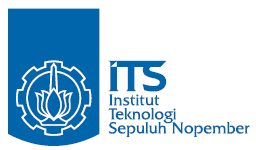
**JURUSAN TEKNIK INFORMATIKA**



**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

**IDENTITAS PENGUSUL**

Nama : Mubdi Nur Hafidh

NRP : 5105100086

Dosen Wali : Sarwosri, S.Kom, MT

**JUDUL TUGAS AKHIR**

**Penerapan Adaptive Artificial Intelligence pada permainan Real Time Strategy Section 52 menggunakan Silverlight**

**PENDAHULUAN**

**LATAR BELAKANG**

Dalam permainan RTS (Real Time Strategy) terdapat AI yang menggerakkan suatu unit, tugas AI ini mengatur strategy untuk memberikan perlawan kepada pemain. Selain pengaturan strategy AI juga mengatur jalannya pergerakan suatu unit dari suatu titik menuju ke titik tujan. Suatu unit dalam melakukan pergerakan menuju titik tujuan akan melakukan *pathfinding* (pencarian jalur) sehingga unit tersebut mampu mencapai titik tujuan dengan benar dan efisien.

Untuk permainan yang kompleks seperti permainan strategy penggabungan AI sulit untuk dilakukan dikarenakan banyaknya pilihan ditiap putaran permainannya. Untuk mengimplementasikan game AI pada permainan yang kompleks, kebanyakan developer menggunakan cara *script*, yaitu list dari rule-rule yang ada dijalankan secara berurutan. *Script* merupakan teknik pilihan pada industri permainan untuk mengimplementasikan game AI, dikarenakan sifatnya yang dapat dimengerti, dapat diprediksi, mampu beradaptasi terhadap kondisi tertentu, mudah di implementasikan, dikembangkan dengan mudah, dan dapat digunakan oleh non-programer.

*Script* biasanya statis dan cenderung panjang serta kompleks, yang mengakibatkan dua masalah. Pertama karena kompleksitasnya, *script* kemungkinan akan mengandung kelemahan yang dapat dimanfaatkan oleh pemain-manusia untuk mengalahkan musuh yang seharusnya memiliki tingkat kesulitan yang tinggi dengan mudah. Kedua karena sifatnya yang statis, *script* tidak dapat menangani strategy pemain-manusia yang tidak tampak, dan tidak dapat mengukur tingkat kesuliatan yang disajikan oleh AI permainan kepada pemain-manusia baik yang masih pemula dan yang mahir.

AI yang terlalu kuat atau terlalu lemah dapat membuat suatu permainan strategy menjadi tidak menarik. Padahal salah satu nilai hiburan pada sebuah permainan yang dapat membuatnya menjadi menarik adalah perlawanan yang diberikan oleh AI dari unit musuh yang digerakkan oleh komputer. Semakin seimbang perlawanan yang diberikan oleh AI akan membuat pemain merasa tertarik untuk memainkan.

Tugas akhir ini membahas mengenai implementasi Adaptive AI permainan dalam sistem pertempuran pada permainan RTS Section 52 untuk mengendalikan pergerakan unit yang dikendalikan oleh komputer. Sehingga unit yang dikendalikan oleh komputer dapat menyesuaikan diri susuai dengan keadaan pertempuran yang sedang berlangsung saat itu. Karena AI yang dapat berkembang sesuai cara bermain pemainnya dapat membuat suatu permainan menjadi menarik. Dalam implementasinya Section 52 akan menggunakan *silverlight* sebagai *framework* pembuatan permainan. *Silverlight* merupakan *framework* aplikasi untuk menulis dan menjalankan *rich internet application*, dengan tujuan dan fitur yang sama seperti *flash*.

*Silverlight* sendiri dipilih sebagai *framework* pembuatan permainan karena permainan berbasis web sudah mulai berkembang dan memiliki peminatnya tersendiri serta masih jarangnya *silverlight* digunakan untuk pembuatan permainan. Selain itu nantinya diharapkan dapat megeksplorasi kemampuan-kemampuan yang dimiliki oleh *silverlight* sehingga mampu membuat permainan berbasis web yang menarik.

