Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 5, Mei 2019, hlm. 4406-4414 http://j-ptiik.ub.ac.id

Penerapan *Procedural Content Generation* untuk Perancangan Level pada 2D *Endless Runner Game* menggunakan *Genetic Algorithm*

Muadz Askarul Muslim[1], Eriq Muhammad Adams Jonemaro[2], Muhammad Aminul Akbar[3]

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

Abstrak

Dengan berkembang pesatnya industri *game*, jumlah konten yang dibutuhkan dalam permainan terus meningkat. Peningkatan jumlah konten diperlukan untuk menjaga agar para pemain tetap tertarik, maka dari itu pekerjaan desain semakin dibutuhkan untuk memenuhi persyaratan ini. *Procedural Content Generation* merupakan solusi dalam menghemat waktu dan uang dan telah berhasil diterapkan dalam beberapa *Endless Runner Game*. Di sini penulis menggunakan metode *Genetic Algorithm* untuk mengimplementasikan *Procedural Content Generation* pada 2D *Endless Runner Game*. *Genetic Algorithm* penulis pilih dikarenakan algoritma dapat melakukan optimisasi yang cocok untuk banyak kasus dari sebuah lingkungan. Selain optimisasi, *Genetic Algorithm* berbentuk modular, sehingga terpisah dari aplikasi dan dapat diaplikasikan ke kasus lainnya tanpa perubahan yang berarti didalamnya. Pembuatan *level* dapat dilakukan dengan dilakukan dengan menggunakan teknik random. Namun hasil *level* yang didapatkan dengan random dapat memiliki masalah seperti tidak tepatnya hasil yang diinginkan karena tidak ada kriteria sebagai pengukur kepantasan dari hasil yang dibuat dengan random seperti dapat dilewatinya *level* yang telah dibuat. Sedangkan dalam *Genetic Algorithm* ada sebuah bagian yang dapat menyeleksi setiap individu dan populasi agar sesuai dengan kriteria yang ditetapkan. Hasil dari pengujian menunjukkan waktu yang dibutuhkan *program* dalam membuat sebuah *level* sangatlah singkat, yaitu 0.02 detik. Dari hasil tersebut menujukkan bahwa algoritma dapat diterapkan dan bekerja dengan baik dalam waktu pembuatan *level*. *Level* yang dihasilkan juga dapat dilewati oleh pemain dengan baik berdasarkan hasil pengujian oleh sampel pemain. Namun kesulitan dari *level* yang dihasilkan tidak dapat dikendalikan menggunakan *Genetic Algorithm* yang digunakan.

Kata kunci: *2D Platform, Endless Runner Game, Procedural Content Generation, Genetic Algorithm*

Abstract

With the rapid development of the gaming industry, the amount of content needed in the game continues to increase. Increasing the amount of content is needed to keep players interested, so design work is increasingly needed to meet these requirements. Procedural Content Generation is a solution to save time and money and has been successfully implemented in several Endless Runner Games. Here the author uses the Genetic Algorithm method to implement the Procedural Content Generation on 2D Endless Runner Game. The author's Geographical Algorithm chooses because the Algorithm can optimize which is suitable for many cases of an environment. In addition to optimization, the Genetic Algorithm is modular, so it is separate from the application and can be applied to other cases without significant changes in it. Making levels can be done by using a random technique. But the results of the randomly obtained level can have problems such as the inappropriate results desired because there are no criteria as a measure of appropriateness from the results that are made randomly as can be passed the level that has been made. Whereas in Genetic Algorithm there is a section that can select each individual and population to fit the specified criteria. The results of the tests show the time needed for the program to make a level very short, which is 0.02 seconds. From these results show that the algorithm can be applied and works well in the creation of levels. The resulting level can also be skipped by players based on the results of testing by a sample of players. But the difficulty of the level produced cannot be controlled using the Genetic Algorithm used.

