Game Real-Time Strategy dengan menggunakan *Artificial Intelligence Quantified Judgement Model* dan *Backpropagation Neural Network*

Owen Ener[1], Gregorius Satia Bhudi M.T.[2], Liliana M. Eng.[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) -8417658
m26412081@john.petra.ac.id[1], greg@petra.ac.id[2], lilian@petra.ac.id[3]

ABSTRAK

Game Real-Time Strategy (RTS) merupakan sebuah *genre video game* yang cukup populer. Keunikan dari RTS adalah sebuah *game* strategi dimana waktu terus berjalan untuk semua pemain, Ini membuat situasi dimana *player* harus menentukan strategi dalam hitungan detik. Untuk mendapatkan sebuah permainan yang baik, maka diperlukan lawan untuk *player*. Cara yang bisa digunakan untuk membuat lawan kepada *player* adalah dengan membentuk suatu AI yang bisa mempertimbangkan *mechanic* dari *game* yang ada. Skripsi ini dibuat untuk mengembangkan pemikiran dalam AI *game* RTS.

Game akan dibuat dengan *Unity Game Engine* 5.1.2f1. AI yang akan diimplementasikan adalah *Quantified Judgement Model* dan *Neural Network Backpropagation*. *Quantified Judgement Model* akan bertindak sebagai *Abstract Controller*, memberikan perintah kepada pasukannnya seperti sebuah general perang. *Neural Network Backpropagation* akan digunakan untuk *Virtual Controller*, dimana AI akan bertindak untuk setiap pasukan mempertimbangkan apa yang harus mereka lakukan. Apakah mereka harus mundur, atau terus maju mengikuti perintah atasan.

Kata Kunci: *RTS, Real-Time Strategy Game, Artificial Intelligence, Quantified Judgement Model, Neural Network Backpropagation.*

ABSTRACT

Real-Time Strategy (RTS) Game is a quite popular video game genre. The uniqueness of RTS Games is that it is a Strategy Game where time will still continue for all the players. This creates situations where the player must determine their strategies in a matter of seconds. To get a good gameplay experience, then we would need enemies for the player. The way we could do that is to create an AI that could take into account the mechanics of the game. This thesis is aimed to broaden our knowledge on how to develop AI for RTS games.

The game will be created in Unity Game Engine 5.1.2f1, where the AI that is going to be implemented is the Quantified Judgement Model and the Neural Network backpropagation. The Quantified Judgement Model will act as Abstract Controller, giving orders to his troops much like a general in a war. Neural Network Backpropagation will be used for the Virtual Character, where the AI will act for each of the troops to think on what they should do. Whether they have to fall back, or keep going forward according to the orders given to him.

Keywords: *Real-Time Strategy Game, Artificial Intelligence, Quantified Judgement Model, Neural Network Backpropagation.*

1. INTRODUCTION

Didalam *Genre* Strategy terdapat *sub-genre Turn-Based Strategy* (TBS) dan *Real-Time Strategy* (RTS). RTS adalah *game* Strategy dimana seluruh tindakan yang dilakukan oleh semua *player* dan semua unit yang berada di dalam *game* tersebut berjalan secara *real-time*.

Untuk mendapatkan sebuah *gameplay* yang diperlukan untuk *game* RTS, maka dibutuhkan pemain kedua untuk melawan pemain pertama. Tetapi, ada beberapa orang yang tidak mau bermain melawan orang lain, atau tidak adanya teman untuk bermain secara langsung. Masalah ini yang membuat dibutuhkannya sebuah *Artificial Intelligence* (AI) untuk menjadi pemain kedua tersebut.

AI dalam Strategy *game* terdapat dua jenis, *Abstract Controller* dan *Virtual Controller*. *Abstract Controller* bertugas untuk mengontrol apa yang akan dilakukan dari pemain kedua yang menjadi lawan dari pemain game tersebut [9]. Sementara Virtual Controller bertugas untuk mengontrol setiap pergerakan dari pasukan yang ada, entah itu berada di pihak musuh ataupun pihak pemain *game* tersebut.

AI *Abstract Controller* dari game ini akan dibuat dengan menggunakan *Quantified Judgement Model*, sementara *Virtual Controller* akan diimplementasikan dengan *Backpropagation Neural Network.*

2. LANDASAN TEORI

2.1 Game

2.1.1 *Video Game*

Virtual environment seperti *video game*, adalah sebuah bentuk dunia buatan yang mensimulasikan keberadaan fisik dari sebuah *user* didalam sebuah tempat yang menirukan dunia nyata, atau merepresentasikan suatu dunia imaginer [12].

