Dynamic Difficulty Adjustment Berbasis Logika Fuzzy Untuk Procedural Content Generation Pada Permainan Roguelike

David Saputra Octadianto Soedargo[1,2*], Hartarto Junaedi[3]

[1] Program Studi Teknik Informatika, Institut Informatika Indonesia, Surabaya, Jawa Timur
[2] Program Studi Teknologi Informasi, Institut Sains dan Teknologi Terpadu Surabaya, Jawa Timur
[3] Program Studi Sistem Informasi, Institut Sains dan Teknologi Terpadu Surabaya, Jawa Timur
Email: [1*] david.wolfhart@gmail.com, [2] hartarto.j@gmail.com

(Naskah masuk: 12 Apr 2022, direvisi: 24 Mei 2022, 8 Jun 2022, diterima: 16 Jun 2022)

Abstrak

Perkembangan industri *video game* sangatlah pesat hingga ada banyak sekali orang yang memainkan *video game*. Setiap orang memiliki tingkat keterampilan yang berbeda dan memiliki kurva belajar yang unik untuk setiap *video game* yang mereka mainkan. Pengembang *video game* pada umumnya memberikan tingkat kesulitan yang bersifat statis sehingga tingkat kesulitan pemain pemula dengan pemain yang berpengalaman itu serupa. Hal ini menimbulkan *video game* menjadi tidak seimbang. *Game balancing* adalah salah satu aspek penting yang dapat meningkatkan minat seseorang untuk memainkan *game* tersebut dan secara adaptif dapat memberikan tingkat kesulitan yang dinamis bagi setiap pemain. Penelitian ini mengintegrasikan *Dynamic Difficulty Adjustment* (DDA) berbasis logika *fuzzy* untuk *Procedural Content Generation* (PCG) pada permainan *roguelike* sehingga dapat menghasilkan tingkat kesulitan yang dinamis. DDA akan mengolah parameter input dari keterampilan pemain ketika menyelesaikan map sebelumnya, seperti lama waktu, sisa darah, banyaknya serangan yang diterima, sisa peluru, akurasi pemain, dan jumlah musuh. Hasil dari DDA akan diolah kembali dengan menggunakan PCG untuk membuat map baru secara prosedural, seperti besar ruangan, jumlah musuh, jumlah item penyembuh, jumlah item peluru, dan jumlah tembok. Hal ini diharapkan dapat menciptakan penyesuaian kesulitan yang dinamis pada setiap map sesuai dengan keterampilan dari pemain. Dalam penelitian ini dilakukan juga perbandingan *video game* dengan tingkat kesulitan yang statis dan dengan tingkat kesulitan yang dinamis untuk dapat mengukur tingkat kepuasan dari pemain. Dari 30 responden, didapatkan hasil bahwa 80% pemain memiliki tingkat kepuasan yang lebih baik ketika memainkan *video game* dengan tingkat kesulitan yang dinamis.

Kata Kunci: *Dynamic Difficulty Adjustment, Fuzzy Logic, Procedural Content Generation, Game Balancing.*

Dynamic Difficulty Adjustment Based on Fuzzy Logic for Procedural Content Generation in Roguelike Games

Abstract

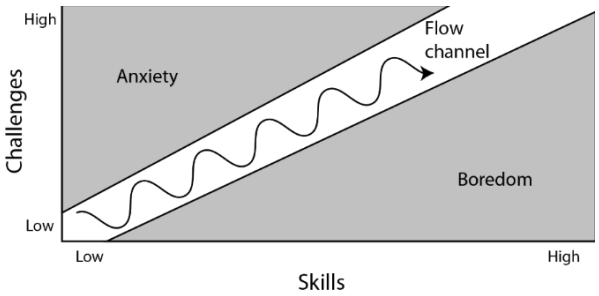
The development of video game industry is so fast that many people play video games nowadays. Everyone has a different skill level and has a unique learning curve for each video game they played. Video game developers generally provide a static difficulty level which means the difficulty level of beginners and experienced players is similar. This causes the video game becomes unbalanced. Game balancing is one of the important aspects that can increase one's interest in playing the game and can adaptively provide a dynamic level of difficulty for each player. This study integrates Dynamic Difficulty Adjustment (DDA) based on fuzzy logic for Procedural Content Generation (PCG) so that it can determine the dynamic difficulty level. The DDA will process the input parameters of the player's skills when completing the previous map, such as the length of time, player's remaining health, player's damage taken, player's remaining bullets, player's accuracy, and number of enemies. The results of the DDA will be reprocessed using PCG to create a new map procedurally, such as the size of room, number of enemies, number of healing items, number of bullet items, and number of walls. This is expected to create dynamic difficulty adjustments on each map according to the skills of the player. In this study, a comparison of video games with a static difficulty level and a dynamic difficulty level was also carried out to measure the level of satisfaction of players. From 30 respondents, it was found that 80% of players had a better level of satisfaction when playing video games with dynamic difficulty levels.

Keywords: *Dynamic Difficulty Adjustment, Fuzzy Logic, Procedural Content Generation, Game Balancing.*

I. PENDAHULUAN

Perkembangan industri *video game* saat ini sangatlah pesat sehingga sejumlah besar *game* dapat diidentifikasi dalam sejumlah genre dan dapat dimainkan di berbagai *platform*. Setiap orang memiliki tingkat keterampilan yang berbeda dan memiliki kurva belajar yang unik untuk setiap *video game* yang mereka mainkan. Dengan demikian menjadi sulit untuk dapat menghibur setiap pemain jika pengembang *video game* menentukan tingkat kesulitan yang sama dalam suatu *game*. Pengembang *game* pada umumnya memberikan tingkat kesulitan yang bersifat statis dan dirancang untuk menargetkan sekumpulan demografis pemain yang umum sehingga dapat menghalangi pengalaman bermain *game* untuk pemain yang tidak dapat diidentifikasi dalam demografis tersebut. Hal ini menimbulkan kebutuhan untuk mengembangkan teknologi dan teknik yang memungkinkan pengembang *video game* untuk membuat *game* yang adaptif.

