Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 11, November 2018, hlm. 4318-4324 http://j-ptiik.ub.ac.id

Penerapan *Particle Swarm Optimization* Untuk *Balancing Ability* Pada *Team Battle Game* RPG

Hilmi Ilyas Rizaldi[1], Eriq Muhammad Adams Jonemaro[2], Muhammad Aminul Akbar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]developer.hilmi@gmail.com, [2]eriq.adams@ub.ac.id, [3]muhammad.aminul@ub.ac.id

Abstrak

Pada Game RPG terdapat 3 profesi yang paling umum yaitu Fighter, Mage, dan Cleric. Dari 3 profesi tersebut terdapat perbedaan yang kompleks yang dapat mempengaruhi permainan RPG. Untuk melakukan penelitian mengenai keseimbangan profesi pada Game RPG, percobaan tersebut memakan waktu yang lama dan cost yang tinggi. Maka dari itu sebuah metode yang mengurangi cost untuk melakukan kegiatan testing ability pada profesi-profesi karakter RPG. Maka dari itu, penelitian ini membahas mengenai bagaimana kita menerapkan sebuah algoritme yang secara umum dapat dipakai dengan mudah yaitu Particle swarm optimization(PSO) dalam melakukan testing ability karakter pada Game RPG secara otomatis sehingga mengurangi cost pada development Game. Penelitian ini menghasilkan sebuah AI bot yang dapat meniru perilaku manusia pada umumnya yang sesuai dengan rules pada Game sehingga dapat membantu Game designer dalam menentukan balancing ability pada Game yang mereka buat. Penelitian ini membahas mengenai penggunaan Artificial neural network(ANN) untuk mengontrol sebuah karakter pada game agar dapat mensimulasikan pertandingan sebagai ukuran balance dari sebuah battle. Kontroler ANN tersebut dilatih tanpa diajari atau belajar sendiri untuk mengerti pergerakan musuh mereka dan dievaluasi untuk dijadikan sebagai kontroler utama pada penelitian ini. Metode pembelajaran ANN sendiri menggunakan PSO untuk menentukan kontroler terbaik dalam masa training. Penelitian ini dilakukan pada game turn based – RPG. Hasil dari penelitian ini adalah proses balancing skill set baru beserta konfigurasi PSO yang mempengaruhi penelitian ini.

Kata kunci: *Game, Artificial Intelligence, Artificial Neural Network, Particle Swarm Optimization.*

Abstract

In RPG Games there are 3 most common professions namely Fighter, Mage, and Cleric. Of the three professions, there are complex differences that can affect the game RPG. To conduct research on the balance of the profession on Game RPG, the experiment takes a long time and high cost. Therefore, a method that reduces the cost of performing the testing ability on the RPG character professions. Therefore, this study discusses how we apply a commonly usable algorithm that is Particle swarm optimization (PSO) in character testing of characters in RPG Games automatically, thereby reducing the cost of game development. This research produces an AI bot that can mimic human behavior in general that matches the rules in the Game so as to help Game designers in determining the balancing ability of the Game they create. This study discusses the use of the Artificial neural network (ANN) to control a character in a game in order to simulate a match as a balanced size of a battle. The ANN controllers were trained without being taught or self-taught to understand their enemy's movements and were evaluated to serve as the primary controllers of the study. ANN own learning method using PSO to determine the best controller in the training. The research was conducted on turn-based games - RPG. The result of this research is a new balancing skill set process along with PSO configuration that influences this research.

Keywords: *Game, Artificial Intelligence, Artificial Neural Network, Particle Swarm Optimization.*

1. PENDAHULUAN

Role-Playing Game (RPG) adalah salah satu genre permainan dan telah membuktikan konsep

yang sangat portabel - mulai dari format live action dan tabletop yang terkandung secara fisik hingga berbagai format digital (Video Games), mobile. Dalam masyarakat kontemporer, Game RPG ada dimana-mana dan digunakan untuk kesenangan dan kepuasan pribadi (Tychsen et al., 2007).