Video game telah menjadi bagian yang besar terhadap kehidupan banyak orang. Berdasarkan informasi dari Entertainment Software Association, 80% rumah tangga di United States memiliki device untuk bermain *video game* dan 42% dari orang Amerika memainkan *video game* tiga jam atau lebih setiap minggunya [6].

2.1.2 *Real-Time strategy Game*

Secara umum, setiap match dalam RTS *game* memiliki dua *player* yang bermulai dengan beberapa unit atau/dan gedung di beberapa lokasi disebuah *two-dimensional terrain* (*map*) [9].

2.2 *Artificial Intelligence*

Menurut Abdennour El Rhalibi, *Artificial Intelligence*, didalam *game* komputer mencakup perilaku dan proses pengambilan

keputusan dari sebuah *game-playing opponents* (dikenal sebagai *non-player character* atau NPC) [7].

2.2.1 *Artificial Intelligence* dalam *Real Time Strategy Game*

Riset dari Artificial Intelligence (AI) telah bergerak arah didalam beberapa tahun terakhir dari Catur dan Go kepada real-time strategy (RTS) game, sebagai test-bed untuk mempelajari tindakan kompleks [4].

Sementara komunitas *machine-learning* dan AI lebih berfokus kepada *real-time strategy game* sebagai *domain of interest* untuk beberapa tahun, sejauh pengetahuan kita, tidak ada satupun riset yang berusaha untuk menjelaskan bagaimana orang mendeskripsikan strategi mereka.

2.2.2 *Quantified Judgement Model*

In Understanding War, Dupuy (1987) memberikan sebuah model matematis disebut sebagai Quantified Judgement Model (QJM) [1].

Ada juga beberapa penambahan yang dapat dilakukan untuk mempertimbangkan beberapa aspek lebih dari QJM dengan memasukan *Theoretical Lethality Index* (TLI) kedalam penghitungan *Power* [2] pada Persamaan 2.1.

Power=Quantity×Quality×TLI×EffectivenessFortification (2.1)

TLI dapat dihitung dengan menggunakan Persamaan 2.2:

$$TLI=RF\times C\times RN\times A\times R \quad (2.2)$$

Dimana:

RF = Rate of Fire setiap menitnya.

C = Target yang bisa mengenai sasaran pada setiap serangan. Sebuah misil anti-tank atau AP round akan memiliki Target 1 melawan armor; sementara HE Fragmentation shell akan memiliki Target 10 melawan pasukan jalan kaki.

RN = Jarak efektif dari senjata tersebut dalam meter.

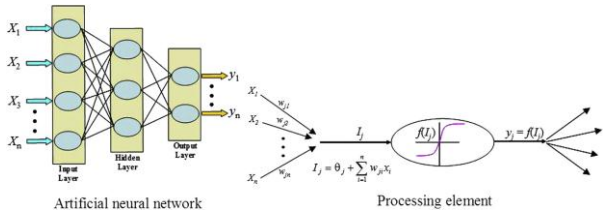
A = Prosentase senjata tersebut mengenai sasaran.

R = Prosentase senjata tersebut bisa menembak.

2.2.3 *Artificial Neural Network*

Neural Network adalah sebuah prosesor yang didistribusikan secara pararel dalam skala masal, dimana dia memiliki kecenderungan untuk menyimpan sebuah pengetahuan pengalaman dan membuatnya tersedia untuk digunakan[4].

Umumnya arsitektur dari ANN memiliki sederet processing elements (PE), atau nodes, yang diatur dalam bentuk layers: input layer, output layer, dan bisa satu atau lebih hidden layers [11].



Gambar 1. Bentuk umum dari sebuah Artificial Neural Network

Untuk *Feed-Forward Neural Network*, *output* dari setiap *neuron* dari setiap layer akan diteruskan maju ke layer yang berada di level selanjutnya, hingga seluruh *output* dari *network* didapatkan [3].

2.2.4 *Backpropagation*

Backpropagation menggunakan teknik *gradient descent* untuk menyesuaikan parameter network yang terbaik dari sebuah training set dari suatu input-output pairs [7].

Backpropagation learning algorithm adalah sebuah *iterative gradient descent algorithm* untuk mengoptimisasi *link weights* untuk meminimalkan *mean squared error* diantara output sesungguhnya dan output ideal. [13].

