Penerapan *Procedural Content Generation* untuk generasi level dalam game *Mythical Maze*

Hansen Krisna Putra, Liliana
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail : m26414003@john.petra.ac.id, Lilian@petra.ac.id

ABSTRAK

Dalam proses pembuatan sebuah aplikasi game yang memiliki banyak level, salah satu kesulitannya adalah mendesain level. Untuk mendesain level, hal-hal yang perlu diperhatikan adalah variasi dari level yang dihasilkan, waktu yang diperlukan untuk mendesain level, dan penempatan dari objek didalam level, seperti titik awal pemain dan tujuan akhir dari level. Metode Procedural Generation dapat digunakan untuk mempermudah pembuatan level yang diinginkan.

Dalam implementasi Procedural Generation, hal pertama yang dilakukan adalah menentukan batasan dari level yang diinginkan, seperti ukuran *terrain*. Ukuran *terrain* akan mempengaruhi kompleksitas sebuah level karena menentukan banyaknya ruangan dan koridor yang dihasilkan. Setelah itu, dibuatlah beberapa ruangan dengan ukuran yang beragam dan ditempatkan kedalam level dengan menggunakan pola yang diinginkan, yang kemudian disambungkan dengan membuat beberapa koridor dimana koridor yang dihasilkan berfungsi sebagai jalan antara dua ruangan. Setelah proses pembuatan level selesai, maka langkah terakhir adalah penempatan objek, seperti pemain dan tujuan akhir, dimana diberi jarak diantara ruangan dimana objek berada.

Dari metode yang digunakan, mampu dihasilkan bentuk level yang beragam dan otomatis. Level dapat dimainkan dengan baik karena tujuan akhir dapat dicapai oleh pemain, dan semua ruangan terhubungkan oleh koridor. Penerapan metode ini selalu menghasilkan sebuah *terrain* dimana terdapat jalan diantara titik awal dan tujuan akhir. Setiap level memiliki tingkat kesulitan yang beragam, yang dipengaruhi oleh tingkat kekompleksan, jarak antara titik awal dan tujuan akhir, dan waktu yang diberikan untuk menyelesaikan level, dimana pemain tidak selalu dapat menyelesaikan level yang besar. Metode ini dapat digunakan untuk menghasilkan varian *Maze*, dan untuk menambah kesulitan, ukuran *terrain* dapat dibesarkan untuk menambahkan kompleksitas.

Kata Kunci: *Procedural Generation, Game, Level Generation, Unity, Maze*

ABSTRACT

Level design is one of the issues in creating a multi-level game application. In level design, some of the important keypoints are the variation of the level, the time to create it, and the locations of objects inside of a level, for example the player's starting point and the exit point. Procedural Generation can be used to simplify the process of designing a level.

In the implementation of Procedural Generation, the first step is to determine the size of the desired level or terrains. The size of a terrain will affect the complexity of a level, where it will affect the amount of rooms and corridors that is made. Next, rooms with various sizes are made and placed within the level with a desired pattern, which is connected by placing corridors that acts as a connector between two rooms. The last step to create a level is to place the objects inside the level, such as player character and the end point, where a distance is given between two objects

From the methods that are used, a level can be automatically made with multiple variations. The levels themselves are well made in which all of the rooms are connected by corridors and the player can reach the exit point from any given poinit in a level. This method always results in a terrain where there is a line or route between the starting and exit points. Each level's difficulty is unique, that is affected by complexity, the distance between objects, and the time given to finish a level. The player doesn't always able to finish a big level. This method can be used to produce variants of a maze, and to add the difficulty, the terrain itself can be enlargen to add complexity

Keywords: *Procedural Generation, Game, Level Generation, Unity, Maze.*

1. PENDAHULUAN

Procedural Content generation adalah sebuah metode dimana sebuah Algoritma dijalankan dengan tujuan untuk menghasilkan konten[1] dengan menggunakan aset yang sudah disiapkan dan dilakukan secara otomatis[2]. Didalam *Game Design*, pada umumnya *Procedural Content Generation* digunakan dengan tujuan untuk membuat sebuah level secara efisien, atau untuk membuat konten yang berbeda setiap kali algoritma dijalankan.