Keywords: *2D Platform, Endless Runner Game, Procedural Content Generation, Genetic Algorithm*

1. PENDAHULUAN

Dengan berkembang pesatnya industri game, jumlah konten yang dibutuhkan dalam permainan terus meningkat. Peningkatan jumlah konten diperlukan untuk menjaga agar para pemain tetap tertarik, maka dari itu pekerjaan desain semakin dibutuhkan untuk memenuhi persyaratan ini. Membuat konten dalam game, seperti musuh, level atau item, membutuhkan waktu dan bisa mahal. Seorang manusia biasanya bekerja sangat lambat dibandingkan dengan komputer. Jika konten seperti itu dapat dikembangkan secara algoritmik, perusahaan yang mengembangkan permainan dapat menghemat banyak waktu dan uang untuk tugas-tugas ini(Andersson & Classon, 2016).

Procedural Content Generation merupakan solusi dalam menghemat waktu dan uang dan telah berhasil diterapkan dalam beberapa endless runner game(Compton & Mateas, 2006). Content generation telah diterapkan kedalam game sejak 2009, saat *CANABALT* dirilis. *CANABALT* merupakan game side-scrolling endless runner yang diciptakan sebagai experiment projek gameplay pada 2009. Game ini dapat membuat platform untuk player lewati dalam jumlah yang tak terhingga tanpa pengawasan oleh manusia.

Baru-baru ini, game seperti *Temple Run* dan *Cookie Run* menerapkan pembuatan platform secara otomatis yang mirip dengan yang dilakukan *CANABALT*. Dengan pesatnya perkembangan game dengan genre endless runner, dibutuhkan sebuah template yang dapat mempermudah dalam pembuatan game dengan genre tersebut. Di sini penulis menggunakan metode *Genetic Algorithm* untuk mengimplementasikan *Procedural Content Generation* pada 2D endless runner game. *Genetic Algorithm* adalah pencarian heuristik yang mencerminkan proses seleksi alam di mana individu yang paling cocok dipilih untuk reproduksi untuk menghasilkan keturunan generasi berikutnya(Vijini Mallawaarachchi, 2017). *Genetic Algorithm* penulis pilih dikarenakan algoritma dapat melakukan optimisasi yang cocok untuk banyak kasus dari sebuah lingkungan. Selain optimisasi, *Genetic Algorithm* berbentuk modular, sehingga terpisah dari aplikasi dan dapat diaplikasikan ke kasus lainnya tanpa perubahan yang berarti didalamnya.

Pembuatan level dapat dilakukan dengan dilakukan dengan menggunakan teknik random. Namun hasil level yang didapatkan dengan random dapat memiliki masalah seperti tidak tepatnya hasil yang diinginkan karena tidak ada kriteria sebagai pengukur kepantasan dari hasil yang dibuat dengan random seperti dapat dilewatinya level yang telah dibuat. Sedangkan dalam *Genetic Algorithm* ada sebuah bagian yang dapat menyeleksi setiap individu dan populasi agar sesuai dengan kriteria yang ditetapkan.

2. LANDASAN KEPUSTAKAAN

2.1 *Platformer*

'Platformers' atau 'game platform' adalah permainan yang terutama berputar di sekitar karakter yang dikontrol oleh pemain, yang berlari dan melompat untuk menghindari rintangan dan / atau untuk mengalahkan musuh(Minkkinen, 2016). Platformer sering diklasifikasikan sebagai sub-genre game aksi, dan dianggap sebagai salah satu genre game pertama. Meskipun menjadi salah satu genre game pertama, game platform telah mempertahankan popularitasnya selama bertahun-tahun.

Platformer game itu sendiri tidak dapat sepenuhnya dianggap sebagai genre standalone karena sering dikombinasikan dengan genre permainan lain seperti endless runner, RPG, dan sebagainya. Genre ini berasal dari arcade di awal tahun 1980 yang memperkenalkan game seperti *Space Panic* dan *Donkey Kong*(Gustafsson, 2014).

2.2 *Endless Runner Game*

Endless Runner Game adalah genre permainan platform di mana karakter pemain terus bergerak maju dunia prosedural yang tak berujung(Jos, 2014). Genre game ini menuntut pemain untuk mendapatkan skor setinggi tingginya. Game ini biasanya juga memiliki tingkat kesulitan yang semakin meningkat seiring bertambahnya skor dari pemain. Di Game 2D Platformer juga biasanya player akan dihadapi dengan berbagai rintangan yang akan menghalangi player untuk mecapai titik akhir dari game, maka dari itu player harus bisa melewati berbagai rintangan agar dapat mencapai skor tertinggi yang mungkin didapat.

