Modifikasi Peta Permainan Bertema Dungeon Crawler Menggunakan Algoritma Genetika Dengan GMS2

[1]Vinza Hedi Satria, [2]Fetty Tri Anggraeny, [3]Pratama Wirya Atmaja

[1]Teknik Informatika, Fakultas Teknologi Elektro & Industri Cerdas,
Institut Teknologi Sepuluh Nopember
[2,3]Teknik Informatika, Fakultas Ilmu Komputer,
Universitas Pembangunan Nasional "Veteran" Jawa Timur
[1]vinzasatria.if@gmail.com, [2]fettyanggraeny.if@upnjatim.ac.id, [3]pratama_wirya.fik@upnjatim.ac.id

Abstrak

Gim atau permainan digital (*Video Game*) memiliki berbagai macam aspek yang mempengaruhi didalamnya, aspek-aspek ini seperti gameplay, visual, story dan banyak aspek lainnya. Dalam tema permainan tertentu, terkadang sebuah aspek lebih menonjol daripada aspek lainnya. *Dungeon Crawler* adalah tema gim yang menonjolkan aspek *gameplay* nya berupa petualangan yang berbeda-beda selama permainan, sebuah kesulitan baru pada sisi pengembang gim (*Developer*) saat hendak mengembangkan sebuah gim *Dungeon Crawler*. Dikarenakan gim lebih mengutamakan aspek petualangan, *developer* diharuskan menciptakan banyak area permainan agar pemain tidak bosan, sedangkan bila diharuskan merubah satu-per-satu akan memakan waktu yang lama, untuk itu diperluhkan cara untuk membuat sebuah area berubah-ubah bentuknya. Pada akhir penelitian ini di ajukan sebuah cara melakukan modifikasi peta gim *Dungeon Crawler* menggunakan algoritma optimisasi, Genetika.Selain aplikasi, rumus yang digunakan juga akan dibuktikan dengan melakukan modifikasi terhadap rumus

Kata kunci: Algoritma Genetika, Permainan Digital, *Dungeon Crawler*

Modification of Dungeon Crawler Themed Game Map Using Genetic Algorithm With GMS2

Abstract

Video games have a variety of aspects that affect them, aspects such as gameplay, visuals, stories and many other aspects. In certain game themes, sometimes one aspect stands out more than others. Dungeon Crawler is a game theme that highlights the gameplay aspects of different adventures during the game, a new difficulty is rising up on the game developer side when trying to develop a Dungeon Crawler game. Because the game prioritizes the adventure aspect, developers are required to create many areas of the game so that players don't get bored, whereas if you are required to change one by one it will take a long time, for that you need to make a map to change its shape. At the end of this study a method for modifying the Dungeon Crawler game map was used using the optimization algorithm, Genetics. Beside application, the formula used will also be proven by modifying the formula value.

Keywords: Genetic Algorithm, Game, Dungeon Crawler

Pendahuluan

Dungeon Crawler atau dalam bahasa Indonesia adalah penjelajahan dungeon merupakan permainan yang berkembang dari tema *adventure* atau petualangan, termasuk tema terlama pada sejarah panjang permainan digital (Cobett, 2017), yang dimulai sejak 1976 oleh William Crowther dan Don Woods yang berbasiskan teks dan berisi cerita petualangan fantasi. Permainan dengan tema ini memang memakan waktu yang sangat lama, berdasarkan peringkat yang disusun oleh Miller (2020), pada 5 peringkat teratas, terdapat 4 judul permainan digital dengan tema *adventure*. Hal ini dikarenakan pada permainan digital dengan tema ini membutuhkan banyak aspek untuk dikembangkan mulai dari gameplay hingga unsur-unsur yang membutuhkan estetika tinggi seperti *story*, *audio* dan *visual*.

Besarnya sumber daya yang diperluhkan dalam pembuatan sebuah gim dengan tema ini membuat pengembang harus memutar otak dalam membuat gimnya, sumber daya harus diatur agar semua aspek dapat dikerjakan dengan baik. Pada dunia *programming*, dikenal PCG atau *Procedural Content Generation* dalam bahasa Indonesia disebut Pembangkitan Peta Prosedural, penggunaan PCG merupakan sebuah jalan keluar yang ideal bagi pengembang gim yang ingin memiliki banyak variasi peta permainan namun membutuhkan sumber daya yang kecil seperti pada *No Man's Sky* (Hello Games, 2016). Konten yang dapat di buat melalui PCG pun bermacam-macam, seperti cerita, gambar dan peta (Togelius et al., 2016b). Meski implementasinya yang beragam, PCG sendiri paling sering di implementasikan pada aspek *gameplay*, seperti yang dilakukan oleh Karavolos, et.al. (2016) atau penempatan karakter dalam permainan *Tower Defense* (Huo et al., 2009). Algoritma yang digunakan pun bermacam-macam, seperti pada penelitian *Tower Defense* dimana peneliti membandingkan antara penggunakan algoritma Genetika dengan PSO. Implementasi langsung pun juga telah dilakukan dengan menggunakan game *Hotline Miami* (Brown et al., 2017, 2018).