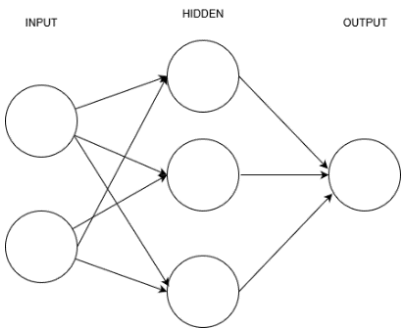
Pada Game RPG terdapat 3 profesi yang paling umum yaitu Fighter, Mage dan Cleric (Voorhees, 2009). Dari 3 profesi-profesi tersebut terdapat perbedaan yang kompleks yang dapat mempengaruhi permainan RPG. Bagaimana pembuatan sebuah karakter yang berbeda tersebut tidak berat sebelah terhadap salah satu profesi merupakan hal yang sangat sulit dilakukan oleh Game designer pada umumnya. Menurut sebuah artikel pada techinasia, gaji rata-rata game tester adalah sebesar Rp. 29,985,417 (Techinasia, 2012). Dibutuhkanya pekerjaan Game tester untuk mencoba secara langsung dan menentukan apakah Game tersebut berat sebelah atau imbang. Untuk melakukan penelitian mengenai keseimbangan profesi pada Game RPG, percobaan tersebut memakan waktu yang lama dan cost yang tinggi. Hal tersebut membuat pada developer Game RPG merasa terbebani untuk membuat Gamenya tidak berat sebelah terhadap salah satu profesi dibandingkan profesi yang lainnya. Maka dari itu sebuah metode yang mengurangi cost untuk melakukan kegiatan testing ability pada profesi-profesi karakter RPG.

Sebuah AI bot pada Game dapat digunakan untuk melakukan testing pada Game-Game saat ini. Terdapat bot *system* pada penelitian "Automated testing: a key factor for success in video Game development. Case study and lessons learned." dengan simple script yang dapat memainkan Game terus menerus untuk melakukan testing. Bot tersebut mengembalikan nilai sukses, gagal dan informasi penuh mengenai kegagalan tersebut sehingga dapat ditelusuri (Buhl dan Gareeboo, 2012). Penelitian tersebut menampilkan sebuah dashboard bot yang memberikan hasil dari *system* yang sedang dijalankan sehingga dapat dilihat Game yang sudah berhasil di testing dan yang tidak. Namun, penelitian tersebut tidak membahas mengenai pembuatan botnya melainkan bagaimana testing secara otomatis tersebut merupakan hal yang sangat membantu dalam pengembangan video Game.

Maka dari itu, penelitian ini akan membahas mengenai bagaimana kita menerapkan sebuah algoritme yang secara umum dapat dipakai dengan mudah yaitu Particle Swarm Optimization dalam melakukan testing ability karakter pada Game RPG secara otomatis sehingga mengurangi cost pada development Game. Penelitian ini akan menghasilkan sebuah AI bot yang dapat meniru perilaku manusia pada umumnya yang sesuai dengan rules pada Game sehingga dapat membantu Game designer dalam menentukan balancing ability pada Game yang akan mereka buat.

Teori penunjang penelitian ini adalah kecerdasan buatan dalam *game*, *artificial neural network*, *particle swarm optimization*, dan skema TABS (*Team Ability Balancing System*). Kecerdasan buatan dalam *Game* (*Game* AI) biasanya digunakan untuk menciptakan lawan pemain. Berlawanan dengan ini, tujuan dari AI adalah membuat pemain tidak mudah bosan dan membiarkannya fokus pada aspek permainan yang menarik. Pemain memberikan command strategis, unit yang dikendalikan komputer menangani detail. Pada saat yang sama, detail dan dinamika permainan dipelihara dengan kontrol detail komputer, bukan hilang melalui abstraksi. Masih ada peluru yang terbang, bukan ubin yang bergeser. Selain itu, AI juga berlaku untuk permainan multi player (Jakulin, 2003).

Artificial neural network (ANN) terdiri dari banyak prosesor sederhana dan terhubung yang disebut neuron, Masing-masing menghasilkan urutan aktivasi bernilai. Input neuron bisa diaktifkan melalui sensor yang dirasakan lingkungan, neuron lain bisa diaktifkan melalui koneksi tertimbang dari neuron aktif sebelumnya.