2.3 *Procedural Content Generation*

Procedural Content Generation adalah penggunaan algoritma komputer dan kekuatan pemrosesan CPU untuk membuat objek secara

cepat(Gu, 2006). Ada beberapa alasan bagi pengembang game untuk tertarik dengan PCG. Itu pertama adalah konsumsi memori - secara prosedural diwakili konten biasanya bisa dikompres dengan menjaga konten agar tetap "tidak terekspansi" hingga kenten dibutuhkan (Togelius, Yannakakis, Stanley, & Browne, 2010).

a. *Online Generation*

Online Generation adalah dimana konten dibuat ketika game sedang dimainkan(Andersson & Classon, 2016). Contoh dari online generation adalah penyediaan *health pack* dan *spawn enemy* yang disesuaikan secara acak berdasakan permainan dan keputusan pemain dari game *Left 4 Dead*. Online Generation memiliki kelebihan untuk menghemat memori secara signifikan.

Gambar 2.1 Healthpack berisi sedikit item dalam Left 4 Dead

Gambar 2.2 Healthpack berisi banyak item dalam Left 4 Dead

b. *Offline Generation*

Offline Generation adalah ketika konten dibuat lalu dipilih dan diperbaiki oleh designer manusia sebelum dirilis(Andersson & Classon, 2016). Ini bertujuan untuk meringankan pekerjaan designer manusia dengan memberikan konten yang telah terpilih dan butuh sedikit penyesuaian. Contoh dari penerapan ini adalah *The Witcher 3*, dimana tumbuhan dan bermacam vegetasi yang ada dibuat oleh komputer dan diperbaiki oleh designer manusia.

Gambar 2.3 Vegetasi dalam *The Witcher 3*

2.4 *Genetic Algorithm*

Genetic Algorithm merupakan salah satu *evolutionary algorithm*, yang terinspirasi dari gagasan seleksi alam dalam pembuatan produk(Andersson & Classon, 2016). Langkah-langkah dasar untuk *Genetic Algorithm* adalah sebagai berikut :

1. Buat satu set solusi awal, populasi.
2. Hitung kesesuaian untuk setiap individu dalam populasi.
3. Pilih subset dari populasi untuk reproduksi. Individu dengan *fitness* yang lebih tinggi nilai memiliki peluang lebih tinggi untuk dipilih dan seorang individu dapat dipilih lebih banyak dari sekali. Mereka yang tidak dipilih mati.
4. Pilih pasangan individu untuk mereproduksi dengan probabilitas p_c.
5. Lakukan crossover pada individu terpilih. Individu yang tidak melalui crossover dilewatkan ke generasi berikutnya tanpa modifikasi.
6. Bermutasi setiap bit pada keturunannya dengan probabilitas p_m.
7. Jika jumlah iterasi atau kriteria lain tidak terpenuhi: ulangi dari langkah 2.

a. *Initial Population*

Proses dimulai dengan satu set individu yang disebut Populasi. Setiap individu adalah solusi untuk masalah yang ingin dipecahkan. Individu dicirikan oleh satu set parameter (variabel) yang dikenal sebagai Gen. Gen bergabung menjadi string untuk membentuk Chromosome (solusi) (Vijini Mallawaarachchi, 2017).

b. *Fitness*

Genetic Algorithm mengambil populasi di mana setiap individu terdiri dari beberapa bagian yang terperinci, misalnya sebuah ukuran dari nol dan satu(Andersson & Classon, 2016). Semua individu dalam populasi awal ini merupakan kemungkinan solusi, tetapi karena dihasilkan secara acak, mudah dibayangkan bahwa beberapa bagian dapat melakukannya dibuat secara acak lebih baik daripada yang lain. Semua individu dalam populasi adalah dianasi dengan *fitness function*.

c. *Method of Selection*

Ketika nilai-nilai *fitness* telah dihitung untuk setiap individu dalam suatu populasi, sebuah metode digunakan untuk menentukan individu mana yang harus digunakan untuk menciptakan keturunan(Andersson & Classon, 2016). Cukup memilih individu dengan nilai *fitness* tertinggi dalam setiap iterasi dapat menyebabkan algoritme berakhir denagn optimal di lokal.