Penggunaan PCG pada gim untuk membangun peta dari nol merupakan hal yang umum dilakukan, namun pembangkitan secara *random* dapat membuat peta yang tidak menyenangkan untuk dimainkan, untuk itu dibutuhkan bentuk modifikasi peta dari masukkan pengembang permainan digital agar peta permainan tetap terjaga kualitasnya karna didesain oleh pengembang namun tetap terasa berbeda dari peta sebelumnya karna telah dimodifikasi. Melakukan modifikasi pun juga harus menggunakan algoritma tertentu, modifkasi yang tidak terukur dapat membuat kualitas dari peta memburuk seperti

tidak terdapat jalan yang terhubung antara jalan masuk sampai ke jalan keluar yang mengakibatkan permainan tidak dapat dimainkan karena terdapat jalan yang tidak terhubung. Untuk menjaga kualitas peta saat melakukan modifikasi dapat dilakukan menggunakan algoritma optimasi, terdapat banyak jenis algoritma Optimasi yang ada seperti *Ant Colony, Particle Swarm Optimization* namun Togelius, et.al. (Togelius et al., 2016b) menyatakan bahwa algoritma evolusi Genetika merupakan algoritma yang paling sering digunakan. Dengan memodifikasi peta menggunakan algoritma optimasi, pemain dapat merasakan peta yang agak berbeda dalam setiap perjalanan, namun tidak akan merubah desain awal dari pengembang gim.

Game Maker merupakan *engine game* yang dibuat oleh yoyogames yang berdiri pada tahun 2007. *Game Maker* dinilai oleh pengguna nya sebagai sebuah *engine* yang mudah digunakan dan sangat cocok digunakan untuk pemula maupun profesional, selain itu keuntungan lain *Game Maker* adalah kemampuannya untuk membuat permainan *multi-platform* dengan hanya menggunakan satu bahasa pemrograman yaitu GML (Game Maker Language) (Cossu, 2019).

Pembahasan

Pertama-tama akan dibahas struktur data dari aplikasi dan alur dari algoritma Genetika yang dirancang beserta rumus-rumus yang digunakan.

A. Jenis Data

Peta yang akan dimodifikasi diambil melalui auto-generate dari algoritma graph grammar dimana pengembang memasukkan peraturan atau rule kedalam graph gramar (Togelius et al., 2016a) lalu aplikasi secara otomatis membuat peta yang nanti akan di modifikasi menggunakan algoritma Genetika, adapun peta yang dibuat membutuhkan tiga jenis struktur data, ketiga struktur data itu adalah:

- Struktur Data List Jenis Node

Struktur data berupa list untuk melambangkan jenis-jenis node apa saja yang berada dalam pembangkitan peta prosedural, data yang disimpan didalam list adalah huruf yang melambangkan jenis ruangan yang muncul didalam peta seperti S untuk Start (mulai) dan F untuk Finish (selesai). Urutan dari list perlu diperhatikan karena akan

berhubungan dengan struktur data selanjutnya.

Tabel 1. Sample Struktur Data List Jenis Node

S	T	T	M	T	A	A	A	T	T	T	M	F

- *Struktur Data Grid Posisi Node*

Struktur data berbentuk grid (matriks 2 dimensi) untuk melambangkan posisi dari node yang dibuat, data yang disimpan didalam posisi node merupakan urutan dari struktur data List jenis node, bukan huruf seperti isi dari list jenis node.

tabel 2. Sample Struktur Data Grid Posisi Node

1	3				
	5	2			
		4	7		
			9		
			6		
			11	10	
				8	
				12	13

- *Struktur Data Grid Posisi Edge*

Struktur data berbentuk grid (matriks 2 dimensi) untuk melambangkan posisi dari edge berfungsi untuk memberitahu hubungan antara dua buah node atau lebih, bentuk penyimpanan data posisi edge berbeda dengan Posisi node, dimana bila posisi node disimpan membentuk sebuah peta, posisi edge menggunakan sumbu X dan Y nya untuk melambangkan urutan node yang sudah dijabarkan pada list jenis node.

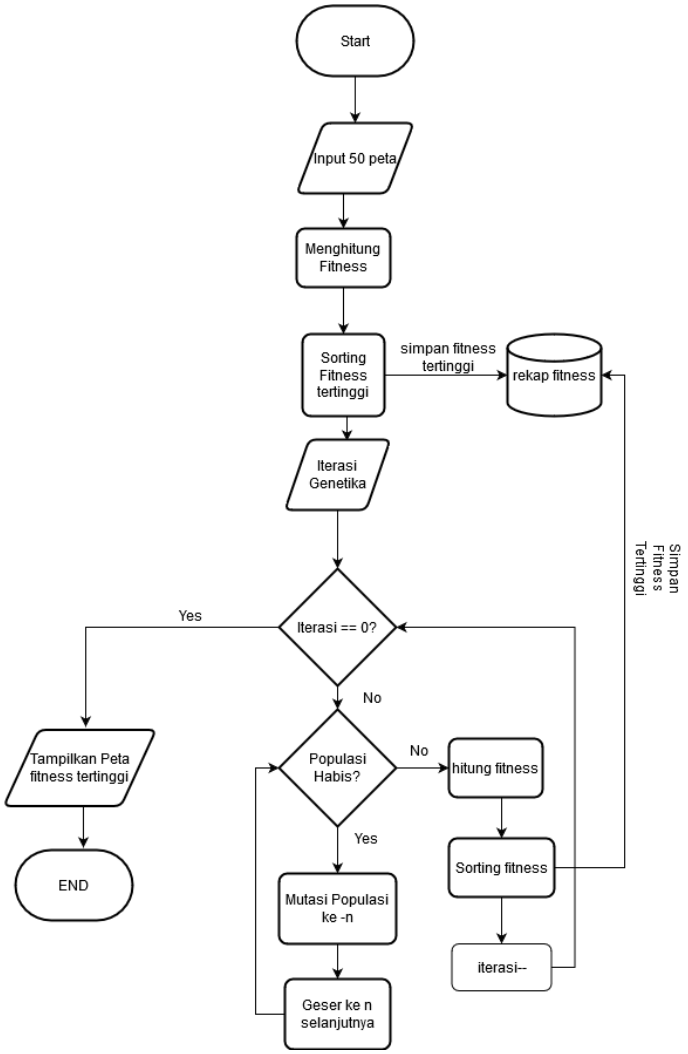
Tabel 3. Sample Struktur Data Grid Posisi Node

