Pemanfaatan Simplex Noise Untuk Menghasilkan Map Yang Natural dengan Unity Engine

Calvin Vionaldy Tjiandra, Rudy Adipranata,Lily Puspa Dewi
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121-131. Surabaya 60236
Telp (031)-2983455, Fax. (031)8417658
E-mail: Vionaldycalvin777@gmail.com,rudya@petra.ac.id,lily@petra.ac.id

ABSTRAK

Dalam pengembangan sebuah game, *terrain* adalah hal pertama yang akan dilihat dalam permainan. *Landscape*, *asset* pada *terrain*, dan bentuk kurvator pada *terrain* akan memberikan suasana yang memberikan pengalaman bermain yang menyenangkan, menegangkan, ataupun menakutkan. Akan tetapi proses pembuatan *terrain* membutuhkan waktu yang lama karena *setting terrain* yang dilakukan oleh *developer*, apalagi jika *developer* butuh membuat beberapa *terrain* dengan parameter yang mirip. Tidak hanya itu, *terrain* yang dihasilkan oleh *developer* sendiri akan menjadi repetitif, karena banyaknya kemiripan antara satu *terrain* dengan yang lain. Agar menghindari hal ini, lebih baik bila menggunakan *Procedural Content Generator* atau PCG untuk membuat sebuah *base terrain* untuk proyek. Simplex noise adalah sebuah metode PCG untuk *generate height map* pada *terrain* untuk dikembangkan oleh *developer*. *Terrain* yang telah berhasil di *generate* dapat dijelajahi oleh *player* , dan untuk menghindari rasa *slump* karena repetitif dari *terrain* setiap kali *terrain* di *generate*, *surface* dari *terrain* akan berubah.

Penelitian kali ini akan berfokus pada pengimplementasian simplex noise dalam men *generate* sebuah base *terrain* untuk *game* 3D dengan menggunakan C# dari unity. Simplex noise akan men *generate* 3 macam *biome* yaitu *forest, savanna, mountain.* Perlin noise akan diimplementasikan sebagai metode pembanding, perlin noise akan men *generate biome* yang sama seperti simplex noise dengan variabel-variabel yang sama juga. Program ini ditujukan untuk melihat *surface* yang di *generate* oleh kedua metode.

Penelitian ini menguji beberapa model *terrain* dengan melihat seberapa hasil dari simplex *generated terrain* dan perlin *generated terrain* dapat membuat variasi dari *terrain*. *Terrain* akan berubah setiap kali metode *generate* dijalankan. Pada akhir penelitian, ditemukan bahwa perlin memiliki keunggulan untuk membuat *terrain* yang memiliki tinggi yang rendah dan simplex memiliki kelebihan untuk membuat *terrain* yang tinggi.

Kata Kunci: *Procedural Content Generator* (PCG), *Simplex noise*, *Perlin Noise*, *Terrain*

ABSTRACT

In the development of game, terrain is the first thing that the player see in the game. Landscape, assets on terrain, and the height kurvator on the terrain will give an environment that can enhance the experience in playing. But the process of making such terrain will take a long time because of the terrain setting which developers do. Moreover, if the developer needs to create many terrain with a similar parameter. not just that, because the similarity from the terrain, the terrain itself will become repetitive. To avoid that, it is better to use the Procedural Content Generator or PCG to create a base terrain with every project. Simplex noise is one of many PCG method to generate a terrain height map to be developed by developer. Terrain that had been generated can be explore by the player , and to diminish the chance of slump because of the repetitive play, the surface will be different each time the method generate.

This paper will be focus on the implementation of simplex noise to generate a base terran for a 3D game with the usage of C# from unity. simplex noise will generate 3 biomes which are the forest, savanna, mountain. Perlin noise will be implemented as well for the purpose of comparison method, perlin noise will generate the same biome as simplex noise. This program will see the result of the surface generated by both methods.

This paper will test some terrain models by how the result of simplex generated terrain and perlin generated terrain can make a variety for terrain. Terrain will change each time the method is running in the program. In the result of the paper, it is found that perlin have an advantage to generate that have a low to middle height terrain and simplex have the advantage to generate a terrain that have a middle to high height.

Keywords: *Procedural Content Generator (PCG), Simplex noise, Perlin Noise, Terrain*

1. PENDAHULUAN

Dalam perkembangan zaman yang modern ini, pembuatan *video game* yang ingin menunjukan kebolehan *developer* untuk membuat sebuah suasana permainan yang lebih natural dan lebih realistis untuk dimainkan juga bertambah. Pemain tidak hanya melihat *Game Play* tetapi juga melihat grafik pada game. Game yang terlihat lebih natural akan memberikan ide dan mengembangkan kreativitas pemain dalam membuat strategi, Terutama bila lingkup yang natural yang ditunjukan adalah dalam pembuatan *Terrain* / *Map* dalam game. Semua organisme yang ikut serta dalam lingkungan sekitarnya. ini lah yang disebut ekosistem [1]. *Map* dapat dibuat dengan berbagai macam cara, cara paling gampang tentunya menggunakan ide dari *developer* sendiri untuk membuat dan mengembangkan *map* tersebut, tetapi jika dikembangkan demikian maka variasi *map* yang ada akan menjadi relatif sedikit dan tidak bervariasi antar *player*. Cara kedua adalah menggunakan *random* dalam pembuatan *map*nya, cara ini sering juga dikenal sebagai *Procedural Content Generation (PCG)*, ini membuat banyak komposisi *map* dan *terrain* tanpa harus mengatur peletakan setiap letak atributnya, hal ini membuat game tampak memiliki banyak *map* tetapi kekurangannya adalah dalam segi naturalis penempatannya. Disinilah *Perlin noise* berperan untuk menentukan *layout* dari

game dengan lebih natural. Untuk mencapai hasil dimana pegunungan dapat dipenuhi oleh pepohonan dan rerumputan, haruslah memiliki posisi yang optimal untuk meletakan asset [3]. Perlin noise sering digunakan sebagai sarana untuk membuat *terrain* dalam *game*. Karena *perlin noise* merupakan *gradient noise* sehingga memiliki perbedaan *value* dari titik 1 ke titik lainnya. *Simplex Noise* adalah sebuah perkembangan dari Perlin noise dimana *noise* ini memiliki skala dimensi yang lebih tinggi. Tetapi masih kebanyakan developer masih menggunakan *perlin noise* sebagai *main source* untuk perkembangan.

