TINGKAT KESULITAN OTOMATIS BERBASIS FUZZY SUGENO PADA PLATFORMER GAME BERTEMA CERITA WAYAN RAMAYANA

Ardiawan Bagus Harisa[1], Husain Ali[2], Hanny Haryanto[3]
[1,2,3]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
Jl. Imam Bonjol No. 207, Semarang, 50131, (023)3517261
E-mail : bagusharisa@arkodestudio.com[1], join.curves@yahoo.com[2], hanny.haryanto@dsn.dinus.ac.id[3]

Abstrak

Minat pemuda terhadap kebudayaan Wayang semakin menurun, padahal Wayang memiliki banyak manfaat dalam bidang pendidikan dan kebudayaan. Penelitian ini mengajukan solusi dengan melalui sebuah game. Game dinilai merupakan media yang interaktif yang dapat digunakan sebagai pelestari Wayang. Mayoritas pemuda memilih jenis Action-Platformer saat bermain game, namun bingung untuk memilih tingkat kesulitan apa yang harusnya dipilih saat bermain game. Untuk itu diperlukan suatu game yang dapat melakukan adaptasi menyesuaikan tingkat kesulitan terhadap kemampuan pemain. Penelitian ini menggunakan metode Fuzzy sebagai pendekatan untuk implementasi tingkat kesulitan otomatis. Parameter input diambil dari atribut karakter pemain saat bermain, kemudian sistem akan mengolah dan mengklasifikasi pemain yang nantinya akan berdampak pada AI/NPC dalam game. Parameter yang digunakan adalah sisa lifepoint (HP), sisa waktu (TIME), kemampuan menyelesaikan tugas (SOLVE), tingkat kesulitan sebelumnya (DIFFICULTY), tingkat keakuratan tembakan pemain (ACCURACY), serta banyaknya pemain melakukan percobaan (TRY). Hasil dari penelitian ini adalah sistem mampu memodelkan tingkat kemampuan pemain sehingga menghasilkan tingkat kesulitan otomatis "BEGINNER", "MEDIUM", "HARD", dan "VERY HARD" dengan menggunakan metode Fuzzy Sugeno pada game bergenre Action-Platformer.

Kata Kunci: *Tingkat Kesulitan Otomatis, Sugeno, Kesulitan Adaptif dan Dinamis, Game Adaptif, Platformer*

Abstract

The interests of Cultural Puppet in of Indonesian youth were decreased, whereas Puppet has many benefits in the field of education and culture. This study suggest a solution through a game. Game assessed as an interactive medium that can be used as a Puppet preserver. The majority of the youth choose Action-Platformer genre while playing the game, but sometimes they confused to choose the difficulty that fit on them. Needed a game that can adapt to adjust the level of difficulty automatically based on player ability. This study uses Fuzzy as an approach to implements auto-difficulty. Input parameters taken from player's character attributes during gameplay, then the system will process and classify the player ability where will have an impact on AI/NPC in the game. The parameters used are the rest of life point (HP), the rest of time (TIME), the ability to complete task (SOLVE), the latest level of difficulty (DIFFICULTY), the accuracy of player shots (ACCURACY), and the number of retry to play (TRY). The result of this study is the system able to model player difficulty level so that produce automatic difficulty level "BEGINNER", "MEDIUM", "HARD", and "VERY HARD" by using Fuzzy-Sugeno on Action-Platformer game genre.

Keywords: *Auto Difficulty, Sugeno, Dynamic Difficulty Adaptive (DDA), Adaptive Game, Platformer*

1. PENDAHULUAN

Minat atau ketertarikan masyarakat khususnya remaja atau generasi muda terhadap pertunjukan Wayang makin menurun [1]. Hal ini juga terjadi pada

pemuda di Semarang. Penelitian ini menggunakan sampel acak 100 orang pemuda yang mayoritas berdomisili di Semarang. Sebanyak 21% responden menyatakan tidak tahu peran pemerintah dalam melestarikan budaya Wayang dan 42% responden menyatakan upaya tersebut masih belum cukup. Ini membuktikan bahwa, upaya yang dilakukan oleh pemerintah tersebut kurang efektif dan efisien.

Implementasi game dengan menggunakan tema Wayang pernah dilakukan oleh Pratama [2] dan Adhitama [3], masing-masing Trading Card Game dan game pada platform iOS. Berdasarkan analisa, dibutuhkan media yang lebih interaktif dan atraktif sehingga minat pemuda akan cerita dan tokoh Pewayangan semakin meningkat karena pembelajaran yang atraktif dan menyenangkan [4] pada saat bermain game lebih mudah di pahami oleh pemain.

Mayoritas pemuda memilih untuk bermain *game* dengan genre Action. Salah satu fitur pada game Action, yaitu pemain dapat memilih tingkat kesulitan permainan yang nanti akan berdampak pada perilaku objek lain yang ada pada game. Namun beberapa pemain yang baru saja memainkan suatu game, tidak mengetahui pada tingkatan apa seharusnya mereka memainkan game yang nantinya sesuai dengan kemampuan mereka. Apabila pemain salah dalam memilih tingkat kesulitan akan menyebabkan imbalance pada permainan, terlalu mudah atau bahkan terlalu susah [5]. Hal itu akan berdampak pada player tidak mendapatkan feedback yang sesuai karena tidak mengetahui berada dimana tingkat kemampuan saat memainkan game yang baru saja dimainkan.

Untuk mengetahui berada pada tingkat kemampuan atau tingkat kesulitan apa seorang pemain, maka diperlukan suatu pemodelan terhadap pemain [6]. Input dari pemain saat memainkan game dapat dijadikan parameter yang berguna sebagai penentu tingkat kesulitan [7]. Menurut Lopes dan Bidarra, komponen adaptive game ada 5 bagian, yaitu : game world, mechanic, AI/NPC, narratives, scenarios/ quest [8]. Berdasarkan uraian latar belakang ini, diperlukan suatu game yang dapat bertindak adaptif menyesuaikan tingkat kemampuan pemain dengan parameter input dari pemain, dimana tingkat kemampuan pemain dapat diseimbangkan dengan bidang AI pada game. enelitian ini hanya berfokus pada bidang AI/NPC.