Sejumlah penelitian telah mengatasi masalah tingkat statis dengan mengusulkan teknik *Dynamic Difficulty Adjustment* (DDA) yang memungkinkan pengelolaan dan pemetaan pengalaman serta keterampilan bermain dari setiap pemain [1]. DDA adalah teknik untuk dapat menyesuaikan skenario, parameter, dan perilaku pemain secara otomatis sehingga dapat mengikuti keterampilan pemain. DDA dibangun diatas *Theory of Flow* oleh Mihaly Csikszentmihalyi yang dapat dilihat pada Gambar 1.



Gambar 1. Konsep *Flow Channel* Diusulkan Oleh Csikszentmihalyi

Flow channel menggambarkan perasaan pemain secara umum yang dibawa oleh rasa tantangan dan keterampilan yang seimbang. Suatu *game* dapat dikatakan seimbang jika tantangan yang diberikan dapat menyesuaikan dengan keterampilan dari pemain. Suatu *game* dapat dikatakan tidak seimbang jika tantangan yang diberikan terlalu tinggi dibandingkan dengan keterampilan dari pemain terlalu rendah sehingga dapat membuat pemain menjadi frustasi, atau jika keterampilan dari pemain terlalu tinggi dibandingkan dengan tantangan yang diberikan terlalu rendah sehingga dapat membuat pemain menjadi bosan. Dengan demikian kita dapat mengetahui bahwa *game balancing* merupakan aspek yang penting untuk dapat meningkatkan minat seseorang untuk memainkan *game* tersebut. DDA dapat membuat pemain tetap terlibat dan terhibur sesuai dengan *teori of flow* [2].

DDA harus dapat memenuhi tiga kebutuhan dasar *video game* sebagai berikut [3]:

- Permainan ini perlu melacak kemampuan pemain secara otomatis dan beradaptasi dengan cepat.
- Permainan harus mengikuti peningkatan atau penurunan level pemain dan menjaga keseimbangan sesuai dengan keterampilan pemain.
- Proses adaptasi tidak boleh dipahami dengan jelas oleh para pemain, dan status permainan yang berurutan harus memiliki koherensi dengan yang sebelumnya.

Penelitian ini mengkaji bagaimana mengintegrasikan *Dynamic Difficulty Adjutment* (DDA) berbasis logika *fuzzy* untuk *Procedural Content Generation* (PCG) pada *game roguelike*. Logika *fuzzy* adalah logika yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar atau salah. Logika *fuzzy* adalah bentuk logika dengan nilai kebenaran bernilai bilangan real berkisar antara 0 dan 1 yang dikenal dengan *fuzzy* (kabur) [4]. Logika *fuzzy* dipilih karena terdapat kemiripan bahasa linguistik yang digunakan dalam menentukan keterampilan pemain ketika memainkan suatu *game* [5]. Salah satu manfaat dari logika fuzzy adalah kesederhanaan dari formulasinya [6]. Untuk dapat menghasilkan keputusan yang tepat dalam pembuatan konten pada level selanjutnya, maka dibutuhkan *input* berupa keterampilan pemain dalam menyelesaikan level sebelumnya. Data *input* yang didapatkan masih terlihat samar, oleh karena itu diperlukan sistem inferensi *fuzzy* untuk dapat merumuskan aturan yang sesuai berdasarkan keputusan yang dibuat.

Procedural Content Generation di industri *video game* adalah teknik pembuatan konten *game* secara otomatis atau semi-otomatis dengan menggunakan algoritma [7]. Hal ini dapat memberikan konten tak terduga dan bervariasi dalam kisaran yang dapat diterima dan dikendalikan. Beberapa konten *game* yang dibuat dapat mempengaruhi *gameplay* seperti jumlah musuh, jumlah item, map, dan masih banyak lagi [8]. Salah satu alasan utama menggunakan PCG dalam pembuatan *video game* adalah untuk menghemat waktu dan sumber daya untuk pengembangan *game*. Selain itu, konten yang dihasilkan menggunakan PCG dapat disesuaikan dengan selera dan kebutuhan pemain dengan cara menggabungkan teknik PCG dengan teknik lainnya seperti DDA.