Backpropagation learning algorithm terdiri dari tiga bagian: *the forwards propagation* dari *input signals*, *the backward propagation* dari *errors*, dan *weight updating*. Dengan menyebarkan *input signals* dengan arah maju, *the backpropagation learning algorithm* pertama mengkalkulasikan output dari setiap *input* vektor dari *training data set*. Lalu, dia mengumpulkan *weight modification* untuk setiap *input vector*. Untuk mengkalkulasikan *weight modification* dari sebuah link tidak terkoneksi dengan *neuron* di *output layer*, *error* dari *output neuron* disebarkan kebelakang dari *output layer* kepada *input layer*.

3. DESIGN SISTEM

Permainan akan dibagi menjadi dua bentuk: *Preparation Phase* dan *Combat Phase*. Preparation Phase berjalan sebelum *Combat Phase.*

Di dalam *Preparation Phase*, *player* akan berjalan bergantian dengan lawannya untuk menentukan apakah yang akan dibawa masuk kedalam *Combat Phase*. Pada fase ini, *player* akan bermain kepintaran melawan musuhnya, dengan berusaha membawa pasukan yang lebih baik daripada musuhnya.

Combat Phase akan berjalan setelah player menekan tombol *ready*, dimana seluruh unit yang telah dipilih pada dapat digunakan untuk mencapai kemenangan. Pada fase ini *gameplay* berjalan secara *real-time* dan semua tindakan musuh dan player akan berjalan secara bersama-sama.Tujuan dari *player* disini hanya untuk menghancurkan musuhnya hingga tidak ada yang bisa melawan *player*.

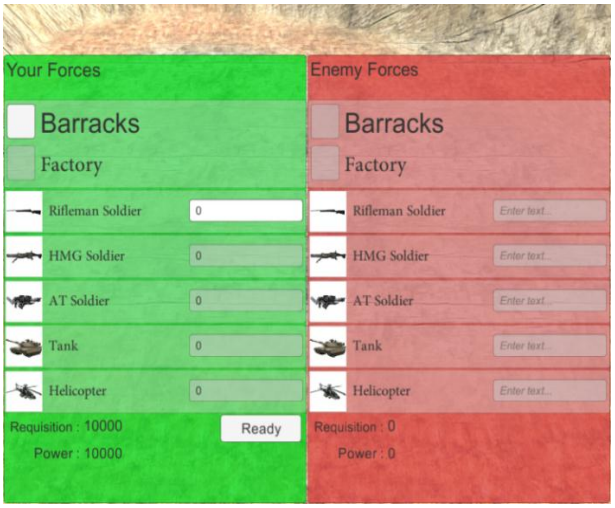
Player akan mengontrol berbagai unit yang telah ditentukan untuk menghancurkan semua unit dari musuhnya. Setiap dari unit tersebut memiliki karakteristik yang berbeda dari berbagai atribut, yaitu: *Health*, *damage*, *rate of fire*, *range*, *movement speed*, *price*, *power*, dan *accuracy*.

Player akan dianggap menang ketika dia berhasil menghancurkan semua unit dan bangunan dari musuhnya. Sebaliknya, jika player kehilangan seluruh unit dan bangunan karena dihancurkan oleh musuhnya, maka player akan dianggap kalah.

AI yang akan digunakan adalah *Quantified Judgement Model* dan *Backpropagation Neural Network*. *Quantified Judgement Model* akan digunakan dalam penentuan strategi dari seluruh unit yang digunakan oleh AI tersebut. *Backpropagation Neural Network* akan digunakan untuk menentukan tindakan yang akan dilakukan oleh setiap unit yang berada dalam map, apakah mereka akan bergerak mengikuti perintah dari player dan AI *Quantified Judgement Model* atau mereka akan melawan perintah dan berusaha mempertahankan diri mereka dengan mundur ke markas mereka.

4. PENGUJIAN PROGRAM

Ketika *player* menjalankan program, maka player akan masuk kedalam *Preparation Phase*. Pada fase ini, *player* akan menentukan jumlah pasukan yang akan dibawa kepada pertarungannya, seperti yang terlihat pada Gambar 2.



Gambar 2. Preparation Phase

Pada Gambar 2, Beberapa Unit tidak dapat dimasukan berapa jumlahnya yang diinginkan, karena diperlukannya gedung yang memproduksi Unit tersebut. HMG dan AT *Soldier* memerlukan Barracks dicentang, sementara Tank dan Helicopter memerlukan Factory untuk dicentang. Setelah gedung yang diperlukan dicentang, maka Unit yang diproduksi oleh gedung tersebut dapat dibawa kepada pertarungan, seperti yang terlihat pada Gambar 3.