d. *Crossover*

Suatu crossover, atau reproduksi, terjadi antara dua item dalam populasi dengan probabilitas tertentu p_c, menciptakan keturunan yang akan menggantikan *parent*. Dalam kasus contoh bit-sequence, crossover antara dua individu akan terjadi di beberapa titik dari sebuah urutan, berpotensi dipilih secara acak(Andersson & Classon, 2016).

e. *Mutation*

Langkah selanjutnya adalah langkah mutasi, di mana setiap bagian keturunannya dimutasi dengan probabilitas tertentu, p_m. Dalam contoh bit-sequence, mutasinya bisa dengan hanya membalikkan satu bit dalam urutan keturunannya. Sekuens baru yang mungkin bermutasi ini akan membentuk populasi baru, dan digunakan dengan cara yang sama seperti generasi sebelumnya(Andersson & Classon, 2016).

f. *Variables*

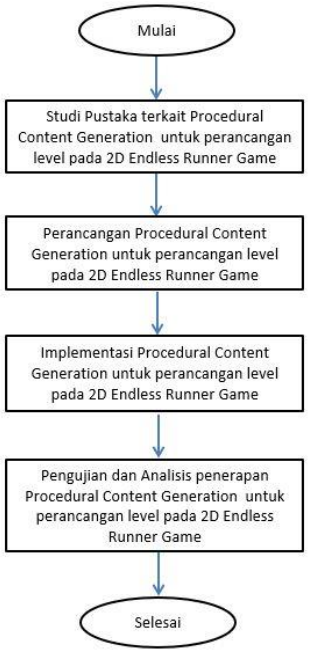
Agar *Genetic Algorithm* berfungsi dengan baik, diperlukan *fitness function*, yaitu rumus matematika yang dalam beberapa cara menjelaskan seberapa baik solusi sesuai dengan masalah yang coba dipecahkannya(Andersson & Classon, 2016). Dalam *fitness function* akan diberikan beberapa variable sebagai batasan dan syarat untuk setiap individu.

g. *Fitness Function*

Bagian penting dari *Genetic Algorithm* adalah *fitness function*(Andersson & Classon, 2016). *Fitness function* membantu memutuskan individu mana yang akan hidup pada generasi berikutnya dan individu mana yang akan dibuang. Oleh karena itu *fitness function* bagian penting dalam *Genetic Algorithm*.

3. METODOLOGI

Pada bab ini akan menjelaskan langkah langah yang akan dilakukan dalam penyususnan skripsi. Penelitian ini bersifat implementatif dengan megimplementasikan Procedural Content Generation pada 2D Endless Runner Game menggunakan Generic Algorithm. Metode penelitian yang digunakan adalah :



Gambar 3.1 Metodologi

Tahap pertama adalah studi pustaka pada tahap ini dilakukan penelusuran pengetahuan dalam rangka menyusun dasar teori yang digunakan sebagai dasar pengetahuan dalam penelitian ini.

Tahap kedua adalah perancangan *procedural content generation* untuk perancangan level pada 2D *endless runner game*.

Pada bagian ini akan dilakukan perancangan pembuatan *genetic algorithm* yang akan digunakan lalu akan melakukan perancangan untuk *level generator* dalam game.

Tahap ketiga adalah implementasi, pada tahap ini akan dilakukan penulisan kode untuk mengimplementasikan *genetic algorithm* dan juga *level generator* yang sudah dirancang sebelumnya.

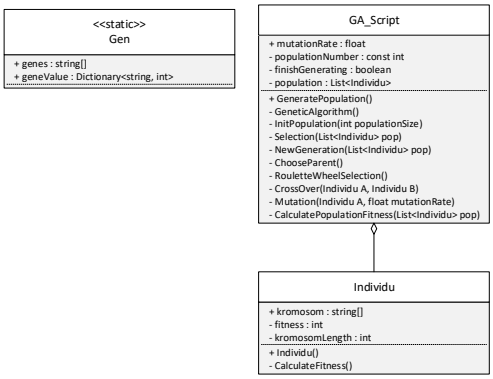
Tahap keempat adalah pengujian dan analisis. Akan dilakukan pengujian terhadap waktu penghasilan level, pengujian validitas dan pengujian performa dari level yang telah dihasilkan.

