pengujian *agent* WClip3 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.32. Tabel 6.33, dan Tabel 6.34.

Tabel 6.26 Hasil Pengujian *Turning Point Agent* WClip1

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	153.2	33.565	153
Defensive	0 %	17.433	12.408	15
Cunning	0 %	10	0	10
Random Team	0 %	13.767	5.171	12

Random Agent	0 %	17.2	9.711	12

Tabel 6.27 Hasil Pengujian Keberagaman *Agent* WClip1

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.857	0.005	0.859
Defensive	0.964	0.052	0.988
Cunning	0.945	0.015	0.942
Random Team	0.919	0.065	0.871
Random Agent	0.877	0.037	0.865

Tabel 6.28 Hasil Pengujian Efektifitas *Agent* WClip1

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	49.967	7.608	52
Defensive	61.3	6.818	63
Cunning	81.767	3.491	82
Random Team	50.1	3.604	51
Random Agent	49.967	3.156	50

Tabel 6.29 Hasil Pengujian *Turning Point Agent* WClip2

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	150.433	36.131	163
Defensive	0 %	14.967	6.117	13
Cunning	0 %	10	0	10
Random Team	0 %	16.1	9.356	13
Random Agent	0 %	17.7	9.064	15

Tabel 6.30 Hasil Pengujian Keberagaman *Agent* WClip2

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.865	0.004	0.866
Defensive	0.982	0.044	1.006
Cunning	0.944	0.012	0.944

Random Team	0.933	0.057	0.927
Random Agent	0.872	0.872	0.018

Tabel 6.31 Hasil Pengujian Efektifitas *Agent* WClip2

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	44.467	6.328	44
Defensive	61.733	4.185	63
Cunning	80.967	3.800	81
Random Team	53.067	5.501	53
Random Agent	45.633	5.629	45

Tabel 6.32 Hasil Pengujian *Turning Point Agent* WClip3

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	151.433	53.809	162
Defensive	0 %	16.933	11.175	12
Cunning	0 %	10.367	2.008	10
Random Team	0 %	14.533	6.072	12
Random Agent	0 %	14.2	7.622	11

Tabel 6.33 Hasil Pengujian Keberagaman *Agent* WClip3

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.867	0.005	0.868
Defensive	0.964	0.0616	1.004
Cunning	0.941	0.011	0.941
Random Team	0.926	0.061	0.899
Random Agent	0.876	0.028	0.869

Tabel 6.34 Hasil Pengujian Efektifitas *Agent* WClip3

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	44.2	6.774	46
Defensive	57.7	7.927	60

Cunning	80.867	3.794	81
Random Team	54.667	4.588	55
Random Agent	47.567	3.857	48

6.1.3.3 Skenario V

Skenario ini melakukan pengujian metode *top culling*. Metode ini digunakan untuk memperoleh *rule* atau *macro action* yang lebih lemah dengan cara membatasi akses ke dalam *rule* atau *macro action* yang terlalu efektif. Pada skenario pengujian ini akan dilakukan penambahan nilai parameter '*wdec*' untuk mendapatkan nilai yang optimal untuk pengaturan tingkat kesulitan *agent*. Tabel 6.35 mejelaskan nilai dari parameter-parameter yang akan diuji di dalam Skenario V yang diimplementasikan ke dalam sebuah *agent*.

Tabel 6.35 Parameter Pengujian Metode *Top Culling*

Nama *Agent*	Parameter				
	winit	*wmin*	*wmax*	*winc*	*wdec*
TC1	100	0	200	5	5
TC2	100	0	200	5	15
TC3	100	0	200	5	30

Hasil pengujian *agent* TC1 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.36, Tabel 6.37, dan Tabel 6.38. Hasil pengujian *agent* TC2 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.39, Tabel 6.40, dan Tabel 6.41. Hasil pengujian *agent* TC3 melawan sejumlah *agent* statis untuk masing-masing parameter ditunjukkan oleh Tabel 6.42, Tabel 6.43, dan Tabel 6.44.

Tabel 6.36 Hasil Pengujian *Turning Point Agent* TC1

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	152.733	48.076	165
Defensive	0 %	14.467	7.555	10
Cunning	0 %	10	0	10
Random Team	0 %	16.967	9.349	14

Random Agent	0 %	15	7.643	11

Tabel 6.37 Hasil Pengujian Keberagaman *Agent* TC1

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.863	0.002	0.863
Defensive	0.862	0.011	0.861
Cunning	0.854	0.071	0.831
Random Team	0.885	0.047	0.863
Random Agent	0.863	0.003	0.864

Tabel 6.38 Hasil Pengujian Efektifitas *Agent* TC1

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	40.5	5.405	40
Defensive	48.167	1.085	48
Cunning	48.333	1.493	48
Random Team	49.1	1.900	49
Random Agent	49.467	2.145	49

Tabel 6.39 Hasil Pengujian *Turning Point Agent* TC2

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	184.333	43.440	193
Defensive	0 %	53.933	64.994	20
Cunning	0 %	10.533	1.737	10
Random Team	0 %	19.967	15.457	12
Random Agent	0 %	27.333	23.263	20

Tabel 6.40 Hasil Pengujian Keberagaman *Agent* TC2

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.866	0.002	0.866
Defensive	0.924	0.027	0.925
Cunning	0.831	0.061	0.816

Random Team	0.887	0.032	0.884
Random Agent	0.891	0.029	0.886

Tabel 6.41 Hasil Pengujian Efektifitas *Agent* TC2

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	32.667	5.141	32
Defensive	39.567	5.805	41
Cunning	35.8	3.977	36
Random Team	41	4.258	40
Random Agent	41.033	3.652	42