Maze atau labirin merupakan sebuah puzzle dimana dimana penyelesaiannya adalah dengan cara mencari jalan keluar dimana pemain mulai dari jalan masuk. Di dalam *Game Design*, sebuah *maze* digunakan sebagai rintangan dimana pemain menjelajahi sebuah lantai yang terdiri dari beberapa ruangan yang disambungkan dengan koridor untuk mencapai tujuan.

Penggunaan *Procedural Content Generation* didalam *Game Design*, terutama didalam proses generasi sebuah *level*, bertujuan untuk menghasilkan bentuk *level* yang berbeda setiap kali iterasi dijalankan. Proses ini dapat digunakan untuk menghasilkan sebuah *game* dimana pemain mendapatkan pengalaman yang berbeda setiap kali *game* dijalankan, seperti misalnya *game no man's sky*, dimana *level* atau *map* yang dibuat mengikuti batasan dan peraturan yang sudah diberikan[3].

Skripsi ini bertujuan untuk mengimplementasikan dan menganalisa salah satu metode *Procedural Content Generation* kedalam algoritma untuk mendesain sebuah game bertipe *maze.*

2. PROCEDURAL CONTENT GENERATION

Procedural Content Generation merupakan sebuah konsep dimana terdapat sebuah algoritma yang dijalankan untuk membuat atau menghasilkan sebuah media[4], contohnya seperti gambar,music atau level dengan menggunakan algoritma. Didalam *Game Design*, *Procedural Content Generation* digunakan sebagai sebuah metode untuk membuat sebuah konten yang dapat digunakan berulang-ulang didalam sebuah *game*, seperti *layout* sebuah *maze*, dimana tujuan dari *maze* adalah untuk mencapai titik akhir[5]. Beberapa varian dari metode ini adalah *Random Room Placement* dan *Random Point Connect*.

Metode *Random Room Placement* menghasilkan beberapa ruangan yang menjadi dasar sebuah *level* dimana setiap ruangan memiliki ukuran yang bervariasi. Jumlah ruangan yang dihasilkan dibatasi oleh ukuran dari *terrain* dan *threshold* yang sudah ditentukan. Ruangan dibuat dengan menentukan panjang dan lebar secara acak dan kemudian dicoba untuk diletakkan kedalam level. Ruangan yang sudah ditempatkan diberi jarak agar dua buah ruangan tidak saling bersebelahan atau menumpuk. Ruangan yang tidak bisa ditempatkan akan dihapus dan proses diulang sampai *threshold* yang ditentukan sudah terpenuhi.

Random Point Connect merupakan salah satu cara untuk menyambungkan semua ruangan didalam level dengan cara membuat jalan diantara dua ruangan. Dua buah ruangan disambungkan dengan cara memilih salah satu sisi dari masing-masing ruangan secara acak, dan menarik garis diantara kedua sisi yang sudah dipilih. Garis bisa ditarik secara vertikal dan kemudian horizontal, atau sebaliknya, dimana arah dari garis yang ditarik dipilih secara acak. Setelah kedua sisi ruangan dihubungkan dengan garis, maka sebuah koridor dibuat sesuai dengan posisi dari garis tersebut. Koridor yang dibuat dapat melewati ruangan atau koridor lainnya yang sudah diletakkan sebelumnya.