Fuzzy Logic merupakan pendekatan yang dapat digunakan untuk memodelkan pemain. Pemodelan dengan menggunakan Fuzzy juga telah diteliti dan sudah digunakan game [9]. Fuzzy merupakan pendekatan yang sederhana, langsung dengan pendekatan natural untuk memindahkan aspek linguistik kedalam model matematis dan dapat digunakan untuk memverifikasi validasi dari penjelasan verbal [10]. Penelitian ini akan menggunakan Fuzzy dengan algoritma Sugeno. Algoritma Sugeno dianggap menjadi algoritma yang cukup sederhana dan kuat untuk implementasi AI pada game.

Sehingga tujuan dari penelitian ini adalah membuat tingkat kesulitan otomatis berdasarkan kemampuan pemain pada game bertema certa Wayang Ramayana dengan mengimplementasikan algoritma Sugeno.

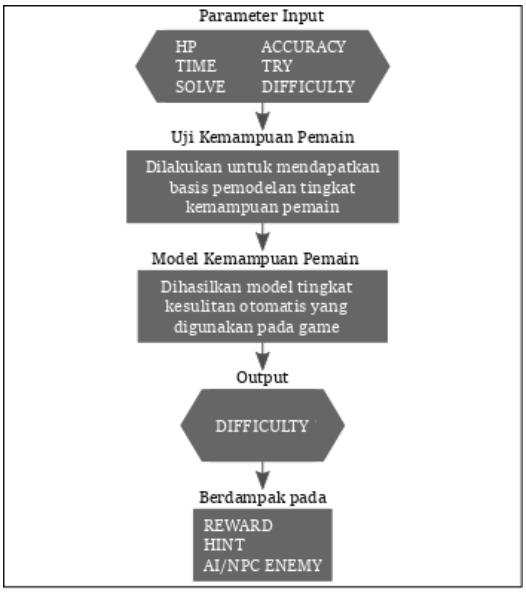
2. METODE

Untuk menyeimbangkan gameplay, pemain harus mendapat feedback yang sesuai dengan kemampuannya. Misalnya saja, pemain dengan golongan tingkat

kemampuan rendah akan mendapat musuh yang lebih mudah, lebih lamban serta reward yang relatif lebih rendah karena hanya menggunakan usaha yang lebih sedikit jika dibandingkan pemain dengan golongan tingkat kemampuan baik. Feedback yang akan diberikan dapat berupa perubahan reward seperti : Hint system (sistem penunjuk) [11], perubahan level, penambahan HP [12]. Dapat juga berupa perubahan kemampuan dan perilaku AI/NPC [7] seperti : penambahan kekuatan dan kecepatan [13], perilaku (patroli, bertahan dan lain-lain) [14]. Perubahan level dan perubahan perilaku pada AI/NPC akan menjaga minat pemain dalam bermain game karena sesuai dengan teori flow. Hint system adalah sistem penunjuk untuk memudahkan pemain dalam memainkan game. Petunjuk dapat berbentuk narasi, tutorial ataupun lainnya.

Sistem yang diajukan pada penelitian ini memiliki 4 range atau kelas dengan berbagai pertimbangan yang ada berdasarkan pada penelitian lain. Untuk mempersingkat proses perhitungan dan mempermudah, maka jumlah range pada variabel output sama dengan jumlah range yang ada pada tiap parameter input. Untuk memungkinkan perhitungan maka range pada tiap kelas harus didefiniskan secara diskrit (dengan nilai *0* hingga *N*). Penelitian ini menggunakan 6 parameter input : sisa lifepoint (HP), sisa waktu (TIME), kemampuan menyelesaikan tugas (SOLVE), tingkat kesulitan sebelumnya (DIFFICULTY), tingkat keakuratan tembakan pemain (ACCURACY), serta banyaknya pemain melakukan percobaan (TRY).

Setelah dipilih parameter yang sesuai, maka akan diujikan pada sampel pemain secara acak sehingga didapat basis pengetahuan untuk pemodelan pemain. Semua parameter tersebut diambil setelah pemain menyelesaikan satu level/ tingkatan saat bermain game.



Gambar 2.1. Model pengembangan Auto Difficulty atau tingkat kesulitan otomatis

2.1 Fuzzy

Setelah didapat parameter, maka perlu dibuat derajat dan range tiap derajat kanggotaan parameter. Berikut adalah derajat keanggotaan parameter input :

1. *HP* : {sedikit, cukup, banyak, penuh}
2. *TIME* : {lambat, cukup, cepat, sempruna}
3. *SOLVE* : {rendah, cukup, tinggi, sempurna}
4. *ACCURACY* : {sangat rendah, rendah, cukup, tinggi}
5. *DIFIFICULTY* : {beginer, medium, hard, very hard}
6. *TRY* : {sangat sering, sering, jarang, tidak pernah}

Berikut adalah derajat keanggotaan pada parameter output :

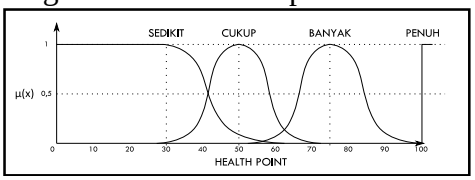
DIFFICULTY : {beginner, medium, hard, very hard}

Setelah didapat derajat keanggotaan, maka selanjutnya adalah membuat range derajat keanggotaan. Fungsi keanggotan menggunakan kombinasi kurva-S dan kurva bentuk Lonceng Beta.

1. *HP*

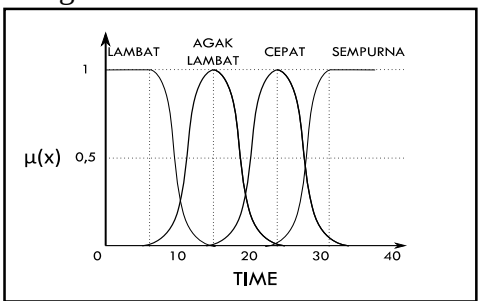
   Merupakan presentase dari HP pada karakter. Jumlah minimal 0% dan

maksimal 100%. Sutanto menggunakan parameter HP atau sisa nyawa sebagai salah satu input untuk sistem cerdas yang dibuat [7]. Beliau tidak menjelaskan bagaimana metode untuk menentukan range pada parameter input tersebut. Range pada parameter input sangat bergantung pada *game designer*. Pada beberapa game seperti The Sims, The Sims 2, menggunakan range 0 – 100% untuk parameter HP.