4. PERANCANGAN

4.1 Perancangan *Genetic Algorithm* dalam Game

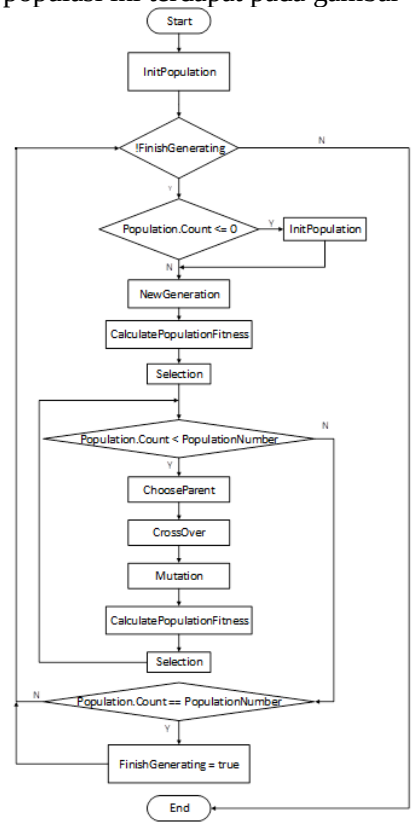
Genetic Algorithm digunakan untuk menghasilkan populasi dalam string yang kemudian akan diterjemahkan dan diubah menjadi gameobject didalam Unity3D. Populasi akan dihasilkan dari pembuatan Individu yang memiliki 10 kromosom berdasarkan Gen yang telah disediakan. Setiap Individu akan memiliki kromosom yang disusun secara acak.

Class Individu akan digunakan sebagai atribut berbentuk list individu dalam *class* GA_Script. Dan Individu akan menyusun kromosom dari *static class* Gen. *Class diagram* dari *Genetic Algorithm* terdapat pada gambar 4.1.



Gambar 4.1 *Class diagram* Genetic Algorithm

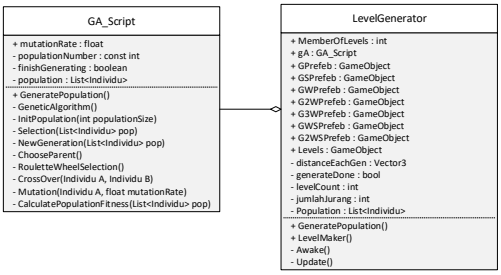
Dalam *class* GA_Script dilakukan inisialisasi populasi dan dilakukan perulangan dalam perhitungan nilai *fitness*, seleksi, perkawinan dan mutasi hingga didapatkan populasi dengan jumlah individu yang diinginkan. Flowchart dari proses pembuatan populasi ini terdapat pada gambar 4.2.



Gambar 4.2 *Flowchart class* GA_Script

4.2 Perancangan *Level Generator* dalam Game

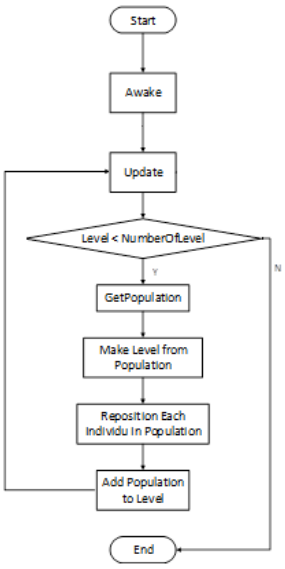
Level Generator digunakan untuk menerjemahkan populasi yang dihasilkan oleh GA_Script menjadi GameObject dalam Unity3D. Object yang dibuat akan dipindahkan kembali lokasi dan diatur hierarkinya. Class diagram dari Level Generator terdapat pada gambar 4.3.



Gambar 4.3 *Class diagram* Level Generator

Diawali dengan pembuatan dictionary yang digunakan untuk menerjemahkan populasi. Selanjutnya masuk dalam perulangan untuk mengecek jumlah level yang ada dalam game

world. Fungsi GenerateLevel akan memasukkan GameObject yang dibuat dalam fungsi LevelMaker kedalam game world, mengatur posisi dan hierarki dari objek yang dibuat. Flowchart dari proses penerjemahan populasi kedalam Game Object ini terdapat pada gambar 4.4.