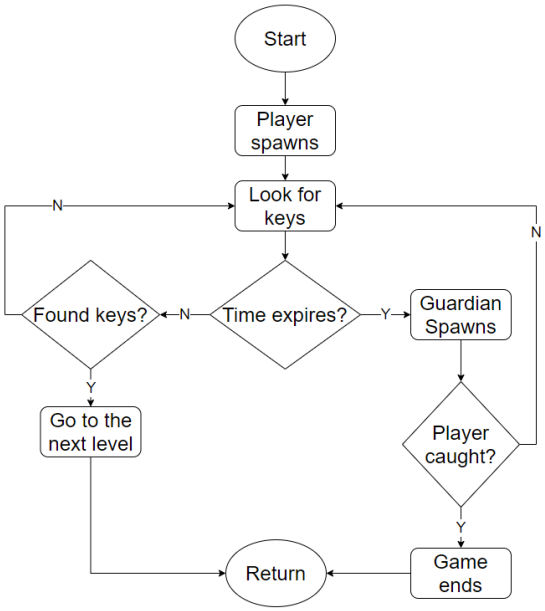
Relasi antara *Random Room Placement* dan *Random Point Connect* ditunjukkan didalam *pseudocode* dibawah ini :

```
    //Random Room Placement
1.  WHILE above Threshold
2.   Create Room
3.   FOR each row on terrain
4.    FOR each column on terrain
5.     If Position (row,column) is free THEN
6.      Place Room
7.      Add Room to List
8.     ENDIF
9.    ENDFOR
10.  ENDFOR
11. ENDWHILE
    //Random Point Connect
12. FOR each Room in List
13.  Get random side in Room A
14.  Get random side in Room B
15.  Make Line between side A and B
16. ENDFOR
```

3. DESAIN SISTEM

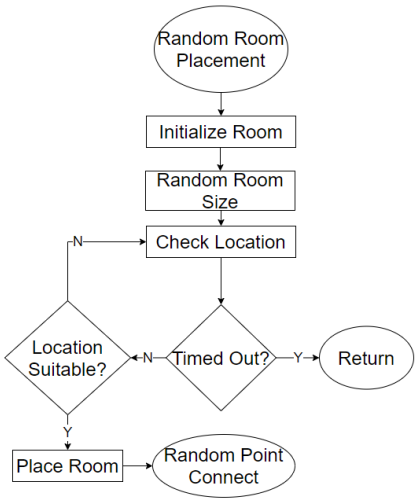
3.1 Desain Sistem

Aplikasi dimulai dengan tampilan awal menu yang terdiri dari beberapa pilihan. Pilihan pertama merupakan fungsi utama aplikasi dimana pengguna dapat memilih tingkat kesulitan dan tipe level dari permainan. Pilihan kedua digunakan untuk memunculkan halaman *how to play* dimana pengguna dapat melihat cara bermain aplikasi. Pilihan ketiga digunakan untuk menutup aplikasi.



Gambar 1. *Flowchart* alur game

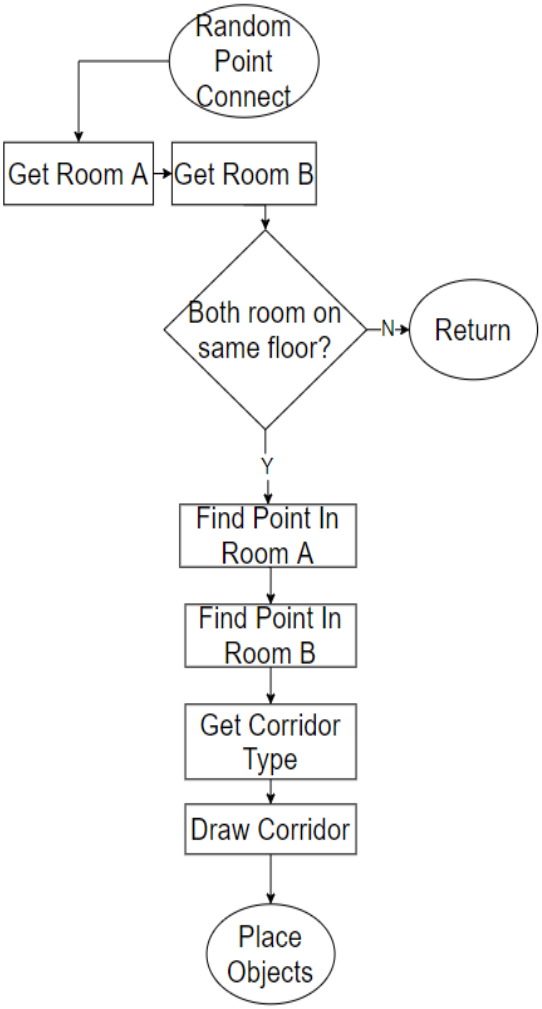
Setelah pengguna memilih tingkat kesulitan dan tipe level, maka pemain akan diarahkan ke level pertama. Pengguna dapat menggerakkan *player* dengan menggunakan *control* yang ditampilkan didalam halaman *how to play*. Setelahw pengguna mendapatkan semua kunci yang ada dialam *level*, maka pemain dapat mencari *exit* dan melanjutkan permainan ke *level* selanjutnya. Apabila waktu didalam sebuah level habis, maka *Guardian* akan muncul dan mencoba untuk menangkap pengguna. Apabila pengguna tertangkap oleh *Guardian*, maka permainan akan selesai dan pengguna diarahkan kembali ke menu utama.



