Implementasi Adaptive Artificial Intelligence pada game Capture The Flag dengan metode Dynamic Scripting

Primawan Satrio[1], Agung Toto Wibowo, S.T., M.T. [2], Bayu Munajat, S.T., M.T.[3]
[1]primawan.satrio@gmail.com, [2]agungtoto@telkomuniversity.ac.id, [3]bayumunajat@telkomuniversity.ac.id

Abstract- ***Artificial Intelligence (AI) sudah dipelajari semenjak tahun 1956 saat John McCarthy mengadakan konferensi akademik pertama nya, dan masih berkembang hingga sekarang karena bidang ini masih sangat sukar dipahami di bidang ilmu komputer. AI mencakup dari mesin yang mampu berpikir, hingga mampu berkembang dan beradaptasi terhadap lingkungan nya dengan tujuan menjadi pemodelan lingkungan yang ada sekarang. Dengan pemodelan yang real-time, diharapkan bisa membantu untuk pengambilan proses keputusan dengan kasus yang lebih kompleks.***
Game Capture The Flag merupakan salah satu metode yang bisa menggambarkan secara real-time karena dengan kedua pihak mempunyai sumber daya yang sama, dan mengandalkan strategi untuk memenangkan permainan. Tetapi salah satu permasalahannya ada pada AI game tersebut. Kebanyakan AI yang ada bersifat statis, sehingga pemain cepat bosan karena pola permainan AI tersebut sudah bisa ditebak.
Dengan alasan tersebut, penulis merekomendasikan Dynamic Scripting sebagai metode untuk mengatasi AI statis dengan meng-implementasikan Adaptive Artificial Intelligence (AAI) pada studi kasus Capture The Flag. Dengan parameter performansi yaitu perbandingan skor antara AAI dengan AI statis.
Berdasarkan hasil pada pengujian pada 4 AI statis, didapatkan bahwa AAI pada AI Balanced dan AI Random kurang memperoleh hasil yang memuaskan dengan tinggi nya skor pada AAI. Sedangkan pada AI Greedy dan AI Defender skor yang diperoleh lebih mencapai harapan dengan seimbang nya skor antara AI statis tersebut dengan AAI.

Kata Kunci : ***reward and punishment, script generation, adaptive artificial intelligence***

I. LATAR BELAKANG

AI dipakai di pembuatan *game*. *Game* yang menarik adalah game yang bisa memberikan pengalaman yang interaktif [1]. *Game* itu menarik dikarenakan beberapa hal : grafis, *gameplay experience*, *replayability value*, dan *artificial intelligence (AI)* nya. AI yang menarik adalah AI yang bisa bersaing dengan pemain, bukan mengalahkan pemain [2]. Kualitas AI yang ada pada *game* sekarang masih rendah, sehingga pemain *game* lebih memilih bermain *multi player* melawan musuh yang dikendalikan oleh manusia, karena kualitas AI-nya masih rendah . [3]

Game Capture The Flag (CTF) merupakan *game* yang kompleks dan adversarial, dengan pemain yang saling berkompetisi & tujuan yang membutuhkan keputusan untuk dibuat di level permainan yang berbeda. Tidak seperti *game* kejar-hidar standar, tujuan dari setiap pemain tidak se-simpel untuk menghindar atau mengejar, tetapi juga bertujuan untuk menyerang dan mempertahankan bendera. Pendekatan yang serupa juga sering digunakan di *game* kejar-hindar yang lebih kompleks, seperti *game* pertarungan udara (eg *Ace Combat*), dimana peran pemain bisa berubah dari waktu ke waktu [4]. Sistem pakar, logika fuzzy, dan mesin automata telah sukses diaplikasikan di real time games [5]. Bagaimanapun, teknik ini biasanya menghasilkan behavior yg statis yang tidak beradaptasi ke strategi lawan. Jika pemain mampu meng-exploit kelemahan AI game, pemain akan kehilangan minatnya setelah bermain beberapa game dikarenakan tantangan yang kurang menarik. Terlebih lagi, saat game semakin lama semakin canggih, mengembangkan sebuah AI game menjadi lebih sulit dan kelemahan yg ditimbulkan akan semakin besar. CTF merepresentasikan sebuah game canggih yang membutuhkan manajemen tim yang cerdas untuk memperoleh dua sub-tujuan yang bertolak belakang (mengambil bendera musuh dan melindungi bendera sendiri) sebagaimana layaknya behavior individu yang cerdas.

AI pada game CTF bisa digunakan untuk simulasi militer [6], kontrol lalu lintas udara, serta logistik yang sifatnya komersil [4]. Dengan penggunaan AI pada tempat yang tergolong vital, serta dibutuhkan pengambilan keputusan yang cepat serta akurat. Semakin banyak pengetahuan dan pengalaman yang dimilki pengambil keputusan, maka keputusan yang dibuat akan semakin baik pula. Oleh karena itu, AI yang bisa berkembang sesuai dengan kondisi di lapangan yang terus berubah akan semakin dibutuhkan [7].

II. DASAR TEORI

A. *Adaptive Artificial Intelligence*

Adaptive Artificial Intelligence (AAI) mengacu ke *non-player character (NPC)* dinamis dimana komputer mampu mengadaptasi *game behavior*-nya dalam merespon musuhnya, baik ketika sesi bermain *game*, atau dalam diantara sesi tersebut. Dalam kasus tertentu, penskalaan tingkat kesulitan pada *game* dinamis menggunakan AAI untuk secara otomatis mengadaptasi parameter *game* dan *behavior* pada kondisi *real time* menurut tingkat keahlian pemain pada *game* (Tan, C.H,. *et al.* 2011)

Dalam paper berjudul *'Artificial Intelligence for Adaptive Computer Games*', [8] mengatakan bahwa tantangan untuk AI *game* adalah :

1. *Complex Decision Spaces*
AI pada *game* umumnya menggunakan kodingan manual dari *programmer game*, sehingga kadang AI tersebut *repetitive*, dan pemain bisa dengan mudah mencari celah, dan mengakali AI tersebut. AAI mampu menangani kemungkinan pengambilan keputusan yang besar.
2. *Knowledge Engineering*
Pembuatan *domain knowledge* memerlukan usaha yang besar dalam membangunnya. Pengembang game harus membuat secara manual kodingan yang ada untuk sebuah spesifik *domain* (baik untuk memperoleh *behavior* strategi atau *behavior* yang cocok untuk RPG).
3. *Authoring Support*
Behavior yang dikembangkan secara manual adalah kode perangkat lunak di sebuah bahasa pemrograman yang kompleks, rentan terhadap kesalahan manusia. *Behavior error* bisa dalam *bug* program, atau tidak mencapai hasil yang diinginkan. *Tools* dibutuhkan untuk mendukung keinginan pengembang *game* yang pada umumnya tidak ahli dalam bidang kecerdasan buatan.
4. *Unanticipated Situations*
Hampir tidak memungkinkan untuk mengantisipasi semua kemungkinan situasi dan strategi pemain saat *game* berlangsung. Ini membuat AI sulit untuk dibuat se-alami mungkin untuk merespon setiap sikap yang pemain lakukan.
5. *User-specific Adaptation*
Pemain mempunyai strategi yang berbeda untuk bermain (untuk *game RPG*), atau cara yang berbeda dalam penyampaian alur cerita (untuk *game* drama interaktif, misalnya *Visual Novel*), tingkah karakter dan interaksi yang berbeda. Setiap *game designer* mulai untuk mendesain kemampuan setiap pengguna, strategi AI dan *behavior* nya harus bisa beradaptasi berdasarkan profil pengguna-nya.
6. *Replayability and variability*
Pemain akan mengalami kebosanan saat bermain game yang ke-*n* kali,dan menjumpai strategi yang sama yang digunakan oleh AI pada game tersebut. Walaupun variasi yang simpel bisa diperoleh melalui seleksi stokastik dari *behavior* atau strategi dari *repository* yang besar.
7. *Rhetorical Objectives*
behavior atau strategi yang dikembangkan secara manual bisanya tidak mencapai tujuan *game* secara tepat. Terutama dalam aplikasi atau *domain* dalam skala yang cukup besar. Tujuan *game* bisa berkisar dari hiburan hingga edukasi, dsb. Sehingga, setiap game tersebut harus menyadari bahwa tujuan nya tidak tercapai di basis per-penggunaan, dan berdaptasi sendiri. Sebagai contoh, beberapa user akan bosan, atau tidak mempelajari pelajaran yang dimaksud.