Node	1	2	3	4	5	6	7	8	9	10	11	12	13
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													

Seperti pada tabel 3, contoh apabila node ke 1 dan node ke 2 memiliki hubungan (edge) maka pada koordinat X=1 dan Y=2 atau X=2 dan Y=1 bernilai 1.

B. Alur Algoritma Genetika

Alur dari algoritma genetika digambarkan ke dalam sebuah *flowchart* untuk menjelaskan secara rinci mengenai cara kerja algoritma, adapun bentuk *flowchart* tersebut terdapat pada gambar 4 dibawah. Genetika akan meminta input peta yang sudah dibangun oleh pengembang dengan jumlah 50 yang akan menjadi populasi (50 peta yang merupakan individu dapat dibuat sama persis). Kemudian sebelum melakukan mutasi, fitness dari populasi dihitung dengan tujuan untuk membandingkan bagaimana perbedaan antara fitness setelah Genetika dan fitness sebelum Genetika.



Gambar 1. Alur Algoritma Genetika

Penghitungan fitness dilakukan dengan dua cara, pertama menghitung jarak terpendek (Shortest Path) menggunakan rumus. Rumus diambil dari penelitian Karavolos, et. al. (2016) di mana ds, e Merupakan jarak terpendek dari Start ke Finish yang didapatkan melalui algoritma DFS (Depth First Search), sedangkan e adalah bilangan euler yaitu 2,719 (pembulatan).

$$f_p = \min\{(1 + e^{3-d_{s,e}})^{-1}, 1 - (1 + e^{13-d_{s,e}})\} \quad (1)$$

Selain rumus tersebut, digunakan juga rumus ke – 2 untuk menghitung variasi dari peta menggunakan rumus. Dimana E_d merupakan hubungan antara dua node yang berbeda sedangkan E adalah jumlah seluruh hubungan node.

$$f_v = \frac{E_d}{E} \quad (2)$$

Kedua *Fitness* ini kemudian di rata-rata, rata-rata *fitness* pada penelitian kali ini menggunakan rata-rata berbobot dengan bobot *fitness 1* sebesar 60% dan *fitness 2* sebesar 40%, hal ini dilakukan dengan pertimbangan variasi pada *node* yang jadi nanti tidak terlalu penting . Setelah semua *fitness* selesai dihitung, selanjutnya adalah melakukan sorting *fitness* tertinggi, nilai *fitness* tertinggi akan diletakkan di kiri sedangkan terendah akan diletakkan di kanan. Nilai tertinggi akan disimpan untuk ditampilkan sebagai hasil. Setelah itu proses mutasi dilakukan, pada dasarnya Genetika memiliki proses mutasi dan rekombinasi, namun rekombinasi dihilangkan karena dapat merusak peta yang sudah dibuat oleh pengembang. Proses mutasi pun juga dilakukan secara hati-hati agar tidak mengacaukan hasil, adapun proses mutasi yang dimodifikasi dari Karavolos, et.al. (2016) adalah sebagai berikut.

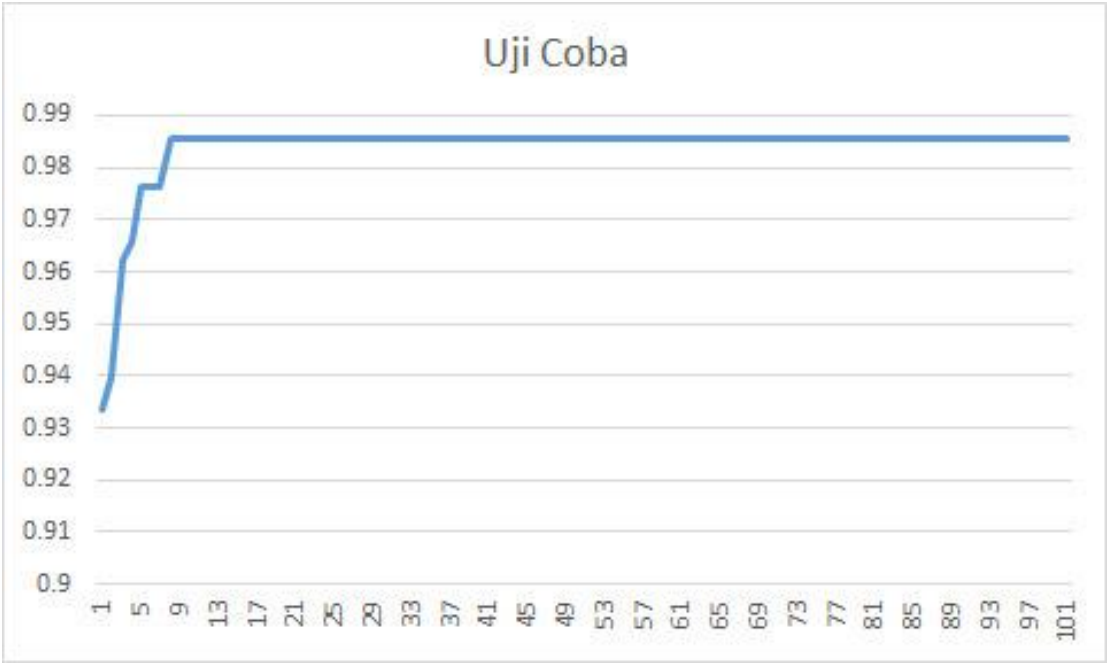
Tabel 4. Daftar Aksi Mutasi (sumber: Karavolos et al., 2016)