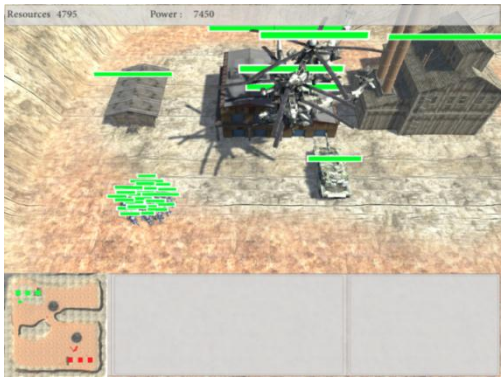
Gambar 3. Preparation Phase dengan Barrack dan Factory dicentang

Disini *player* dapat mengisi jumlah yang diinginkan kepada bagian yang berwarna putih. Ketika *player* selesai memilih, maka *player* dapat menekan tombol *Ready* dan memberikan musuhnya untuk memilih pasukannya. Pada posisi ini, musuh player, yang dikontrol oleh AI *Quantified Judgement Model* akan menentukan pasukannya yang dilawan oleh pasukan *player*.

Gambar 4. Preparation Phase setelah player menekan tombol Ready

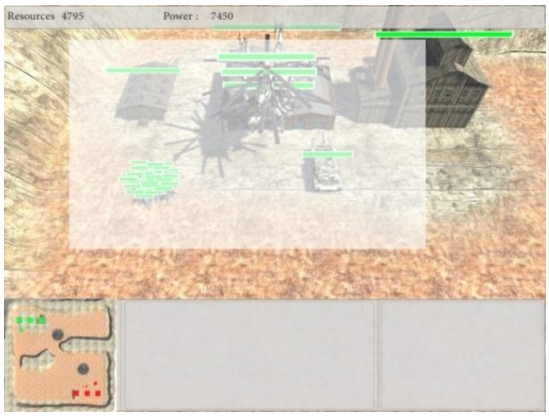
Pada Gambar 4 terlihat bahwa pasukan musuh yang berada pada kotak merah telah terisi. Pada posisi ini, *player* dapat melakukan perubahan untuk melawan kembali pasukan musuhnya. Setelah *player* selesai melakukan perubahan, maka player dapat menekan *Ready* untuk masuk ke *Combat Phase*.

Combat Phase dimulai dengan player melihat pasukannya yang telah dipilih pada *Preparation Phase*. Gambar 5 menunjukan posisi awal dari *Combat Phase* dengan semua pasukan yang telah ditentukan pada Gambar 4.



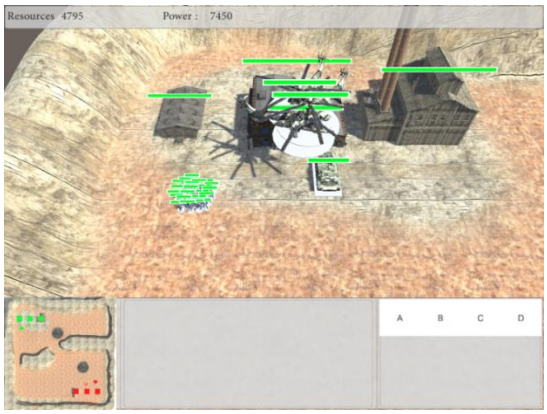
Gambar 5. *Combat Phase* pada posisi awal

Pada fase ini, *player* akan menggerakan Unit-nya untuk menghancurkan seluruh pasukan dan bangunan musuh. *Player* dapat menggerakan Unit-nya dengan memilih Unit yang diinginkan dengan left-drag kepada seluruh unit yang ada, seperti yang terlihat pada Gambar 6.