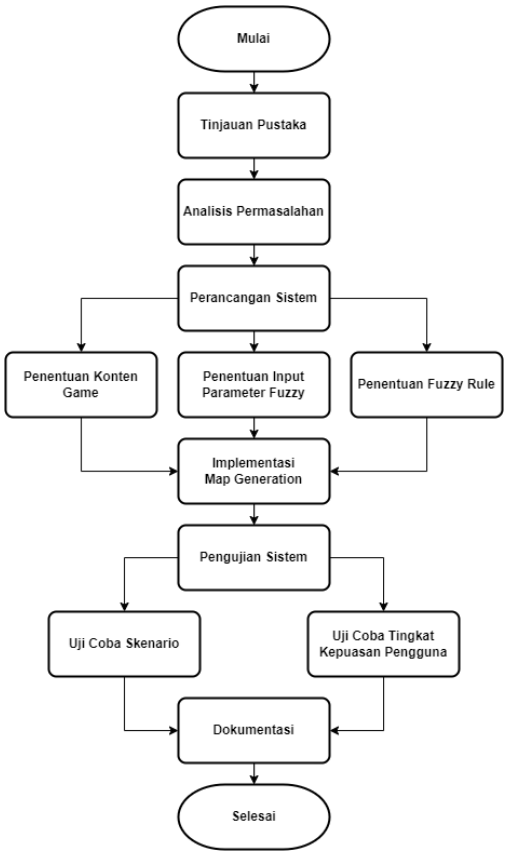
Roguelike adalah salah satu genre *video game* dimana mengharuskan pemainnya dapat bermain secara efisien dengan meminimalisir resiko hilangnya *progress* permainan [9]. Tingkat kesulitan pada permainan ini terletak pada bagaimana pemain dapat mengatur langkah dan sumber daya yang terbatas untuk dapat menyelesaikan suatu level dalam permainan agar dapat melanjutkan ke level selanjutnya. Genre *roguelike* ini dipilih karena fakta bahwa genre ini dapat memiliki map atau level yang tak terbatas dan dapat dibuat dinamis dengan menggunakan teknik PCG [10]. Selain itu, dengan menggunakan teknik DDA maka konten yang akan dibuat dapat disesuaikan dengan keterampilan dari pemain saat menyelesaikan map sebelumnya.

Pengujian pada penelitian ini akan dilakukan dengan uji coba pengguna memainkan *game*, lalu pengguna akan mengisikan *survey* berbasis kuisioner untuk mengukur tingkat kepuasan pengguna terhadap *game* yang dikembangkan. Hasil dari penelitian ini diharapkan dapat memberikan solusi bagi pengembang dan industri *video game* untuk pembuatan konten

di dalam *game* secara prosedural dengan tingkat kesulitan yang dinamis menyesuaikan kemampuan pemain. Selain itu, berdasarkan pada penelitian-penelitian sebelumnya yang juga membahas tentang peran DDA pada *video game*, maka penelitian ini juga dapat menjadi fakta bahwa DDA berbasis logika *fuzzy* dapat menciptakan tingkat kesulitan yang dinamis sesuai dengan kemampuan pemain dalam membuat konten pada permainan secara prosedural [11]. Hal ini akan didukung oleh hasil uji coba tingkat kepuasan pengguna sebagai bukti bahwa *game balancing* adalah aspek yang penting dalam pengembangan *game* [12, 13].

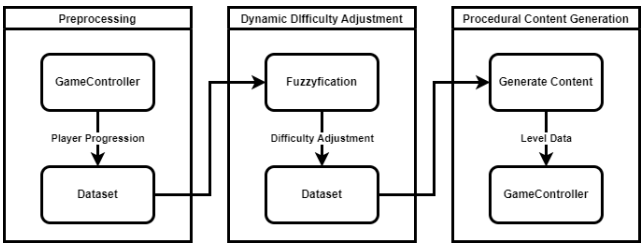
II. METODOLOGI PENELITIAN

Proses pengembangan penelitian ini terbagi atas beberapa tahapan yang dapat dilihat pada Gambar 2 berikut ini:



Gambar 2. Metodologi Penelitian

Penelitian ini diawali dengan tahapan studi pustaka yang dilakukan dengan cara mempelajari teori-teori literatur dari buku-buku referensi dan jurnal penelitian yang berkaitan dengan permasalahan dalam penelitian ini. Selanjutnya dilakukan analisis permasalahan dan menentukan batasan pada penelitian. Tahapan selanjutnya adalah perancangan sistem untuk akan menentukan teknik dan pendekatan apa yang digunakan agar dapat memberikan solusi terhadap permasalahan yang diangkat pada penelitian ini. Tahapan ini terbagi dalam tiga proses, yaitu penentuan konten pada *game* yang akan diteliti, penentuan *input* parameter *fuzzy* yang digunakan, dan penentuan *fuzzy rule*. Tahapan selanjutnya adalah mengimplementasikan sistem yang telah ditentukan sebelumnya untuk proses *map generation*. Sistem ini dapat dilihat pada Gambar 3 diagram blok.



Gambar 3. Diagram Blok

Terdapat tiga langkah implementasi yaitu *preprocessing*, *Dynamic Difficulty Adjustment* (DDA), dan *Procedural Content Generation* (PCG). Pada awal *preprocessing*, *GameController* akan menyimpan *progress* dari pemain ke dalam *dataset*. *Progress* yang disimpan terdiri dari lama waktu bermain, sisa darah dari pemain, banyaknya luka yang diterima pemain, sisa peluru dari pemain, akurasi serangan dari pemain, dan banyaknya jumlah musuh pada level ini. *Dataset* akan diolah pada proses DDA dengan melakukan *fuzzifikasi* terlebih dahulu terhadap *dataset*. Hasil *fuzzifikasi* akan diolah kembali dengan teknik DDA untuk menghasilkan *dataset* pada proses PCG. *Dataset* yang dihasilkan adalah parameter untuk konten pada level selanjutnya, seperti besar ruangan, banyaknya musuh, banyaknya *item* penyembuh, dan banyaknya *item* peluru. Proses PCG akan mengolah *dataset* untuk membuat level selanjutnya. Level yang dihasilkan ini akan dikembalikan ke *GameController* agar dapat dimainkan oleh pemain.

Tahapan selanjutnya adalah pengujian sistem. Tahapan ini terbagi menjadi dua proses, yaitu uji coba skenario permainan dan uji coba kuisioner tingkat kepuasan pengguna. Uji coba skenario permainan dilakukan dengan mencoba beberapa skenario untuk menguji kemampuan sistem dalam membuat level yang sesuai. Sedangkan uji coba kuisioner dilakukan terhadap responden yang akan mencoba memainkan *game* yang dikembangkan lalu mengisikan kuisioner untuk dapat mengukur tingkat kepuasan pengguna. Tahapan terakhir adalah mendokumentasikan hasil penelitian dan pengujian. Hasil dari pengujian akan dianalisa untuk menghasilkan kesimpulan dan saran dari penelitian.