Gambar 4.2 *Flowchart class* Level Generator

5. PENGUJIAN

5.1 Pengujian Waktu Dalam Menghasilkan Level

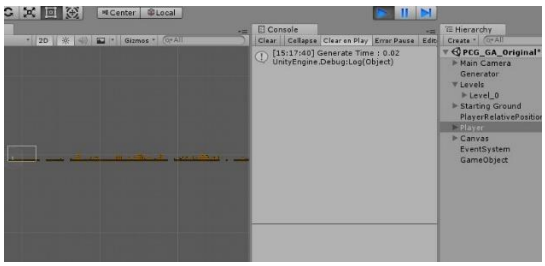
Pengujian akan dilakukan dimulai dengan pengadaan populasi dimana setiap populasi yang dihasilkan beranggotakan 10 individu. Pengujian akan dilakukan sebanyak 3 iterasi. Iterasi pertama akan dihasilkan 1 populasi, iterasi kedua akan dihasilkan 3 populasi dan iterasi ketiga akan dihasilkan 5 populasi. Waktu yang dibutuhkan dalam menghasilkan level pada pengujian akan disajikan dalam bentuk grafik dan tabel.

Gambar 6.2 Waktu Penghasilan Level

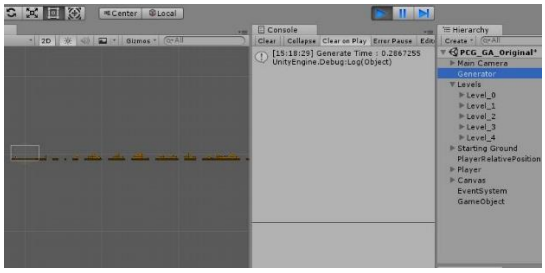
Iterasi	Jumlah Populasi	Waktu (s)
1	1	0.020
2	5	0.286
3	10	0.366

Tabel 5.1 Pengujian Waktu

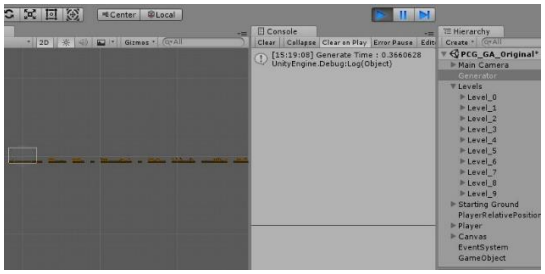
Terdapat perbedaan waktu yang tidak konsisten dari data yang didapatkan seperti yang ditampilkan dalam table 6.1. Perbedaan waktu ini disebabkan oleh jumlah populasi yang dihasilkan, banyaknya iterasi dalam membuat populasi dan penerjemaahan populasi kedalam bentuk game object dalam game engine Unity3D.



Gambar 5.2 Iterasi 1 Penghasilan Level



Gambar 5.3 Iterasi 2 Penghasilan Level



Gambar 5.4 Iterasi 3 Penghasilan Level

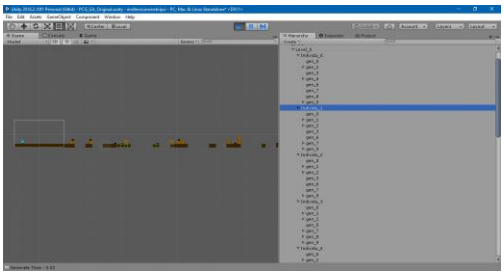
5.2 Pengujian Validitas

Pengujian ini dilakukan dengan tujuan mengetahui apakah populasi yang

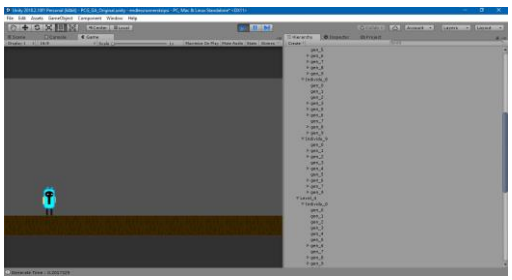
dihasilkan memenuhi struktur dan syarat yang telah ditentukan.