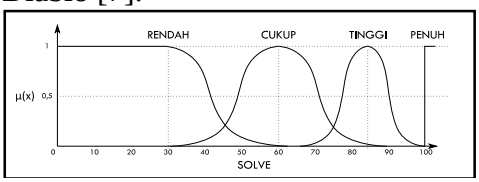
Gambar 2.2. Himpunan Fuzzy pada variabel HP

2. *TIME*
   Diberikan T sebagai batas waktu untuk tiap *level*. Diberikan TIME sebagai sisa waktu yang dimiliki pemain untuk menyelesaikan *level* permainan terhadap T. Satuan untuk T dan TIME adalah detik. Untuk mendapatkan range yang sesuai, maka perlu dilakukan pengujian secara langsung. Karena range tiap kelas sangat bergantung dengan hasil dari uji coba pemain, tingkat kesulitan, panjang area pada tiap level. Dengan pendekatan *try and error*, didapat range untuk tiap kelas sebagai berikut :



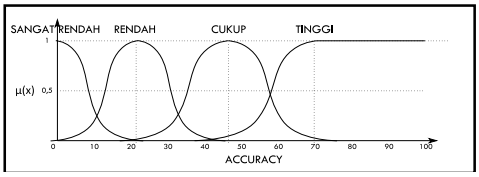
Gambar 2.3. Himpunan Fuzzy pada variabel TIME

3. *SOLVE*
   Parameter ini adalah presentase dari tugas atau *quest* yang harus diselesaikan oleh pemain. Jika pemain berhasil menyelesaikan semua tugas, maka SOLVE akan bernilai 100%. Penelitian menggunakan range antara 0 hingga 100 % berdasarkan beberapa *game* yang telah menggunakan range tersebut seperti contohnya Canaan, Diablo [7].



Gambar 2.4. Himpunan Fuzzy pada variabel SOLVE

4. ACCURACY
   Presentase jumlah serangan pemain yang mengenai musuh dibagi dengan jumlah total serangan yang dikeluarkan. *Game Virtua Cop* yang Berjaya pada era windows 98, menggunakan parameter ACCURACY sebagai parameter untuk sistem agar dapat menilai kemampuan pemain. Setelah kemampuan pemain diketahui maka sistem dapat memberi *reward* sesuai dengan tingkat keakuratan dari akurasi pemain dalam menembak lawan. Penelitian tentang penggunaan parameter ACCURACY ini juga telah dibuktikan pada penelitian yang dilakukan oleh Bruno [15].



Gambar 2.5. Himpunan Fuzzy pada variabel ACCURACY

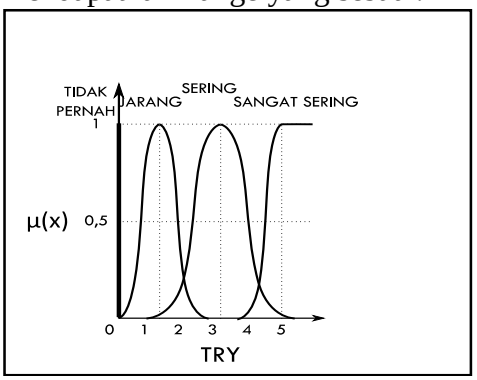
5. *DIFFICULTY*
   Nilai dari parameter ini diambil dari atribut pada pemain. Parameter ini akan menjadi parameter sumber untuk dilakukan penghitungan kembali klasifikasi tingkat

kemampuan pemain pada setiap *levelnya*.

$$DIFFICULTY = \begin{cases} beginer;\ DIFFICULTY = beginer \\ medium;\ DIFFICULTY = medium \\ hard;\ DIFFICULTY = hard \\ very\ hard;\ DIFFICULTY = very\ hard \end{cases} \quad (2.1)$$

6. *TRY*

Didapat dari jumlah percobaan ulang pemain untuk memainkan suatu level pada *game*. Performa pemain dan waktu yang dihabiskan pemain dalam bermain akan berpengaruh pada *experience* yang didapat pada saat bermain *game*. Yanakakkis menerpakan konsep flow Csikszentmihalyi untuk pemodelan pemain, dimana apabila kemampuan atau performa pemain rendah, maka challenge atau tingkat kesulitan diturunkan sesuai performa pemain [6]. Pada penelitian ini menggunakan range 0 hingga 5 x percobaan pemain pada saat bermain game. Penentuan tersebut dilakukan setelah uji coba pada pemain untuk mendapatkan range yang sesuai.



Gambar 2.6. Himpunan Fuzzy pada variabel TRY

2.2 Rule pada Fuzzy

Derajat keanggotaan untuk tiap parameter telah ditentukan. Langkah selanjutnya adalah pembuatan rule Fuzzy untuk parameter output yang diberi nama *DIFFICULTY_RESULT*. Dari *rule* diatasa diketahui terdapat 6 variabel dan 4 range derajat keanggotaan. Untuk *rule* optimal seharusnya terdapat 6^4 *rule* yang terbentuk. Namun karena jumlah *rule* yang terlalu besar, penelitian ini menggunakan *rule* yang telah dibentuk secara eksplisit. Rule yang akan dibentuk menggunakan aturan kombinasi :

$$ {}_nC_r = \frac{n!}{r!\,(n-r)!} \quad (2.2)$$

Sehingga didapat

$$ {}_6C_4 = \frac{6!}{4!\,(6-4)!} = 15$$

15 *rule* untuk masing-masing derajat keanggotaan pada variabel. Masing masing dapat dilihat pada Tabel 1, Tabel 2, Tabel 3, dan Tabel 4.

Untuk mengatasi jika ada kondisi yang terlewat dari rule yang telah dibentuk, maka dibentuk rule ke 61, yaitu :

$$\text{DR} = \bigvee \frac{\Sigma\,\mu\text{D BEGINER}}{\text{n}}, \frac{\Sigma\,\mu\text{D MEDIUM}}{\text{n}}, \frac{\Sigma\,\mu\text{D HARD}}{\text{n}}, \frac{\Sigma\,\mu\text{D VERYHARD}}{\text{n}} \quad (2.3)$$

Dimana μD adalah derajat keanggotaan dari variabel (DR) DIFFICULTY_RESULT, dan n adalah banyaknya variabel. DR adalah nilai maksimum dari μD BEGINER, μD MEDIUM, μD HARD, μD VERY HARD.

$$DIFFICULTY\_RESULT = \begin{cases} beginer;\ DR = beginer \\ medium;\ DR = medium \\ hard;\ DR = hard \\ very\ hard;\ DR = very\ hard \end{cases} \quad (2.4)$$

3. IMPLEMENTASI

Pada bab ini akan menjelaskan tentang tahapan dan implementasi Algoritma Sugeno pada *game* Ramayana. Dalam hal ini basis pengetahuan telah didapat secara intuitif pada rule-rule yang telah dibuat.