**RUMUSAN MASALAH**

Permasalahan yang diangkatdalam tugas akhir ini adalah sebagai berikut:

* Bagaimana merancang dan mengaplikasikan *Adaptive game AI* pada tiap unit agar dapat mengambil keputusan saat akan melakukan action.
* Bagaimana menerapkan *pathfinding* pada pergerakan unit dari suatu titik menuju titik tujuan.
* Bagaimana membuat *battle system* yang bisa digunakan dalam *Real Time Strategy*
* Bagaimana membuat *battle system* tersebut bisa di uji tersendiri dan bisa berjalan dengan baik sewaktu digabung dengan modul utama.

**BATASAN MASALAH**

Ruang lingkup permasalahan yang dibahas dalam Tugas Akhir ini adalah sebagai berikut:

* Menggunakan A\* untuk *pathfinding*.
* Implementasi menggunakan *Silverlight 3*.
* Menggunakan *Dynamic Scripting* untuk *Adaptive game AI*.

**TUJUAN**

Tugas akhir ini memiliki tujuan sebagai berikut:

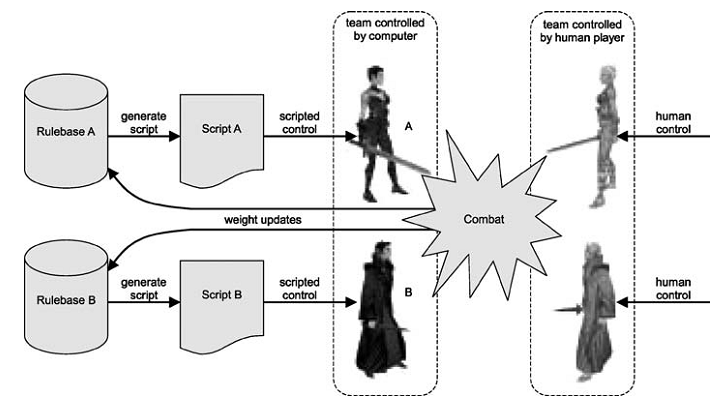
* Mengimplementasikan *Adaptive game AI* pada tiap unit sehingga dapat menyusun strategy sesuai situasi yang terjadi saat itu.
* Membuat tiap unit dapat bergerak menuju titik yang telah ditentukan dengan benar.
* Membuat *battle system* RTS yang sesuai untuk permainan Section 52.
* Membuat modul *battle system* tersebut dapat diuji secara terpisah dan bisa berjalan baik sewaktu digabung dengan modul utama.

**RINGKASAN**

*Adaptive AI* permainan merupakan *AI* permainan yang dapat berkembang atau berubah – ubah sesuai dengan kondisi/situasi yang sedang terjadi, sehingga nantinya dapat memberikan perlawanan yang seimbang sesuai dengan pemainnya. Untuk menerapkan adaptive AI permainan dapat dilakukan dengan beberapa teknik, salah satunya dengan *Dynamic Scripting*.

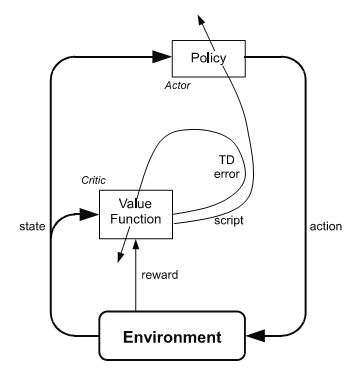
*Dynamic scripting* merupakan sebuah teknik untuk memperoleh pembelajaran secara online oleh unit yang digerakkan komputer. Dalam *dynamic scripting*, *script* dibuat secara online, yaitu selama permainan berlangsung, berdasarkan *rule-rule* yang diekstrak dari sebuah *rulebase*. Pada teknik ini proses adaptasi berjalan dengan pemberian *reward* atau pemberian *penalty* suatu *rule-rule* sesuai dengan peranannya terhadap hasil yang diperoleh.[1]