2. TINJAUAN STUDI

2.1 Procedural Content Generator

Procedural Content Generator merupakan sebuah topik yang diteliti lebih dari tiga puluh tahun, dan memberikan hasil model yang memiliki kualitas yang tinggi dan untuk *terrain* yang memiliki fitur yang spesifik, seperti *landscape*, model tanaman dan distribusi vegetasi [2]. Proses otomatis *noise layer masking* di desain untuk membolehkan *developer* membuat procedural *terrain* yang komplex [7]. .Karena *Noise* adalah kumpulan angka, maka kita perlu meng *assign* tujuan dari angka tersebut sehingga *noise* tersebut akan dibuat elevasi yang akan dipanggil. Penerapan *perlin noise* dan *simplex noise* untuk menciptakan atau mensimulasikan permukaan dataran dapat dilakukan pada *noise* dimensi dua dan tiga [9]. Dalam *genre game* yang dapat bercampur dengan *genre* yang lain, banyak aspek yang dapat dipersatukan dari masing masing *genre*. Hal ini termasuk *resource* yang terbatas, musuh, bahaya lingkungan dan juga *terrain* yang berbahaya.[8]

2.2 Perlin Noise

Perlin noise dikembangkan oleh ken perlin pada tahun 1985 sebagai teknik penghalusan (*Smoothing*) pada noise yang pada jaman itu sangat kasar untuk digunakan pada proses penciptaan *terrain*.[4]. Metode ini adalah salah satu metode paling sering digunakan oleh *developer* untuk membuat PCG. Dalam unity, *perlin noise* dapat dipanggil menggunakan Mathf.PerlinNoise(float x,float y). Dalam *function* ini, *noise* yang dihasilkan tidak sepenuhnya *random* melainkan tersusun dari gelombang yang nilainya semakin bertambah dan berkurang. *Value* yang di *return* adalah *float* yang memiliki nilai antara 0.0 dan 1.0. *Value* ini akan menjadi warna, dari 0.0 = putih sampai 1.0 = hitam, *value* di antara 0.0 dan 1.0 akan menjadi warna keabu-abuan semakin mendekati 0.0 akan semakin memutih begitu juga sebaliknya. *Perlin Noise* dapat menghasilkan *map* yang terlihat lebih natural dan *balance* dari pada *random noise*[5]

2.3 Simplex Noise

Simplex Noise dikembangkan oleh Ken Perlin yang bertujuan menggantikan algoritma *noise* yang klasik miliknya *Perlin noise* yang membuat dia memenangkan penghargaan *academy*. *Simplex noise* juga memiliki skala dimensi yang lebih tinggi dan memiliki komputasi yang lebih rendah dengan rumus O(N^ 2) dibandingkan *classic noise*. *Simplex noise* juga tidak memiliki *directional artifacts* yang terlalu kelihatan. *Simplex noise*, dalam bidang 2d merupakan tile berbentuk semacam Segitiga [6]. Unity memiliki *library* untuk *simplex noise* bernama *Fastnoise*. *Fastnoise* memiliki 2 parameter *float* untuk koordinat yang dituju sama seperti *perlin noise*. tetapi nilai *return* yang dihasilkan oleh *simplex noise* adalah -1.0 sampai 1.0, sehingga hasil dari *simplex noise* harus dinormalkan terlebih dahulu sehingga menghasilkan nilai antara 0.0 sampai 1.0.

3. DESAIN SISTEM

3.1 Desain Game

Game yang akan dibuat adalah sebuah *game* simulasi bernama *Scenery* yang membawa pemain masuk kedalam *game* untuk melihat hasil *terrain* yang telah di-generate oleh kedua metode. *Terrain* akan di-generate menggunakan perlin noise atau simplex noise sesuai dengan pilihan *player*.

3.2 Alur Simulasi

Ketika mulai simulasi player dapat memilih *Play* atau *Exit*. Untuk menunjukan pengujian simplex noise, diberikan metode perlin noise sebagai metode pelopor dan pembanding untuk responden. Karena itu jika *player* memilih *play*, maka *player* akan diberikan pilihan untuk melihat metode simplex atau perlin. Setelah *player* menentukan metode yang ingin digunakan, player akan memilih ekosistem yang akan menjadi *base* dari *terrain* yang akan di-generate. Dalam simulasi ini, *player* dapat melihat hasil *render* yang telah di-generate sesuai dengan ekosistem yang dipilih *player*. *Player* dapat memilih untuk keluar kapanpun *player* mau.