Tabel 6.42 Hasil Pengujian *Turning Point Agent* TC3

Taktik	*Turning Point*			
	TP>241	Avg.	St.Dev	Med.
Assault	0 %	145.8	54.337	152
Defensive	0 %	84.2	79.936	58
Cunning	0 %	11.433	4.492	10
Random Team	0 %	19.367	14.646	12
Random Agent	0 %	25.267	17.510	22

Tabel 6.43 Hasil Pengujian Keberagaman *Agent* TC3

Taktik	Keberagaman		
	Avg.	St.Dev	Med.
Assault	0.867	0.003	0.866
Defensive	0.926	0.031	0.932
Cunning	0.830	0.067	0.814
Random Team	0.882	0.039	0.883
Random Agent	0.902	0.037	0.907

Tabel 6.44 Hasil Pengujian Efektifitas *Agent* TC3

Taktik	Efektifitas		
	Avg.	St.Dev	Med.
Assault	31.933	4.432	32
Defensive	39.967	5.573	41

Cunning	34.333	5.046	34
Random Team	40.2	5.074	41
Random Agent	41.767	4.761	42

6.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian metode *dynamic scripting* menggunakan *macro action*. Proses analisis yang dilakukan meliputi analisis hasil pengujian dari masing-masing *agent* yang mengimplementasikan metode *dynamic scripting* menggunakan *macro action* dan analisis mengenai teknik *difficulty scaling* yang telah dilakukan ditinjau dari parameter-parameter yang diujikan.

6.2.1 Analisis Skenario I

Skenario I melakukan pengujian tahap pertama, yaitu metode *dynamic scripting*. Analisis dilakukan dengan cara melihat parameter-parameter hasil pegujian *agent* DS1 dan *agent* DS2.

Dari hasil pengujian *agent* DS1 menunjukkan nilai rata-rata *turning point* yang cukup baik, yaitu sebesar 167 ketika melawan *agent* Assault, 64.210 melawan *agent Defensive*, 14 melawan *agent Cunning*, 33.1 melawan *Random Team*, dan 41.967 melawan *Random Agent*.

Agent DS1 juga menunjukkan nilai rata-rata keberagaman yang cukup tinggi, yaitu 0.771 ketika melawan *agent Assault*, 0.711 melawan *agent Defensive*, 0.766 melawan *agent Cunning*, 0.77 melawan *Random Team*, dan 0.773 ketika melawan *Random Agent*.

Nilai rata-rata efektifitas yang ditunjukkan *agent* DS1 cukup tinggi yaitu 27.833 ketika melawan *agent Assault*, 35.333 melawan *agent Defensive*, 49.867 melawan *agent Cunning*, 38.267 melawan *Random Team*, dan 38.433 melawan *Random Agent*.

Pengujian *agent* DS2 menunjukkan nilai rata-rata *turning point* yang lebih baik dari DS1 ketika melawan beberapa *agent*. Nilai *turning point* rata-rata yang dihasilkan dari pengujian *agent* DS2 yaitu sebesar 39.5 ketika melawan *agent*

Defensive, 12.467 melawan *agent Cunning*, 35.867 melawan *Random Team*, dan 33.133 melawan *Random Agent*. Akan tetapi, *agent* DS2 menunjukkan hasil rata-rata *turning point* yang kurang memuaskan ketika melawan *agent Assault*, yaitu sebesar 197.367. Perbedaan nilai yang terjadi disebabkan karena penambahan nilai parameter *MaxPenalty* yang lebih besar jika dibandingkan dengan *agent* DS1. Penambahan nilai parameter *MaxPenalty* kepada *agent* DS2 menyebabkan pemberian *penalty* kepada *rule* yang telah di*generate* oleh *agent* DS2 lebih besar. Dengan nilai *penalty* yang lebih besar, namun tanpa diimbangi dengan proses seleksi urutan antar *rule* di dalam *script* yang baik, menyebabkan *agent* DS2 membutuhkan waktu yang cenderung lebih banyak untuk memproduksi *script* yang efektif jika dibandingkan *agent* DS1 ketika melawan *agent Assault* yang mempunyai *script* yang cenderung kuat.

Nilai rata-rata keberagaman yang ditunjukkan *agent* DS2 tidak jauh berbeda dari *agent* DS1 yaitu sebesar 0.658 melawan *agent Assault*, 0.733 ketika melawan *agent Defensive*, 0.768 melawan *agent Cunning*, 0.775 melawan *Random Team*, dan sebesar 0.776 ketika melawan *Random Agent*. Perbandingan nilai yang tidak jauh berbeda ini disebabkan karena penambahan parameter *MaxPenalty* belum bisa meningkatkan nilai keberagaman rata-rata secara signifikan.

Agent DS2 menunjukkan rata-rata nilai efektifitas yang lebih tinggi dari rata-rata nilai efektifitas yang ditunjukkan *agent* DS1 ketika melawan beberapa *agent*, yaitu sebesar 44.133 ketika melawan *agent Defensive*, 52.367 melawan *agent Cunning*, 44.767 melawan *Random Team*, dan 44.733 melawan *Random Agent*. Akan tetapi, *agent* DS2 menunjukkan nilai efektifitas rata-rata yang kurang memuaskan ketika melawan *agent Assault*, yaitu hanya sebesar 3.333 dengan nilai median 0. Perbedaan nilai yang terjadi disebabkan karena penambahan nilai parameter *MaxPenalty* yang lebih besar jika dibandingkan dengan *agent* DS1. Penambahan nilai parameter *MaxPenalty* kepada *agent* DS2 menyebabkan pemberian *penalty* kepada *rule* yang telah di*generate* oleh *agent* DS2 lebih besar. Dengan nilai *penalty* yang lebih besar, namun tanpa diimbangi dengan proses seleksi urutan antar *rule* di dalam *script* yang baik, menyebabkan *agent* DS2

melakukan *penalty* yang lebih besar terhadap *rule* yang sebenarnya memiliki tingkat efektifitas yang tinggi jika di*generate* pada urutan yang lebih tepat.