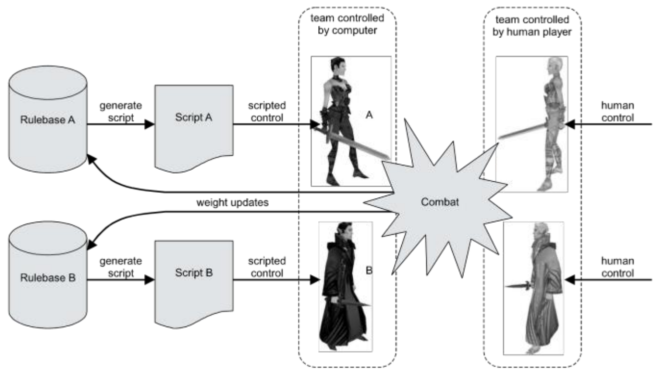
Dan efek implementasi AAI pada game adalah :

- Meningkatkan pengalaman bermain, karena AAI bisa beradaptasi ke tiap individu untuk menyesuaikan cara bermain, dan tujuan pemain memainkan game tersebut.
- Mengurangi waktu dan biaya pengembangan, karena jika sebuah *game* mampu beradaptasi sendiri, pihak pengembang *game* membutuhkan usaha yang lebih minimal untuk memprediksi segala kemungkinan yang akan terjadi.

B. Dynamic Scripting

Dynamic scripting (DS) memenuhi lima dari delapan kebutuhan komputasional dan fungsional, yaitu :

- *Speed* (computational): *DS* cepat secara komputasional, karena DS hanya membutuhkan ekstraksi dari *rulebase* dan mengganti bobot setiap perulangan.
- *Effectiveness* (computational): Efektivitas dari DS teruji ketika semua aturan di *rulebase* masuk akal**,** berdasarkan *domain knowledge* yang sesuai. Setiap aksi dimana *agent* mengeksekusi melalui skrip yang mengandung aturan-aturan ini, adalah aksi yang paling tidak efektif (walaupun itu mungkin tidak sesuai untuk beberapa situasi). Dengan catatan jika seorang pengembang *game* membuat kesalahan dan memasukkan sebuah aturan yang kurang cerdas di *rulebase*, teknik DS akan segera meng-alokasi-kan aturan ini ke bobot nilai terendah. Oleh karena itu, kebutuhan dari efektivitas haruslah terpenuhi bahkan jika *rulebase* terdapat beberapa aturan yang kurang cerdas. Singkatnya, jika kebanyakan aturan di *rulebase* adalah kurang cerdas**,** algoritma DS tidak akan mampu untuk membangkitkan *game* AI yang cukup.
- *Robustness* (computational): algoritma DS merupakan algoritma yang handal, karena bobot dari sebuah aturan di *rulebase* merepresentasikan sebuah kegunaan secara statis, diturunkan dari beberapa sampel *Fitness* yang diharapkan dari skrip yang terdapat dari aturan tersebut. Sebuah hukuman (*penalty*) yang kurang tepat tidak akan menghilangkan sebuah aturan dari *rulebase*, dan akan diberi keringanan ketika aturan tersebut dipilih kembali, atau bahkan ketika aturan lain terkena hukuman. Sama halnya ketika sebuah hadiah (*reward*) mengakibatkan sebuah aturan kurang cerdas dipilih lebih sering, dimana hanya mengakibatkan itu akan mengumpulkan hukuman yang kurang cepat lebih cepat.
- *Clarity* (functional): algoritma DS membangkitkan skrip, dimana skripnya bisa dengan mudah dipahami oleh pengembang game.
- *Variety* (functional): algoritma DS membangkitkan skrip baru untuk setiap *agent*, dan skrip baru ini menyediakan variasi di *behavior* nya.



Gambar 1 Dynamic Scripting [9]

Dynamic scripting adalah teknik *online competitive machine-learning* untuk game AI, sehingga bisa di-klasifikasi-kan sebagai optimasi stokastik. *Dynamic scripting* menjaga beberapa *rulebase,* satu dari setiap kelas *agent* di dalam game. Setiap kali sebuah *instance* sebuah *agent* dibangkitkan, *rulebase* digunakan untuk membuat skrip baru yang mengendalikan *behavior* setiap *agent.* Probabilitas setiap aturan yang ditentukan untuk sebuah skrip dipengaruhi oleh nilai bobot yang dipasang di setiap aturan. Tujuan dari *dynamic scripting* adalah untuk mengadaptasikan bobot di setiap *rulebase* sehingga nilai *fitness* yang diharapkan dari setiap *behavior* didefinisikan dari skrip yang dibangkitkan meningkat secara drastis, bahkan di lingkungan yang cenderung berubah-berubah.

Rumus yang digunakan untuk *team-fitness* yaitu :

$$( )\ \Sigma\ \{ \text{—}(\ \frac{( )}{( )}) \{ \ \} \ \} \quad \text{………………..(1)}$$

Dalam rumus *team-fitness*, *g* merujuk ke tim, *c* merujuk ke *agent*, ***Ng*** N adalah total jumlah *agents* di tim *g*, dan ***ht***(c) N adalah nyawa *agent c* di waktu *t.* Hasil dari rumus ini adalah, apabila sebuah tim memiliki *fitness* 0, maka dianggap kalah. Dan apabila tum yang memiliki *fitness* > 0 maka dianggap menang.

Rumus yang digunakan untuk *agent-fitness* yaitu :

$$(\ ) \ \text{—}(\ ( ) \ \ ( ) \ \ ( ) \ \ ( )) \quad \text{………………..(2)}$$

Dimana *g* adalah tim dimana agent *a* berada. Rumus ini mengandung empat komponen : (1) *F(g),* adalah *fitness* untuk tim *g,* diturunkan dari rumus *team-fitness.* (2) *A(a)* [0,1], dimana adalah rata-rata dari kemampuan bertahan dari agent *a*, (3) *B(g)* [0,1] dimana adalah parameter nyawa dari semua *agents* di tim *g*, dan (4) *C(g)* [0,1] dimana adalah parameter besarnya kerusakan yang telah diberikan semua *agent* di tim lawan *g*.