Gambar 1. *Artificial neural network*

Algoritme *Particle swarm optimization* (PSO) adalah metode untuk optimasi fungsi nonlinier kontinu (Kennedy dan Eberhart, 1995). Algoritme ini terinspirasi oleh pengamatan

perilaku sosial dan kolektif terhadap pergerakan burung betina untuk mencari makanan atau kelangsungan hidup serta usaha budidaya ikan. Algoritme PSO terinspirasi dari pergerakan anggota terbaik populasi dan pada saat bersamaan juga dengan pengalaman mereka sendiri. Metafora tersebut menunjukkan bahwa satu set solusi bergerak dalam ruang pencarian dengan tujuan untuk mencapai posisi atau solusi terbaik. Untuk kecepatan dari particle ke - *i* menggunakan Persamaan 1 (Kennedy dan Eberhart, 1995),

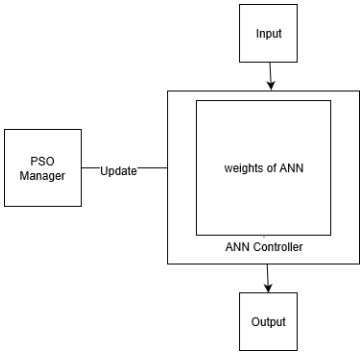
$$v_i^{t+1} = wv_i^t + c_1 r_1 (p_i - x_i^t) + c_2 r_2 (p_g^t - x_i^t), \quad (1)$$

Persamaan PSO memiliki v yang merupakan kecepatan dan x yang merupakan posisi. Pada persamaan ini terdapat variabel yang memiliki nilai tidak pasti pada setiap case penggunaan partikel swarm optimization. Variabel tersebut merupakan w (inertia weight), c1 (individual constant weight) dan c2(social constant weight). Untuk posisi dari particle ke - *i* menggunakan Persamaan 2 (Kennedy dan Eberhart, 1995),

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \quad (2)$$

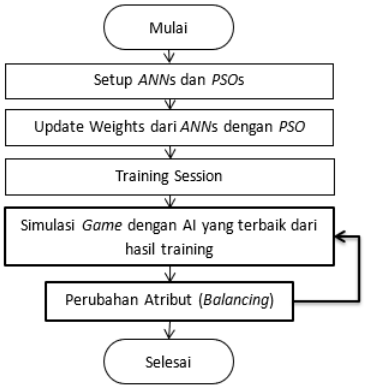
Training ANN menggunakan PSO adalah dengan cara menyamakan weights pada ANN dengan posisi partikel pada PSO. Posisi tersebut dimasukan kedalam weights ANN lalu dijadikan sebagai fungsi perhitungan ANN. Hasil dari perhitungan tersebut akan menjadi output dari ANN yang nantinya akan dihitung sebuah fitness sebagai dasar perhitungan velocity baru pada PSO.

Menggunakan controller ANN untuk mengendalikan perilaku karakter dan menggunakan PSO manajer untuk mengatur kawanan partikel untuk kesesuaian nilai dari karakter tersebut (Fang dan Wong, 2012).



Gambar 2. *Training ANN menggunakan PSO*

TABS merupakan system yang dibuat untuk menangani *battle* dari dua tim dan mengevaluasi tim tersebut dalam *Game* RPG. Menggunakan controller ANN untuk mengendalikan perilaku karakter dan menggunakan PSO manajer untuk mengatur kawanan partikel untuk kesesuaian nilai dari karakter tersebut (Fang dan Wong, 2012). Alur kerja TABS adalah sebagai berikut.



Gambar 3. *Team Ability Balancing* System (TABS)

2. PERANCANGAN

Kode program yang diadopsi dan didapatkan dari sumber github pada link berikut ini (https://github.com/yabara/MCTSFor*Turn*Based*RPG*). Spesifikasi kebutuhan hardware, software, ketika melakukan implementasi adalah Core i7 7700HQ, RAM 16 GB, NVIDIA GeForce GT 1050, Windows 10, Visual Studio 2017 dan Unity 2017.

Pertama pada perancangan, didefinisikan skill-skill pada *game*. *Skill-skill* tersebut merupakan skill dari 3 class yaitu *warrior*, *wizard*, dan healer. Selanjutnya terdapat perancangan neural network, *input* merupakan hal yang dapat di lihat oleh player seperti Hit Point(HP)p (P = merupakan index player termasuk diri sendiri), Available Skillk (K= merupakan index Skill, tiap profesi memiliki

jumlah Skill yang berbeda membuat tambahan input), Debuffp (Apakah player terkena debuff 1 atau tidak 0) dan Output merupakan target dan Skill. Apabila Skill yang terpilih adalah Skill support, maka Skill terebut akan menuju friendly target. Semua *input* pada *Neural Network* di normalisasi menjadi interval [0,1]. Sehingga didapatkan data *input*, *output*, dan *hidden* pada Tabel 1.