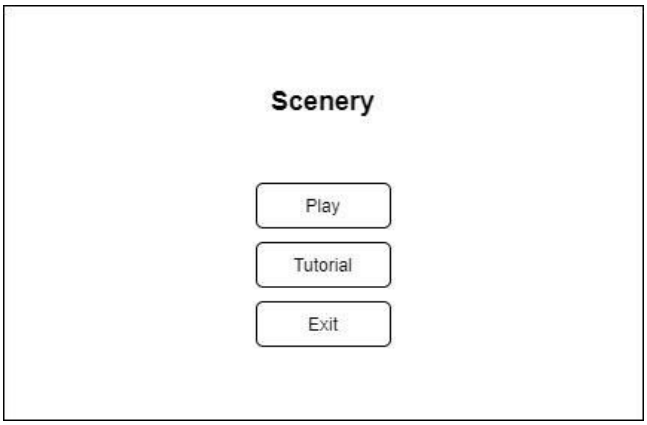
3.3 Simplex Noise Generation

Set width, *height*, dan *depth* untuk *terrain*, dalam simplex noise dibutuhkan *adjuster* dan *widden*. Adjuster akan berperan sebagai *ancore* untuk membuat nilai yang di *return* memiliki nilai 0 sampai 1 dan *widden* untuk memperbanyak *area height* dalam noise. Setelah *set* telah ditetapkan, perlu mengambil komponen melalui GetComponent, untuk mengakses data dari *terrain*. kemudian *set* data dari *terrain* dalam fungsi *generate terrain*, *size* dari data *terrain* menggunakan vector3 dari *width*, *height*, dan *depth*. Setelah *size* telah selesai di *set*, akan fungsi SetHeight dengan *parameter* (0,0,GenerateHeight). Dalam fungsi akan dibuat *float array* 2D *height* untuk membuat setiap titik dari *terrain* memiliki *float* yang berhubungan dengannya. Kemudian akan dilakukan *loop* di setiap poin itu untuk, akan di *set float value* sama seperti *value* dari perlin noise. Untuk perlin noise akan dibuat value dari *Calculate Height*, sehingga Height[x,y] = CalculateHeight. CalculateHight akan mereturn *value* dari Simplex noise. Jika ingin perlin yang tampak *random*, dapat digunakan *variabel* bebas *offsetX* dan *offsetY* untuk merender *terrain* yang berbeda setiap kali play. terrain yang telah digenerate dapat diberikan *rigid body* untuk memberikan *mass* kepada *terrain*.

3.4 Perlin Noise Generation

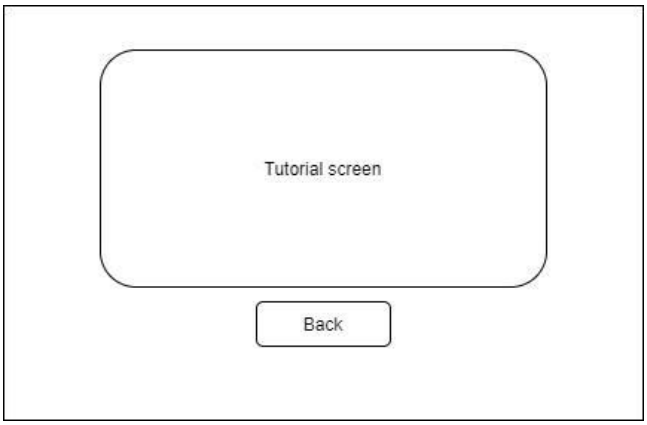
Set width, *height*, dan *depth* untuk *terrain*. Setelah *set* telah ditetapkan, perlu mengambil komponen melalui GetComponent, untuk mengakses data dari *terrain*. kemudian *set* data dari *terrain* dalam fungsi *generate terrain*, *size* dari data *terrain* menggunakan vector3 dari *width*, *height*, dan *depth*. Setelah *size* telah selesai di *set*, akan fungsi SetHeight dengan parameter 0,0,GenerateHeight(). Fungsi dari GenerateHeight akan mereturn 2D *float array*. Kemudian akan dilakukan *loop* di setiap poin itu untuk, akan diset *float value* sama seperti *value* dari perlin noise. Untuk perlin noise akan dibuat *value* dari *Calculate Height*, sehingga Height[x,y] = CalculateHeight. CalculateHight akan mereturn *value* dari Perlin noise. Jika ingin perlin yang tampak *random*, dapat digunakan variabel bebas *offsetX* dan *offsetY* untuk merender *terrain* yang berbeda setiap kali *play*. *terrain* yang telah di *generate* dapat diberikan *rigid body* untuk memberikan *mass* kepada *terrain*.

3.5 Desain User Interface



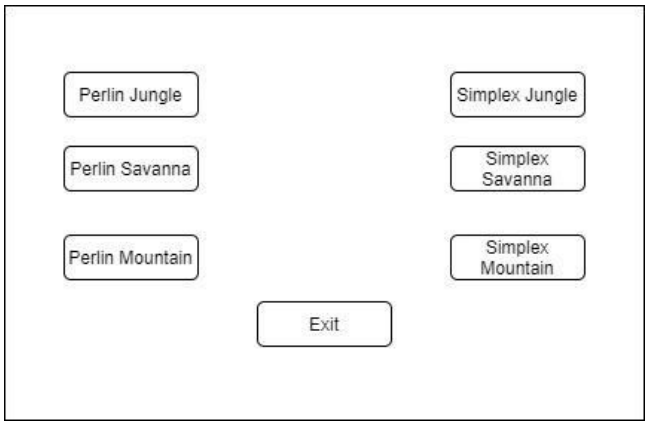
Gambar 1. Main Menu

Program akan menampilkan home menu yang memiliki 3 *button*, *Play* , *Tutorial* dan *Exit* seperti pada Gambar 1.



Gambar 2. Tutorial Menu

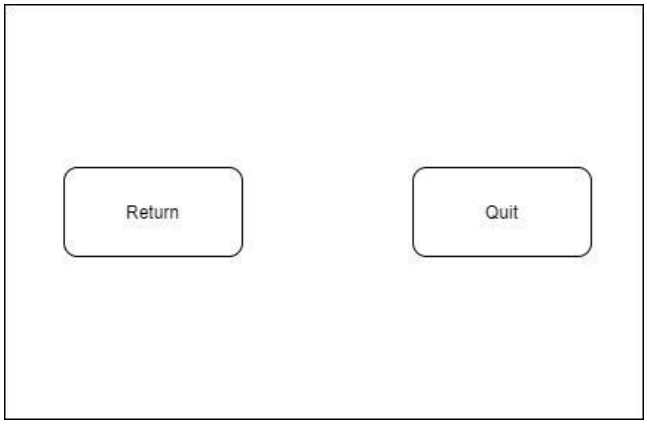
Jika *Tutorial* ditekan, *menu* akan berpindah ke *menu* untuk melihat *tutorial* seperti pada Gambar 2. Terdapat 1 *button* pada menu *Tutorial* yaitu *back button* dan 1 window untuk *text tutorial*. Jika *button back* ditekan, maka akan kembali ke *menu* awal.