*Dynamic scripting* mengatur beberapa *rulebase*, dimana tiap unit memiliki sebuah *rulebase* tersendiri. Setiap sebuah unit baru dihasilkan, *rulebase-rulebase* digunakan untuk membuat *script* baru yang mengendalikan tindakan unit tersebut. *Rule-rule* yang meliputi sebuah *script* *controlling* sebuah unit tertentu diekstrak dari *rulebase* yang sesuai dengan unit tersebut. Kemungkinan suatu rule dipilih untuk sebuah *script* dipengaruhi oleh nilai *weight* yang tercantum pada tiap rule. Tujuan *dynamic scripting* ini adalah untuk mengadaptasi *weight* pada *rulebase* sehingga kesesuaian perilaku yang diharapkan didefinisikan oleh *script* yang dihasilkannya meningkat secara terus menerus, walaupun dalam situasi yang berubah-ubah. Kesesuaian biasanya didefinisikan sebagai kemungkinan suatu tim yang merupakan asal dari suatu unit memenangkan permainan. Adaptasi suatu *rulebase* berlangsung dengan mengubah nilai *weight* untuk menggambarkan peluang keberhasilan atau kegagalan suatu *rule* yang terkait dalam *script*. Perubahan *weight* sendiri ditentukan oleh *weight-update function*.[1]



Gambar 1. Ilustrasi Dynamic Sripting [2]

Gambar 1 menjelaskan mengenai terjadinya sebuah pertempuran menggunakan *dynamic scripting*. Diawal sebuah pertempuran, sebuah *script* baru dihasilkan untuk tiap unit yang dikendalikan komputer, secara acak memilih sejumlah *rule-rule* dari *rulebase* yang berkaitan. *Dynamic scripting* menggunakan sebuah mekanisme pemilihan, yaitu terdapat relasi linier antara kemungkinan suatu rule dipilih dan *weight*-nya yang berkaitan. Urutan *rule* mana yang ditempatkan pada *script* bergantung pada bagian aplikasi. Sebuah mekanisme *priority* dapat digunakan agar suatu *rule* diutamakan daripada *rule* lainnya. Mekanisme yang seperti ini hanya dibutuhkan bila urutan *rule-rule* dan *action* didiskripsikan terlebih dahulu oleh pengetahuan bagian.[1]



Gambar 2. Arsitektur umum Dynamic scripting [2]

Gambar 2 menjelaskan mengenai proses belajar dalam *dymanic scripting*. Dalam penyelesaian suatu pertempuran, *weight-weight* dari *rule-rule* digunakan selama pertempuran diadaptasi sesuai kontribusinya terhadap hasil yang diperoleh. *Rule-rule* yang mengantarkan pada keberhasilan diberi *reward* dengan penambahan *weight*, seedangkan *rule-rule* yang mengantarkan pada kegagalan diberi *penalty* dengan pengurangan *weight*. Penambahan atau pengurangan setiap *weight* dikompensasikan untuk menjaga total *weight* konstan dengan cara mengurangi atau menambah semua sisa *weight*. Pendistribusian *weight-weight* inilah yang merupakan fitur penting dari *dynamic scripting*, karena membuat semua *rule* pada *rulebase* belajar di tiap *update*.[1]

Mulanya, *dynamic scripting* digunakan untuk membuat *script* pada tim musuh dalam *computer role-playing games* (CRPG). Dikarenakan adanya perbedaan antara *script* pada permainan CRPG dan RTS, dilakukanlah sbuah modifikasi terhadap algoritma *dynamic scripting* untuk sebuah permainan RTS yang pada intinya mengenai dua aspek sebagai berikut:

1. Pada CRPG *rule-rule* dibedakan berdasarkan perbedaan tipe musuh (misalnya: *warrior, wizzard*, dan sebagainya), pada RTS *rule-rule* dibedakan berdasarkan perbedaan *game states* (kondisi permainan yang berlangsung).
2. *Rule-rule* untuk AI musuh diadaptasi saat sebuah terjadi perubahan *state* dan *rule-rule* dievaluasi berdasarkan *fitness* untuk *state* yang sebelumnya dan *fitness* untuk keseluruahn permainan. [2]