Untuk rumus ( ) ( ) ( ) didefinisikan sebagai berikut :

$$( ) \ - \ \{ \ (\frac{( )}{}) \{ ( ) \} ; \ \frac{( )}{( )} \{ ( ) \}$$

$$( ) \ \text{—} \Sigma \ \{ \ \{ ( ) \} ; \ \frac{( )}{( )} \{ ( ) \}$$

$$( ) \ \text{—} \Sigma \ \{ \ \{ ( ) \} ; \ \frac{( )}{( )} \{ ( ) \}$$

adalah total jumlah dari *agents* di tim yang mempunyai *g*, *D(a)* N adalah jumlah dari 'kematian' dari *agent a*, dan adalah konstanta (di set angka maksimal dari jumlah babak permainan)

Hasil dari *agent-fitness* diterjemahkan menjadi *weight adjustment* untuk aturan di skrip. Nilai bobot berkisar antara . Nilai bobot baru dikalkulasikan sebagai , dimana adalah nilai bobot asal, dan *weight adjustment* dituliskan di rumus berikut (rumus ini merupakan implementasi dari fungsi *'CalculateAdjustment'* dari algoritma *weight adjustment* yang akan dilampirkan dibawah)

$$\{ \ \lfloor \ \text{—} \rfloor \{ \ \} ; \ \lfloor \ \text{—} \rfloor \{ \ \} \quad \text{………………..(3)}$$

asdas adalah *maximum reward,* dan adalah *maksimum penalty, F* adalah *agent-fitness,* dan ⟨ ⟩ adalah nilai *break-event point.*

Pseudocode untuk *fitness function*, serta *weight adjustment* dilampirkan sebagai berikut :

Algorithm 1 Script Generation

```
ClearScript()
sumweights = 0
for i = 0 to rulecount − 1 do
  sumweights = sumweights + rule[i].weight
end for
{Repeated roulette wheel selection}
for i = 0 to scriptsize − 1 do
  try = 0; lineadded = false
  while try < maxtries and not lineadded do
    j = 0; sum = 0; selected = −1
    fraction = random(sumweights)
    while selected < 0 do
      sum = sum + rule[j].weight
      if sum > fraction then
        selected = j
      else
```

```
        j=j+1
      end if end
        while
      lineadded = InsertInScript(rule[selected].line)
      try = try + 1
    end while
end for
FinishScript()
```

Algoritma 1 merepresentasikan prosedur pembangkitan skrip. Di algoritma ini, fungsi *'InsertInScript'* menambah sebuah baris skrip ke skrip. jika baris skrip tersebut sudah terdapat di skrip, fungsi tersebut tidak mempunyai efek, dan mengembalikan nilai *'false'*. Di sisi lain, baris skrip tersebut dimasukkan dan fungsi nya mengembalikan nilai *'true'*. Algoritma ini bertujuan untuk meletakkan baris *'scriptsize'* di dalam skrip, tapi mungkin berakhir dengan baris skrip yang lebih sedikit jika membutuhkan lebih dari percobaan *'maxtries'* untuk menemukan sebuah baris baru. Fungsi '*FinishScript*' menambahkan satu atau lebih bari skrip yang secara umum bisa diaplikasikan ke skrip, untuk meyakinkan bahwa skrip tersebut akan selalu menemukan cara untuk mengeksekusi.

Algorithm 2 Weight Adjustment

```
active = 0
for i = 0 to rulecount − 1 do
        if rule[i].activated then
          active = active + 1
        end if
end for
if active <= 0 or active >= rulecount then
        return {no updates are needed.}
end if
nonactive = rulecount − active
adjustment = CalculateAdjustment (Fitness)
compensation = −active*adjustment/nonactive
remainder = 0
{Credit assignment}
for i = 0 to rulecount − 1 do
        if rule[i].activated then
          rule[i].weight = rule[i].weight + adjustment
        else
          rule[i].weight = rule[i].weight + compensation
        end if
        if rule[i].weight < minweight then
          remainder = remainder + (rule[i].weight −
minweight)
          rule[i].weight = minweight
        else if rule[i].weight > maxweight then
          remainder = remainder + (rule[i].weight −
maxweight)
          rule[i].weight = maxweight
        end if

end for
DistributeRemainder();
```

Fungsi *'CalculateAdjustment'* meng-kalkulasi-kan hadiah, atau hukuman untuk setiap diterimanya aturan yang telah diaktifkan. Parameter *'Fitness'* adalah tolak ukur dari peformansi skrip tersebut ketika dijalankan. Fungsi *'DistributeRemainder'* mendistribusikan perbedaan antara total bobot yang sekarang, dan total semua bobot asli. Secara umum, *'Fitness'* ini akan diimplementasikan sebagai perulangan di atas semua bobot, menyerahkan sebagian kecil pecahan dari sisa untuk setiap bobot jika itu tidak menyebabkan bobot tersebut melebihi batas bobot, hingga *'remainder'* nya nol. Ketika banyak dari bobot di *rulebases* mencapai batas bobot, ini bisa menjadi proses yang menghabiskan sangat banyak waktu yang benar-benar mengganggu *gameplay*. Sebagai solusi, bagian dari *'remainder'* bisa di bawa ke panggilan penyesuaian bobot berikutnya.

C. *Capture The Flag*

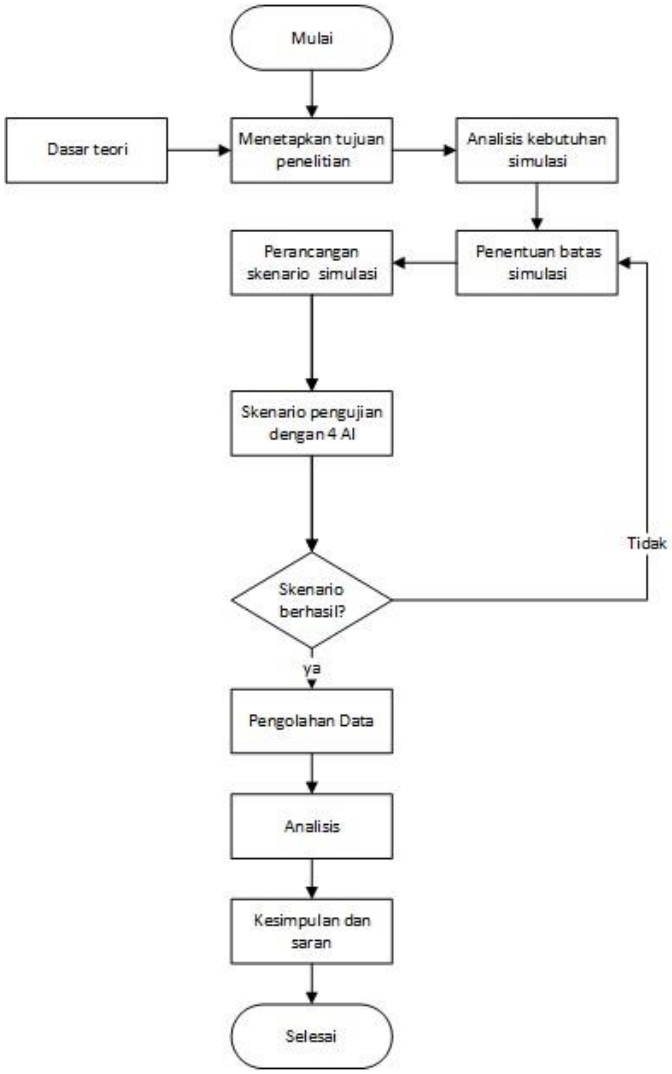
Game dengan genre CTF diadaptasi dari *genre Real Time Strategy (RTS)*, atau *Turn Based Strategy (TBS)*. *Game* CTF merupakan game yang bertujuan dimana setiap tim harus menguasai satu tempat yang dikategorikan sebagai tempat bendera musuh untuk memenangkan permainan. Pemenang bisa didapat dari tim mana yang paling banyak menguasai tempat bendera musuh, atau hingga waktu yang ditentukan habis. *Game* CTF memerlukan strategi tim untuk bertahan, dan menyerang tim musuh. *Game* ini juga membutuhkan kemampuan individual seperti kecepatan berlari, menganalisa medan permainan, dll.