Hasil pengujian *agent* DS2 yang lebih baik dari *agent* DS1 di dalam beberapa kasus pengujian kurang sebanding dengan performa *agent* DS2 tersebut ketika melawan *agent Assault*. Dengan kata lain, bisa dikatakan bahwa *agent* DS1 masih lebih baik jika dibandingkan *agent* DS2.

Berdasarkan hasil pengujian *agent* DS1 dan *agent* DS2, bisa dikatakan bahwa dengan nilai parameter yang tepat, metode *dynamic scripting* mampu menyajikan taktik yang mampu beradaptasi dengan cukup cepat dan memiliki nilai efektifitas dan tingkat keberagaman yang cukup tinggi.

6.2.2 Analisis Skenario II

Skenario II melakukan pengujian metode *dynamic scripting* dan *macro action* dengan cara membandingkan data hasil pengujian metode *dynamic scripting* dan *macro action* dengan data hasil pengujian metode *dynamic scripting* di dalam Skenario I.

Dari hasil pengujian *agent* DSMA1 menunjukkan nilai rata-rata *turning point* yang lebih baik dari pada *agent* DS1 maupun *agent* DS2 untuk beberapa *agent* AI statis yang dilawannya. Hal ini disebabkan oleh penggunaan *macro action* yang efektif untuk beberapa jenis *agent* AI statis yang dilawan. Besarnya nilai *turning point* tersebut antara lain 49.333 melawan *agent Defensive*, 10.433 melawan *agent Cunning*, 32.333 melawan *Random Team*, dan 55 melawan *Random Agent*. *Agent* DSMA1 memiliki nilai rata-rata *turning point* yang kurang baik jika dibandingkan *agent* DS1 dan *agent* DS2, yaitu sebesar 201.633 ketika melawan *agent Assault*. Akan tetapi, *turning point* yang ditunjukkan *agent* DSMA1 mempunyai nilai standar deviasi yang lebih kecil dari pada *agent* DS1 dan *agent* DS2, yang berarti bahwa *agent* DSMA1 memberikan hasil *turning point* yang lebih stabil jika dibandingkan *agent* DS1 dan *agent* DS2 ketika melawan *agent Assault*.

Agent DSMA1 juga menunjukkan nilai rata-rata keberagaman yang lebih tinggi daripada *agent* DS1 maupun *agent* DS2 untuk semua *agent* AI statis yang

dilawannya. Hal tersebut disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cukup tinggi. Besarnya nilai keberagaman rata-rata yang didapatkan dari hasil pengujian *agent* DSMA1 yaitu 0.823 ketika melawan *agent Assault*, 0.864 melawan *agent Defensive*, 0.766 melawan *agent Cunning*, 0.77 melawan *Random Team*, dan 0.773 ketika melawan *Random Agent*.

Nilai rata-rata efektifitas yang ditunjukkan *agent* DSMA1 cukup tinggi yaitu 32.733 ketika melawan *agent Assault*, 50.167 melawan *agent Defensive*, 71.667 melawan *agent Cunning*, 45.4 melawan *Random Team*, dan 42.5 melawan *Random Agent*. Nilai efektifitas rata-rata tersebut lebih tinggi dari pada nilai rata-rata efektifitas yang ditunjukkan *agent* DS1 dan *agent* DS2. Tingginya nilai efektifitas rata-rata yang ditunjukkan *agent* DSMA1 jika dibandingkan dengan *agent* DS1 dan *agent* DS2 disebabkan penggunaan *macro action* yang efektif untuk beberapa *agent* AI statis yang dilawan.

Pengujian *agent* DSMA2 menunjukkan nilai rata-rata *turning point* yang lebih baik dari DSMA1. Hal tersebut disebabkan oleh pemberian nilai parameter *MaxPenalty* yang lebih besar jika dibandingkan dengan *agent* DSMA1. Dengan penambahan besarnya nilai *MaxPenalty* sebesar 8, *agent* DSMA2 lebih cepat beradaptasi dalam menyajikan taktik yang lebih efektif jika dibandingkan dengan *agent* DSMA1. Nilai *turning point* rata-rata yang dihasilkan dari pengujian *agent* DSMA2 yaitu sebesar 149 ketika melawan *agent Assault*, 14.933 ketika melawan *agent Defensive*, 10 melawan *agent Cunning*, 14.333 melawan *Random Team*, dan 12.333 melawan *Random Agent*. Nilai *turning point* rata-rata yang ditunjukkan *agent* DSMA2 menunjukkan perbedaan yang signifikan jika dibanding dengan *agent* DS1 dan *agent* DS2 sebelumnya yang merepresentasikan metode *dynamic scripting* tanpa menggunakan *macro action*.

Nilai rata-rata keberagaman yang ditunjukkan *agent* DSMA2 sangat tinggi, yaitu sebesar 0.859 melawan *agent Assault*, 0.999 ketika melawan *agent Defensive*, 0.941 melawan *agent Cunning*, 0.978 melawan *Random Team*, dan sebesar 0.955 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata ini disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cukup tinggi.