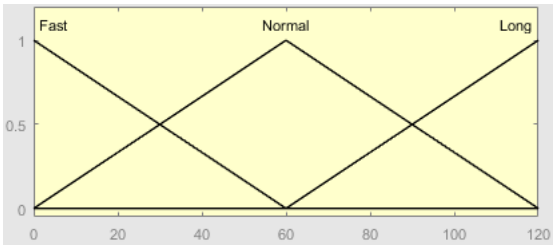
III. HASIL DAN PEMBAHASAN

A. *Input Fuzzy*

Untuk dapat menghasilkan keputusan yang tepat dalam pembuatan konten pada level selanjutnya, maka dibutuhkan *input* berupa keterampilan pemain dalam menyelesaikan level sebelumnya. Data *input* yang didapatkan masih terlihat samar, oleh karena itu diperlukan sistem inferensi *fuzzy* untuk dapat merumuskan aturan yang sesuai berdasarkan keputusan yang dibuat [14, 15]. Data *input* parameter yang akan digunakan adalah *time*, *health*, *damage taken*, *bullet*, *accuracy*, dan *enemy*. Parameter serta derajat keanggotaan dari parameter ini ditentukan berdasarkan dari beberapa contoh penelitian

terdahulu. Berikut ini adalah grafik fungsi keanggotaan *fuzzy* untuk setiap *input* parameter yang akan diteliti.

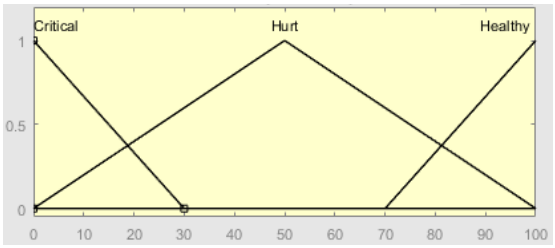
1. *Time*



Gambar 4. Derajat Keanggotaan *Time*

Input parameter *time* adalah lama waktu dari pemain ketika menyelesaikan level sebelumnya. Derajat keanggotaan parameter ini dikelompokkan menjadi tiga anggota, yaitu *fast*, *normal*, dan *long*.

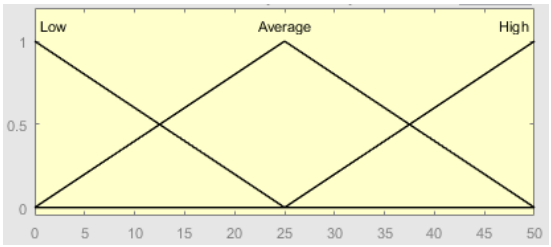
2. *Health*



Gambar 5. Derajat Keanggotaan *Heallth*

Input parameter *health* adalah sisa darah dari pemain ketika menyelesaikan level sebelumnya. Derajat keanggotaan parameter ini dikelompokkan menjadi tiga anggota, yaitu *critical*, *hurt*, dan *healthy*.

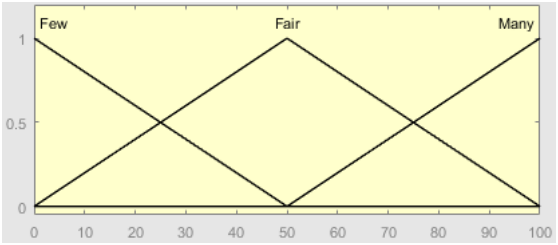
3. *Damage Taken*



Gambar 6. Derajat Keanggotaan *Damage Taken*

Input parameter *damage taken* adalah banyaknya serangan yang diterima pemain ketika menyelesaikan level sebelumnya. Derajat keanggotaan parameter ini dikelompokkan menjadi tiga anggota, yaitu *low*, *average*, dan *high*.

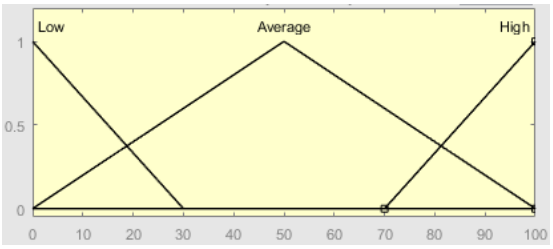
4. *Bullet*



Gambar 7. Derajat Keanggotaan *Bullet*

Input parameter *bullet* adalah banyaknya peluru yang ditembakkan pemain ketika menyelesaikan level sebelumnya. Derajat keanggotaan parameter ini dikelompokkan menjadi tiga anggota, yaitu *few*, *fair*, dan *many*.

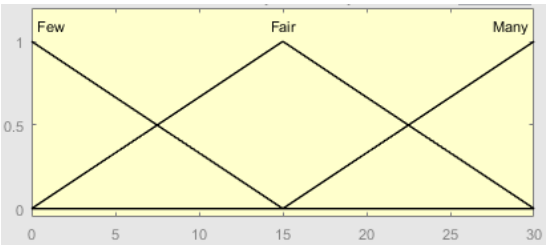
5. *Accuracy*



Gambar 8. Derajat Keanggotaan *Accuracy*

Input parameter *accuracy* adalah banyaknya peluru dari pemain yang mengenai musuh. Derajat keanggotaan parameter ini dikelompokkan menjadi tiga anggota, yaitu *low*, *average*, dan *high*.