Gambar 3. *Ecosystem Menu*

Pada *menu* seperti gambar 3, terdapat 7 *button*. *Button* perlin *Forest*, perlin *savana*, perlin *Mountain*, simplex *jungle*, simplex *savanna*, simplex *mountain* dan *Back*. Sesuai dengan nama buttonnya, jika "Perlin *Forest*" ditekan maka game akan membuat simulasi dengan tema *forest* dengan metode perlin noise, begitu juga dengan ekosistem lainnya. Setelah ekosistem dipilih, *Player* akan diberikan *window* dimana *player* menjadi karakter dalam simulasi dengan *style FPS.*

Karakter dapat menekan tombol *Tab* untuk mengeluarkan *menu* seperti gambar 4.



Gambar 4. *Return Menu*

Pada *window* FPS ini *player* dapat melihat *terrain* dan seluruh aspek dari *terrain* dari perspektif *player*.

4. IMPLEMENTASI SISTEM

Bab ini akan membahas lebih lanjut mengenai implementasi *code* dari pembahasan – pembahasan yang terdapat pada Bab 3. *Tools* menggunakan C# dari unity dengan *library* UnityEngine, dan FastNoise

Implementasi sistem dilakukan pada komputer dengan spesifikasi:

- RAM: 16GB, DDR4
- Memory: 1TB HDD, 256GB SSD
- CPU: Intel Core i7 8750H
- GPU: NVIDIA GeForce GTX 1060
- OS Windows 10

5. PENGUJIAN SISTEM

Bab ini akan menjelaskan lebih lanjut mengenai hasil *render terrain* dari metode perlin dan simplex noise dalam mengolah *heightmap* untuk menghasilkan *terrain* yang sesuai dan natural. Terdapat berbagai macam bioma yang akan ditampilkan dengan bentuk heightmap yang berbeda-beda.

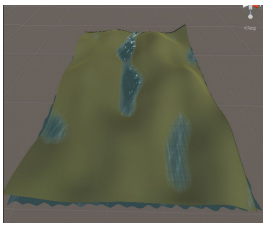
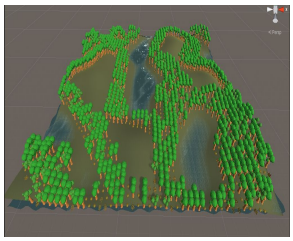
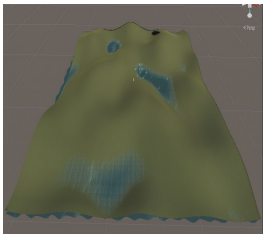
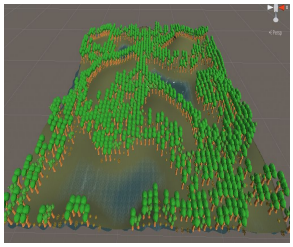
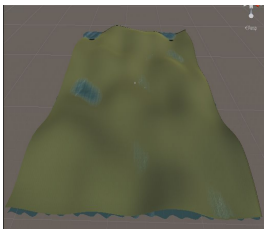
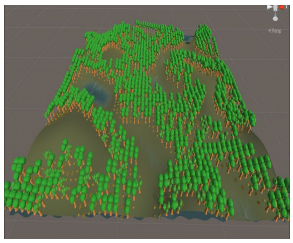
5.1 Pengujian Hasil *Terrain* Yang di-*Generate* Menggunakan Perlin Noise

Bagian ini mengulas mengenai percobaan menggunakan Perlin Noise sebagai model noise untuk mengembangkan *terrain*.

5.1.1 Jungle In Perlin

Tabel 1 adalah pengujian hasil dari *map* dengan ukuran 256 x 256 yang di-generate menggunakan perlin noise dengan 3 jenis model *terrain* yang telah di-generate. *Terrain* yang dihasilkan memiliki *depth* 24, *scale* , *resource* yang akan di *generate*. Dalam Perlin *terrain*, *height* yang di bawah 8 poin merupakan tinggi dari air. Objek - objek akan di *spawn* apabila ketinggian(Y) yang didapat dari *raycast*, memiliki ketinggian(Y) lebih dari 8 dan kurang dari 15 poin. Dalam *Spawn*, ada *chance* objek yang akan di *spawn*, *chance* dari 10%, 20%, 25%, 40% dan 5%. Dalam *terrain jungle* perlin ini, 10% akan *spawn* pohon tipe 2, 20% akan *spawn grass*, 25% akan *spawn bush*, 40% akan *spawn* pohon tipe 1. sekangkan 5% sisanya akan *spawn null* atau tidak *spawn* sama sekali.

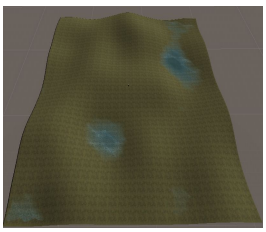
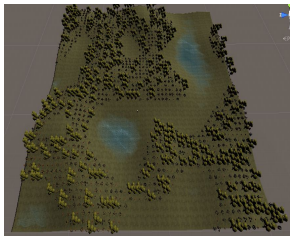
Tabel 1. Tabel Perlin Noise Jungle

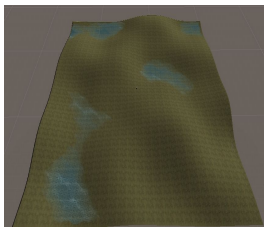
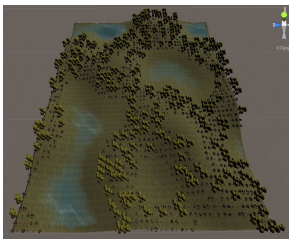
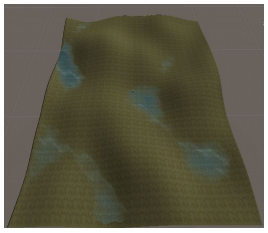
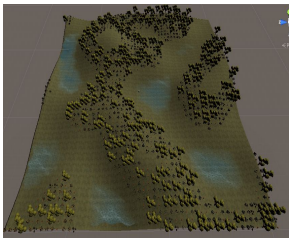
Model Terrain	Model With Asset
	