Agent DSMA2 menunjukkan rata-rata nilai efektifitas yang sangat tinggi, yaitu sebesar 49.033 ketika melawan *agent Assault*, 62.533 melawan *agent Defensive*, 84 melawan *agent Cunning*, 58.167 melawan *Random Team*, dan 59.333 melawan *Random Agent*. Tingginya nilai efektifitas rata-rata *agent* DSMA2 jika dibandingkan dengan *agent* DSMA1 disebabkan oleh peningkatan nilai parameter *MaxPenalty* sebesar 8. Dengan nilai parameter *MaxPenalty* yang lebih tinggi dan disertai seleksi urutan *rule* di dalam *script* yang baik yang dimiliki sebuah *macro action*, *agent* DSMA2 mampu beradaptasi dengan cepat dalam menyajikan taktik yang sangat efektif.

Dari data hasil pengujian Skenario II *agent* DSMA2 menunjukkan performa yang lebih baik sebagai *agent* AI yang merepresentasikan metode *dynamic scripting* menggunakan *macro action*. Dari data hasil pengujian ini juga bisa dikatakan bahwa metode *dynamic scripting* dan *macro action* mampu menyajikan taktik yang mampu beradaptasi dengan lebih cepat dengan efektifitas dan tingkat keberagaman yang lebih tinggi dibandingkan metode *dynamic scripting*.

6.2.3 Analisis Difficulty Scaling

Analisis pada tahap ini dilakukan untuk membandingkan beberapa metode *difficulty scaling* yang diimplementasikan ke dalam *dynamic scripting* dan *macro action*.

6.2.3.1 Analisis Skenario III

Skenario ini melakukan pengujian terhadap metode *high-fitness penalising* sebagai metode *difficulty scaling* atau pengatur tingkat kesulitan *agent* AI dengan cara membandingkannya dengan data hasil pengujian metode *dynamic scripting* menggunakan *macro action* yang terbaik, yaitu *agent* DSMA2 di dalam Skenario II.

Dari hasil pengujian *agent* HFP1 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* DSMA2 untuk beberapa *agent* AI statis yang dilawan, 16.433 melawan *agent Defensive*, 10.1 melawan *agent Cunning*, 15.933

melawan *Random Team*, dan 15.467 melawan *Random Agent*. Pada pengujian melawan *agent Assault*, *agent* HFP1 menunjukkan hasil *turning point* rata-rata yang lebih baik dari pada *agent* DSMA2 yaitu sebesar 126.5 dengan nilai standar deviasi yang lebih besar jika dibandingkan dengan *agent* DSMA2, yaitu sebesar 9.957. Perbedaan nilai *turning point* yang tidak signifikan jika dibandingkan *agent* DSMA2 dikarenakan proses pemberian *penalty* terhadap nilai *fitness* yang dilakukan *agent* HFP1 belum bisa memperlambat kecepatan adaptasi *agent* HFP1 yang merepresentasikan metode *high-fitness penalising*.

Agent HFP1 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.859 ketika melawan *agent Assault*, 0.866 melawan *agent Defensive*, 0.979 melawan *agent Cunning*, 0.869 melawan *Random Team*, dan 0.885 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* HFP1 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* HFP1 masih menunjukkan hasil yang tidak jauh berbeda dari *agent* DSMA2, yaitu sebesar 45.467 ketika melawan *agent Assault*, 66.433 melawan *agent Defensive*, 81.733 melawan *agent Cunning*, 59.367 melawan *Random Team*, dan 59.4 melawan *Random Agent*. Masih tingginya nilai efektifitas rata-rata *agent* HFP1 jika dibandingkan *agent* DSMA2 dikarenakan proses pemberian *penalty* terhadap nilai *fitness* yang dilakukan *agent* HFP1 masih memungkinkan terpilihnya *macro action* yang memiliki tingkat efektifitas yang terlampau tinggi untuk suatu *agent* AI statis yang dilawan.

Dari data hasil pengujian, *agent* HFP2 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* DSMA2 untuk beberapa *agent* AI statis yang dilawan, 19.5 melawan *agent Defensive*, 10 melawan *agent Cunning*, 17.267 melawan *Random Team*, dan 15.367 melawan *Random Agent*. Pada pengujian melawan *agent Assault*, *agent* HFP2 masih menunjukkan hasil *turning point* rata-rata yang lebih baik dari pada *agent* DSMA2 yaitu sebesar 137.2. Perbedaan nilai *turning point* yang tidak signifikan jika dibandingkan *agent* DSMA2 dikarenakan proses pemberian *penalty* terhadap nilai *fitness* yang dilakukan *agent* HFP2 belum

bisa memperlambat kecepatan adaptasi *agent* HFP2, meskipun dengan nilai *fdec* yang lebih tinggi jika dibandingkan *agent* HFP1.

Agent HFP2 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.907 ketika melawan *agent Assault*, 0.872 melawan *agent Defensive*, 0.976 melawan *agent Cunning*, 0.951 melawan *Random Team*, dan 0.975 melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* HFP2 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* HFP2 masih menunjukkan hasil yang cenderung sama dari *agent* HFP1, yaitu sebesar 44.433 ketika melawan *agent Assault*, 64.333 melawan *agent Defensive*, 80.833 melawan *agent Cunning*, 57.867 melawan *Random Team*, dan 54.433 melawan *Random Agent*. Perbedaan nilai efektifitas rata-rata *agent* HFP2 yang tidak jauh berbeda dari *agent* HFP1 dikarenakan proses pemberian *penalty* terhadap nilai *fitness* yang dilakukan *agent* HFP2 masih kurang mampu memperkecil kemungkinan terpilihnya *macro action* yang memiliki tingkat efektifitas yang terlampau tinggi untuk suatu *agent* AI statis yang dilawan.