6. *Enemy*



Gambar 9. Derajat Keanggotaan *Enemy*

Input parameter *enemy* adalah banyaknya musuh yang dilawan pemain ketika menyelesaikan level sebelumnya. Derajat keanggotaan parameter ini dikelompokkan menjadi tiga anggota, yaitu *few*, *fair*, dan *many*.

B. *Fuzzy Rule*

Aturan *fuzzy* digunakan dalam logika *fuzzy* untuk menyimpulkan *output* berdasarkan parameter *input*. Terdapat 3 aturan yang digunakan pada penelitian ini, yaitu aturan *enemy spawn rule* digunakan untuk menentukan jumlah musuh yang akan dibuat pada level selanjutnya, aturan *health item spawn rule* digunakan untuk menentukan jumlah *health item*

yang akan dibuat pada level selanjutnya, dan aturan *ammo item spawn rule* digunakan untuk menentukan jumlah *ammo item* yang akan dibuat pada level selanjutnya. Berikut ini adalah tabel-tabel aturan *fuzzy* pada penelitian ini.

1. *Enemy Spawn*

Tabel 1. *Enemy Spawn Rule*

No	Input												Output
	Time			Damage Taken			Accuracy			Enemy			Enemy Spawn
	F	N	L	L	A	H	L	A	H	Fe	Fa	M	
1				✓	✓			✓	✓	✓			Increase
2	✓			✓					✓	✓	✓		Increase
3		✓		✓					✓		✓		Increase
4		✓				✓	✓				✓		Decrease
5			✓			✓	✓				✓	✓	Decrease
6					✓	✓	✓	✓				✓	Decrease

Aturan pada Tabel 1 digunakan untuk menentukan jumlah musuh yang akan dibuat pada level selanjutnya. Terdapat 4 *input* parameter yang digunakan, yaitu parameter *time* dengan derajat keanggotaannya F = *fast*, N = *normal*, dan L = *long*, parameter *damage taken* dengan derajat keanggotaannya L = *low*, A = *average*, dan H = *high*, parameter *accuracy* dengan derajat keanggotaannya L = *low*, A = *average*, dan H = *high*, dan parameter *enemy* dengan derajat keanggotaannya Fe = *few*, Fa = *fair*, dan M = *many*. *Output* parameter dari aturan ini ada 2, yaitu *increase* dimana jumlah musuh akan ditambahkan, dan *decrease* dimana jumlah musuh akan dikurangi.

2. *Health Item Spawn*

Tabel 2. *Health Item Spawn Rule*

No	Input									Output
	Health			Damage Taken			Enemy			Health Item Spawn
	C	Hu	He	L	A	H	Fe	Fa	M	
1	✓									Increase
2						✓		✓	✓	Increase
3		✓			✓	✓		✓	✓	Increase
4				✓			✓	✓		Decrease
5		✓		✓	✓		✓	✓		Decrease
6			✓							Decrease

Aturan pada Tabel 2 digunakan untuk menentukan jumlah *health item* yang akan dibuat pada level selanjutnya. Terdapat 3 *input* parameter yang digunakan, yaitu parameter *health* dengan derajat keanggotaannya C = *critical*, Hu = *hurt*, dan He = *healthy*, parameter *damage taken* dengan derajat keanggotaannya L = *low*, A = *average*, dan H = *high*, dan parameter *enemy* dengan derajat keanggotaannya Fe = *few*, Fa = *fair*, dan M = *many*. *Output* parameter dari aturan ini ada 2, yaitu *increase* dimana jumlah *health item* akan ditambahkan, dan *decrease* dimana jumlah *health item* akan dikurangi.

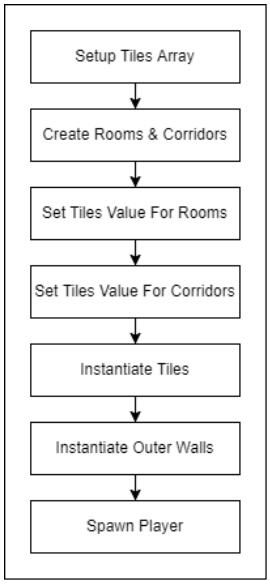
3. *Ammo Item Spawn*

Tabel 3. *Ammo Item Spawn Rule*

No	Input									Output
	Bullet			Accuracy			Enemy			Ammo Item Spawn
	Fe	Fa	M	L	A	H	Fe	Fa	M	
1	✓									Increase
2				✓	✓			✓	✓	Increase
3		✓			✓				✓	Increase
4		✓			✓		✓			Decrease
5					✓	✓	✓	✓		Decrease
6			✓							Decrease

Aturan pada Tabel 3 digunakan untuk menentukan jumlah *ammo item* yang akan dibuat pada level selanjutnya. Terdapat 3 *input* parameter yang digunakan, yaitu parameter *bullet* dengan derajat keanggotaannya Fe = *few*, Fa = *fair*, dan M = *many*, parameter *accuracy* dengan derajat keanggotaannya L = *low*, A = *average*, dan H = *high*, dan parameter *enemy* dengan derajat keanggotaannya Fe = *few*, Fa = *fair*, dan M = *many*. *Output* parameter dari aturan ini ada 2, yaitu *increase* dimana jumlah *ammo item* akan ditambahkan, dan *decrease* dimana jumlah *ammo item* akan dikurangi.

C. *Map Generation*



Gambar 10. Diagram Blok *Map Generation*

Map Generation adalah proses dimana membuat *map* secara prosedural menggunakan PCG berdasarkan hasil dari *fuzzy rule* yang telah dibahas sebelumnya. Konten yang akan dibuat adalah *map* pada level tersebut yang terdiri dari *room* atau jumlah ruangan pada *map*, *enemy zombie* atau jumlah musuh pada setiap *room*, *health item* atau jumlah *item* penyembuh, *ammo item* atau jumlah *item* peluru, dan *wall* atau tembok.