Gambar 2. *Flowchart Random Room Placement*

Langkah pertama yang dilakukan saat proses pembuatan level adalah membuat *terrain* dengan ukuran yang disesuaikan dengan tingkat kesulitan yang dipilih. Selanjutnya, metode *Random Room Placement* dijalankan.

Proses pertama yang dilakukan adalah membuat sebuah ruangan dengan ukuran yang diacak. Setelah ruangan dibuat, maka algoritma akan mencari sebuah lokasi didalam level untuk menempatkan ruangan. Pengecekan dimulai dari bagian kiri atas level dan berakhir pada bagian kanan bawah. Apabila terdapat tempat kosong, maka ruangan akan ditempatkan dan diberi jarak lebih untuk menghindari dua ruangan yang terlalu berdekatan. Apabila tidak terdapat tempat kosong yang cocok untuk ruangan, maka ruangan akan dihapus dan aplikasi akan mengulang proses hingga mencapai *Threshold* yang ditentukan.



Gambar 3. *Flowchart Random Point Connect*

Setelah semua ruangan ditempatkan didalam level, maka proses *Random Point* Connect dijalankan untuk menyambungkan ruangan dengan cara membuat koridor. Proses dimulai dengan memilih dua buah ruangan kemudian dan kemudian mengambilksalah satu titik dari masing-masing ruangan yang akan disambungkan dengan jalur yang ditentukan secara acak, baik secara vertikal dan kemudian horizontal, atau sebaliknya. Jalur yang dibuat dapat melewati ruangan dan koridor lainnya yang sudah dibuat.

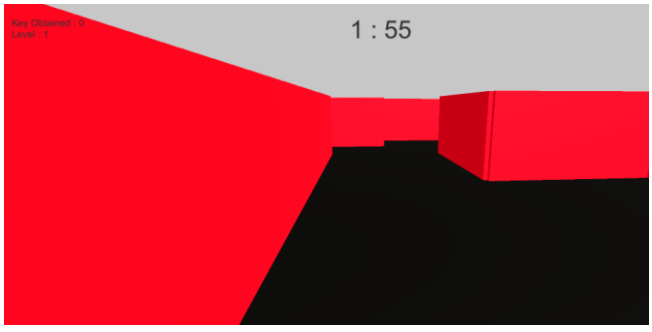
3.2 Desain Interface

Gambar 4. Desain Interface *Main Menu*

Gambar 4. menampilkan tentang *interface Main Menu*, yang merupakan tampilan menu utama ketika aplikasi dibuka. Terdapat tiga buah tombol pada tampilan, dimana tombol pertama akan membuka sebuah tampilan untuk mengatur opsi permainan.

Gambar 5. Desain *Interface* opsi *Play Game*

Gambar 5. menunjukkan tampilan apabila tombol *Play Game* dipilih, dimana akan muncul sebuah tampilan yang berisi beberapa tombol dan opsi untuk memilih tipe *Maze*. Tiga buah tombol yang ada berfungsi sebagai opsi untuk memilih tingkat kesulitan yang diinginkan, yang digunakan sebagai parameter untuk mementukan tingkat kesulitan dari *level* yang dibuat.



Gambar 6. Desain Interface didalam game

Gambar 6. menunjukkan tampilan *interface* ketika *user* sedang bermain, dimana terdapat sebuah *timer* untuk *level* yang dimainkan, dan terdapat informasi tentang level di bagian kiri atas layar yang berisi jumlah kunci yang didapatkan didalam level dan pada level berapa *user* berada.