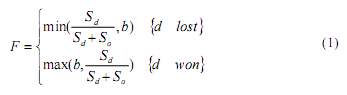


Gambar 3. Arsitektur umum Permainan Section 52

Gambar 3 menjelaskan mengenai arsitektur umum permainan section 52 dimana permainan dapat dilakukan secara multiplayer. Perhitungan AI dilakukan terpusat pada server permainan, sehingga client tidak perlu melakukan perhitungan cukup dengan mengirimkan data yang dibutuhkan dalam perhitungan saja maka server yang akan melakukan perhitungan AI dan mengirimkan hasil perhitungan kepada client. Perhitungan yang dilakukan meliputi pengecekan state menggunakan persamaan (1) dan persamaan (2). Setelah pengecekan state dilakukan akan ditentukan keputusan pergerakan yang akan diambil, setelah itu dilakukan pemberian skor menggunakan persamaan (3). Selanjutnya melakukan update weight yang nantinya digunakan dalam mengubah state dan membantu pengambilan keputusan menggunakan persamaan (4). Selain itu saat terjadi pergerakan unit dari suatu titik menuju titik tujuan akan dilakukan perhitungan pathfinding menggunakan persamaan (5).

Berikut beberapa fungsi yang digunakan dalam *dynamic scripting*.

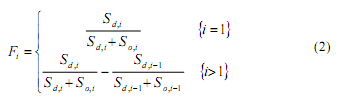
*Overall fitness function* *F* untuk pemain *d* yang dikendalikan *dynamic scripting* (untuk menyingkat disebut “pemain dinamis”) memiliki sebuah nilai dalam range [0,1]. Didefinisikan sebagai berikut:



Dalam persamaan (1), *Sd* merepresentasikan skor untuk pemain dinamis, *So* merepresentasikan skor untuk pemain dinamis milik musuh, dan *b* [0,1] merupakan *break-event point*. Pada *break event point,* *weight* tidak berubah.[2]



Untuk pemain dinamis, *state* *fitness* *Fi* untuk *state* *i* didefinisikan sebagai berikut:



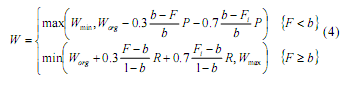
Dalam persamaan (2), *Sd,x* merepresentasikan skor pemain dinamis setelah state *x* dan *So,x* merepresentasikan skor pemain dinamis milik musuh setelah state *x*.[2]

*Score function* merupakan *domain-dependent*, dan harus mampu menggambarkan kekuatan relatif dari dua pemain yang berhadapan dalam permainan. Didefinisikan skor *Sx* untuk pemain *x* sebagai berikut:

*Sx* = 0.7*Mx* + 0.3*Bx* (3)

Dalam persamaan (3), *Mx* merepresentasikan *military point* untuk pemain *x*, yaitu banyaknya point yang diberikan untuk membunuh unit dan mengancurkan bangunan. *Bx* merepresentasikan *building point* untuk pemain *x*, yaitu banyaknya point yang diberikan untuk membuat pasukan dan membangun bangunan.[2]

*Weight-update function* mengubah *fitness function* menjadi *weight adaptation* untuk tiap *rule* dalam *script*. *Weigh–update function* *W* untuk pemain dinamis didefinisikan sebagai berikut:



Dalam persamaan (4), *W* merupakan nilai weight yang baru, *Worg* merupakan nilai weight asal, *P* merupakan *penalty* maksimum, *R* merupakan *reward* maksimum, *Wmax* merupakan nilai *weight* maksimum, *Wmin*merupakan nilai *weight* minimum, *F* merupakan *overall fitness* dari pemain dinamis, *Fi*merupakan *state fitness* untuk pemain dinamis pada *state i*, dan *b* merupakan *break-event point*.[2]

*Pathfinding* merupakan metode pencarian jalur(*plotting*)yang dilakuakn oleh aplikasi komputer. Dalam sebuah game metode *pathfinding* yang sering digunakan adalah A\*(dibaca “*A star*”). A\* menggunakan sebuah best-first search dan mencari jalur yang least-cost dari initial node menuju goal node. A\* menggunakan sebuah fungsi *distance-plus-cost heuristic* (biasanya dinotasikan dengan *f(x)*) untuk menentukan *node-node* yang dilalui dalam *tree*.[3]

Fungsi distance-plus-cost heuristic merupakan jumlah dari dua fungsi berikut:

* Fungsi path-cost, yang merupakan cost dari starting-node ke current (biasanya dinotasikan dengan g*(x)*).
* Dan sebuah perkiraan heuristic ke goal (tujuan )yang dapat diterima (biasanya dinotasikan dengan *h(x)*).