III. SKENARIO SIMULASI

A. *Analisis Kebutuhan Sistem*

Untuk tahap perancangan sistem, simulasi membutuhkan sebuah tools yang mampi mengimplementasikan *'script generation'* & *'weight adjustment'* secara *online learning*, dan parameter performansi yang jelas. Dalam hal ini, parameter performansi yang dimaksud adalah 'skor'.

Proses tahapan dalam membangun simulasi dapat di lihat pada gambar berikut :



Gambar 2 Flowchart simulasi

B. Perancangan Simulasi

Pengantar simulator AI Sandbox

AI Sandbox yang digunakan bertujuan sebagai simulator untuk pengujian AI statis dengan *Adaptive Artificial Intelligence. Game* dengan genre ini mengimplementasikan tiga teknik AI, yaitu : *planning*, *reasoning*, dan *path-finding*. Akan terdapat 20 peta yang keluar secara acak, untuk memeriksa kehandalan AAI nanti nya.
Desain AI Sandbox ini adalah :

1. *Blue bot vs red bot*

Player akan mengendalikan pihak *blue bot* untuk melawan *red bot.* Pemain akan berperan sebagai *AI Commander*, yang artinya hanya akan memerintahkan secara global, bukan secara mikro seperti game RTS pada umumnya.

2. Aturan dasar dari AI Sandbox yang digunakan

Beberapa perubahan seperti jarak pandang bot, kecepatan berjalan, waktu tunda saat bot berputar untuk mengambil posisi menembak akan diseimbangkan lagi apabila diperlukan. Ada tujuh aturan dasar dari game ini, yaitu :

- Satu bot yang mengambil posisi bertahan di arah yang tepat, akan selalu menang melawan bot yang menyerang.
- Satu bot yang menyerang dengan arah yang tepat, akan selalu menang melawan bot yang sedang menyerbu.
- Dua bot yang menyerang satu bot akan menang, setelah terdapat satu korban dari pihak yang menyerang
- Dua bot yang menyerbu satu bot akan menang, setelah terdapat satu korban dari pihak yang menyerbu.
- Satu bot yang menyerbu akan menang melawan bot yang bertahan menghadap arah yang kurang tepat.

Untuk konsep bendera, ada dua aturan dasar, yaitu :

- Saat ada bendera yang jatuh setelah bot yang membawa ditembak oleh pihak lawan, bendera hanya bisa diambil oleh pihak bot yang membawa sebelumnya.
- Setelah sekian detik bendera tersebut tidak ada yang mengambil, bendera tersebut akan kembali lagi ke tempat semula.

Untuk konsep bertarung, bot akan otomatis meng-*handle* ini. Ada tiga aturan dasar yaitu :

- Jeda menembak dari berjalan atau berlari (jeda menembak dari berlari akan lebih lama)
- Jeda menembak untuk berputar, dan mengambil garis lurus pada target.
- Waktu jeda menembak yang konstan (0,5 detik)

3. *AI Commander*

AI Commander berperan sebagai otak dari kubu merah atau biru. Disini, *AI Commander* berperan sebagai *strategic level* untuk AI pada game ini.
Beberapa aturan yang diimplementasikan untuk *AI Commander* ini adalah :

- Informasi penuh tentang peta.
- Informasi penuh akan setiap bot nya (bertahan, menyerang, menyerbu, dan berjalan).
- Sebagian informasi tentang bot musuh. Informasi hanya dapat diketahui apabila bot musuh masuk dalam jarak pandang (menyerang, atau membawa bendera).
- Informasi penuh terhadap posisi bendera di peta.

4. Empat perintah global untuk bot

Sebagai *AI Commander,* perintah yang bisa diberikan untuk bot adalah :

- ***Defend*** - Bot akan mengambil posisi jongkok, dan fokus membidik musuh di sekitarnya. Membidik berakibat tembakan sangat akurat, dan mematikan. Akan tetapi jarak pandang terbatas, dan butuh waktu untuk di-*set* ke posisi seperti ini.

- ***Attack*** – Bot akan menyerang ke posisi yang ditentukan. Jika ada bot musuh yang terlihat, bot akan langsung menembak
- ***Charge*** – Bot akan menyarang ke posisi yang telah ditentukan dengan kondisi berlari. Gerakan ini lebih cepat dari Attack tetapi mendapat pinalti waktu tunda untuk menembak.
- ***Move*** - Berjalan meng-eksplorasi peta. Dalam kondisi ini, bot tidak akan menyerang musuh yang masuk dalam jangkauan tembak.

1.1.1Ruang Lingkup Simulasi

Simulasi dijalankan dengan spesifikasi sebagai berikut :

Perangkat lunak

a) AI Sandbox 0.20.9 dengan 4 AI yang diujikan.
b) Eclipse dengan *plugin* PyDev untuk memprogram AAI dengan bahasa Python.
c) Microsoft Office untuk pengolahan data dan penulisan laporan.

Perangkat keras

a) Single CPU
b) 50 GB HDD
c) 2048 Mb RAM

C. Skenario Simulasi

Pada simulasi yang dijalankan terdapat berbagai macam faktor yang akan mempengaruhi *output* yang dihasilkan, baik secara langsung maupun tidak langsung. Untuk itu, perlu didefinisikan terlebih dahulu pemodelan simulasi secara umum untuk mengetahui faktor apa saja yang berpengaruh saat proses simulasi berlangsung.

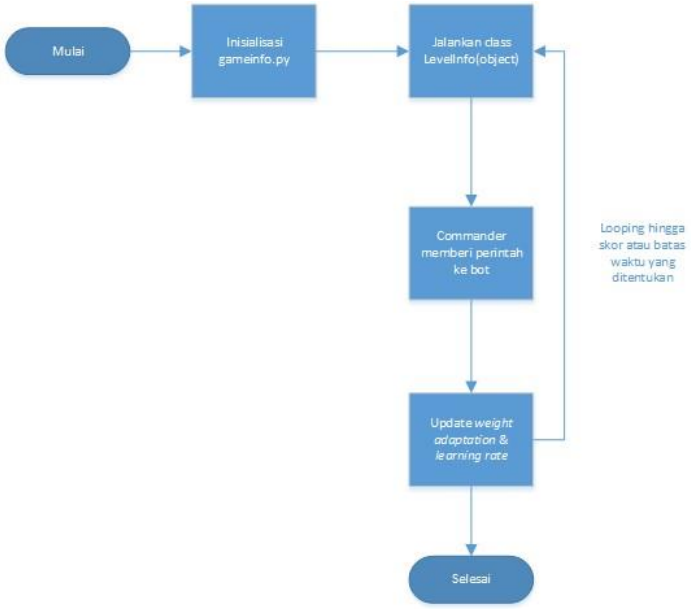
Capture The Flag SDK adalah simulasi yang digunakan untuk menguji *commander* yang beradaptasi lewat AI buatan pemain, dan dijalankan dengan *AI Sandbox*™. AI yang dibuat akan memerintah bot dari tim sendiri, dan beradaptasi untuk mengalahkan tim lawan di beberapa variasi level.