4. PENGUJIAN SISTEM

4.1 Spesifikasi Sistem

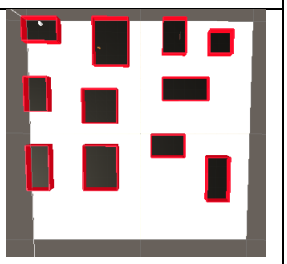
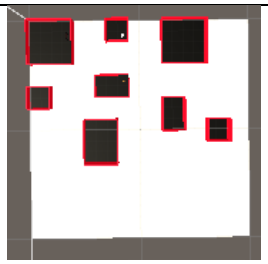
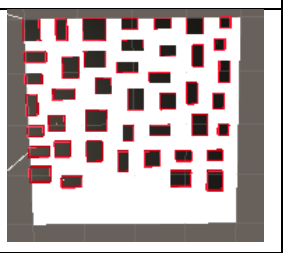
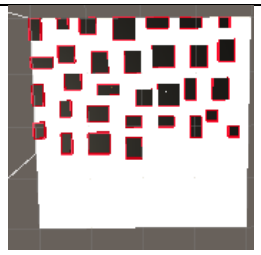
Pengujian dilakukan dengan menggunakan *personal computer* dengan spesifikasi sebagai berikut :

Prosesor : Intel Core i5-4690K CPU @ 3.50GHz
RAM : 8192MB RAM
Kartu Grafis : 2047MB NVIDIA GeForce GTX 1060 6GB
OS : Windows 10 Pro 64-Bit
Resolusi : 1920 x 1060
DirectX : DirectX12
Program dibuat dengan menggunakan Unity versi 2020.1.0f1

4.2 Pengujian Random Room Placement

Tingkat kesulitan mempengaruhi besar dari sebuah level dan jumlah ruangan yang bisa ditempatkan, dimana semakin tinggi tingkat kesulitan yang digunakan, maka level akan semakin besar. Tabel 1. menunjukkan rata – rata hasil dari percobaan pada tingkat kesulitan *Easy* dan *Medium* dan contoh bentuk dari penempatan ruangan.

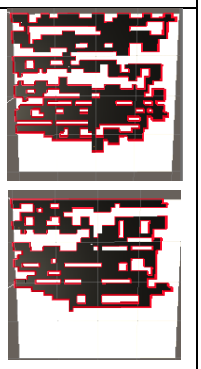
Tabel 1. Pengujian pada tingkat kesulitan yang berbeda.

Tingkat Kesulitan	Jumlah Ruangan	Total Free Space %	Pola Level
Easy (20%)	10-12	14-19.25	
Easy (40%)	8	33.25 – 39.25	
Medium (20%)	40-43	18.25 % - 19.68	
Medium (40%)	29-32	38.43 – 39.56	

4.3 Pengujian Random Point Connect

Jumlah ruangan yang ada mempengaruhi bentuk dan jumlah dari koridor yang ada. Tabel 2. menjelaskan tentang rata – rata dari hasil percobaan pada tingkat kesulitan *Easy* dan *medium* dimana terdapat dua buah Batasan yang berfungsi sebagai pembatas untuk penempatan ruangan.

Tabel 2. Pengujian pada tingkat kesulitan yang berbeda

Tingkat Kesulitan	Jumlah Ruangan (20% / 40%)	Free Space % (20% / 40%)	Corridor % (20% / 40%)	Pola Level
Easy	10-12 / 8-9	13-18 / 35-39.75	23.5 / 11.5-15.75	
Medium	40-43/31-32	16.875-19.5625 / 37.25-38.6875	22.3125-29.5 / 19.8125-22.3125	