Tabel 1. Jumlah neuron pada setiap test case

Test Case	*Input*	*Hidden*	*Output*	Weight
1 VS 1 *WARRIOR, WARRIOR*	9	8	7	143
1 VS 1 *WIZARD, WIZARD*	9	8	7	143
1 VS 1 *WARRIOR, WIZARD*	9	8	7	143
2 VS 2 *WARRIOR HEALER, WIZARD HEALER*	13	11	9	262

Selanjutnya, perancangan *particle swarm optimization* dengan menggunakan variabel PSO w, c1, c2 menjadi nilai 0.5, 1.5, dan 1,5. Batas posisi dari PSO adalah [4, -4]. Perhitungan *fitness* pada *training* adalah adanya penambahan apabila melakukan skill, mengeluarkan *damage* terhadap musuh, memenangkan pertandingan dan menggunakan mana point saat memenangkan pertandingan. *Fitness* akan berkurang apabila salah saat menggunakan skill dan kalah dari pertandingan.

3. IMPLEMENTASI

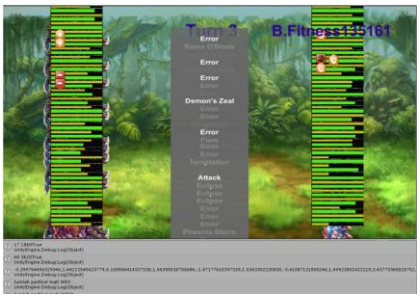
Pada ANN *Controller*, setiap profesi pada *Game* RPG Turn Based ini diberikan controller masing-masing. Dalam controller tersebut dapat beberapa fungsi utama Neural Network yaitu ComputeOutputs dan LogSigmoid.

Pada PSO, akan dibuat partikel yang memiliki jumlah diantara 20-100 berdasarkan penelitian yang menyatakan jumlah partikel tersebut tidak berpengaruh signifikan antara 1 dengan yang lainnya (Rohler dan Checn,2011). Maka didapatkan jumlah partikel dan itterasi pada tabel 2. Turn maksimal ini dibuat untuk membatasi apabila terdapat partikel yang terhenti saat *training* sehingga iterasi tidak menggunakan waktu yang lama.

Tabel 2. Jumlah partikel

Test Case	**Partikel**	**Itterasi Training**	**Turn Maksimal**
WARRIOR VS *WARRIOR*	56	1000	15
WIZARD VS *WIZARD*	56	1000	15
WARRIOR VS *WIZARD*	*WARRIOR* = 28 *WIZARD* = 28	1000	15
WARRIOR HEALER VS *WIZARD HEALER*	*WARRIOR* = 28 *WIZARD = 28* *HEALER = 56*	1000	30

Selama *training*, partikel akan bertarung secara terus menerus hingga menang atau mencapai batas *turn.* Gambar 3 menggambarkan training yang dilakukan antara *warrior* dengan *wizard* dengan informasi dibawahnya apabila terjadi perubahan best fitness maka akan diberikan info dengan menyebutkan angka index partikel, jumlah fitness, dan apakah partikel tersebut menang atau tidak. Informasi tersebut di ikuti dengan informasi isi posisi dari partikel terbaik dan jumlah partikel yang mati.



Gambar 4. *Training*

Game akan dimainkan oleh kontroller terbaik dari hasil training. Gambar 4 menggambarkan testing yang dilakukan oleh

agen ANN yang sudah ditraining. Data yang didapatkan dari testing ini merupakan error win ratio yang nanti akan di lihat pada pengujian lalu menjadi dasar dari perubahan skill set pada permainan.

Gambar 5. Training

4. PENGUJIAN DAN ANALISIS

Pada subbab ini, akan dibahas mengenai hasil pengujian berdasarkan win ratio masing-masing case. Dari case tersebut akan di dapatkan nilai error yang nantinya akan menjadi dasar perubahan skill set sebuah class yang ada. Setelah skill set tersebut dibuat maka akan dilakukan pengujian berdasarkan win rate lagi bagi case yang terpengaruh dan akan dibandingkan dengan nilai error yang sebelumnya. Tabel 3 mendeskripsikan hasil pengujian win ratio pada saat skill set belum diubah.