Nama	Keterangan
Insert Node	Menyisipkan sebuah node random diantara dua node yang terhubung, untuk melakukan ini ukuran edge harus berjumlah 3
Add Node	Menambahkan sebuah node random dan sebuah edge melalui node lain.
Change Node	Sebuah Node dipilih secara random namun dengan ketentuan bukan node S ataupun F dan berubah menjadi node lain
Delete Node	Sebuah node dipilih untuk dihapus dengan kondisi : Bila node memiliki sebuah edge maka edge juga akan dihapus, bila node berada di antara 2 edge,maka 2 edge disambung sehingga membentuk edge yang panjang
Add Ege	Sebuah edge ditambah diantara dua node yang belum memiliki edge, penambahan ini mengakibatkan sebuah rute baru terbentuk
Delete Edge	Sebuah edge dipilih untuk bersiap dihapus, sebelumnya dilakukan pencarian path dari node yang edgenya akan dihapus dengan node start untuk memastikan bahwa penghapusan edge tidak akan berakibat fatal

Mutasi dipilih secara acak menggunakan algoritma random biasa, setelah ke 50 populasi di mutasi, fitness dihitung kembali dan di lakukan sorting dari fitness. fitness tertinggi akan disimpan, hal ini dilakukan terus menerus hingga iterasi yang sudah ditentukan oleh pengembang. Penyimpanan nilai Fitness hanya berfungsi sebagai variabel penelitian, dalam proses pengembangan gim, hal ini tidak diperlukan dan dapat dihapus.