No	Jumlah Populasi	Hasil yang Diharapkan	Hasil	Status
1	1	Setiap populasi memiliki 10 individu dan setiap individu memiliki 10 Gen	Setiap populasi memiliki 10 individu dan setiap individu memiliki 10 Gen	Valid
2	5	Setiap populasi memiliki 10 individu dan setiap individu memiliki 10 Gen	Setiap populasi memiliki 10 individu dan setiap individu memiliki 10 Gen	Valid
3	10	Setiap populasi memiliki 10 individu dan setiap individu memiliki 10 Gen	Setiap populasi memiliki 10 individu dan setiap individu memiliki 10 Gen	Valid

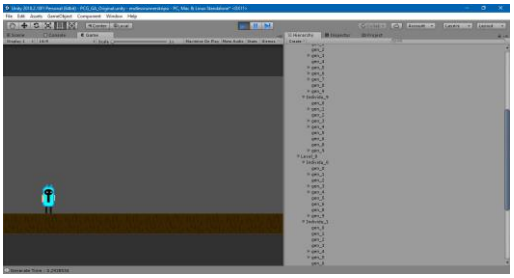
Tabel 5.2 Pengujian Validitas Struktur Populasi



Gambar 5.5 Validitas struktur dari penghasilan 1 populasi



Gambar 5.6 Validitas struktur dari penghasilan 5 populasi



Gambar 5.7 Validitas struktur dari penghasilan 10 populasi

5.3 Pengujian Performa

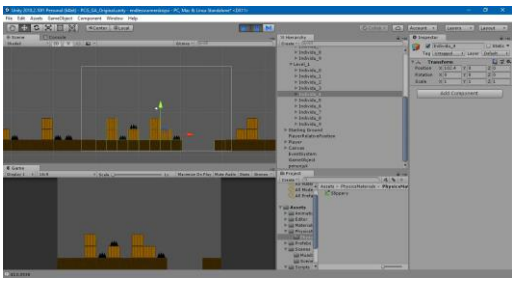
Pengujian dilakukan dengan pengambilan data dari 3 pemain yang memainkan level yang telah dihasilkan. Setiap pemain akan memainkan level sebanyak 10 kali dan data akan disajikan dalam bentuk table.

Pemain dalam pengambilan data memiliki perbedaan pengalaman dalam bermain game 2D *endless runner*. Riyan merupakan pemain yang sudah terbiasa dengan game 2D *endless runner* dan sudah bermain dibeberapa game dengan genre yang sama lainnya. Mayovio merupakan pemain yang sangat jarang memainkan game 2D *endless runner*. Yosua merupakan pemain yang terkadang bermain game 2D *endless runner*.

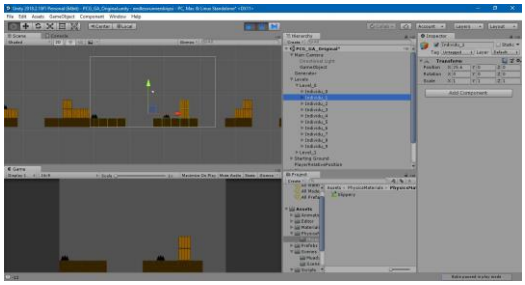
Nama Pemain	Individu terakhir yang dilewati pemain disetiap iterasi										Mean
	1	2	3	4	5	6	7	8	9	10	
Riyan	2	3	3	5	5	9	10	12	15	6	7
Mayovio	3	1	2	1	2	1	2	1	2	3	1.8
Yosua	2	2	4	1	2	5	6	5	3	3	3.3

Tabel 5.3 Pengujian Performa

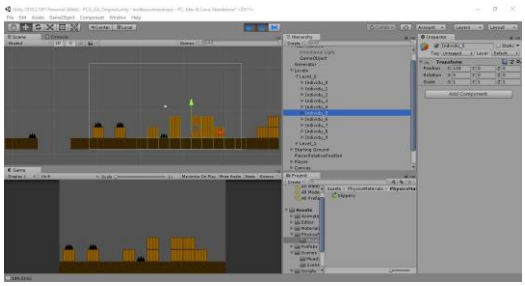
Pemain yang melakukan pengujian memiliki kesulitan dalam memainkan level yang telah dihasilkan. Hal ini dikarenakan factor sifat karakter yang digunakan untuk melewati level cukup sulit untuk dikendalikan. Sehingga hanya pemain Riyan dengan game ber-genre 2D endless runner saja yang dapat melewati level dengan progress setiap percobaannya yang baik.