Tabel 2.1: Rule untuk "BEGINNER" pada variabel DIFFICULTY_RESULT

No	Rule	TRY	DIFFICULTY	ACCURACY	SOLVE	TIME	HP	DIFFICULTY_RESULT
1	1	SANGAT SERING	BEGINER	SANGAT RENDAH	RENDAH	X	X	BEGINER
2	2	X	BEGINER	SANGAT RENDAH	RENDAH	LAMBAT	X	BEGINER
3	3	X	X	SANGAT RENDAH	RENDAH	LAMBAT	SEDIKIT	BEGINER
4	4	SANGAT SERING	X	X	RENDAH	LAMBAT	SEDIKIT	BEGINER
5	5	SANGAT SERING	BEGINER	SANGAT RENDAH	X	X	SEDIKIT	BEGINER
6	6	X	BEGINER	X	RENDAH	LAMBAT	SEDIKIT	BEGINER
7	7	SANGAT SERING	BEGINER	X	X	LAMBAT	SEDIKIT	BEGINER
8	8	SANGAT SERING	BEGINER	SANGAT RENDAH	X	LAMBAT	X	BEGINER
9	9	X	BEGINER	SANGAT RENDAH	X	LAMBAT	SEDIKIT	BEGINER
10	10	X	BEGINER	SANGAT RENDAH	RENDAH	X	SEDIKIT	BEGINER
11	11	SANGAT SERING	X	SANGAT RENDAH	RENDAH	LAMBAT	X	BEGINER
12	12	SANGAT SERING	X	SANGAT RENDAH	RENDAH	X	SEDIKIT	BEGINER
13	13	SANGAT SERING	BEGINER	X	RENDAH	LAMBAT	X	BEGINER
14	14	SANGAT SERING	BEGINER	X	RENDAH	X	SEDIKIT	BEGINER
15	15	SANGAT SERING	X	SANGAT RENDAH	X	LAMBAT	SEDIKIT	BEGINER

Tabel 2.2: Rule untuk "MEDIUM" pada variabel DIFFICULTY_RESULT

No	Rule	TRY	DIFFICULTY	ACCURACY	SOLVE	TIME	HP	DIFFICULTY_RESULT
1	16	SERING	MEDIUM	RENDAH	CUKUP	X	X	MEDIUM
2	17	X	MEDIUM	RENDAH	CUKUP	AGAK LAMBAT	X	MEDIUM
3	18	X	X	RENDAH	CUKUP	AGAK LAMBAT	CUKUP	MEDIUM
4	19	SERING	X	X	CUKUP	AGAK LAMBAT	CUKUP	MEDIUM
5	20	SERING	MEDIUM	RENDAH	X	X	CUKUP	MEDIUM
6	21	X	MEDIUM	X	CUKUP	AGAK LAMBAT	CUKUP	MEDIUM
7	22	SERING	MEDIUM	X	X	AGAK LAMBAT	CUKUP	MEDIUM
8	23	SERING	MEDIUM	RENDAH	X	AGAK LAMBAT	X	MEDIUM
9	24	X	MEDIUM	RENDAH	X	AGAK LAMBAT	CUKUP	MEDIUM
10	25	X	MEDIUM	RENDAH	CUKUP	X	CUKUP	MEDIUM
11	26	SERING	X	RENDAH	CUKUP	AGAK LAMBAT	X	MEDIUM
12	27	SERING	X	RENDAH	CUKUP	X	CUKUP	MEDIUM
13	28	SERING	MEDIUM	X	CUKUP	AGAK LAMBAT	X	MEDIUM
14	29	SERING	MEDIUM	X	CUKUP	X	CUKUP	MEDIUM
15	30	SERING	X	RENDAH	X	AGAK LAMBAT	CUKUP	MEDIUM

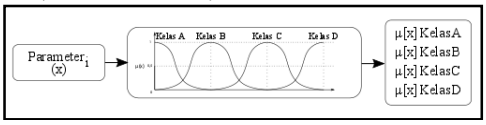
Tabel 2.3: Rule untuk "HARD" pada variabel DIFFICULTY_RESULT

No	Rule	TRY	DIFFICULTY	ACCURACY	SOLVE	TIME	HP	DIFFICULTY_ RESULT
1	31	JARANG	HARD	CUKUP	TINGGI	X	X	HARD
2	32	X	HARD	CUKUP	TINGGI	CEPAT	X	HARD
3	33	X	X	CUKUP	TINGGI	CEPAT	BANYAK	HARD
4	34	JARANG	X	X	TINGGI	CEPAT	BANYAK	HARD
5	35	JARANG	HARD	CUKUP	X	X	BANYAK	HARD
6	36	X	HARD	X	TINGGI	CEPAT	BANYAK	HARD
7	37	JARANG	HARD	X	X	CEPAT	BANYAK	HARD
8	38	JARANG	HARD	CUKUP	X	CEPAT	X	HARD
9	39	X	HARD	CUKUP	X	CEPAT	BANYAK	HARD
10	40	X	HARD	CUKUP	TINGGI	X	BANYAK	HARD
11	41	JARANG	X	CUKUP	TINGGI	CEPAT	X	HARD
12	42	JARANG	X	CUKUP	TINGGI	X	BANYAK	HARD
13	43	JARANG	HARD	X	TINGGI	CEPAT	X	HARD
14	44	JARANG	HARD	X	TINGGI	X	BANYAK	HARD
15	45	JARANG	X	CUKUP	X	CEPAT	BANYAK	HARD