Tabel 3. Pengujian Win Rate Sebelum skill diubah.

Case Pengujian	Win rate 1	Win rate 2	Win rate 3	Win rate 4	Win rate 5
1 VS 1 *WARRIOR, WARRIOR*	500:500	500:500	500:500	500:500	500:500
	Avg 500:500 (Error = 0%)				
1 VS 1 *WIZARD, WIZARD*	492:508	526:474	499:501	499:501	482:518
	Avg 499.6:500.4 (Error = 0.08%)				
1 VS 1 *WARRIOR, WIZARD*	275:725	302:698	285:715	288:712	287:713
	Avg 287.4:712.6 (Error = 42.52%)				
2 VS 2 *WARRIOR HEALER, WIZARD HEALER*	498:502	499:501	497:503	498:502	498:502
	Avg 498:502 (Error = 0.4%)				

Pada Tabel 3 dapat di lihat bahwa pada case 3 terdapat error yang lebih besar dibandingkan dengan case lainnya. Dari case tersebut di telusuri skill yang digunakan *wizard* untuk melawan *warrior* pada 1 vs 1. Pattern skill yang digunakan *wizard* adalah Eclipse. Dimana pada skill Eclipse memberikan status effect mengurangi akurasi sebesar 40% yang membuat *warrior* gagal melaksanakan actionnya. Pada case 4 walaupun *wizard* menggunakan skill yang sama, jumlah error lebih sedikit dikarenakan terdapat serangan yang berjalan dari healer. Walaupun action *warrior* gagal, healer tetap dapat menggunakan actionnya tanpa masalah. Dari analisis tersebut maka dilakukan penurunan status effect skill eclipse dari 40% menjadi 15% dan penurunan damage maksimal, minimal sebanyak 2 poin. Dengan perubahan skill tersebut didapatkan hasil pengujian pada case 3 dan 4 yang akan di deskripsikan pada Tabel 4.

Tabel 4. Pengujian Win Rate Setelah skill diubah.

No	Case Pengujian	Win *rate* 1	Win *rate* 2	Win *rate* 3	Win *rate* 4	Win *rate* 5
3	1 VS 1 *WARRIOR, WIZARD*	432:568	418:582	421:579	430:570	435:565
		Avg 427.2:572.8 (*Error* = 10.06%)				
4	2 VS 2 *WARRIOR HEALER, WIZARD HEALER*	497:503	497:503	499:501	499:501	498:502
		Avg 498:502 (*Error* = 0.4%)				

1.1 Pengujian Pengaruh Parameter PSO w, c1, dan c2

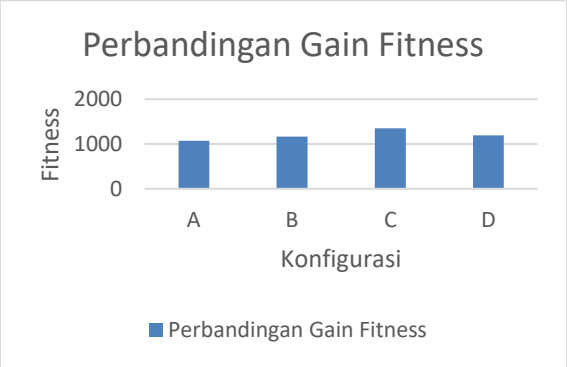
Pengujian ini akan menghasilkan perbedaaan gain fitness yang di dapatkan ketika menggunakan konfigurasi PSO yang berbeda-beda. Sebagaimana pada bab 2 subbab 2.4 PSO memiliki konfigurasi parameter yang berbeda-beda yang dapat mempengaruhi proses training ANN. Tabel 5 akan mendeskripsikan parameter yang akan di uji pada pengujian ini

Tabel 5. Pengujian Win Rate Sebelum skill diubah.

Konfigurasi	W	C1	C2
A	0.4	1.5	2
B	0.5	1.5	1.5
C	1.4	1.5	2.5
D	0.9	1.75	3

Nilai pada perubahan parameter PSO tidak menunjukan perbedaan yang signifikan. Namun pada nilai konfigurasi C terdapat penambahan fitness yang dapat membantu untuk proses training dalam mendapatkan partikel terbaik dengan waktu yang lebih cepat. Gambar 5 menunjukan perbandingan gain fitness pada

pengujian ini.