Sehingga diperoleh persamaan *f(x)* = *g(x)* + *h(x)* (5)

**METODOLOGI**

Metodologi yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

1. Studi Literatur

Tahap dimana pencarian dan pemahaman literatur, teori-teori yang berkaitan dengan rumusan masalah yang bisa digunakan untuk pembuatan permainan.

1. Perancangan perangkat lunak

Tahap dimana dilakukan perancangan terhadap hasil analisis dari studi literatur yang telah dilakukan.

1. Pembuatan perangkat lunak

Pada tahap ini dilakukan implementasi dari rancangan yang telah dibuat menjadi perangkat lunak.

1. Uji coba dan evlauasi

Tahap untuk melakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui masalah - masalah yang muncul dan melakukan perbaikan apabila hasil yang dinilai tidak sesuai dengan tujuan.

1. Penyusunan laporan tugas akhir

Tahap akhir yaitu menyusun laporan yang berisi dokumentasi mulai dari dasar teori, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

Sistematika laporannya sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi latar belakang masalah, tujuan pembuatan

tugas akhir, permasalahan, batasan masalah, metodologi

yang digunakan dan sistematika penyusunan tugas akhir.

BAB II. TINJAUAN PUSTAKA

Bab ini membahas beberapa teori penunjang yang

berhubungan dengan pokok pembahasan dan mendasari

pembuatan tugas akhir ini.

BAB III. METODOLOGI PENGEMBANGAN

PERANGKAT LUNAK

Bab ini membahas secara detail tentang analisis sistem,

desain dan perancangan peragkat lunak serta implementasi

perangkat lunak dan disertai dengan potongan source code

yang penting dalam sistem tersebut.

BAB IV. UJI COBA DAN EVALUASI

Bab ini membahas uji coba dari aplikasi yang dibuat

dengan melihat output yang dihasilkan oleh aplikasi, dan

evaluasi untuk mengetahui kemampuan aplikasi.

BAB V. PENUTUP

Bab ini berisi kesimpulan dari hasil uji coba yang

dilakukan serta saran untuk pengembangan aplikasi

selanjutnya.

**DAFTAR ACUAN**

[1] Ponsen Marc, Spronck Pieter, Kuyper Ida Sprinkhuizen, Postma Eric “**Adaptive game AI with dynamic scripting**”, SpringerScience

[2] Ponsen Marc, Spronck Pieter, “**IMPROVING ADAPTIVE GAME AI WITH EVOLUTIONARY LEARNING**”, Lehigh University, USA, Maastricht University, The Netherlands

[3] Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "**A Formal Basis for the Heuristic Determination of Minimum Cost Paths**"

**JADWAL PEMBUATAN TUGAS AKHIR**

Tugas akhir ini dikerjakan menurut jadwal sebagai berikut:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Tahapan | Bulan | | | | | |
| Maret | April | | Mei | | Juni |
| 1 | Studi literatur |  |  |  |  |  |  |
| 2 | Analisis dan Desain Perangkat Lunak |  |  |  |  |  |  |
| 3 | Implementasi Perangkat Lunak |  |  |  |  |  |  |
| 4 | Uji Coba dan Evaluasi |  |  |  |  |  |  |
| 5 | Penyusunan Buku Tugas Akhir |  |  |  |  |  |  |

**LEMBAR PENGESAHAN**

Surabaya, 06 Maret 2012

Menyetujui,

Pembimbing 1 Pembimbing 2

Imam Kuswardayan, S.Kom, MT Sarwosri, S.Kom, MT

NIP: 132306543 NIP: 132297166