Tabel 2.4: Rule untuk "VERYHARD" pada variabel DIFFICULTY_RESULT

No	Rule	TRY	DIFFICULTY	ACCURACY	SOLVE	TIME	HP	DIFFICULTY_ RESULT
1	46	TIDAK PERNAH	VERY HARD	TINGGI	SEMPURNA	X	X	VERY HARD
2	47	X	VERY HARD	TINGGI	SEMPURNA	SEMPURNA	X	VERY HARD
3	48	X	X	TINGGI	SEMPURNA	SEMPURNA	PENUH	VERY HARD
4	49	TIDAK PERNAH	X	X	SEMPURNA	SEMPURNA	PENUH	VERY HARD
5	50	TIDAK PERNAH	VERY HARD	TINGGI	X	X	PENUH	VERY HARD
6	51	X	VERY HARD	X	SEMPURNA	SEMPURNA	PENUH	VERY HARD
7	52	TIDAK PERNAH	VERY HARD	X	X	SEMPURNA	PENUH	VERY HARD
8	53	TIDAK PERNAH	VERY HARD	TINGGI	X	SEMPURNA	X	VERY HARD
9	54	X	VERY HARD	TINGGI	X	SEMPURNA	PENUH	VERY HARD
10	55	X	VERY HARD	TINGGI	SEMPURNA	X	PENUH	VERY HARD
11	56	TIDAK PERNAH	X	TINGGI	SEMPURNA	SEMPURNA	X	VERY HARD
12	57	TIDAK PERNAH	X	TINGGI	SEMPURNA	X	PENUH	VERY HARD
13	58	TIDAK PERNAH	VERY HARD	X	SEMPURNA	SEMPURNA	X	VERY HARD
14	59	TIDAK PERNAH	VERY HARD	X	SEMPURNA	X	PENUH	VERY HARD
15	60	TIDAK PERNAH	X	TINGGI	X	SEMPURNA	PENUH	VERY HARD

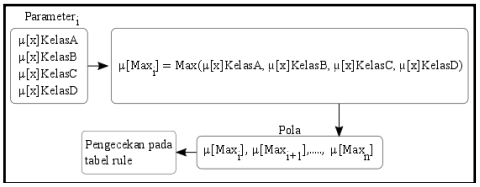
Yakni sesuai dengan kelas pada variabel yang telah disajikan pada Gambar 2.2, Gambar 2.3, Gambar 2.4, Gambar 2.5,dan Gambar 2.6.



Gambar 3.1. Alur Fuzzyfikasi untuk pemodelan pemain

Untuk setiap parameter yang akan dijadikan sebagai input akan dicari nilai derajat keanggotaannya pada masing-masing kelas yang ada sesuai dengan desain *Fuzzy* yang ada. Penelitian ini menggunakan 6 input parameter dan masing-masing parameter memiliki 4 kelas.

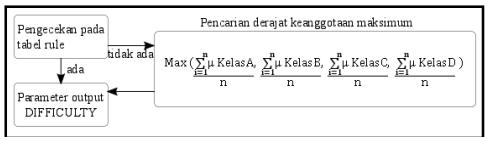
1. Pembentukan basis pengetahuan
   Telah diutarakan sebelumnya, basis pengetahuan ditentukan secara intuitif. Terdapat 4 derajat keanggotaan untuk setiap parameter input. Nilai parameter output (DIFFICULTY_RESULT) sesuai dengan ***Tabel 3.1*** untuk derajat keanggotaan BEGINNER, ***Tabel 3.2*** untuk derajat keanggotaan MEDIUM, ***Tabel 3.3*** untuk derajat keanggotaan HARD, dan ***Tabel 3.4*** untuk derajat keanggotaan VERY HARD.



Gambar 3.2. Alur pencarian dan pengecekan pola yang terbentuk pada rule

Setelah didapat nilai derajat keanggotaan tiap kelas pada masing-masing parameter, maka dicari nilai maksimum di antara kelas-kelas yang ada pada tiap parameter. Dengan begitu akan terbentuk pola. Pola tersebut akan diuji pada tabel rule yang sudah ada.

2. Mesin Inferensi dan Defuzzyfikasi
   Pada tahap ini dihitung seluruh nilai derajat keanggotaan untuk setiap parameter input dan keluaran dari tahap ini dapat ditentukan secara tegas (crisp). Untuk melakukan kedua tahap ini, menggunakan Metode Rata-rata (*Average*) dan fungsi implikasi *Max*. Sehingga parameter output akan didapat dari rata-rata derajat keanggotaan dengan nilai terbesar.



Gambar 3.3. Hasil pemodelan pemain

Jika pola yang diujikan terbaca oleh tabel rule yang ada, maka akan langsung menghasilkan output berupa nilai DIFFICULTY yang baru. Jika belum ada maka akan dilakukan penghitungan dengan menggunakan fungsi implikasi *Max*. Sistem akan mencari nilai maksimum dari hasil penjumlahan tiap golongan kelas dibagi dengan jumlah parameter. Dengan begitu sistem akan mendapat output meskipun tabel tidak mengenal pola yang menjadi input.

$$\mu D\ BEGINER = (\mu[HP]Sedikit + \mu[TIME]Lambat + \mu[TRY]\ SangatSering + \mu[SOLVE]Rendah + \mu[ACCURACY]SangatRendah + \mu[DIFFICULTY])/n \quad (2.5)$$

$$\mu D\ MEDIUM = (\mu[HP]Cukup + \mu[TIME]AgakLambat + \mu[TRY]Sering + \mu[SOLVE]Cukup + \mu[ACCURACY]Rendah + \mu[DIFFICULTY])/n \quad (2.6)$$

$$\mu D\ HARD = (\mu[HP]Banyak + \mu[TIME]Cepat + \mu[TRY]Jarang + \mu[SOLVE]Tinggi + \mu[ACCURACY]Cukup + \mu[DIFFICULTY])/n \quad (2.7)$$

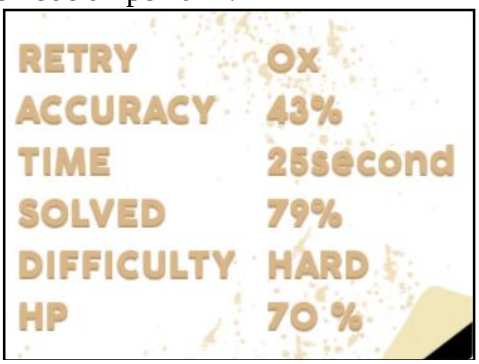
μD VERYHARD = (μ[HP]Penuh
+ μ[TIME]Sempurna
+ μ[TRY]TidakPernah
+ μ[SOLVE]Penuh
+ μ[ACCURACY]Tinggi
+ μ[DIFFICULTY])/n
(2.8)

DR = V▒[(Σ μD BEGINER)/n],
(Σ μD MEDIUM)
/n, (Σ μD HARD)/n,
(Σ μD VERYHARD)/n
(2.9)

Dimana μD adalah derajat keanggotaan dari variabel (DR) DIFFICULTY_RESULT, dan n adalah banyaknya variabel. DR adalah nilai maksimum dari μD BEGINER, μD MEDIUM, μD HARD, μD VERY HARD.