Dari data hasil pengujian, *agent* HFP3 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* HFP1 dan HFP2, yaitu sebesar 17.767 melawan *agent Defensive*, 10 melawan *agent Cunning*, 14.9 melawan *Random Team*, dan 11.967 melawan *Random Agent*. Pada pengujian melawan *agent Assault*, *agent* HFP2 masih menunjukkan hasil *turning point* rata-rata yang lebih baik dari pada *agent* DSMA2 yaitu sebesar 138.933. Perbedaan nilai *turning point* yang tidak signifikan jika dibandingkan *agent* HFP1 dan *agent* HFP2 dikarenakan proses pemberian *penalty* terhadap nilai *fitness* yang dilakukan *agent* HFP3 belum bisa memperlambat kecepatan adaptasi *agent* HFP2, meskipun dengan nilai *fdec* yang lebih tinggi jika dibandingkan *agent* HFP1 dan *agent* HFP2.

Agent HFP3 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.976 ketika melawan *agent Assault*, 0.883 melawan *agent Defensive*, 0.971 melawan *agent Cunning*, 0.973 melawan *Random Team*, dan 0.993 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan

agent HFP3 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* HFP3 masih menunjukkan hasil yang cenderung sama dari *agent* HFP1 dan HFP2, yaitu sebesar 44.433 ketika melawan *agent Assault*, 64.333 melawan *agent Defensive*, 80.833 melawan *agent Cunning*, 57.867 melawan *Random Team*, dan 54.433 melawan *Random Agent*. Perbedaan nilai efektifitas rata-rata *agent* HFP3 yang tidak jauh berbeda dari *agent* HFP1 dan *agent* HFP2 dikarenakan proses pemberian *penalty* terhadap nilai *fitness* yang dilakukan *agent* HFP3 masih kurang mampu membatasi kemungkinan terpilihnya *macro action* yang memiliki tingkat efektifitas yang terlampau tinggi untuk suatu *agent* AI statis yang dilawan

Dari data hasil pengujian Skenario III, metode *high-fitness penalising* masih menyajikan taktik dengan tingkat efektifitas yang cukup tinggi. Tingkat efektifitas rata-rata yang terlalu tinggi tersebut disebabkan karena metode *high-fitness penalising* masih belum mampu memperkecil kemungkinan terpilihnya *macro action* yang memiliki tingkat efektifitas yang terlampau tinggi. Tingginya nilai efektifitas rata-rata metode *high-fitness penalising* juga disebabkan oleh penggunaan *macro action* di dalam *macro action base* yang cenderung tangguh jika di*generate* ke dalam suatu *script*.

Untuk parameter keberagaman dan *turning point*, metode ini masih cenderung mampu beradaptasi dengan cepat dan menyajikan tingkat keberagaman yang tinggi.

Dari data hasil pengujian *agent* HFP1, *agent* HFP2 dan *agent* HFP3, penambahan nilai *fdec* menghasilkan nilai efektifitas yang lebih kecil, akan tetapi hasil yang diperoleh masih belum menujukkan perbedaan yang signifikan.

6.2.3.2 Analisis Skenario IV

Skenario ini melakukan pengujian terhadap metode *weight clipping* sebagai metode *difficulty scaling* atau pengatur tingkat kesulitan *agent* AI. Analisis Skenario IV dilakukan dengan cara membandingkannya dengan data

hasil pengujian metode *dynamic scripting* menggunakan *macro action* yang terbaik, yaitu *agent* DSMA2 di dalam Skenario II.

Dari hasil pengujian *agent* WClip1 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* DSMA2 untuk beberapa *agent* AI statis yang dilawan. Perbedaan yang tidak terlalu signifikan tersebut dikarenakan pembatasan nilai *WMax* dan *Wmin* yang dilakukan *agent* WClip1 masih belum bisa memperlambat kecepatan adaptasi *agent* WClip1 yang merepresentasikan metode *weight clipping*. Nilai *turning point* rata-rata yang ditunjukkan dari hasil pengujian *agent* WClip1 yaitu sebesar 153.2 ketika melawan *agent Assault*, 17.433 melawan *agent Defensive*, 10 melawan *agent Cunning*, dan sebesar 15.467 ketika melawan *Random Agent*. Pada pengujian melawan *Random Team*, *agent* WCLip1 menunjukkan hasil *turning point* rata-rata yang lebih baik dari pada *agent* DSMA2 yaitu sebesar 13.767. Secara keseluruhan, *agent* WClip1 menunjukkan hasil *turning point* rata-rata yang lebih kecil dari pada *agent* DSMA2 dengan perbedaan yang tidak signifikan.