Game ini dimainkan oleh dua tim oleh bot, dimana poin nya diperoleh kapanpun sebuah tim mengambil bendera tim lain dan mengembalikan nya ke lokasi dimana bendera mencetak skor. Pemenangnya adalah tim yang mencetak paling banyak poin. Bendera dan *bot* akan kembali hidup secara sinkron pada interval yang teratur sampai *game* berakhir. *Bot* di *game* ini tidak bisa dikontrol secara interaktif, tetapi mereka dikendalikan oleh AI commander strategis yang dibuat di Python.

Blok diagram ini menjelaskan tentang bagaimana seluruh

proses pada Tugas Akhir ini.

Gambar III.1 Blok diagram simulasi CTF



Gambar III.2 Flowchart keseluruhan sistem

Penggambaran proses implementasi untuk AAI pada game CTF ini dijabarkan menjadi lima tahap, yaitu *insialisasi gameinfo.py, jalankan class LevelInfo(object), commander memberi perintah(informasi) ke bot, update weight adaptation & learning rate,* dan looping hingga skor atau batas waktu yang ditentukan.

Pemodelan AAI dengan Dynamic Scripting

Seiring dengan berjalannya waktu, *knowledge* AAI akan didesain sehingga mampu mempelajari area permainan, mana area yang berbahaya untuk dilewati sehingga akan cenderung menghindari jalur tersebut, dan mana area yang menguntungkan AAI akan cenderung lebih sering dilewati.

Proses yang akan memperbaharui *knowledge* nya dijabarkan dalam proses berikut :

Gambar III.3 Gambaran proses secara keseluruhan

Data Training

Informasi yang didapat dari pertarungan akan diproses melalu *fitness function* dan *weight adjustment.* Data akan terus di-update setiap event yang ditentukan terjadi.

***Data Fitness* didapat**

Setelah nilai *fitness* dan *weight* didapat, masukkan data yang telah di proses ke algoritma Commander.

Update knowledge

Tahap akhir, perbaharui *knowledge* AAI yang ada ke *script game*.

if bot.flag:

```
    # this bot has flag
    if bot.distanceFrom(scoreLocation) > 10:
        # still far from scorelocation

greedy_move_using_heuristic_knowledge(scoreLocation)
    else:
        # bug-handle on being close from targeted-location
        go_directly_to_score_location
else if team_flag_is_stolen:
    prevent_opponent_from_scoring
else if bot.instructed_to_revenges:
    revenge_team_mate
attacker and defender
else:
    if bot.attacker:
        if team.has_not_found_flag:
            search_flag
            # a random exploration
            # explored area has less chance to be re-explored.
        else:

            if not team.have_flag:
                if bot.distanceFrom(flagLocation) > 10:

greedy_move_using_heuristic_knowledge(flagLocation)
                else:
                    # handle on bug when bot is close to target_location
                    go_directly_to_flag
            else:
                # someone on the team has a flag, but not this bot
                cover_the_flag_carrier
    else:
        # for defender
        if bot.far_from_team_flag:
            pick_random_position_around_the_flag
        after_being_close_to_flag:

pick_facing_direction_according_to_event_locations
        after_looking_at_the_direction:
            standby, do nothing until enemy is seen
```

4 kelas yang akan menjadi kelas utama untuk AAI ini adalah sebagai berikut :

a) Class Field

Class ini merupakan class yang digunakan untuk menginisiasi array2d, serta indexing [x,y]

b) Class KnowledgeField

Melakukan perubahan nilai fitness suatu lokasi, dimana area sekitarnya ikut berubah.

c) Class helper

Merupakan class yang dibuat untuk membantu pengimplementasian AAI pada map dengan berupa penyediaan dictionary, tuple dan index field.

d) Class Strategy

Class yang mengontrol dari kelas KnowledgeField (Model), dan Instruction (View?)

IV. HASIL DAN ANALISIS

A. Analisis Skenario 1

AAI vs AI Greedy

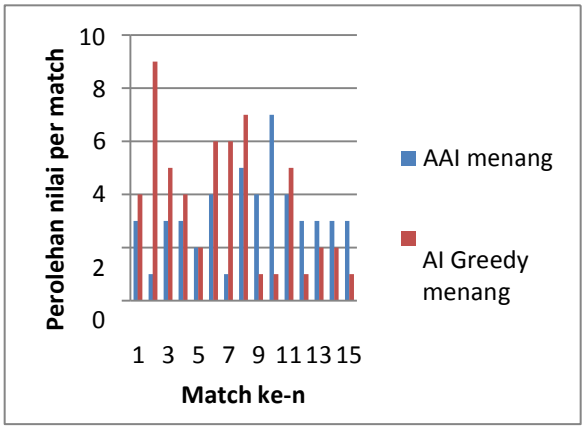
Pengujian pertama dilakukan sebanyak 101 pertandingan.

:

{'AIGreedy': 4, 'AAI': 3}{'AIGreedy': 9, 'AAI': 1}{'AIGreedy': 5, 'AAI': 3}{'AIGreedy': 4, 'AAI': 3}{'AIGreedy': 2, 'AAI': 2}{'AIGreedy': 6, 'AAI': 4}{'AIGreedy': 6, 'AAI': 1}{'AIGreedy': 7, 'AAI': 5}{'AIGreedy': 1, 'AAI': 4}{'AIGreedy': 1, 'AAI': 7}{'AIGreedy': 5, 'AAI': 4}{'AIGreedy': 1, 'AAI': 3}{'AIGreedy': 2, 'AAI': 3}{'AIGreedy': 2, 'AAI': 3}{'AIGreedy': 1, 'AAI': 3}{'AIGreedy': 5, 'AAI': 1}{'AIGreedy': 1, 'AAI': 2}{'AIGreedy': 3, 'AAI': 5}{'AIGreedy': 7, 'AAI': 3}{'AIGreedy': 2, 'AAI': 4}{'AIGreedy': 2, 'AAI': 1}{'AIGreedy': 5, 'AAI': 3}{'AIGreedy': 5, 'AAI': 3}{'AIGreedy': 3, 'AAI': 4}{'AIGreedy': 1, 'AAI': 5}{'AIGreedy': 2, 'AAI': 4}{'AIGreedy': 6, 'AAI': 1}{'AIGreedy': 1, 'AAI': 5}{'AIGreedy': 4, 'AAI': 2}{'AIGreedy': 5, 'AAI': 2}{'AIGreedy': 2, 'AAI': 2}{'AIGreedy': 0, 'AAI': 2}{'AIGreedy': 5, 'AAI': 5}{'AIGreedy': 5, 'AAI': 5}{'AIGreedy': 8, 'AAI': 2}{'AIGreedy': 2, 'AAI': 4}{'AIGreedy': 2, 'AAI': 2}{'AIGreedy': 8, 'AAI': 1}{'AIGreedy': 2, 'AAI': 4}{'AIGreedy': 5, 'AAI': 3}{'AIGreedy': 4, 'AAI': 4}{'AIGreedy': 2, 'AAI': 1}{'AIGreedy': 7, 'AAI': 5}{'AIGreedy': 4, 'AAI': 1}{'AIGreedy': 6, 'AAI': 3}{'AIGreedy': 2, 'AAI': 3}{'AIGreedy': 3, 'AAI': 5}{'AIGreedy': 0, 'AAI': 1}{'AIGreedy': 6, 'AAI': 4}{'AIGreedy': 5, 'AAI': 10}{'AIGreedy': 4, 'AAI': 2}{'AIGreedy': 3, 'AAI': 3}{'AIGreedy': 6, 'AAI': 2}{'AIGreedy': 4, 'AAI': 2}{'AIGreedy': 5, 'AAI': 4}{'AIGreedy': 0, 'AAI': 5}{'AIGreedy': 2, 'AAI': 6}{'AIGreedy': 6, 'AAI': 0}{'AIGreedy': 3, 'AAI': 3}{'AIGreedy': 0, 'AAI': 2}{'AIGreedy': 2, 'AAI': 5}{'AIGreedy': 5, 'AAI': 5}{'AIGreedy': 3, 'AAI': 4}{'AIGreedy': 3, 'AAI': 4}{'AIGreedy': 3, 'AAI': 3}{'AIGreedy': 5, 'AAI': 2}{'AIGreedy': 5, 'AAI': 3}{'AIGreedy': 1, 'AAI': 3}{'AIGreedy': 2, 'AAI': 3}{'AIGreedy': 7, 'AAI': 2}{'AIGreedy': 1, 'AAI': 4}{'AIGreedy': 6, 'AAI': 2}{'AIGreedy': 5, 'AAI': 5}{'AIGreedy': 3, 'AAI': 3}{'AIGreedy': 5, 'AAI': 12}{'AIGreedy': 4, 'AAI': 1}{'AIGreedy': 3, 'AAI': 1}{'AIGreedy': 3, 'AAI': 5}{'AIGreedy': 3, 'AAI': 3}{'AIGreedy': 1, 'AAI': 4}{'AIGreedy': 3, 'AAI': 2}{'AIGreedy': 4, 'AAI': 0}{'AIGreedy': 2, 'AAI': 0}{'AIGreedy': 5, 'AAI': 5}{'AIGreedy': 7, 'AAI': 0}{'AIGreedy': 6, 'AAI':

3}{'AIGreedy': 2, 'AAI': 4}{'AIGreedy': 10, 'AAI': 1}{'AIGreedy': 3, 'AAI': 5}{'AIGreedy': 1, 'AAI': 3}{'AIGreedy': 2, 'AAI': 4}{'AIGreedy': 7, 'AAI': 10}{'AIGreedy': 4, 'AAI': 3}{'AIGreedy': 4, 'AAI': 3}{'AIGreedy': 4, 'AAI': 3}{'AIGreedy': 3, 'AAI': 3}{'AIGreedy': 5, 'AAI': 2}{'AIGreedy': 7, 'AAI': 5}{'AIGreedy': 4, 'AAI': 4}{'AIGreedy': 6, 'AAI': 3}{'AIGreedy': 8, 'AAI': 3}

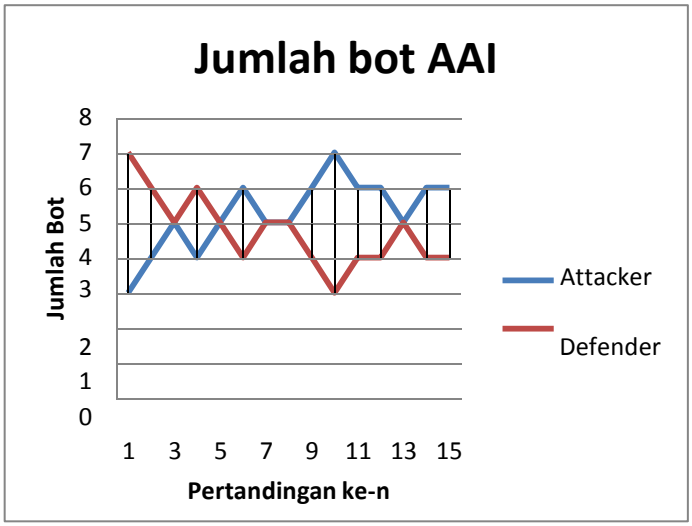
Hasil yang didapat, AI Greedy mendapat perolehan nilai 61, AAI mendapat perolehan nilai 43, dengan seri sebanyak 18 kali.



Gambar IV.1 Diagram AAI vs AI Greedy

1.1.2Analisa komposisi Attacker & Defender bot

Perubahan bot dari match pertama ke match berikutnya tidak begitu signifikan. Ini terlihat dari perubahan attacker dan defender yang tidak begitu jauh karena menyesuaikan bot lawan yang menggunakan strategi menyerang.



Gambar IV.2 Diagram jumlah bot AAI vs AI Greedy

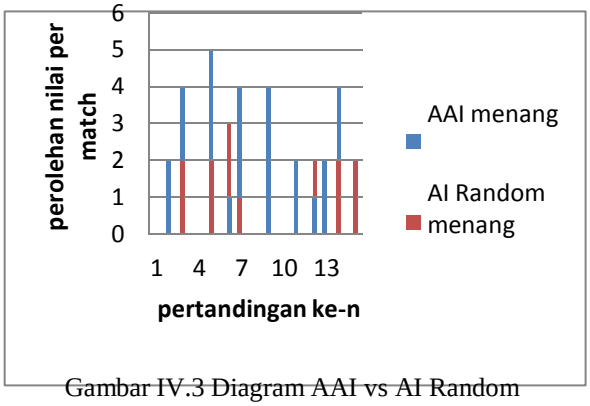
B. Analisis Skenario 2

AAI vs AI Balanced

Pengujian dilakukan sebanyak 4 kali dengan total 101 kali pertandingan

{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 2, 'AAI': 4}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 2, 'AAI': 5}{'AIRandom': 3, 'AAI': 1}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 1, 'AAI': 4}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 2, 'AAI': 1}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 2, 'AAI': 4}{'AIRandom': 2, 'AAI': 2}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 5}{'AIRandom': 0, 'AAI': 1}{'AIRandom': 0, 'AAI': 4}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 0, 'AAI': 3}{'AIRandom': 1, 'AAI': 4}{'AIRandom': 1, 'AAI': 1}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 4}{'AIRandom': 1, 'AAI': 0}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 2, 'AAI': 4}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 0, 'AAI': 7}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 1, 'AAI': 4}{'AIRandom': 1, 'AAI': 1}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 1, 'AAI': 0}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 4, 'AAI': 1}{'AIRandom': 3, 'AAI': 1}{'AIRandom': 1, 'AAI': 5}{'AIRandom': 2, 'AAI': 3}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 0, 'AAI': 3}{'AIRandom': 2, 'AAI': 4}{'AIRandom': 2, 'AAI': 2}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 1, 'AAI': 0}{'AIRandom': 1, 'AAI': 5}{'AIRandom': 2, 'AAI': 1}{'AIRandom': 0, 'AAI': 1}{'AIRandom': 1, 'AAI': 0}{'AIRandom': 2, 'AAI': 2}{'AIRandom': 2, 'AAI': 3}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 1, 'AAI': 4}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 2, 'AAI': 0}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 2, 'AAI': 3}{'AIRandom': 3, 'AAI': 3}{'AIRandom': 1, 'AAI': 0}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 3}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 2, 'AAI': 4}{'AIRandom': 3, 'AAI': 0}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 2, 'AAI': 1}{'AIRandom': 0, 'AAI': 0}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 0, 'AAI': 3}{'AIRandom': 3, 'AAI': 4}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 0, 'AAI': 5}{'AIRandom': 0, 'AAI': 2}{'AIRandom': 4, 'AAI': 2}{'AIRandom': 0, 'AAI': 4}{'AIRandom': 2, 'AAI': 1}{'AIRandom': 2, 'AAI': 3}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 0, 'AAI': 6}{'AIRandom': 1, 'AAI': 3}{'AIRandom': 1, 'AAI': 2}{'AIRandom': 1, 'AAI': 1}{'AIRandom': 0, 'AAI': 5}{'AIRandom': 0, 'AAI': 3}