4.4 Pengujian panjang dari sebuah level

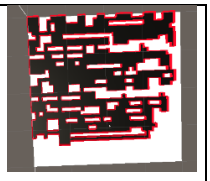
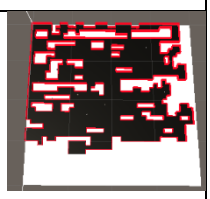
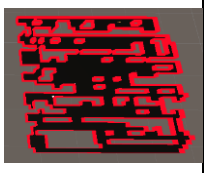
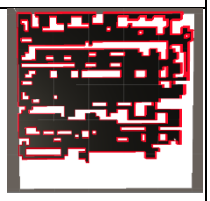
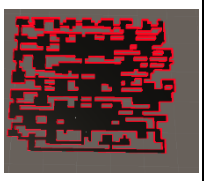
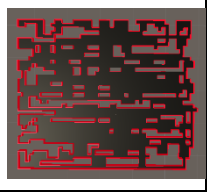
Pengujian dilakukan untuk mengetahui rata – rata panjang dari sebuah level, yaitu dari awal permainan sampai dengan penguji menyelesaikan level dalam hitungan detik dan pengujian dilakukan pada Batasan 20%. Tabel 3. menjelaskan tentang hasil dari pengujian yang dilakukan.

Tabel 3. Pengujian panjang dari sebuah level

Tingkat Kesulitan	Jumlah Ruangan	Durasi Level	Level Clear %	Layout Level
Easy /2D	9-12	20.02-42.52	100	
Easy / 3D	20-22	54.4763-87.05319	100	

Tabel 3. Pengujian panjang dari sebuah level

Medium / 2D	39-43	63.77–124.1976	100	
Medium / 3D	81-87	239.004-256.0245	33	 
Hard / 2D	60-66	34.42767-59.63	100	
Hard / 3D	193-201	288.3122/335.4158	33	  

5. KESIMPULAN

Dari hasil perancangan dan pembuatan program Penerapan *Procedural Content Generation* untuk generasi level dalam game *Mythical Maze*, dapat diambil kesimpulan :

- Rancangan Random Room Placement dan Random Point Connect dapat berjalan dengan mengikuti parameter – parameter yang diberikan dan sesuai dengan rancangan awal.
- Program dapat menghasilkan sebuah level yang terdiri dari beberapa ruangan yang kemudian disambungkan oleh koridor dan berisi beberapa macam *Key Objects*, seperti *Player*, *Key*, *Exit Door*, dan *Teleporter*.
- Bentuk level dengan tipe 2D cenderung lebih gampang atau lebih bisa diselesaikan jika dibandingkan dengan level dengan tipe 3D, baik dalam tingkat kesulitan Easy, Medium, maupun Hard.
- Hasil dari Random Room Placement mengikuti sebuah pola dimana ruangan yang ditempatkan di bagian kiri atau di atas cenderung mendekati ujung dari wilayah yang sudah ditentukan dan ruangan lainnya ditempatkan di tengah secara acak.
- Dari hasil pengujian didapatkan sebuah level dengan parameter yang sama memiliki jumlah ruangan, bentuk ruangan, dan waktu penyelesaian level yang berbeda. Hal ini memungkinkan pemain dapat merasakan pengalaman yang berbeda setiap kali bermain yang menghasilkan tingkat *replayability* yang tinggi.

6. REFERENCES

[1] Alba Amato. 2017. Procedural Content Generation in the Game Industry. In Oliver Korn, Newton Lee Game Dynamics. Best Practices in Procedural and Dynamic Game Content Generation, Springer. 15-25.

[2] Jessica R. Baron. 2017. Procedural Dungeon Generation Analysis and Adaptation. In Proceedings of the SouthEast Conference (ACM SE '17). Association for Computing Machinery, New York, NY, USA, 168–171. DOI:https://doi.org/10.1145/3077286.3077566

[3] Nathan Hilliard, John Salis, and Hala ELA arag. 2017. Algorithms for procedural dungeon generation. J. Comput. Sci. Coll. 33, 1 (October 2017), 166–174.

[4] Nicholas A. Barriga. 2018. A Short Introduction to Procedural Content Generation Algorithms for Videogames. International Journal on Artificial Intelligence Tools.

[5] P. Kim et al. 2019. Design-centric maze generation. In Proceedings of the 14th International Conference on the Foundations of Digital Games (FDG '19). Association for Computing Machinery, New York, NY, USA, Article 83, 1–9. DOI:https://doi.org/10.1145/3337722.3341854