3.2 Contoh Kasus

Berikut adalah contoh kasus yang mungkin dapat terjadi pada saat pemodelan pemain :



Gambar 3.4. Contoh kasus yang mungkin terjadi

Tahap pertama yang dilakukan yaitu menentukan atau mencari nilai derajat keanggotaan pada tiap kelas tiap parameter input. Untuk parameter HP, akan dicari nilai derajat keanggotaannya dengan *Fuzzy* sesuai pada Gambar 2.2, untuk parameter TIME sesuai pada Gambar 2.3, untuk parameter SOLVE sesuai pada Gambar 2.4, untuk parameter ACCURACY sesuai pada Gambar 2.5 dan untuk untuk parameter TRY sesuai pada Gambar 2.6. Dengan demikian maka akan diperoleh hasil sebagai berikut :

Penghitungan derajat keanggotaan (μ) pada parameter HP :

$$\mu[70]\mathrm{HP}_{\mathrm{SEDIKIT}} = 0$$

$$\mu[70]\mathrm{HP}_{\mathrm{CUKUP}} = 0$$

$$\mu[70]\mathrm{HP}_{\mathrm{BANYAK}} = 1 - 2\left(\frac{75-70}{75-50}\right)^2 = 0.92$$

$$\mu[70]\mathrm{HP}_{\mathrm{PENUH}} = 0$$

Penghitungan derajat keanggotaan (μ) pada parameter ACCURACY :

$$\mu[43]\mathrm{ACCURACY}_{\mathrm{SANGATRENDAH}} = 0$$

$$\mu[43]\mathrm{ACCURACY}_{\mathrm{RENDAH}} = 0$$

$$\mu[43]\mathrm{ACCURACY}_{\mathrm{CUKUP}} = 1 - 2\left(\frac{45-43}{45-20}\right)^2 = 0.9872$$

$$\mu[43]\mathrm{ACCURACY}_{\mathrm{TINGGI}} = 2\left(\frac{43-40}{70-40}\right)^2 = 0.02$$

Penghitungan derajat keanggotaan (μ) pada parameter TIME :

$$\mu[25]\mathrm{TIME}_{\mathrm{LAMBAT}} = 0$$

$$\mu[25]\mathrm{TIME}_{\mathrm{AGAKLAMBAT}} = 0$$

$$\mu[25]\mathrm{TIME}_{\mathrm{CEPAT}} = 1 - 2\left(\frac{25-22}{32-22}\right)^2 = 0.82$$

$$\mu[25]\mathrm{TIME}_{\mathrm{SEMPURNA}} = 2\left(\frac{25-23}{30-23}\right)^2 = 0.163$$

Penghitungan derajat keanggotaan (μ) pada parameter SOLVE :

$$\mu[79]\mathrm{SOLVE}_{\mathrm{RENDAH}} = 0$$

$$\mu[79]\mathrm{SOLVE}_{\mathrm{CUKUP}} = 2\left(\frac{90-79}{90-60}\right)^2 = 0.268$$

$$\mu[79]\mathrm{SOLVE}_{\mathrm{TINGGI}} = 1 - 2\left(\frac{85-79}{85-70}\right)^2 = 0.68$$

$$\mu[79]\mathrm{SOLVE}_{\mathrm{SEMPURNA}} = 0$$

Berdasarkan kasus diatas, pemain belum pernah mencoba ulang (TRY = 0), maka parameter DIFFICULTY_INPUT akan diatur secara default.

$\mu DIFFICULTY\_INPUT_{BEGINNER} = 1$
$\mu DIFFICULTY\_INPUT_{MEDIUM} = 0$
$\mu DIFFICULTY\_INPUT_{HARD} = 0$
$\mu DIFFICULTY\_INPUT_{VERY\ HARD} = 0$

Penghitungan derajat keanggotaan (μ) pada parameter TRY :
$\mu[0]TRY_{TIDAK\ PERNAH} = 1$
$\mu[0]TRY_{JARANG} = 0$
$\mu[0]TRY_{SERING} = 0$
$\mu[0]TRY_{SANGATSERING} = 0$

Tabel 3.1: Hasil Fuzzyfikasi

TRY	DIFFICULTY	ACCURACY	SOLVE	TIME	HP
TIDAK PERNAH	BEGINER	CUKUP	TINGGI	CEPAT	BANYAK

Berdasarkan kasus diatas maka pola yang terbentuk adalah seperti yang ditampilkan pada Tabel 3.1. Setelah diketahui nilai dari tiap derajat keanggotaan pada suatu variabel, maka dipilih nilai terbesar / maksimum yang kemudian dengan menggunakan table rule yang sudah ada dapat diketahui outputnya.

Berdasarkan rule ke-33 pada tabel 2.3, maka akan menghasilkan DIFFICULTY_RESULT : HARD.

Untuk memastikan dapat dilakukan penghitungan dengan menggunakan rule ke-61.

$$\mu BEGINNER = (\mu[70]HP_{SEDIKIT} + \mu[25]TIME_{LAMBAT} + \mu[0]TRY_{SANGATSERING} + \mu[79]SOLVE_{RENDAH} + \mu[43]ACCURACY_{SANGATRENDAH} + \mu DIFFICULTY\_INPUT_{BEGINNER}) / 6$$

$$\mu BEGINNER = \frac{(0+0+0+0+0+1)}{6} = 0.166$$

$$\mu MEDIUM = (\mu[70]HP_{CUKUP} + \mu[25]TIME_{AGAKLAMBAT} + \mu[0]TRY_{SERING} + \mu[79]SOLVE_{CUKUP} + \mu[43]ACCURACY_{RENDAH} + \mu DIFFICULTY\_INPUT_{MEDIUM})/6$$

$$\mu MEDIUM = \frac{(0+0+0+0.268+0+0)}{6} = 0.044$$

$$\mu HARD = (\mu[70]HP_{BANYAK} + \mu[25]TIME_{CEPAT} + \mu[0]TRY_{JARANG} + \mu[79]SOLVE_{TINGGI} + \mu[43]ACCURACY_{CUKUP} + \mu DIFFICULTY\_INPUT_{HARD})/6$$