Gambar 6. Player melakukan pemilihan untuk pasukannya

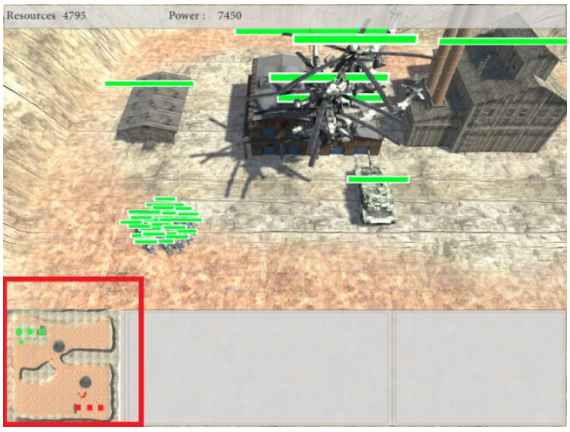
Unit yang telah dipilih akan memiliki notifikasi berwarna putih disekitarnya, seperti yang terlihat pada Gambar 7.



Gambar 7. Kondisi setelah player selesai memilih

Player dapat menggerakkan Unitnya ke posisi yang diinginkan dengan cara melakukan right-click kepada lokasi yang diinginkan. Seperti yang terlihat pada Gambar 8.

Unit yang dimiliki *player* berwarna hijau, sementara Unit yang dimiliki musuh *player* yang dikontrol AI memiliki warna merah. *Player* dapat mengetahui lokasi musuh melalui Minimap, lokasi minimap ditunjukan pada Gambar 8.

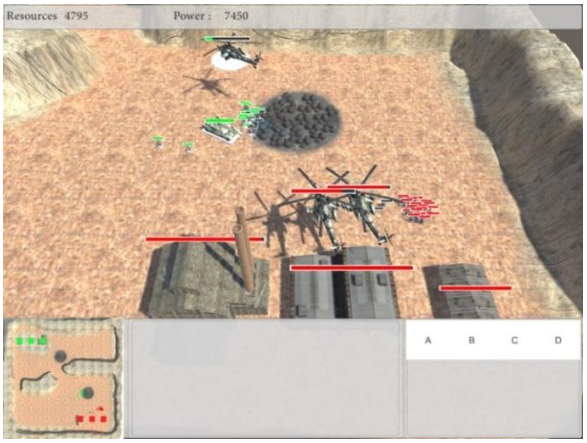


Gambar 8. Lokasi Minimap

Player dapat menyuruh perintah kepada Unit sesuai strategi yang telah dipikiran oleh *player*. Gambar 9 menunjukan pergerakan dari unit *player* (yang berwarna hijau) mendekati unit lawan (yang berwarna merah).

Gambar 9. Unit player bergerak mendekati musuhnya

Ketika Unit mendekati musuhnya maka Unit akan mulai menembak Unit musuhnya. Seperti yang terlihat pada Gambar 10.



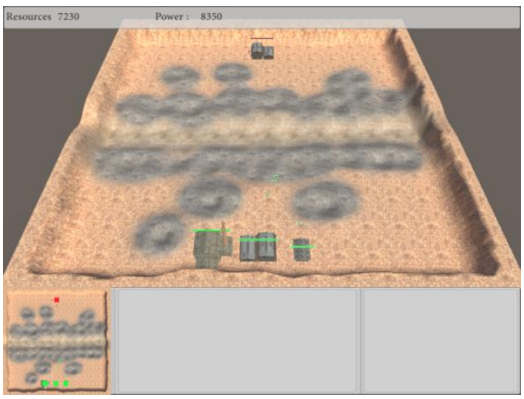
Gambar 10. *Unit* player menyerang lawannya

Ketika musuh ataupun unit *player* ditembak, maka *Health* Unit yang ditembak berkurang.

Ketika *Health* sebuah Unit kosong, maka Unit tersebut akan mati, atau hancur, seperti yang terlihat pada Gambar 10 disebelah kanan.

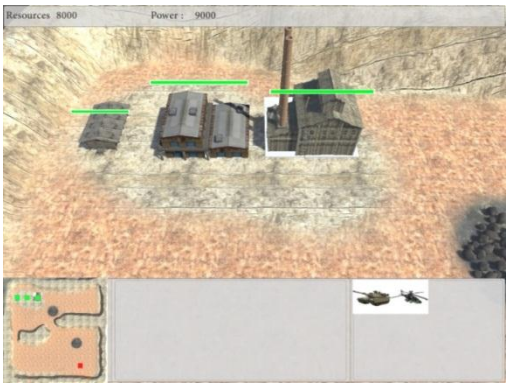
Gambar 11. Berbagai kondisi Health (kiri - belum kosong, kanan – kosong)

Player akan melawan musuhnya hingga semua gedung musuh dihancurkan. Ketika *player* sudah berhasil menghancurkan seluruh pasukan musuh, maka *player* masuk ke *level* berikutnya. Gambar 11 menunjukan *level* berikut jika player telah mengalahkan musuhnya.