Gambar 6. Perbandingan Gain Fitness

5. KESIMPULAN

Berdasarkan dengan analisis dari pengujian yang sudah dilakukan terhadap penerapan Particle Swarm Optimizaition pada game RPG, maka dapat diambil sebuah kesimpulan sebagai berikut ini:

1. Penerapan metode Artificial neural network pada simulasi pertandingan pada game ini berjalan sesuai dengan tujuan dengan hasil pada pengujian yang menciptakan AI yang dapat memberikan aksi berdasarkan inputan yang ditentukan.
2. Penerapan Particle Swarm Optimization ini sesuai dengan tujuan pada penelitian. PSO dapat mencari weights yang tepat pada saat penerapan training kontroller. Pada hasil pengujian weights yang didapatkan dari training menggunakan PSO dapat menjalankan AI yang mengerti peraturan pada game.
3. Penerapan TABS yang dilakukan pada game RPG ini menghasilkan pengujian yang dapat dianalisis dan ditemukan ketidakseimbangan anatara kedua tim pada 2 case.
4. Ditemukan error pada win rate sebesar 42,52% pada case 3 dan error win rate yang kecil 0% pada case 1 yang dapat disimpulkan metode ini memiliki tingkat efektifitas yang tinggi dalam menemukan ketidakseimbangan pada game ini. Error pada case tersebut dapat diturunkan hingga mencapai angka 10,06%.
5. Konfigurasi yang berbeda pada PSO mempengaruhi pada pendapatan fitness pada saat training. Konfigurasi C pada pengujian merupakan konfigurasi yang terbaik pada training kasus penelitian ini.

6. DAFTAR PUSTAKA

Antonio dan Chen, 2011. An analysis of sub-swarms in multi-swarm systems. Proceeding AI'11 Proceedings of the 24th international conference on Advances in Artificial Intelligence Pages 271-280.

Buhl dan Gareeboo, 2012. Automated testing: a key factor for success in video Game development. Case study and lessons learned. PNSQC 2012.

Ernest, Adams, 2014. Fundamentals of Role-Playing Game Design. New Riders Publishing.

Fang dan Wong, 2012. Game team balancing by using particle warm optimization. Knowledge-Based System 34 91-96.

Han dan Moraga, 2005. The influence of the sigmoid function parameters on the speed of backpropagation learning. IWANN 1995: From Natural to Artificial Neural Computation pp 195-201.

Heaton, jeff. 2017. The Number of Hidden Layers. http://www.heatonresearch.com/2017/06/01/hidden-layers.html. Diakses pada 14 Desember 2017.

Jakulin, Aleks, 2003. Artificial Intelligence in Games: Food for Thought Series. http://www.stat.columbia.edu/~jakulin/FT/. diakses pada 13 Agustus 2017.

Kennedy dan Eberhart, 1995. Particle Swarm Optimization. IEEE. Perth, WA, Australia.

McCaffrey, James. 2013. Neural Network Training Using Particle Swarm Optimization. https://visualstudiomagazine.com/Articles/2013/12/01/Neural-Network-Training-Using-Particle-Swarm-Optimization.aspx?Page=2. Diakses pada 15 Desember 2017.

Putra, Yuka, 2017. Penerapan Adaptive Ai pada Game Turn Based Rpg dengan Menggunakan Metode Monte Carlo Tree Search. Universitas Brawijaya. Malang.

Putra, Yuka. 2017. MCTS For Turn Based RPG. https://github.com/yabara/MCTSForTurnBasedRPG. Diakses 15 November 2017.

Salim, Hendri. 2012. Yuk Intip Gaji Orang-Orang Bekerja Di Dunia Gaming. https://id.techinasia.com/yuk-intip-gaji-orang-orang-bekerja-di-dunia-gaming. Diakses 16 November 2017.

Tychsen, Smith, Hitchens, dan Tosca, 2006. Communication in Multi-Player Role Playing Games – The Effect of Medium. Technologies for Interactive Digital Storytelling and Entertainment. Lecture Notes in Computer Science, vol 4326. Springer, Berlin, Heidelberg

Voorhees, Gerald, 2009. The Character of Difference: Procedurality, Rhetoric, and Roleplaying Games. the international journal of computer Game research, volume 9 issue 2 November 2009 ISSN:1604-7982.