Agent WCLip1 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.857 ketika melawan *agent Assault*, 0.964 melawan *agent Defensive*, 0.945 melawan *agent Cunning*, 0.919 melawan *Random Team*, dan 0.877 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* WClip1 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* WClip1 masih menunjukkan hasil yang tidak jauh berbeda dari *agent* DSMA2 untuk beberapa *agent* AI statis, yaitu sebesar 49.967 ketika melawan *agent Assault*, 61.3 melawan *agent Defensive*, dan sebesar 81.767 ketika melawan *agent Cunning*. Hasil efektifitas rata-rata yang lebih kecil ditunjukkan *agent* WClip1 ketika melawan beberapa *agent*, yaitu sebesar 50.1 ketika melawan *Random Team*, dan 49.967 melawan *Random Agent*. *Agent* WClip1 menunjukkan perbedaan nilai efektifitas rata-rata yang signifikan ketika melawan *Random Team* dan *Random Agent*. Masih tingginya nilai efektifitas rata-rata *agent* WClip1 jika dibandingkan *agent* DSMA2 ketika melawan beberapa *agent* AI statis dikarenakan proses pembatasan

nilai *WMin* dan *WMax* yang dilakukan *agent* WClip1 masih memungkinkan terpilihnya *macro action* yang memiliki tingkat efektifitas yang tinggi untuk suatu *agent* AI statis yang dilawan. Akan tetapi, ketika melawan *agent* AI statis yang mengalami proses pemilihan secara acak, yaitu *Random Team* dan *Random Agent*, *agent* WClip1 sudah bisa mampu mengurangi frekuensi terpilihnya m*acro action* yang memiliki tingkat efektifitas yang tinggi.

Dari hasil pengujian *agent* WCLip2 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* WClip1, yaitu sebesar 150.433 ketika melawan *agent Assault*, 14.967 melawan *agent Defensive*, 10 melawan *agent Cunning*, dan sebesar 16.1 ketika melawan *Random Agent*, dan sebesar 17.7 ketika melawan *Random Team*. Secara keseluruhan, nilai rata-rata *turning point* yang ditunjukkan *agent* WClip2 lebih kecil dengan perbedaan yang tidak signifikan jika dibandingkan dengan *agent* DSMA2. Perbedaan nilai *turning point* yang tidak signifikan tersebut dikarenakan pembatasan nilai *WMax* dan *Wmin* yang dilakukan *agent* WClip2 masih belum bisa memperlambat kecepatan adaptasi *agent* WClip2 yang merepresentasikan metode *weight clipping*, meskipun dengan nilai parameter *wdec* yang lebih besar.

Agent WClip2 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.865 ketika melawan *agent Assault*, 0.982 melawan *agent Defensive*, 0.944 melawan *agent Cunning*, 0.933 melawan *Random Team*, dan 0.872 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* WClip2 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* WCLip2 masih menunjukkan hasil yang tidak jauh berbeda dari *agent* WClip1, yaitu sebesar 44.467 ketika melawan *agent Assault*, 61.733 melawan *agent Defensive*, dan sebesar 80.967 ketika melawan *agent Cunning*, 53.067 ketika melawan *Random Team*, dan 45.633 ketika melawan *Random Agent*. Perbedaan nilai efektifitas rata-rata yang tidak jauh berbeda tersebut disebabkan karena penambahan nilai parameter *wdec* pada *agent* WClip2 masih memungkinkan terpilihnya *macro*

action yang memiliki tingkat efektifitas yang tinggi dengan frekuensi yang tidak jauh berbeda dari *agent* WClip1.

Dari hasil pengujian *agent* WClip3 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* WClip1 dan *agent* WClip2. Perbedaan yang tidak terlalu signifikan tersebut dikarenakan pembatasan nilai *WMax* dan *Wmin* yang dilakukan *agent* WClip3 masih belum bisa memperlambat kecepatan adaptasi *agent* WClip3 yang merepresentasikan metode *weight clipping*. Nilai *turning point* rata-rata yang ditunjukkan dari hasil pengujian *agent* WClip3 yaitu sebesar 151.433 ketika melawan *agent Assault*, 16.933 melawan *agent Defensive*, 10.367 melawan *agent Cunning*, dan sebesar 14.533 ketika melawan *Random Agent*, dan sebesar 14.2 ketika melawan *Random Team*.

Agent WClip3 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.867 ketika melawan *agent Assault*, 0.964 melawan *agent Defensive*, 0.941 melawan *agent Cunning*, 0.926 melawan *Random Team*, dan 0.876 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* WClip3 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* WClip3 masih menunjukkan hasil yang tidak jauh berbeda dari *agent* WClip1 dan *agent* WClip2, yaitu sebesar 44.2 ketika melawan *agent Assault*, 57.7 melawan *agent Defensive*, dan sebesar 80.867 ketika melawan *agent Cunning*, 54.667 ketika melawan *Random Team*, dan 47.567 ketika melawan *Random Agent*. Perbedaan nilai efektifitas rata-rata yang tidak jauh berbeda tersebut disebabkan karena penambahan nilai parameter *wdec* pada *agent* WClip3 masih memungkinkan terpilihnya *macro action* yang memiliki tingkat efektifitas yang tinggi dengan frekuensi yang tidak jauh berbeda dari *agent* WClip1 dan *agent* WClip2.

Dari hasil pengujian Skenario IV yang didapatkan, metode *weight clipping* mampu menghasilkan taktik dengan efektifitas yang lebih rendah dari metode *dynamic scripting* dan *macro action*. Akan tetapi, pengujian terhadap beberapa *agent* AI statis masih menunjukkan tingkat efektifitas yang cukup tinggi. Secara

keseluruhan, metode *weight clipping* masih kurang mampu menseleksi *macro action* yang memiliki tingkat efektifitas yang cukup rendah dengan cepat.

Metode ini juga masih mempunyai tingkat keberagaman yang tinggi dan mampu beradaptasi secara cepat.

Dari hasil yang pengujian *agent* WClip1, *agent* WClip2, dan *agent* WClip3, menunjukkan bahwa penambahan nilai *wdec* pada metode ini pada beberapa pengujian masih menunjukkan nilai efektifitas yang lebih besar dengan perbedaan yang tidak signifikan.