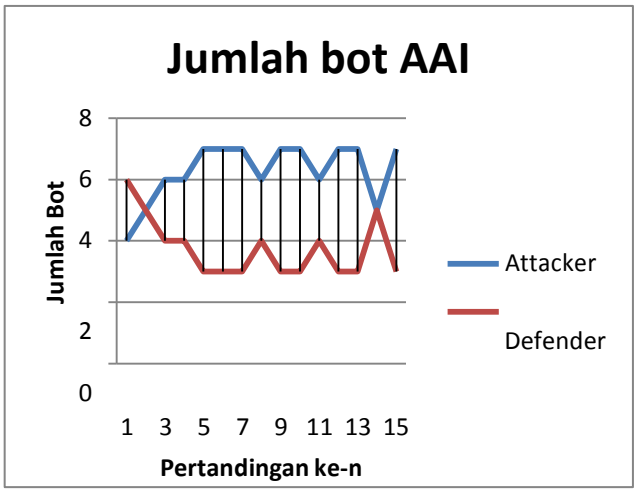
Hasil yang didapat, AI Random memperoleh perolehan nilai 15, AAI memperoleh nilai 66, dengan seri sebanyak 19 kali.



Gambar IV.3 Diagram AAI vs AI Random

1.1.3Analisa komposisi Attacker & Defender bot

Saat bot mengeksploitasi arena, bot AAI kurang bisa membaca pola bot lawan (AI Random) yang ada di arena pertandingan. Sehingga AAI terus mengksploitasi pertandingan dan banyak memenangkan babak demi babak.



Gambar IV.4 Diagram jumlah bot AAI vs AI Random.

C. Analisis Skenario 3

AAI vs AI Balanced

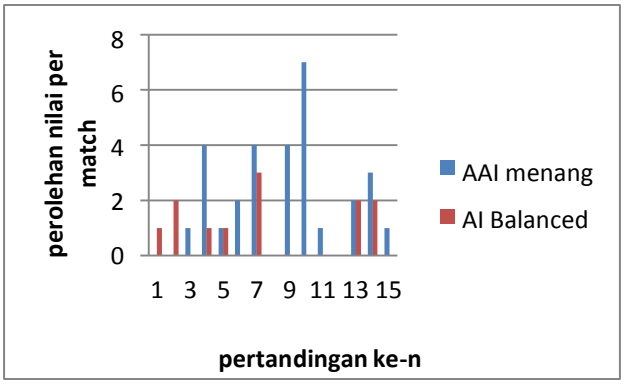
Pengujian dilakukan sebanyak 2 kali, dengan 101 pertandingan

{'AIBalanced': 1, 'AAI': 0}{'AIBalanced': 2, 'AAI': 0}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 1, 'AAI': 4}{'AIBalanced': 1, 'AAI': 1}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 3, 'AAI': 4}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 0, 'AAI': 4}{'AIBalanced': 0, 'AAI': 7}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 2, 'AAI': 3}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 2, 'AAI':

2}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 2, 'AAI': 1}{'AIBalanced': 1, 'AAI': 1}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 0, 'AAI': 8}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 5, 'AAI': 2}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 1, 'AAI': 3}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 1, 'AAI': 5}{'AIBalanced': 0, 'AAI': 6}{'AIBalanced': 1, 'AAI': 5}{'AIBalanced': 2, 'AAI': 1}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 0, 'AAI': 5}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 2, 'AAI': 0}{'AIBalanced': 0, 'AAI': 3}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 1, 'AAI': 3}{'AIBalanced': 1, 'AAI': 3}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 1}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 1}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 2, 'AAI': 3}{'AIBalanced': 3, 'AAI': 2}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 3}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 1}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 2, 'AAI': 5}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 0, 'AAI': 0}{'AIBalanced': 3, 'AAI': 3}{'AIBalanced': 2, 'AAI': 4}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 2, 'AAI': 4}{'AIBalanced': 2, 'AAI': 3}{'AIBalanced': 1, 'AAI': 5}{'AIBalanced': 4, 'AAI': 2}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 1, 'AAI': 2}{'AIBalanced': 4, 'AAI': 4}{'AIBalanced': 0, 'AAI': 3}{'AIBalanced': 0, 'AAI': 6}{'AIBalanced': 3, 'AAI': 5}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 3, 'AAI': 4}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 1, 'AAI': 1}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 0, 'AAI': 1}{'AIBalanced': 1, 'AAI': 2}{'AIBalanced': 2, 'AAI': 1}{'AIBalanced': 1, 'AAI': 5}{'AIBalanced': 3, 'AAI': 3}{'AIBalanced': 0, 'AAI': 3}{'AIBalanced': 1, 'AAI': 2}{'AIBalanced': 0, 'AAI':

0}{'AIBalanced': 0, 'AAI': 5}{'AIBalanced': 0, 'AAI': 5}{'AIBalanced': 4, 'AAI': 3}{'AIBalanced': 2, 'AAI': 2}{'AIBalanced': 2, 'AAI': 0}{'AIBalanced': 3, 'AAI': 1}{'AIBalanced': 0, 'AAI': 5}{'AIBalanced': 0, 'AAI': 2}{'AIBalanced': 4, 'AAI': 2}

Hasil yang didapat, AI Balanced memperoleh perolehan nilai 14, AAI memperoleh perolehan nilai 58, dan seri sebanyak 28 kali.

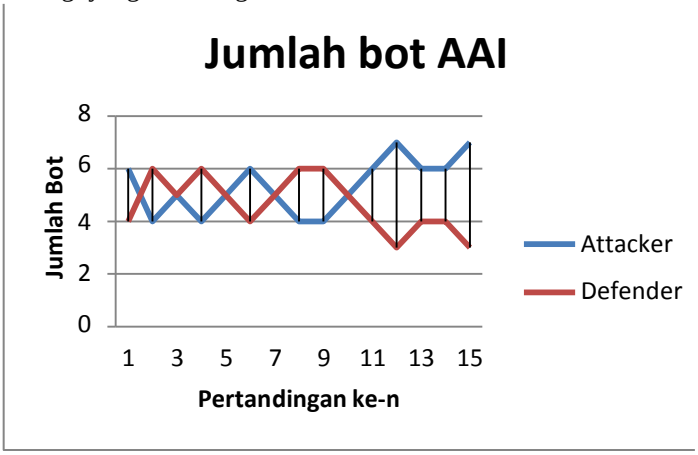


Gambar IV.5 Diagram AAI vs AI Balanced

1.1.4Analisa komposisi Attacker & Defender bot

Penyesuaian AAI dengan AI Balanced ada penyesuaian berkala di setiap matchnya. AI Balanced menggunakan strategi yang seimbang antara bot penyerang dengan yang bertahan. AAI menggunakan strategi yang terus

mengeksploitasi arena karena AI Balanced menggunakan strategi yang seimbang.