	
	

5.1.2 Savanna In Perlin

Tabel 2 adalah pengujian hasil dari *map* dengan ukuran 256 x 256 yang di-generate menggunakan perlin noise dengan 3 jenis model *terrain* yang telah di-generate. Hasi ada pada Tabel 5.2. *Terrain* yang dihasilkan memiliki *depth* 19, *scale* -3.4, *resource* yang akan di generate. Dalam Perlin *terrain*, *height* yang di bawah 8 poin merupakan tinggi dari air. Objek - objek akan di *spawn* apabila ketinggian(Y) yang didapat dari *raycast*, memiliki ketinggian(Y) lebih dari 8 dan kurang dari 15 poin. Dalam *Spawn*, ada *chance* objek yang akan di *spawn*, *chance* dari 10%, 20%, 25%, 40% dan 5%. Dalam *terrain jungle perlin* ini, 10% akan RockGroup2, 20% akan *spawn* pohon tipe 3, 25% akan *spawn* batang pohon yang telah di tebang (*Stump*), 40% akan *spawn* Rumput. Sekangkan 5% sisanya akan *spawn null* atau tidak *spawn* sama sekali.

Tabel 2. Tabel Perlin Savanna

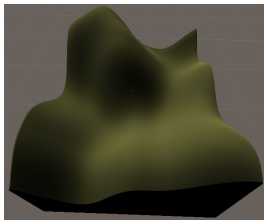
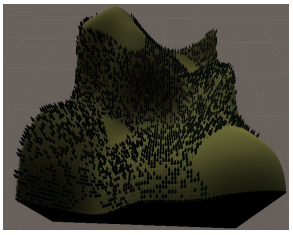
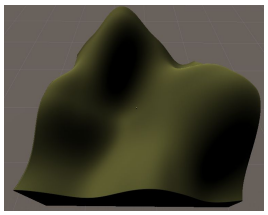
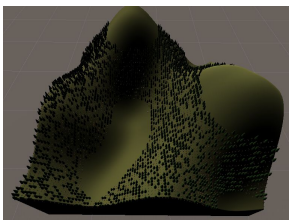
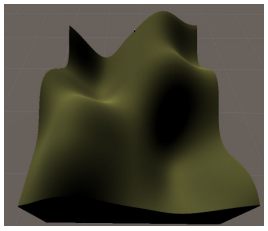
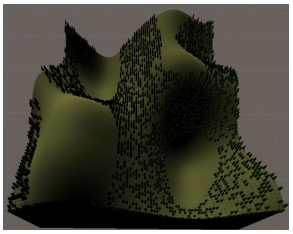
Model *Terrain*	Model With Asset
	

5.1.3 Mountain In Perlin

Tabel 3 adalah pengujian hasil dari *map* dengan ukuran 256 x 256 yang di-generate menggunakan *perlin noise* dengan 3 jenis model *terrain* yang telah di-generate. Hasi ada pada Tabel 5.3. *Terrain* yang dihasilkan memiliki *depth* 19, *scale* -3.4, *resource* yang akan di *generate*. Dalam *Perlin terrain*, *height* yang di bawah 8 poin merupakan tinggi dari air. Objek - objek akan di *spawn* apabila ketinggian(Y) yang didapat dari *raycast*, memiliki ketinggian(Y) lebih dari 8 dan kurang dari 15 poin. Dalam *Spawn*, ada *chance objek* yang akan di *spawn*, *chance* dari 10%, 20%, 25%, 40% dan 5%. Dalam *terrain mountain perlin* ini, 10% akan RockGroup2, 20% akan *spawn* pohon tipe 3, 25% akan *spawn* batang pohon yang telah di tebang (*Stump*), 40% akan *spawn* Rumput. Sekangkan 5% sisanya akan *spawn null* atau tidak *spawn* sama sekali.

Tabel 3. Tabel Perlin Mountain

Model Terrain	Model With Asset
	
	
	

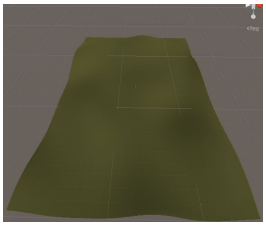
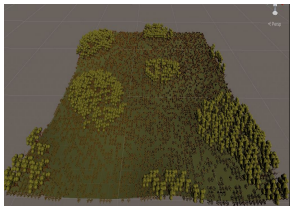
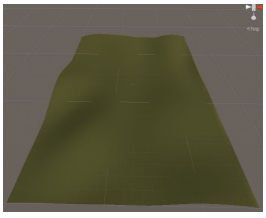
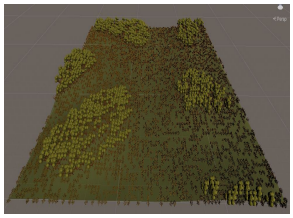
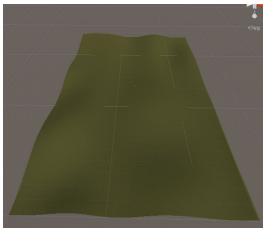
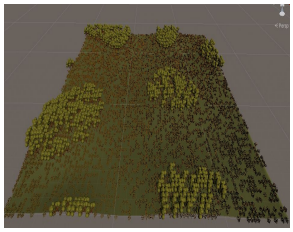
5.2 Pengujian Hasil *Terrain* Yang di-Generate Menggunakan Simplex Noise

Bagian ini mengulas mengenai percobaan menggunakan *Simplex Noise* sebagai model *noise* untuk mengembangkan *terrain*. *Simplex noise* memiliki *scale noise* yang lebih kecil dari pada *perlin noise*, karena itu *scale* yang diberikan juga akan lebih besar dalam *generator* ini.