6.2.3.3 Analisis Skenario V

Skenario ini melakukan pengujian terhadap metode *top culling* sebagai metode *difficulty scaling* atau pengatur tingkat kesulitan *agent* AI. Analisis Skenario V dilakukan dengan cara membandingkannya dengan data hasil pengujian metode *dynamic scripting* menggunakan *macro action* dengan performa terbaik, yaitu *agent* DSMA2 di dalam Skenario II.

Dari hasil pengujian *agent* TC1 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* DSMA2. Hal tersebut dikarenakan pembatasan seleksi *macro action* yang memiliki tingkat efektifitas yang tinggi yang dilakukan di dalam metode *top culling* masih belum bisa mengurangi kecepatan adaptasi *agent* TC1 yang merepresentasikan metode ini. Nilai *turning point* rata-rata yang ditunjukkan *agent* TC1 yaitu sebesar 152.733 ketika melawan *agent Assault*, 14.467 melawan *agent Defensive*, 10 melawan *agent Cunning*, 16.967 ketika melawan *Random Agent*, dan sebesar 15 ketika melawan *Random Team*. Secara keseluruhan, nilai rata-rata *turning point* yang ditunjukkan *agent* TC1 lebih kecil dengan perbedaan yang tidak signifikan jika dibandingkan dengan *agent* DSMA2.

Agent TC1 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.863 ketika melawan *agent Assault*, 0.862 melawan *agent Defensive*, 0.854 melawan *agent Cunning*, 0.885 melawan *Random Team*, dan 0.863 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* TC1 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* TC1 menunjukkan hasil yang lebih kecil dibandingkan *agent* DSMA2, yaitu sebesar 40.5 ketika melawan *agent Assault*, 48.167 melawan *agent Defensive*, dan sebesar 48.333 ketika melawan *agent Cunning*, 49.1 ketika melawan *Random Team*, dan 49.467 melawan *Random Agent*. Secara keseluruhan, *agent* TC1 menunjukkan perbedaan nilai efektifitas rata-rata yang signifikan dibanding *agent* DSMA2. Perbedaan yang signifikan tersebut disebabkan karena metode ini mampu secara langsung membatasi terpilihnya *macro action* yang memiliki tingkat efektifitas yang cenderung tinggi untuk di*generate* ke dalam sebuah *script*.

Dari hasil pengujian *agent* TC2 menunjukkan nilai rata-rata *turning point* yang lebih besar dari *agent* DSMA2 untuk beberapa *agent* AI statis, yaitu sebesar 184.333 ketika melawan *agent Assault*, 53.933 melawan *agent Defensive*, 10.533 melawan *agent Cunning*, 19.967 ketika melawan *Random Agent*, dan sebesar 27.333 ketika melawan *Random Team*. Secara keseluruhan, nilai rata-rata *turning point* yang ditunjukkan *agent* TC2 lebih besar dengan perbedaan yang signifikan jika dibandingkan dengan *agent* DSMA2 ketika *agent Assault*, *agent Defensive*, *Random Team*, dan *Random Agent*. Perbedaan yang tidak terlalu signifikan tersebut dikarenakan pembatasan seleksi *macro action* yang memiliki tingkat efektifitas yang tinggi yang dilakukan di dalam metode *top culling* masih belum bisa mengurangi kecepatan adaptasi *agent* TC2, meskipun dengan nilai parameter *wdec* yang lebih besar.

Agent TC2 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.866 ketika melawan *agent Assault*, 0.924 melawan *agent Defensive*, 0.831 melawan *agent Cunning*, 0.887 melawan *Random Team*, dan 0.891 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* TC2 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* TC2 menunjukkan hasil yang lebih kecil dibandingkan *agent* TC1, yaitu sebesar 32.667 ketika melawan *agent Assault*, 39.567 melawan *agent Defensive*, dan sebesar 35.8 ketika melawan *agent Cunning*, 41 ketika melawan *Random Team*, dan 41.033 ketika melawan

Random Agent. Secara keseluruhan, hasil nilai efektifitas rata-rata *agent* TC2 menunjukkan perbedaan yang signifikan dibanding *agent* TC1. Perbedaan yang signifikan tersebut dikarenakan pemberian nilai parameter *wdec* yang lebih besar dari pada *agent* TC1. Penambahan nilai *wdec* mampu membuat nilai yang menjadi pembatas sebuah *macro action* untuk bisa terpilih untuk di*generate* ke dalam sebuah *script* cenderung lebih kecil.

Dari hasil pengujian *agent* TC3 menunjukkan nilai rata-rata *turning point* yang tidak jauh berbeda dari *agent* TC2 untuk beberapa *agent* AI statis. Perbedaan yang tidak terlalu signifikan tersebut dikarenakan pembatasan seleksi *macro action* yang memiliki tingkat efektifitas yang tinggi yang dilakukan di dalam metode *top culling* masih belum bisa mengurangi kecepatan adaptasi *agent* TC3, meskipun dengan nilai parameter *wdec* yang lebih besar. Nilai *turning point* rata-rata yang ditunjukkan *agent* TC3 yaitu sebesar 11.433 melawan *agent Cunning*, dan sebesar 19.367 ketika melawan *Random Agent*, dan sebesar 25.267 ketika melawan *Random Team*. Ketika melawan *agent Assault*, *agent* TC3 menunjukkan nilai *turning point* rata-rata yang lebih kecil dibanding *agent* TC2, mendekati nilai yang ditunjukkan *agent* DSMA2, yaitu sebesar 145.8. Hasil ketika melawan *agent Defensive* menunjukkan nilai *turning point* rata-rata yang lebih besar dan signifikan jika dibandingkan *agent* TC2, yaitu sebesar 84.2.