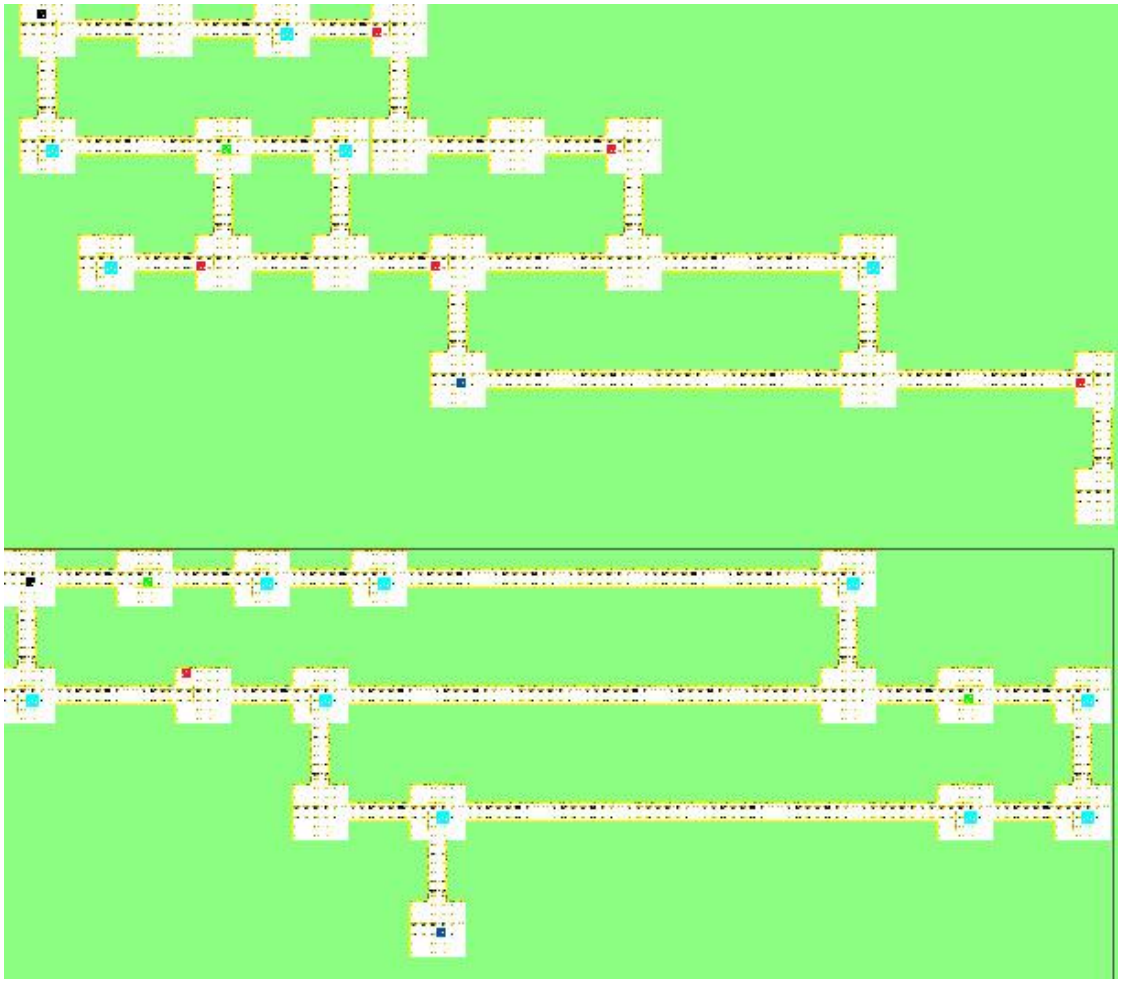
C. Uji Coba Aplikasi

Ditunjukkan hasil output dari aplikasi yang berfungsi sebagai variabel penelitian berupa fitness sebanyak 100 kali mutasi genetika yang dimasukkan ke dalam aplikasi pengolahan data Microsoft Excell sehingga berbentuk seperti gambar dibawah.



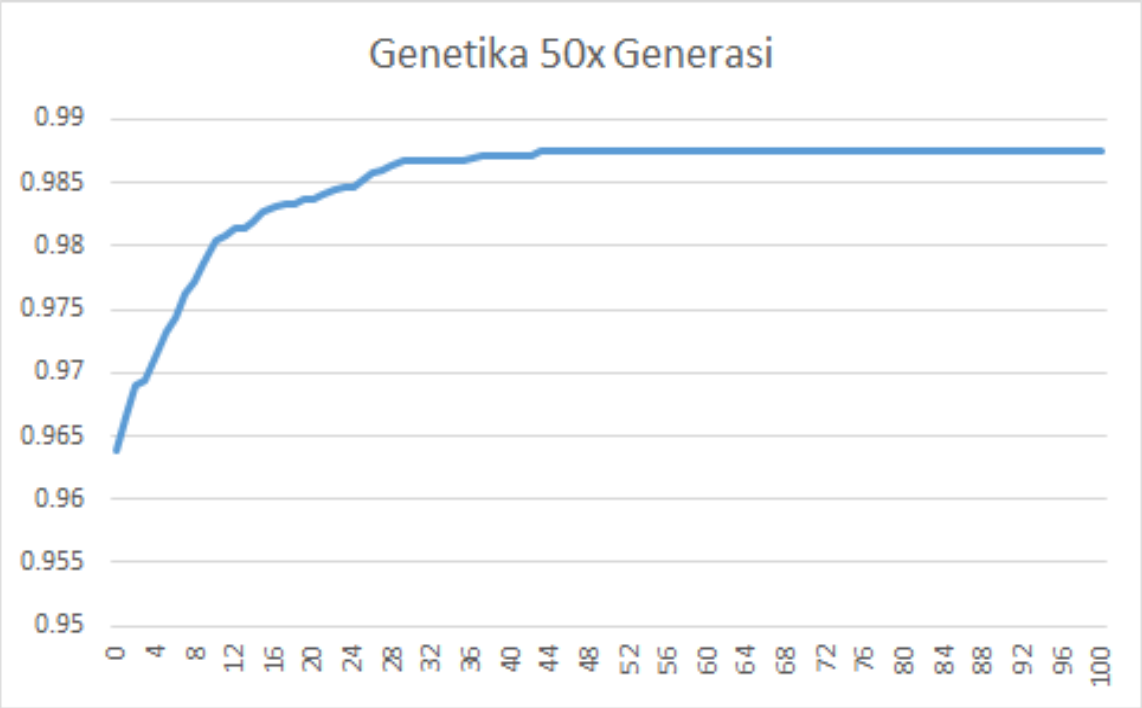
Gambar 2. Uji Coba Fitness 1x generasi

Melalui fitness yang telah dipaparkan sebelumnya dapat dilihat bahwa fitness mengalami kenaikan apabila dibandingkan pada uji coba 1 (sebelum dilakukan genetika) hingga uji coba ke 100. Selain fitness di tampilkan juga contoh hasil peta hasil modifikasi menggunakan algoritma Genetika, pada penelitian kali ini modifikasi genetika di implementasikan ke dalam sebuah gim ber *tema Dungeon Crawler* yang dipadukan dengan Quiz menggunakan Game Maker Studio 2 seperti pada gambar dibawah.



Gambar 3. Hasil Uji Coba Peta Menggunakan Permainan Prototip Quiz

Gambar 3 merupakan hasil dua kali percobaan modifikasi peta menggunakan algoritma Genetika, dapat dilihat bahwa pada generasi 1 dan 2 mengalami perbedaan yang signifikan. Selanjutnya *variabel* dari *fitness* diuji untuk menunjukkan bahwa rumus *fitness* merupakan rumus ideal. Pengujian ini hanya dilakukan kepada persamaan (1) karena hanya variabel dari persamaan tersebut yang dapat dirubah, sedangkan persamaan dua akan tetap sama, bobot pada rata-rata pun juga akan tetap dibiarkan sama. Berbeda dengan gambar 3 yang dilakukan hanya dalam 1 kali generasi, pengujian kali ini akan dilakukan sebanyak 50 kali generasi kemudian di rata-rata untuk membuat hasil penghitungan yang lebih akurat. Pengujian dilakukan sebanyak 3 kali, pertama adalah mengurangi rentang dari rumus *fitness* dan kedua adalah menambah rentang dari rumus *fitness* dan terakhir adalah menambah variabel *fitness* namun rentang tidak berubah.