5.2.1 Jungle In Simplex

Tabel 4 adalah pengujian hasil dari *map* dengan ukuran 256 x 256 yang di-generate menggunakan *simplex noise* dengan 3 jenis model *terrain* yang telah di-generate. Hasi ada pada Tabel 4. *Terrain* yang dihasilkan memiliki *depth* 24, *scale* 116. Dalam *Perlin terrain*, *height* yang di bawah 3 poin merupakan tinggi dari air. Objek - objek akan di *spawn* apabila ketinggian(Y) yang didapat dari *raycast*, memiliki ketinggian(Y) lebih dari 8 dan kurang dari 15 poin. Dalam *Spawn*, ada *chance* objek yang akan di *spawn*, *chance* dari 10%, 20%, 25%, 40% dan 5%. Dalam *terrain jungle simplex* ini, 10% akan rumput hijau, 20% akan *spawn* pohon kering, 25% akan *spawn* pohon, 40% akan *spawn* rumput kuning. Sekangkan 5% sisanya akan *spawn null* atau tidak *spawn* sama sekali.

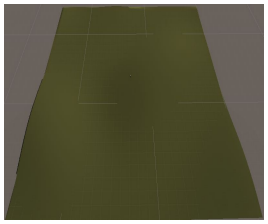
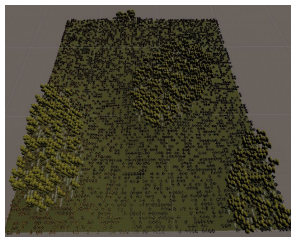
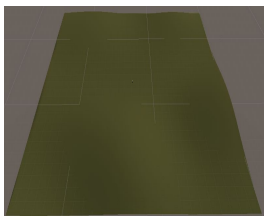
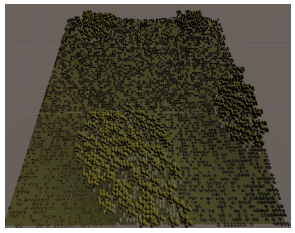
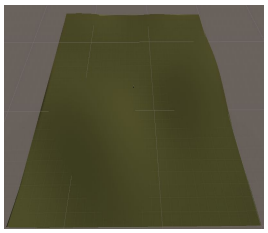
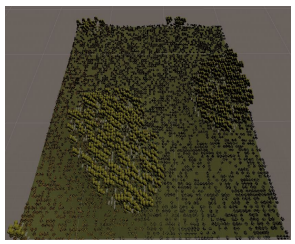
Tabel 4. Tabel Simplex Jungle

Model Terrain	Model With Asset
	
	
	

5.2.2 Savanna In Simplex

Tabel 5 adalah pengujian hasil dari *map* dengan ukuran 256 x 256 yang di-generate menggunakan *simplex noise* dengan 3 jenis model *terrain* yang telah di-generate. Hasi ada pada Tabel 5. *Terrain* yang dihasilkan memiliki *depth* 19, *scale* -3.4, *resource* yang akan di *generate*. Dalam simplex *terrain*, *height* yang di bawah 8 poin merupakan tinggi dari air. Objek - objek akan di *spawn* apabila ketinggian(Y) yang didapat dari *raycast*, memiliki ketinggian(Y) lebih dari 8 dan kurang dari 15 poin.

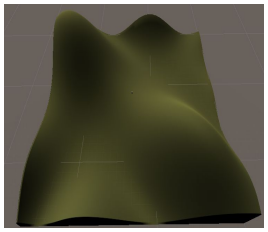
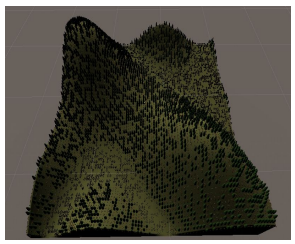
Tabel 5. Tabel Simplex Savanna

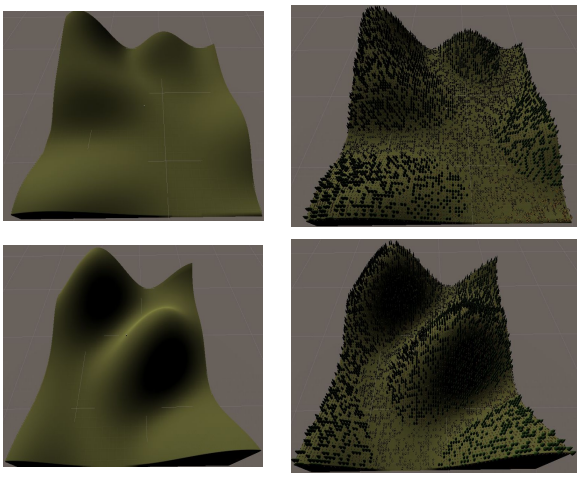
Model Terrain	Model With Asset
	
	
	

5.2.3 Mountain In Simplex

Tabel 6 adalah pengujian hasil dari *map* dengan ukuran 256 x 256 yang di-generate menggunakan *simplex noise* dengan 3 jenis *model terrain* yang telah di-generate. Hasi ada pada Tabel 6. *Terrain* yang dihasilkan memiliki *depth* 19, *scale* -3.4, *resource* yang akan di *generate*. Dalam *Simplex terrain*, *height* yang di bawah 8 poin merupakan tinggi dari air. Objek - objek akan di *spawn* apabila ketinggian(Y) yang didapat dari *raycast*, memiliki ketinggian(Y) lebih dari 8 dan kurang dari 15 poin. Dalam *Spawn*, ada chance objek yang akan di *spawn*, chance dari 10%, 20%, 25%, 40% dan 5%. Dalam *terrain mountain simplex* ini, 10% akan RockGroup2, 20% akan *spawn* pohon tipe 3, 25% akan *spawn* batang pohon yang telah di tebang (Stump), 40% akan *spawn* Rumput. Sekangkan 5% sisanya akan *spawn null* atau tidak *spawn* sama sekali.