Gambar IV.6 Diagram jumlah bot AAI vs AI Random

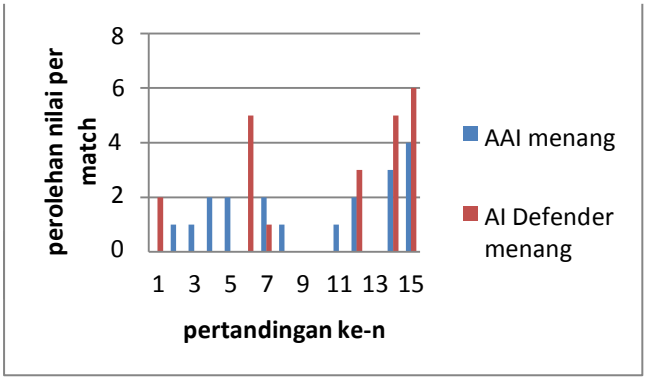
D. Analisis Skenario 4

AAI vs AI Random

Pengujian dilakukan sebanyak 3 kali, dengan total 101 pertandingan

{'AAI': 2, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 5}{'AAI': 2, 'AIDefender': 1}{'AAI': 1, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 3}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 5}{'AAI': 4, 'AIDefender': 6}{'AAI': 1, 'AIDefender': 3}{'AAI': 3, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 2}{'AAI': 2, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 1}{'AAI': 0, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 3}{'AAI': 3, 'AIDefender': 5}{'AAI': 3, 'AIDefender': 0}{'AAI': 4, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 6}{'AAI': 2, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 4}{'AAI': 0, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 5, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 4}{'AAI': 2, 'AIDefender': 1}{'AAI': 0, 'AIDefender': 1}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 3}{'AAI': 0, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 5}{'AAI': 4, 'AIDefender': 1}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 1}{'AAI': 0, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 3}{'AAI': 1, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 1}{'AAI': 1, 'AIDefender': 6}{'AAI': 1, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 1}{'AAI': 4, 'AIDefender': 9}{'AAI': 3, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 5, 'AIDefender': 3}{'AAI': 1, 'AIDefender': 0}{'AAI': 4, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 1}{'AAI': 1, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 2, 'AIDefender': 1}{'AAI': 3, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 3}{'AAI': 2, 'AIDefender': 5}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 9}{'AAI': 0, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 5}{'AAI': 3, 'AIDefender': 2}{'AAI': 2, 'AIDefender': 1}{'AAI': 0, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 2}{'AAI': 2, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 1, 'AIDefender': 0}{'AAI': 4, 'AIDefender': 0}{'AAI': 4, 'AIDefender': 6}{'AAI': 2, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 3}{'AAI': 4, 'AIDefender': 0}{'AAI': 3, 'AIDefender': 0}{'AAI': 0, 'AIDefender': 0}{'AAI': 4, 'AIDefender': 0}

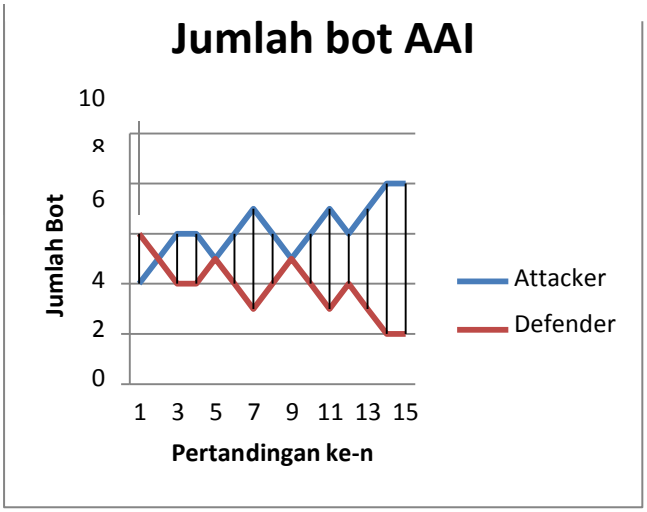
Hasil yang didapat, AI Defender memperoleh nilai 43, AAI memperoleh nilai 25, dengan seri sebanyak 33 kali.



Gambar IV.7 Diagram AAI vs AI Defender

1.1.5Analisa komposisi Attacker & Defender bot

Jumlah bot *attacker* dan *defender* mengalami cukup banyak perubahan. Perubahan ini didasari oleh penyesuaian strategi AAI terhadap AI Defender yang bot nya banyak melakukan strategi pertahanan, sehingga bot AAI menambah jumlah bot *attacker*.



Gambar IV.8 Diagram jumlah bot AAI vs AI Random

V. KESIMPULAN DAN SARAN

Kesimpulan

Hasil pengujian menggambarkan bahwa AAI memperoleh nilai paling baik saat diuji dengan (sesuai urutan) AI Defender, AI Greedy, AI Balanced, dan AI Random.

1. AAI tidak begitu baik melawan AI Random karena pola nya yang acak.
2. AAI sangat baik mengahadapi AI Defender karena hanya satu yang menyerang, dan lebih banyak yang bertahan. Disini, bobot heuristik peta kurang ter--*update* karena minim konfrontasi di arena simulasi.

Saran

1. Dibutuhkan riset lebih lanjut terhadap prediksi karakter pemain/AI yang ada. Karena jika hanya menggunakan pengetahuan *heuristic, reward and punishment* saja tidak cukup.
2. Diperlukan penanaman *knowledge* ke *database AAI* seiring dengan seringnya bertemu dengan lawan yang sama.
3. Untuk memperluas hasil pengujian, gunakan versi AI Sandbox yang lebih lama, atau gunakan framework lain untuk menguji AI ini.

DAFTAR PUSTAKA

[1] Alexander Nareyek , *Game AI Is Dead. Long Live Game AI!*: National Univesity of Singapore, 2007.

[2] J Hagelback and S.J. Johansson , *Measuring player experience on runtime dynamic difficulty scaling in an RTS game*., 2009.

[3] Schaeffer J, "'A Gamut of Games'," *Artificial Intelligence Magazine 22(3)*, pp. 29–46, 2001.

[4] Haomiao Huang and Jerry Ding, *A Differential Game Approach to Planning in Adversarial Scenarios: A*. Berkeley: NASA (NNA06CN22A) AFOSR, 2011.

[5] J van Waveren and J Rothkrantz, "Artificial player for Quake III arena," *International Journal of Intelligent Games & Simulation 1*, 2003.

[6] Marc S Atkin , David L Westbrook , and Paul R Cohen , *Capture the Flag: Military Simulation Meets Computer Games*.: IN PROCEEDINGS OF AAAI SPRING SYMPOSIUM SERIES ON AI AND COMPUTER GAMES, 1999.

[7] Bijan Fazlollahi and Mihir A Parikh , *Adaptive Decision Support Systems*.: Georgia State University, 1997.

[8] A Ram , S Ontanon , and M Mehta , *Artificial Intelligence for Adaptive Computer Games*., 2007.

[9] P Spronck and D.W Aha , *Automatically Acquiring Domain Knowledge For Adaptive Game AI Using*

Evolutionary Learning., 2005.