$$\mu HARD = \frac{(0.92+0.82+0+0.68+0.9872+0)}{6} = 0.567$$

$$\mu VERYHARD = (\mu[70]HP_{PENUH} + \mu[25]TIME_{SEMPURNA} + \mu[0]TRY_{TIDAKPERNAH} + \mu[79]SOLVE_{SEMPURNA} + \mu[43]ACCURACY_{TINGGI} + \mu DIFFICULTY\_INPUT_{VERYHARD})/6$$

$$\mu VERYHARD = \frac{(0+0.163+1+0+0.02+0)}{6} = 0.197$$

$$DIFFICULTY\_RESULT = \bigvee 0.166, 0.044, 0.567, 0.197 = 0.567\ (\mu HARD)$$

μBEGINNER merupakan variabel untuk menampung jumlah nilai derajat keanggotaan dari berbagai variabel yang telah ditentukan secara intuitif yang digolongkan pada **DIFFICULTY_RESULT** = "BEGINNER". Pada tahap terakhir

didapat parameter output "BEGINNER", "MEDIUM", "HARD" atau "VERYHARD". Output tersebut merupakan model atau klasifikasi pemain berdasarkan performa pemain saat memainkan game.

Berdasarkan penghitungan hasil menunjukan bahwa μ**HARD** adalah derajat keanggotaan terbesar, sehingga dapat dipastikan bahwa model pemain dengan input sedemikian seperti contoh kasus diatas menghasilkan tingkat kesulitan "**HARD**".

4. HASIL DAN PEMBAHASAN

Berikut adalah paparan hasil penelitian.

4.1 Hasil Pengujian

Tabel 4.1: Hasil pengujian black-box

No	Output yang diharapkan	Output	Hasil
1	Sistem ARD dapat membaca parameter input.	Sistem ARD dapat membaca parameter input.	Sesuai
2	Sistem ARD dapat menghasilkan output berupa DIFFICULTY yang baru.	Sistem ARD dapat menghasilkan output berupa DIFFICULTY yang baru.	Sesuai
3	Sistem ARD dapat menyimpan parameter output (model pemain) yang telah dikalkulasi.	Sistem ARD dapat menyimpan parameter output (model pemain) yang telah dikalkulasi.	Sesuai
4	GameController mengecek dan melakukan pengisian data secara default jika data masih kosong.	GameController mengecek dan melakukan pengisian data secara default jika data masih kosong.	Sesuai
5	Enemy dapat melakukan penyesuaian sesuai dengan model pemain.	Enemy dapat melakukan penyesuaian sesuai dengan model pemain.	Sesuai

Sistem cerdas yang digunakan untuk melakukan klasifikasi atau pemodelan pemain atau untuk mendapatkan tingkat kesulitan otomatsi ini diberi nama *ARD (Auto Ramayana Difficulty*) Pengujian yang dilakukan akan berfokus pada sistem *ARD*. Pengujian *black-box* dilakukan untuk menguji apakah system *ARD* telah berfungsi sesuai kebutuhan. Tabel 4.1 adalah hasil dari pengujian black-box.

Setelah didapat sistem untuk mengetahui tingkat kemampuan pemain, maka dilakukan uji coba pada pemain. Pengujian ini dilakukan pada pemain dengan memainkan level 1 pada game Ramayana.

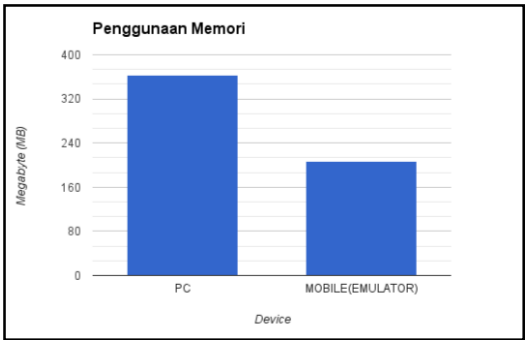
Tabel 4.2: Data ujicoba pada pemain

No	HP (%)	TIME (detik)	SOLVE (%)	ACCURACY (%)	TRY (kali)	DIFFICULTY_RESULT (output)
1	100	100	86	29	1	VERYHARD
2	100	90	96	90	0	VERYHARD
3	100	96	93	58	0	VERYHARD
4	100	65	100	100	1	VERYHARD
5	100	70	96	76	0	VERYHARD
6	100	117	0	0	7	BEGINNER
7	100	75	61	80	7	VERYHARD
8	10	16	61	31	7	MEDIUM
9	70	25	79	43	0	HARD
10	30	30	30	50	0	BEGINNER
11	10	30	30	50	0	BEGINNER
12	100	78	76	96	2	VERYHARD
13	90	72	76	83	0	VERYHARD
14	30	50	30	76	0	BEGINNER
15	90	72	79	83	1	VERYHARD
16	10	41	82	47	8	HARD
17	10	41	61	30	7	BEGINNER
18	50	41	76	96	9	BEGINNER
19	90	58	89	100	10	BEGINNER
20	90	65	89	90	10	VERY HARD

Berdasarkan pengujian jumlah model pemain "VERY HARD" dan "BEGINNER" lebih banyak dibandingkan model pemain "HARD" dan "MEDIUM". Hal tersebut dikarenakan desain *Fuzzy* yang dibuat. Range atau luas daerah derajat keanggotaan 1 pada parameter "MEDIUM" dan "HARD" (μMEDIUM dan μHARD) lebih kecil dari pada luas daerah untuk derajat keanggotaan 1 pada "VERY HARD" dan "BEGINNER" (μVERYHARD dan μBEGINNER).