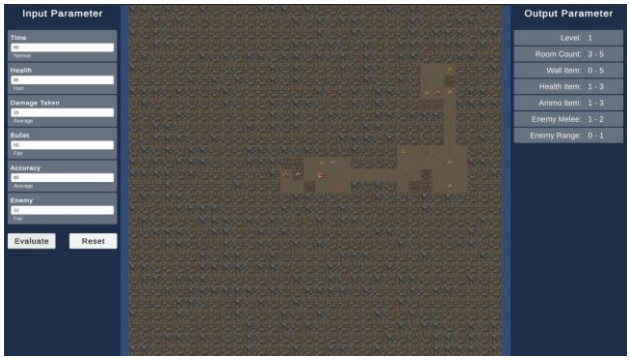
Alur untuk membuat *map* secara prosedural dapat dilihat pada gambar 10. Berikut ini adalah penjelasan dari masing-masing langkah:

1. *SetupTilesArray*
   Sistem menyiapkan *array* 2D untuk menampung besar *map* berdasarkan ukuran *width* dan *height* yang telah ditentukan oleh DDA. *Tiletype* adalah enumerasi yang dibuat sebagai penanda peletakan konten nantinya. *Tiletype* terdiri dari *floor*, *wall*, *item*, dan *enemy*. *Tiles* adalah variabel yang akan digunakan untuk menampung *map* yang akan dibuat.
2. *CreateRoomsAndCorridors*
   Sistem akan membuat objek *rooms* dan *corridors*. *Room* adalah ruangan atau wilayah dimana *enemy*, *item*, dan *wall* akan dibuat didalam disana. Banyaknya *room* yang dibuat berdasarkan *roomCount* yang telah ditentukan oleh DDA. Besar *width* dan *height* dari tiap *room* juga telah ditentukan oleh DDA. *Corridor* adalah lorong yang menghubungkan dua atau lebih *room*.
3. *SetTilesValuesForRooms*
   Sistem akan menentukan letak koordinat setiap *tiles* pada setiap *room*. *Tiles* yang ditentukan adalah *wall*, *item*, dan *enemy*. Banyaknya *tiles* tersebut telah ditentukan oleh DDA dan diberikan sedikit unsur acak (*random*) agar jumlahnya dapat bervariasi tiap *room*.
4. *SetTilesValuesForCorridors*
   Sistem akan menentukan letak koordinat setiap *corridor* pada setiap *room*. *Corridor* yang dibuat akan memiliki arah hadap (*north*, *east*, *south*, dan *west*), dan panjang dari *corridor* tersebut. *Tile* pada *corridor* ditentukan sebagai *floor*.
5. *InstantiateTiles*
   Sistem akan membuat objek-objek dari *tiles* yang telah ditentukan sebelumnya. Iterasi i dan j adalah titik koordinat pada *map*. Sistem akan membaca pada array *tiles* lalu membuat objek di *board* berdasarkan *array prefab objects*.
6. *InstantiateOuterWalls*
   Sistem akan membuat objek *outer walls* diluar ukuran map sebagai pembatas agar pemain tidak dapat keluar melebihi ukuran *map* yang telah ditentukan.
7. *SpawnPlayer*
   Sistem akan meletakkan karakter pemain pada salah satu *room* secara acak.

D. Uji Coba

Uji coba dilakukan dengan dua proses, yaitu uji coba skenario permainan dan uji coba tingkat kepuasan pengguna. Uji coba skenario dilakukan untuk menguji kemampuan sistem dalam membuat level yang sesuai, sedangkan uji coba tingkat kepuasan pengguna dilakukan untuk menguji hipotesa penelitian dan menghasilkan kesimpulan serta saran dari penelitian.

Uji coba skenario permainan dilakukan dengan mencoba beberapa skenario untuk menguji kemampuan sistem dalam membuat level yang sesuai. Berikut ini penjelasan beberapa skenario yang diujikan beserta penjelasannya.

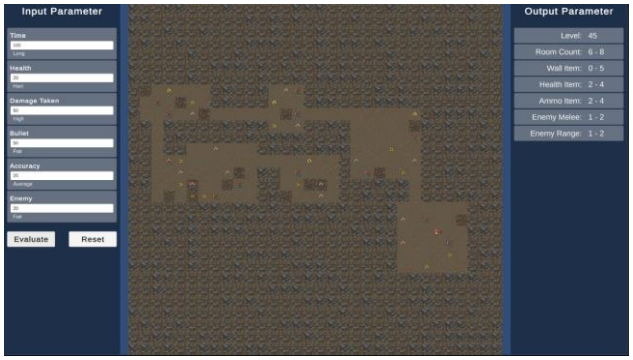


Gambar 11. Skenario Awal Permainan

Gambar 11 adalah contoh skenario awal pada level 1. Pada level ini, *map* yang dibuat menggunakan parameter awal yang sudah ditentukan dikarenakan masih belum ada parameter input keterampilan pemain yang dapat diolah. Setelah permain menyelesaikan level ini, DDA akan bekerja mengolah input keterampilan pemain untuk membuat level selanjutnya. Meski tidak memiliki data yang diolah, sistem tetap dapat membuat *map* secara acak, sehingga level 1 pun dapat menghasilkan bentuk *map* yang beragam.