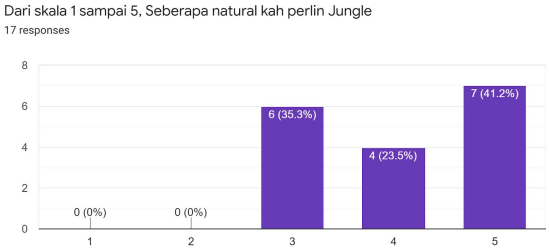
Tabel 6. Tabel Simplex Mountain

Model Terrain	Model With Asset
	



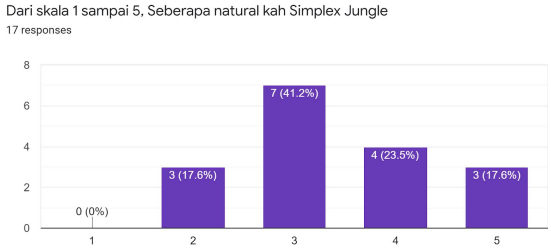
5.3 Pengujian Hasil Kuisoner

Setelah program telah dibuat, 17 responden berusia antara *23* sampai *25* tahun, dan memiliki pengalaman bermain dengan game bergenre 3D setidaknya 1 tahun diminta untuk menjalankan dan memainkan simulasi *Scenery*. Responden diminta menjalankan 6 simulasi *terrain* mulai dari *Perlin Jungle* sampai *Simplex Mountain*. Setelah selesai memainkan simulasi, responden diminta mengisi kuisioner.



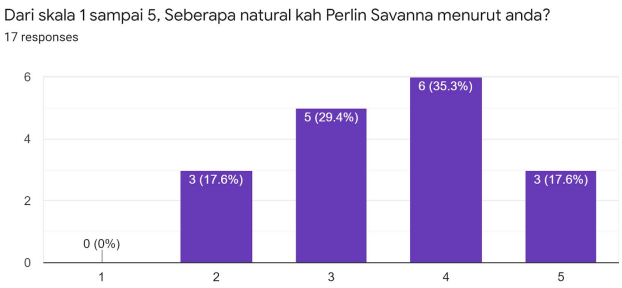
Gambar 5. Bar Chart Responden Perlin Jungle

Pada Gambar 5, dari 17 responden, 41,2% berpendapat bahwa *terrain jungle* yang dihasilkan oleh perlin sangat natural. 23,5% responden berpendapat bahwa *terrain* yang dihasilkan sudah natural. sedangkan 35,3% menyatakan bahwa *terrain* yang dihasilkan biasa saja.



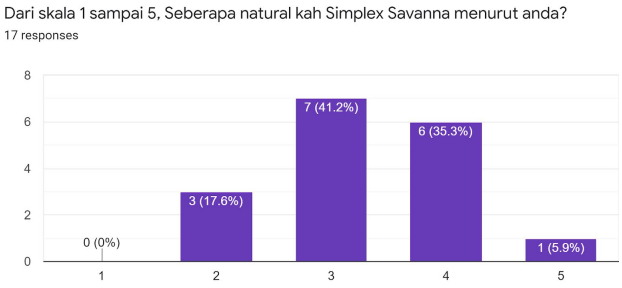
Gambar 6. Bar Chart Responden Simplex Jungle

Pada gambar 6, dari 17 responden, 17,6% berpendapat bahwa *terrain jungle* yang dihasilkan oleh *simplex* sangat natural. 23,5% responden berpendapat bahwa *terrain* yang dihasilkan sudah natural. sedangkan 41,2% menyatakan bahwa *terrain* yang dihasilkan biasa saja, dan 17,6% berpendapat bahwa *terrain* yang dihasilkan kurang natural.



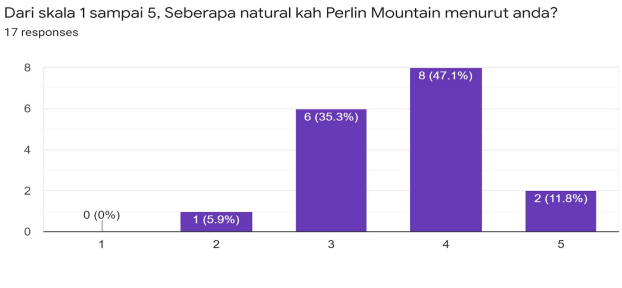
Gambar 7. Bar Chart Responden Perlin Savanna

Pada Gambar 7, dari 17 responden, 17,6% berpendapat bahwa *terrain savanna* yang dihasilkan oleh *perlin* sangat natural. 35,5% responden berpendapat bahwa *terrain* yang dihasilkan sudah natural. sedangkan 29,4% menyatakan bahwa *terrain* yang dihasilkan biasa saja, dan 17,6% berpendapat bahwa *terrain* yang dihasilkan kurang natural.



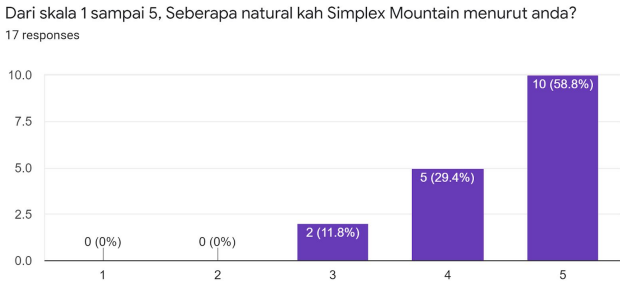
Gambar 8. Bar Chart Responden Simplex Savanna

Pada Gambar 8, dari 17 responden, 5,9% berpendapat bahwa *terrain savanna* yang dihasilkan oleh *simplex* sangat natural. 35,5% responden berpendapat bahwa *terrain* yang dihasilkan sudah natural. sedangkan 41,4% menyatakan bahwa *terrain* yang dihasilkan biasa saja, dan 17,6% berpendapat bahwa *terrain* yang dihasilkan kurang natural.



Gambar 9. Bar Chart Responden Perlin Mountain

Pada Gambar 9, dari 17 responden, 11,8% berpendapat bahwa *terrain mountain* yang dihasilkan oleh *perlin* sangat natural. 47,1% responden berpendapat bahwa *terrain* yang dihasilkan sudah natural. sedangkan 35,3% menyatakan bahwa *terrain* yang dihasilkan biasa saja, dan 5,9% berpendapat bahwa *terrain* yang dihasilkan kurang natural.