Pada platform PC, untuk melakukan

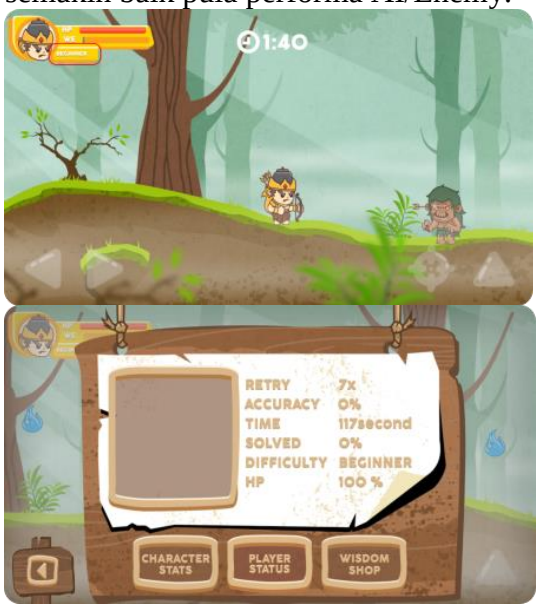
update gameplay dan output tingkat kesulitan otomatis atau pemodelan pemain memerlukan waktu rata-rata 0.002361 detik. Berikut adalah penggunaan memori pada saat game Ramayana dijalankan.



Gambar 4.1. Perbedaan penggunaan memori pada PC dan Mobile

4.2 Screenshot game Ramayana

Performa AI/Enemy dalam game ini dipengaruhi oleh kemampuan pemain. Semakin baik performa pemain, maka semakin baik pula performa AI/Enemy.



Gambar 4.2. Screenshoot model pemain "BEGINNER"

4.3 Pembahasan

Berdasarkan hasil pengujian *black-box* dan pengujian game Ramayana oleh pemain implementasi algoritma Sugeno pada game Ramayana dapat disimpulkan bahwa sistem ARD yang dibuat dapat memenuhi kebutuhan fungsionalitas dan dapat mendukung sistem dalam memodelkan pemain. Sehingga pemain tidak perlu secara manual memilih tingkat kesulitan yang dimainkan.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut :

1. Sistem dapat membuat pemodelan pemain dan menentukan tingkat kesulitan dari game secara otomatis berdasarkan kemampuan pemain dalam menyelesaikan permainan. Terdapat 4 kelas tingkat kesulitan yang diusulkan dalam penelitian ini : "BEGINNER", "MEDIUM", "HARD" dan "VERY HARD"
2. Sistem *ARD* menggunakan algoritma *Fuzzy-Sugeno* pada game bertema Wayang Ramayana yang dapat menghasilkan pemodelan pemain. Dimana model pemain tersebut akan digunakan untuk penghitungan pemodelan selanjutnya dan untuk merubah performa objek-objek yang ada didalam game seperti Enemy atau NPC.

2. Saran

Penelitian ini menghasilkan sebuah *game prototype* bernama "RAMAYANA". Namun masih terdapat kekurangan dalam penelitian ini yang dapat dikembangkan dalam penelitian selanjutnya. Saran untuk penelitian selanjutnya yaitu sebagai berikut :

1. Penelitian ini menggunakan metode yang sederhana dan intuitif. Pada penelitian berikutnya dapat dilakukan dengan menggunakan algoritma yang lebih cerdas atau akurat seperti *Neural-Net* ataupun kombinasi dari *Neural-Net* dan *Fuzzy (ANN)*.
2. Dalam pemodelan pemain, memerlukan resource yang cukup besar untuk penghitungan parameter

output. Diperlukan metode yang lebih ringan dari metode yang dipakai pada penelitian ini dalam hal penggunaan memori prosessor.

DAFTAR PUSTAKA

[1] N. E. Wardani and E. Widiyastuti, "Mapping Wayang Traditional Theatre as A Form of Local Wisdom of Surakarta Indonesia," *Asian Journal of Social Sciences & Humanities*, vol. 2, pp. 314-321, 2013.

[2] W. Y. A. Pratama and A. Zpalanzani, "PERANCANGAN TRADING CARD GAME WAYANG "WAYANG WARFARE"," *Jurnal Tingkat Sarjana bidan Senirupa dan Desain*, pp. 1-7, 2012.

[3] A. K. Nugraha1, K. I. Satoto and R. Kridalukmana, "Perancangan Permainan Gelembung Huruf (Tokoh Wayang) Berbasis Sistem Operasi IOS Menggunakan Gamesalad," Universitas Diponegoro, Semarang, 2012.

[4] A. M. Hussaan, K. Sehaba and A. Mille, "Tailoring Serious Games with Adaptive Pedagogical Scenarios," in *International Conference on Advanced Learning Technologies*, Lyon, 2011.

[5] R. Lopes and R. Bidarra, "Adaptivity Challenges in Games and Simulations : A Survey," *IEEE Transactions on Computational Intelligence And Ai in Games*, pp. 85-99, 2011.

[6] G. N. Yannakakis, P. Spronck, D. Loiacono and E. Andre, "Player Modeling," Dagstuhl Publishing, 2013.

[7] N. Peirce, O. Conlan and V. Wade, "Adaptive Educational Games: Providing Non-invasive Personalised Learning Experiences," in *Second IEEE International Conference on Digital Games and Intelligent Toys Based Education*, Dublin, 2008.

[8] E. M. Carneiro and A. M. Cunha, "An Adaptive Game AI Architecture," *SBC - Proceedings of SBGames*, pp. 21-24, 2012.

[9] E. Tron and M. Margaliot, "Mathematical modeling of observed natural behavior : a fuzzy logic approach," *Fuzzy Sets and Systems*, pp. 437-450, 2004.

[10] P. A. Nogueira, R. Aguiar, R. Rodrigues, E. Oliveira and L. E. Nacke, "Fuzzy Affective Player Models : A Physiology-Based Hierarchical Clustering Method," in *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2014)*, Ontario, 2014.

[11] E. L.-C. Law and M. D. Rust-Kickmeier, "80Days: Immersive Digital Educational Games with Adaptive Storytelling," University of Graz, 2008.

[12] A. Hallengren and M. Svensson, "Dynamic difficulty adjustment for roleplaying games," Blekinge Institute of Technology, Sweden, 2013.

[13] L. Ermi and F. Mayra, "Fundamental Components of the Gameplay Experience : Analysing Immersion," in *Digital Games Research Association's Second International Conference*, 2005.

[14] S. Kurniawan Sutanto, "Dynamic Difficulty Adjustment in Game Based On Type of Player with Anfis Method," *Journal of Theoretical and Applied Information Technology*, pp. 254-260, 2005.

[15] B. B. P. L. d. Araujo and B. Feijo, "Evaluating dynamic difficulty adaptivity in shoot'em up games," in *SBC - Proceedings of SBGames 2013*,

Saou Paulo, 2013.