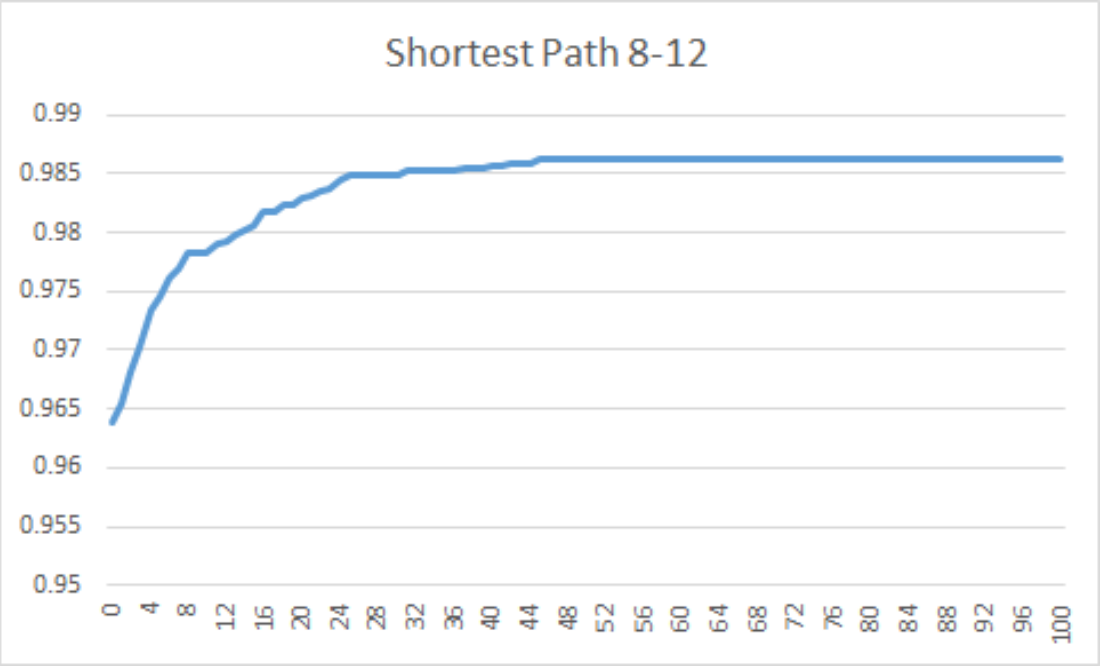


Gambar 4. Hasil fitness 50 kali generasi

Pertama ditampilkan hasil *fitness* dengan generasi sebanyak 50x sebagai perbandingan. Pada gambar 8 dapat dilihat bahwa *fitness* tertinggi yang didapatkan sebelum dilakukan mutasi adalah 9,63 dan *fitness* tertinggi sebelum terjadi titik jenuh atau konvergen adalah +0,985 atau lebih tepatnya berada pada 0.,987 yang berada pada iterasi ke 43. Pada rumus ini, rentang antara nilai persamaan kiri (berjumlah 3) dan kanan (berjumlah 13) adalah 10, selanjutnya dilakukan modifikasi berupa pengurangan rentang dari *fitness* sehingga persamaan (1) berubah menjadi:

$$f_p = \min\{(1 + e^{8-d_{s,e}})^{-1}, 1 - (1 + e^{13-d_{s,e}})\} \quad (3)$$

Nilai dirubah dengan meningkatkan nilai kiri menjadi 8 dan mempertahankan nilai kanan yaitu 13, sehingga terdapat jarak 5 poin dari sebelumnya 10. Setelah rumus dirubah, uji coba dilakukan lagi dengan total jumlah 50 kali generasi sehingga menghasilkan grafik pada gambar 8 sebagai berikut:

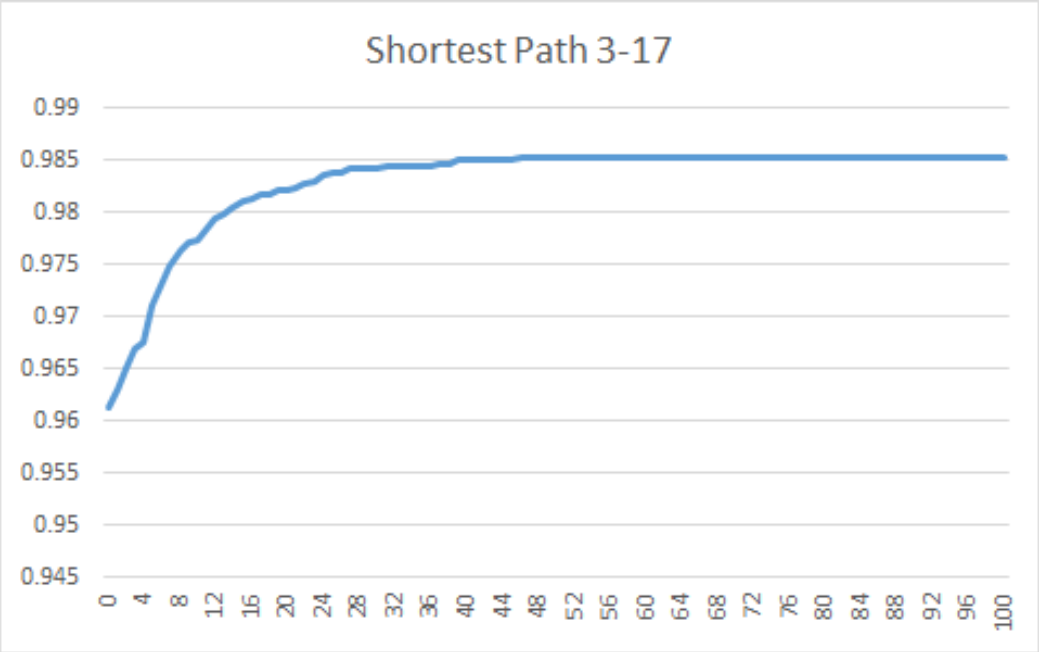


Gambar 5. Hasil fitness 50 kali generasi rumus 8-12

Pada uji coba selanjutnya, hasil fitness sebelum dilakukan mutasi bersifat sama dari sebelumnya 0,963 menjadi tetap 0,963. Fitness tertinggi sebelum konvergen terjadi pun juga mengalami penurunan sehingga menjadi 0,986 dan mengalami penurunan 0,001 poin dari rumus asli persamaan (1). Rumus di modifikasi kembali, kali ini rentang dinaikkan sehingga rumus fitness berubah menjadi :

$$f_p = \min\{(1 + e^{3-d_{s,e}})^{-1}, 1 - (1 + e^{17-d_{s,e}})\} \quad (4)$$

Nilai dirubah dengan meningkatkan nilai kanan menjadi 17 dari awalnya 13, sedangkan nilai kiri mengunakan nilai orisinil dari rumus tanpa modifikasi yaitu 3, sehingga terbentuk rentang baru yaitu sebesar 14, berbeda 4 poin dari rentang asli yang berjumlah 10. Menggunakan rumus yang telah dimodifikasi, uji coba sebanyak 50 kali iterasi diulangi kembali, hasil *fitness* di olah kembali menggunakan *Microsoft Excell* sehingga mendapatkan hasil berupa grafik pada gambar 9

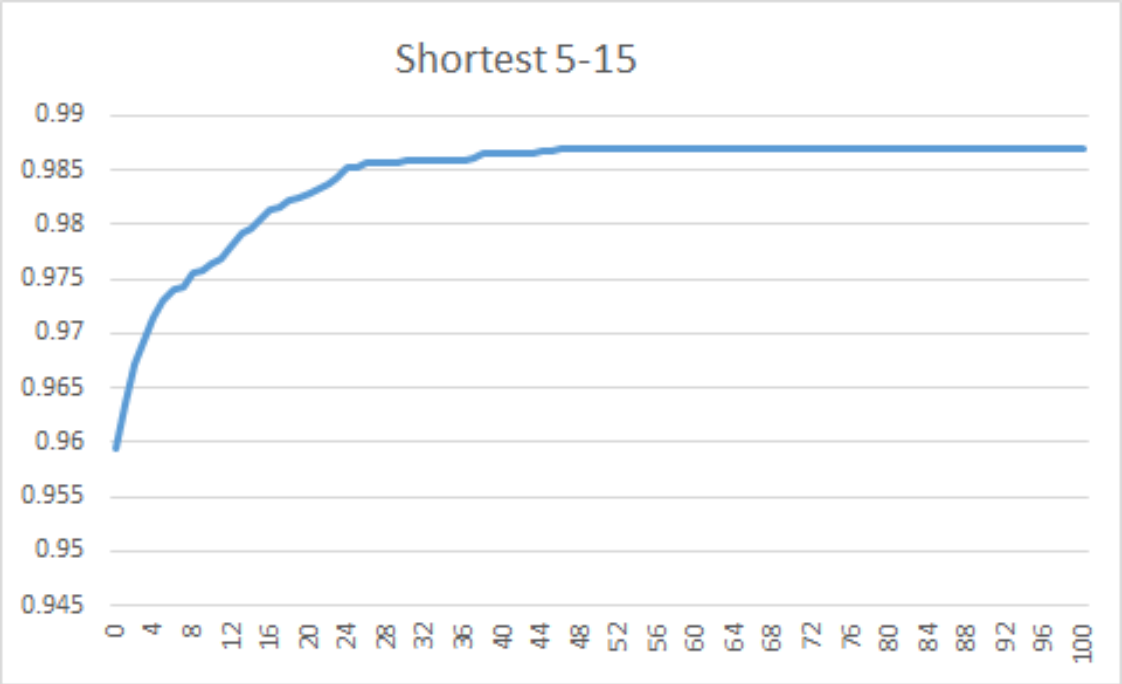


Gambar 6. Hasil fitness 50 kali generasi rumus 3-17

Grafik pada gambar 10 menunjukkan bahwa fitness sebelum mutasi dilakuan berada pada angka 0,961, jauh lebih rendah 0,002 poin daripada angka rumus orisinil sebelum di modifikasi yaitu 0,963. Turunnya nilai fitness awal ini diikuti juga dengan turunnya nilai fitness tertinggi yaitu 0,985, turun 0,002 dari nilai fitness sebelum dilakukan modifikasi. Selanjutnya adalah melakukan modifikasi rumus untuk merubah nilai pada rumus namun rentang tetap tidak berubah, sehingga persamaan berubah menjadi :

$$f_p = \min\{(1 + e^{5-d_{s,e}})^{-1}, 1 - (1 + e^{15-d_{s,e}})\} \quad (5)$$

Nilai dirubah menjadi 5 dan 15, namun rentang antara kedua nilai masih sama, yaitu 10. Dengan menggunakan rumus yang sudah di modifikasi tersebut, dilakukan lagi percobaan sebanyak 50 kali generasi, hasil dari percobaan tersebut ditampilkan pada gambar 11 dibawah ini.