Agent TC3 masih menunjukkan nilai rata-rata keberagaman yang tinggi, yaitu 0.867 ketika melawan *agent Assault*, 0.926 melawan *agent Defensive*, 0.830 melawan *agent Cunning*, 0.882 melawan *Random Team*, dan 0.902 ketika melawan *Random Agent*. Tingginya nilai keberagaman rata-rata yang ditunjukkan *agent* TC3 disebabkan oleh penggunaan *macro action* yang memiliki tingkat keberagaman yang cenderung tinggi.

Nilai efektifitas rata-rata yang ditunjukkan *agent* TC3 masih menunjukkan hasil yang tidak jauh berbeda dari *agent* TC2, yaitu sebesar 31.933 ketika melawan *agent Assault*, 39.967 melawan *agent Defensive*, dan sebesar 34.333 ketika melawan *agent Cunning*, 40.2 ketika melawan *Random Team*, dan 41.767 ketika melawan *Random Agent*. Secara keseluruhan, nilai efektifitas rata-rata *agent* TC3 lebih kecil dengan perbedaan yang tidak signifikan jika dibandingkan

dengan *agent* TC2. Perbedaan yang tidak terlalu signifikan tersebut dikarenakan penambahan nilai parameter *wdec* tidak bisa merubah besarnya frekuensi terpilihnya *macro action* yang memiliki tingkat efektifitas yang cenderung rendah yang dimiliki *agent* TC2.

Berdasarkan data hasil Skenario V, metode *top culling* mampu menyajikan taktik dengan efektifitas yang lebih rendah jika dibandingkan metode *dynamic scripting* dan *macro action* untuk semua jenis *agent* AI statis yang diuji.

Tingkat keberagaman yang dimiliki metode ini masih tinggi dengan kemampuan beradaptasi yang cepat.

Penambahan nilai *wdec* pada metode ini mampu menghasilkan nilai efektifitas yang lebih kecil, akan tetapi hasil yang diperoleh masih belum menujukkan perbedaan yang signifikan.

Dari ketiga hasil pengujian algoritma *dificulty scaling*, metode *top culling* menunjukkan hasil perbedaan efektifitas yang signifikan jika dibandingkan dengan dua metode lainnya. Hasil nilai efektifitas metode *top culling* juga mempunyai standar deviasi yang lebih kecil jika dibandingkan dua metode yang lainnya, hal itu berarti metode *top culling* lebih bisa diandalkan daripada kedua metode yang lain.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Implementasi metode *dynamic scripting* dan *macro action* mampu menghasilkan *agent* AI adaptif dan mampu menyajikan taktik yang beragam tanpa mengesampingkan efektifitas taktik tersebut.
2. Hasil pengujian nilai *turning point* yang dilakukan terhadap tiga jenis *agent* statis menunjukkan bahwa penggunaan *macro action* di dalam metode *dynamic scripting* mampu meningkatkan kecepatan adaptasi *agent* dalam menyajikan taktik yang efektif.
3. Hasil pengujian nilai keberagaman yang dilakukan terhadap tiga jenis *agent* statis menunjukkan bahwa penggunaan *macro action* di dalam metode *dynamic scripting* mampu menyajikan taktik yang lebih beragam dibanding metode *dynamic scripting* pada umumnya.
4. Hasil pengujian nilai efektifitas yang dilakukan terhadap tiga jenis *agent* statis menunjukkan bahwa penggunaan *macro action* di dalam metode *dynamic scripting* mampu menyajikan taktik yang lebih efektif dibanding metode *dynamic scripting* pada umumnya.
5. Dibandingkan dengan metode *high-fitness penalising* dan *weight clipping*, metode *top culling* sebagai pengatur tingkat kesulitan *agent* berhasil menunjukkan hasil yang terbaik.
6. Peningkatan nilai parameter *fdec* pada metode *high-fitness penalising* dan *wdec* pada metode *weight clipping* dan *top culling* untuk nilai tertentu tidak menujukkan penurunan nilai efektifitas yang signifikan. Perubahan yang tidak signifikan tersebut disebabkan oleh penggunaan *macro action* yang cenderung terlalu efektif.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan *agent* AI ini antara lain:

1. Untuk pengembangan lebih lanjut metode ini dapat dikembangkan untuk jenis game yang lain.
2. Untuk implementasi algoritma *dynamic scripting* dan *macro action* yang lebih optimal diperlukan perancangan dan proses *balancing skill* yang lebih baik.

DAFTAR PUSTAKA

[SZI-09] Szita, Istv´an, et al. 2009. Effective and Diverse Adaptive Game AI. Universiteit Maastricht, Institute for Knowledge and Agent Technology.

[SPR-05] Spronck, Pieter, et al. 2005. Adaptive Game AI with Dynamic Scripting. Universiteit Maastricht, Institute for Knowledge and Agent Technology.

[TIM-06] Timuri, Timor. 2006. Automatic Rule Ordering for Dynamic Scripting. Master Thesis at Maastricht ICT Competence Centre Institute for Knowledge and Agent Technology Maastricht, The Netherlands

[MIL-06] Millington, Ian. 2006. Artificial Intelligence for Games. The Morgan Kaufmann Series in Interactive 3D Technology.

[SUL-11] Sullivan, Anne. 2011. Extending *CRPG*s as an Interactive Storytelling Form. Center for Games and Playable Media, UC Santa Cruz.