Gambar 12. Level berikut jika player mengalahkan musuhnya

Player dapat menambahkan jumlah pasukannya dengan melakukan *left-click* kepada sebuah Gedung, seperti yang terlihat pada Gambar 12.



Gambar 13. Player memilih sebuah gedung untuk membuat unit baru

Pada pojok kanan bawah, ditunjukan Unit apa saja yang dapat dibuat oleh gedung tersebut. *Player* dapat membuat Unit yang diinginkan dengan menekan gambar dari unit yang diinginkan. Setelah lewat beberapa waktu, Unit yang telah dipilih akan keluar dari gedung dan dapat digunakan untuk melawan musuhnya

Didalam map tersedia *Heavy Cover*. Jika Unit berada pada posisi *Heavy Cover*, maka semua damage yang diterima oleh Unit tersebut akan berkurang sejumlah 46%. *Heavy Cover* dimarkai dengan bentuk *obstacle* (bebatuan) atau bentuk kawah seperti yang terlihat pada Gambar 14.

Gambar 14. Jenis Heavy Cover (kiri – bebatuan, kanan – kawah)

Kondisi jika ditembak diluar *Heavy Cover* tergambarkan pada Gambar 15. Tetapi, Jika Unit berada pada sebuah *Heavy Cover*, maka akan terlihat seperti Gambar 16. Kedua Gambar 15 dan Gambar 16 adalah kondisi yang terlihat ketika jenis Unit yang sama, yaitu Rifleman, diserang oleh Unit AT.

Gambar 15. Damage yang terlihat jika diluar Heavy Cover

Gambar 16. Damage yang terlihat jika dalam Heavy Cover

5. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian sistem yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

- Teori *Quantified Judgement Model* dan *Neural Network Backpropagation* dapat diimplementasikan kedalam sebuah *game* RTS.
- *Quantified Judgement Model* memerlukan beberapa perubahan dalam perhitungan TLI untuk mengakomodir atribut yang dimiliki oleh *game* sebelum bisa digunakan.

6. DAFTAR PUSTAKA

[1] Desmond, M. S.. 2007 Implementation of The Quantified Judgement Model to examine the impact of Human Factors on Marine Corps Distributed Operations. Thesis. Naval Post-Graduate School 2007.

[2] Dickheiser, M.. 2006. *Game Programming Gems 6 (Game Development Series)*. Charles River Media, Inc..

[3] Ding, M., Wang, L., & Bi, R.. 2001. An ANN-based Approach for Forecasting the Power Output of Photovoltaic System. *Procedia Environmental Sciences 11*, 2011. 1308 – 1315.

[4] Hajek, M. 2005. *Neural Networks*. South Africa: University of KwaZulu-Natal

[5] Metoyer, R.,et. al.. 2010. Explaining how to play real-time strategy games. *Knowledge-Based Systems*, 23 295–301.

[6] Murias, K., et. al.. 2016. The Effects of Video Game Use on Performance in a Virtual Navigation Task. *Computers in Human Behavior*, 2016. 398-406.

[7] Qun, Dai. 2014. A two-phased and Ensemble scheme integrated Backpropagation algorithm. *Applied Soft Computing 24*. 1124–1135.

[8] Rhalibi, A. E., Wong, K. W., & Price, M.. 2008. Artificial Intelligence for Computer Games. *International Journal of Computer Games Technology*. Hindawi Publishing Corporation.

[9] Robertson, G., & Watson, I. D. 2014. *A Review of Real-Time Strategy Game AI. AI Magazine*, 35(4), 75-104.

[10] Sanchez, D., Dalmau, C.. 2004. *Core Techniques and Algorithm in Game Programming.* Indiana: New Riders Publishing.

[11] Shanin, M. A.. 2014. State-of-the-art Review of Some Artificial Intelligence. *Applications in Pile Foundations. Geoscience Frontiers 7*, 2016. 33-44.

[12] Stanney, K. M., Mourant, R. R., Kennedy, R. S.. 1998. Human factors issues in virtual environments: are view of the literature. *Presence Teleoperators Virtual Environ.7*, 327–351.

[13] Takizawa, H. & Chida, Tatsuya. 2009. Evaluating Computational Performance of Backpropagation Learning on Graphics Hardware. *Electronic Notes in Theoretical Computer Science*, 2009. 379–389.