Gambar 12. Skenario Permainan Semakin Sulit

Gambar 12 adalah contoh skenario yang telah dilakukan uji coba pembuatan level hingga ke level 39. Angka level 39 dipilih secara acak sebagai contoh skenario permainan yang telah berjalan dari skenario sebelumnya level 1 hingga ke 39 dengan diberikan *input* parameter keterampilan pemain yang mahir. *Input* parameter keterampilan pemain yang diberikan yaitu *time* 60 (normal), *health* 25 (*hurt*), *damage taken* 20 (*average*), *bullet* 10 (*few*), *accuracy* 90 (*high*), *enemy* 30 (*many*). *Input* ini akan diolah dengan algoritma *fuzzy* dan berdasarkan *fuzzy rule* yang telah ditetapkan, maka menghasilkan *output* parameter yaitu *room count* 11-13, *wall item* 0-5, *health item* 3-5, *ammo item* 1-3, *enemy melee* 2-3, *enemy range* 0-1. Berdasarkan hasil tersebut akan diolah kembali oleh untuk membuat konten secara procedural seperti *wall*, *health item*, *ammo item*, dan *enemy*. Level yang dihasilkan akan semakin sulit dilihat dari *input* parameter keterampilan pemain pada skenario ini dimana pemain cukup ahli dalam menyelesaikan *map* sebelumnya.



Gambar 13. Skenario Permainan Semakin Mudah.

Gambar 13 adalah skenario yang telah dilakukan uji coba pembuatan level hingga ke level 45. Angka level 45 dipilih secara acak sebagai contoh skenario permainan yang telah berjalan dari skenario sebelumnya level 39 hingga level 45 dengan diberikan *input* parameter keterampilan pemain yang kurang mahir. *Input* parameter keterampilan pemain yang diberikan yaitu *time* 100 (*long*), *health* 25 (*hurt*), *damage taken* 50 (*high*), *bullet* 50 (*fair*), *accuracy* 20 (*average*), *enemy* 20 (*fair*). *Input* ini akan diolah dengan algoritma *fuzzy* dan berdasarkan *fuzzy rule* yang telah ditetapkan, maka menghasilkan *output* parameter yaitu *room count* 6-8, *wall item* 0-5, *health item* 2-4, *ammo item* 2-4, *enemy melee* 1-2, *enemy range* 1-2. *Map* yang dihasilkan akan semakin mudah dilihat dari input parameter keterampilan pemain pada skenario ini dimana pemain kurang ahli dalam menyelesaikan *map* sebelumnya.

Berdasarkan hasil skenario yang telah diuji sebelumnya, sistem yang dikembangkan dapat menghasilkan level yang dinamis sesuai dengan *input* parameter keterampilan pemain. Jika pemain yang mahir dalam menyelesaikan level, maka level selanjutnya akan dibuat lebih sulit. Jika pemain merasa kesulitan atau kurang mahir, maka level selanjutnya akan dibuat lebih mudah. Hal ini akan dapat menciptakan *game balancing* untuk pemain yang mahir dengan yang tidak mahir. Selain itu sistem yang dikembangkan juga dapat menciptakan aliran yang seimbang, sehingga tidak membuat pemain menjadi frustrasi jika setiap level semakin sulit atau membuat pemain menjadi bosan jika setiap level semakin mudah.

Tahap pengujian selanjutnya adalah uji coba tingkat kepuasan pengguna. Uji coba ini dilakukan terhadap 30 orang responden. Masing-masing responden akan mencoba *game* dengan mode normal dan mode dinamis. *Game* dengan mode normal adalah *game* yang dikembangkan tanpa DDA dimana setiap level selanjutnya dibuat lebih sulit dari level sebelumnya. Tidak ada penyesuaian tingkat kesulitan pada mode ini. *Game* dengan mode dinamis adalah *game* yang dikembangkan dengan DDA dimana setiap level selanjutnya dibuat menyesuaikan dengan keterampilan pemain pada level sebelumnya. Setelah mencoba, masing-masing responden akan diberikan kuisioner untuk mengukur tingkat kepuasan pengguna dalam memainkan *game* dengan mode normal dan mode dinamis. Hasil dari kuisioner ini akan dievaluasi dan menghasilkan kesimpulan serta saran dari penelitian. Berikut ini adalah hasil dari kuisioner.

Tabel 4. Frekuensi Responden Terhadap Mode yang Disukai

No	Mode Yang Disukai	Frekuensi	Persentase	Kumulatif Persen
1	Mode normal	6	20,0%	20,0%
2	Mode dinamis	24	80,0%	100,0%

Berdasarkan Tabel 4 dapat diketahui bahwa responden yang telah mencoba *game* dengan mode normal dan dinamis lebih memilih mode dinamis yang paling disukai dengan frekuensi 80%. Sebanyak 20% responden memilih mode normal yang lebih disukai karena menurut mereka lebih banyak tantangan karena tiap level pasti lebih sulit daripada sebelumnya. Hal ini membuktikan bahwa tantangan dalam permainan tidak dapat menjadi tolak ukur bahwa permainan itu dapat disukai oleh setiap pemain, melainkan keseimbangan tantangan pada permainan.

Tabel 5. Hasil Evaluasi Kuisioner

Jenis Evaluasi Permainan	Jawaban					Skor	Skor Akhir
	1	2	3	4	5	T*Pn	(T*Pn) / (n * 5)
Tingkat kesenangan mode normal	3	7	7	9	4	94	62,67%
Tingkat kemudahan mode normal	11	6	7	6	0	68	45,33%
Tingkat kesenangan mode dinamis	1	1	3	10	15	127	84,67%
Tingkat kemudahan mode dinamis	0	0	6	11	13	127	84,67%

Berdasarkan pada Tabel 5 hasil evaluasi diatas dapat ditarik kesimpulan bahwa *game* dengan mode yang normal memiliki tingkat kesenangan sebesar 62.67% dan tingkat kemudahannya sebesar 45.33%. Sedangkan *game* dengan mode yang dinamis memiliki tingkat kesenangan sebesar 84.67% dan tingkat kemudahannya sebesar 84.67%. Hal ini membuktikan bahwa *game* dengan mode dinamis lebih disenangi dan lebih mudah dibandingkan dengan *game* dengan mode normal atau tanpa menggunakan algoritma DDA.