Gambar 5.8 Pengujian Performa Oleh Riyan



Gambar 5.9 Pengujian Performa Oleh Mayovio



Gambar 5.10 Pengujian Performa Oleh Yosua

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Penelitian tentang *Procedural Content Generation* untuk perancangan level pada 2D *Endless Runner Game* menggunakan *Genetic Algorithm* berhasil dilakukan sesuai dengan metode yang sudah ditentukan sebelumnya. Hasil dari dapat disimpulkan sebagai berikut.

1. *Procedural Content Generation* menggunakan *Genetic Algorithm* dapat diterapkan. Dimana *level* yang dihasilkan menggunakan *Genetic Algorithm* dapat dibuat dengan benar dalam 2D *Endless Runner Game*.
2. Kinerja algoritma sebagai *Procedural Content Generation* pada 2D *Endless Runner Game* dapat disimpulkan baik. Berdasarkan pengujian waktu, algoritma dapat memenuhi kebutuhan dari pembuatan konten dengan waktu yang singkat. Tetapi perbedaan waktu setiap pembuatan *level* bergantung dengan banyaknya generasi yang dibutuhkan algoritma untuk mendapatkan hasil yang diharapkan.
3. Dari pengujian performa, *level* yang dihasilkan dinilai dapat dilewati oleh pemain dan dapat dinilai sebagai level dengan kualitas yang baik. Kesimpulan diambil berdasarkan data hasil *sample* pemain yang menguji *program.* Tetapi tingkat kesulitan medan yang sangat acak membuat sulit pemain yang tidak terbiasa dengan *game* dengan genre 2D *Endless Runner*. Terdapat juga faktor bagaimana sikap dari karakter seperti kecepatan lari karakter, kekuatan lompat karakter dan mekanisme jatuh karakter yang digunakan untuk melewati *level* yang dihasilkan.

6.2 Saran

Berdasarkan penelitian yang sudah dilakukan yaitu *Procedural Content Generation* untuk perancangan *level* pada 2D *Endless Runner Game* menggunakan *Genetic Algorithm*, maka penulis memberikan saran sebagai berikut :

Penerapan *Procedural Content Generation* untuk perancangan *level* pada 2D *Endless Runner Game* menggunakan *Genetic Algorithm* dapat dipakai, tapi perancangan penilaian *fitness* yang digunakan sebagai parameter nilai dari *level* cukup kompleks dan butuh untuk diperbaiki. Selain itu sifat karakter yang digunakan untuk melewati *level* yang dihasilkan juga merupakan factor cukup besar yang menentukan jika *level* yang dihasilkan menggunakan *Genectic Algorithm* dapat dilewati penain atau tidak. Pada penelitian selanjutnya disarankan untuk mencoba algortima lain yang dapat mengatur tingkat kesulitan dari level yang dihasilkan.

7. DAFTAR PUSTAKA

Andersson, V., & Classon, J. (2016). Procedural Generation of Levels with Controllable Difficulty for a Platform Game Using a Genetic Algorithm.

Compton, K., & Mateas, M. (2006). Procedural Level Design for Platform Games. *Artificial Intelligence for Interactive Digital Entertainment*, 109–111. https://doi.org/10.1111/j.1468-2958.1976.tb00706.x

Gu, B. (2006). Procedural Content Generation.

Gustafsson, A. (2014). Bachelor Thesis in Computer Science An Analysis of Platform Game Design.

Jos, R. (2014). Procedural Level Balancing in Runner Games, 797–801.

Minkkinen, T. (2016). Basics of Platform Games, 1–42.

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2010). Search-Based Procedural Content Generation, (Acornsoft 1984), 141–142.

Vijini Mallawaarachchi. (2017). Introduction to

Genetic Algorithms — Including Example Code. Retrieved August 28, 2018, from https://towardsdatascience.com/introdctio n-to-genetic-algorithms-including-example-code-e396e98d8bf3