Gambar 7. Hasil fitness 50 kali generasi rumus 5-15

Nilai fitness sebelum dilakukan mutasi memiliki nilai 0,958 mengalami penurunan sebesar 0,005 poin apabila dibandingkan dengan nilai asli yaitu 0,963. Sedangkan nilai fitness tertinggi mengalami penurunan yaitu pada 0,986, sedangkan pada rumus asli berada pada titik 0,987.

Setelah semua pengujian sudah dilakukan, data diolah kembali untuk disimpulkan ke dalam sebuah tabel untuk mempermudah penjelesan. Tabel menampilkan jenis modifikasi, nilai fitness awal, nilai fitness tertinggi yang didapatkan sebelum berada pada titik jenuh atau konvergen dan titik konvergen yang merupakan indikasi dimulai pada iterasi ke-berapa konvergen terjadi.

Tabel 5. Hasil Akhir Uji Coba

No	Modifikasi Rumus	Fitness Sebelum Mutasi	Fitness Tertinggi	Titik Konvergen
1	Tanpa Modifikasi	0,963871	0,98748584	43
2	Shortest Path 8-12	0,96383422	0,98616094	45
3	Shortest Path 3-17	0,96120252	0,98533414	39
4	Shortest Path 5-15	0,9594783	0,9868813	44

Melalui uji coba yang telah dilakukan, dapat ditarik hasil akhir pada tabel 5. Melalui tabel 2 dapat ditarik kesimpulan bahwa berdasarkan modifikasi rumus yang telah dilakukan terhadap *fitness shortest path* menunjukkan bahwa rumus *fitness* tanpa

modifikasi memiliki nilai *fitnes* paling tinggi dan titik konvergen yang berada di posisi tengah. Maka dapat disimpulkan bahwa rumus yang digunakan pada penelitian kali ini merupakan rumus yang ideal

Kesimpulan

Penelitian pada kesempatan kali ini menghasilkan sebuah aplikasi modifikasi peta menggunakan algoritma Genetika dengan menggunakan Game Engine Game Maker Studio 2. Selain berjalannya aplikasi, rumus *fitness* juga diuji dengan mengubah variabel yang ada didalam rumus. Melalui aplikasi ini diharapkan pengembang dapat memodifikasi peta yang sudah ia buat agar tidak diperluhkan lagi usaha tambahan dalam pembuatan peta.

Saran penelitian kedepan adalah untuk melakukan pemutakhiran aplikasi, Genetika tidak hanya digunakan dalam modifikasi tetapi digunakan untuk membuat peta dari nol apabila pengembang memang memiliki niat untuk membuat sebuah peta yang benar-benar membutuhkan nol usaha. Selain itu pemutakhiran genetika juga dapat dilakukan pada penelitian kedepan agar selisih *fitness* dapat terlihat jauh lebih jelas daripada penelitian yang telah dilakukan saat ini, dimana selisih berada pada kisaran 0,001 – 0,02.

Ucapan Terima Kasih

Kami ucapkan terima kasih kepada pihak-pihak yang membantu dalam terciptanya laporan penelitian ini. Kami juga ucapkan terima kasih kepada tim penerbit *Journal of Animation and Game Studies* yang telah meluangkan waktunya dalam melakukan *review* sampai proses-proses selanjutnya.

Referensi

Brown, J. A., Lutfullin, B., & Oreshin, P. (2017). Procedural content generation of level layouts for Hotline Miami. *2017 9th Computer Science and Electronic Engineering (CEEC)*, 106–111.

Brown, J. A., Lutfullin, B., Oreshin, P., & Pyatkin, I. (2018). Levels for Hotline Miami 2: Wrong Number Using Procedural Content Generation. *MDPI*, 1–16.

Cobett, R. (2017). *The history of RPGs*. PC Gamer. https://www.pcgamer.com/the-complete-history-of-rpgs/

Cossu, S. M. (2019). *Game Development With Game Maker Studio 2*. Apress.

Hello Games. (2016). *No Man's Sky*.

Huo, P., Niu, B., Wang, H., & Shiu, S. C. (2009). Application and Comparism of Particle Swarm Optimization and Genetic Algorithm in Strategy Defense Game. *2009 Fifth International Conference On Natural Computation*.

Karavolos, D., Liapis, A., & Yannakakis, G. (2016). Evolving Missions to Create Game Spaces. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*.

Miller, B. (2020). *10 Modern Games With The Longest Development Time, Ranked by How Long They Took To Make*. The Gamer. https://www.thegamer.com/games-with-long-development-time/

Togelius, J., Shaker, N., & Dormans, J. (2016a). Grammars and L-systems with applications to. In *Procedural Content Generation in Book* (hal. 1–26). Springer.

Togelius, J., Shaker, N., & Dormans, J. (2016b). *Procedural Content Generation In Game*. Springer.