IV. KESIMPULAN

Berdasarkan hasil pengujian yang telah dilakukan dapat diperoleh beberapa kesimpulan, yang pertama adalah *Dynamic Difficulty Adjustment* berbasis logika *Fuzzy* untuk *Procedural Content Generation* dapat menciptakan *game balancing* sesuai dengan keterampilan dari pemain. Kesimpulan kedua adalah tingkat kepuasan pemain dalam bermain *game* juga dapat dipengaruhi oleh keseimbangan pada *game* tersebut. Hal ini

didukung oleh hasil kuisioner yang membuktikan bahwa 80% pemain lebih menyukai permainan dengan mode dinamis dibandingkan dengan mode yang normal. Kesimpulan ketiga yaitu *Game* dengan tingkat kesulitan yang dinamis lebih mudah untuk dimainkan dengan persentase 84,67% tingkat kemudahannya. Sedangkan *game* dengan tingkat kesulitan yang normal lebih sulit untuk dimainkan dengan persentase 45,33% tingkat kemudahannya.

Berdasarkan hasil pengujian yang telah dilakukan juga dapat diperoleh beberapa saran untuk penelitian lebih lanjut. Saran pertama adalah menambahkan lebih banyak parameter *input* dari pemain agar dapat mengukur keterampilan pemain lebih baik lagi. Saran kedua adalah menambahkan lebih banyak parameter *output* selain konten seperti status dari pemain dan musuh agar *game* dapat lebih dinamis. Saran yang ketiga yaitu dapat menambahkan fitur multiplayer dan mengkombinasikan keterampilan semua pemain lalu menerapkan DDA agar dapat menciptakan *game* yang dinamis bagi semua pemain.

REFERENSI

[1] M. Zohaib, "Dynamic Difficulty Adjustment (DDA) in Computer *Game*s A Review" *in Hindawi*, Karnataka, 2018.

[2] M. G. Duque, R. B. Palm, D. Ha, dan S. Risi, "Finding *Game* Levels with the Right Difficulty in a Few Trials through Intelligent Trial-and-Error", *Conf. IEEE on Games*, 2020.

[3] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic *game* balancing," *in Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artifcial Intelligence (IJCAI)*, 12, 7 pages, Edinburgh, United Kingdom, 2005.

[4] H. Junaedi, M. Hariadi, dan I. K. E. Purnama, "Penerapan Sinematografi dalam Penempatan Posisi Kamera dengan Menggunakan Logika Fuzzy", *Jurnal Ilmu Komputer dan Informatika Khazanah Informatika*, vol. 4, no. 2, pp. 55-61, Desember 2018.

[5] J. Pranata, E. M. Yuniarno, S. M. Susiki, dan H. Thuan, "DDA Pada Musuh Berbasis Skor Menggunakan Logika Fuzzy" *in Seminar Nasional Informatika*, Yogyakarta, 2015, pp. 187-194.

[6] R. Rismanto, R.Ariyanto, A. Setiawan, dan M. E. Zari, "Sugeno Fuzzy for Non-Playable Character Behaviors in a 2D Platformer *Game*" *J. in Int. Journal of Engineering & Technology*, vol. 7, no. 4.36, pp. 222-227, 2018.

[7] R. Parekh, "Staying in the Flow Using Procedural Content Generation and Dynamic Difficulty Adjustment", thesis, Worcester Polytechnic Institute, United State, 2017.

[8] B. M. F. Viana dan S. R. Santos, "Procedural Content Generation: A Survey", *J. on Interactive Systems*, vol. 12, no. 1, pp. 83-101, 2021.

[9] H. Hermawan dan H. Setiyani, "Implementasi Algoritma A-Star Pada Permainan Komputer Roguelike Berbasis Unity", *J. Algoritma, Logika dan Komputasi*, vol. 2, no. 1, pp. 111-120, 2019.

[10] A. Gellel dan P. Sweetser, "A Hybrid Approach to Procedural Generation of Roguelike Video *Game* Levels", *Proc. Foundation of Digital Games*, no. 3, pp. 1-10, September 2020.

[11] G. Nwankwo, S. Mohammed, dan J. Fiaidhi, "Procedural Content Generation for Dynamic Level Design and Difficulty in a 2D *Game* Using UNITY" *in Int. Journal of Multimedia and Ubiquitos Engineering*, vol. 12, no. 9, pp. 41-52, 2017.

[12] S. Xue, M. Wu, J. Kolen, N. Aghdaie, dan K. A. Zaman, "Dynamic Difculty Adjustment for Maximized Engagement in Digital *Game*s," *in Proc. of the the 26th International Conference*, pp. 465–471, Perth, Australia, April 2017.

[13] G. Andrade, G. Ramalho, A. S. Gomes, dan V. Corruble, "Dynamic *Game* Balancing an Evaluation of User Satisfaction", *Proc. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 2, no. 1, pp. 3-8, 2006.

[14] C. V. N. Segundo, K. E. A. Calixto, dan R. P. D. Gusmao, "Dynamic Difficulty Adjustment through Parameter Manipulation for Space Shooter *Game*", *Proc. SBGames*, September 2016.

[15] J. Pranata, "DDA Based on Fuzzy Logic for Adventure *Game*'s Score", thesis, Institut Teknologi Sepuluh Nopember, 2016.