Gambar 10. Bar Chart Responden Simplex Mountain

Pada Gambar 10, dari 17 responden, 58,8% berpendapat bahwa *terrain mountain* yang dihasilkan oleh *simplex* sangat natural. 29,1% responden berpendapat bahwa *terrain* yang dihasilkan sudah natural. sedangkan 11,8% menyatakan bahwa *terrain* yang dihasilkan biasa saja.

Dari Hasil ke-6 chart yang kita dapatkan, dapat diberi poin untuk setiap jawaban skala dari responden. Untuk skala 1 diberikan poin -2, untuk skala 2 diberikan poin -1, untuk skala 3 diberikan poin 0, untuk skala 4 diberikan poin +1, dan untuk skala 5 diberikan poin +2.

Tabel 7. Tabel poin Perlin Noise

Model Terrain	Model With Asset
Perlin Jungle	14
Perlin Savanna	9
Perlin Mountain	11

Tabel 8. Tabel poin Simplex Noise

Model Terrain	Model With Asset
Simplex Jungle	7
Simplex Savanna	5
Simplex Mountain	25

Dari hasil perhitungan pada Tabel 7 dan Tabel 8 dapat dilihat bahwa dalam *generate* dataran yang memiliki aspek *jungle*, *Perlin* jauh lebih unggul. Untuk generate terrain *savanna*, *perlin* juga lebih unggul 4 poin daripada *simplex*. Sedangkan bila merender *terrain* yang memiliki aspek *mountain*, *Simplex* jauh lebih unggul dalam generatenya. Tetapi jika dalam keseluruhan hasil *render* didapat bahwa total poin yang dimiliki oleh *Perlin* adalah 34 poin, sedangkan *Simplex* memiliki 37 poin.

6. KESIMPULAN DAN SARAN

Bab ini memuat kesimpulan dan saran dalam skripsi ini.

6.1 Kesimpulan

Setelah dilakukan eksperimentasi dan analisa menggunakan 3 buah model *Perlin noise* dan 3 buah model *Simplex noise* dalam membuat *Terrain*, dapat ditarik beberapa kesimpulan sebagai berikut :

- Metode *Perlin Noise* memiliki variatif *height* yang lebih rapat dibandingkan *Simplex noise* sehingga *Perlin noise* menghasilkan dataran yang lebih bergelombang dan kurvator yang tinggi. Hal ini mengakibatkan asset seperti sungai dan danau sangat mudah diimplementasikan pada PCG dari perlin noise.
- Metode *Simplex Noise* memiliki pola yang ber-artefak membuat setiap ada *height* yang dihasilkan, *height* tersebut akan dikelilingi oleh *surface* yang memiliki *value* 0 yang berarti akan memberikan dataran yang benar-benar datar dan tidak bergelombang. Ini mengakibatkan *simplex noise* menjadi kurang *appealing* saat menghasilkan *terrain* seperti *jungle* dan *savanna*. Sedangkan *value* 0 yang berada disekitar *surface* yang memiliki kurvator, membuat asset tampil berlebihan, seperti *asset* air yang seakan akan meluap karena banyaknya *water level* pada *terrain*. Tetapi jika digunakan untuk membuat terrain yang memiliki kuravator yang tinggi seperti *mountain*, *simplex* mampu memberikan hasil pencapaian yang bagus karena *value* 0 pada *simplex* memberikan kesan bahwa *terrain mountain* memiliki *base* dataran yang disekitar *mountain*nya.
- *Perlin* dan *Simplex* memiliki keunggulan dalam masing-masing *terrain* yang di-generate tetapi dapat disimpulkan *perlin noise* lebih variatif dan dapat mengembangkan lebih banyak *terrain* natural daripada *simplex noise*.

6.2 Saran

Setelah dilakukan eksperimentasi dan analisa ,dapat ditarik beberapa saran sebagai berikut :

- Karena 2 metode ini merupakan *predecessor* dan *successor*, penelitian selanjutnya bisa mencoba untuk menggabungkan kedua metode menjadi satu *biome* yang besar. Contohnya : percampuran antara 3 biome, yaitu *jungle*, *savanna*, dan *mountain* menjadi satu *biome* besar. *Jungle* dan *savanna* menggunakan *perlin* sedangkan mountain menggunakan *simplex*.
- Memberikan *Random spawn* metode dalam penentuan *spawn rate asset* dalam *terrain*. Dalam skripsi ini, asset hanya di *spawn* berdasarkan tinggi dari *height map* yang didapatkan dan dicek berdasarkan point-point dari *terrain*. Hal ini membuat *player* dapat melihat pola *spawn* dalam *terrain*, untuk penelitian selanjutnya, dapat diberikan variabel "*random*" pada spawnernya.

7. DAFTAR PUSTAKA

[1] Balasubramanian.A. 2017.*Ecosystem And Its Component.*

[2] Bidarra. Rafael 2010. Procedural Natural System for Level Design.

[3] Cho Kuk, Baeg SeungHo, dan Park Sangdeok. 2013. *Natural Terrain Detection and SLAM using LIDAR for UGV*.

[4] Erstu Erich, Sell Janar, dan Valli Suido. 2012 Perlin Noise Generator.

[5] Ginove Joshua Hertanto W. 2019.Game Membangun Kerajaan Dengan *Procedural Generated Map* menggunakan *perlin noise*.

[6] Gustavson Stevan. 2005. *Simplex Noise Demystified*. DOI= https://www.researchgate.net/publication/216813608

[7] Hyttinen Tuomo, Makinrn Erkki, dan Poranen Timo. 2017. Terrain Synthesis Using Noise by Example.

[8] Reid Samantha, dan Downing Steven 2018.*Survival Theme Video Game and Cultural Constructs of Power*.

[9] Wijaya Wandi, dan Rahman Abdul. 2018. Analisi Perbandingan Perlin noise dan Simplex noise Untuk Menciptakan Permudaan Dataran pada Pembuatan Game.