**BAB I**

**PENDAHULUAN**

* 1. Latar Belakang

Pada era modern ini, teknologi semakin maju seiring dengan berkembangnya zaman, sudah banyak teknologi yang bermunculan yang dimanfaatkan untuk bidang – bidang tertentu seperti di bidang medis, pendidikan, pariwisata, transportasi, maupun di bidang game itu sendiri. Generator untuk game merupakan salah satu contohnya. Sudah cukup banyak generator yang bermunculan dan diterapkan dalam pembuatan game maupun aplikasi lainnya karena dengan menggunakan generator pembuatan berbagai macam game maupun aplikasi lainnya jauh lebih mudah.

Pada beberapa kondisi, plugin yang sudah ada digunakan untuk beberapa tujuan, serta dapat dimanfaatkan untuk beragam keperluan, diantaranya membantu developer untuk membuat aplikasi maupun game berbasis VR (Virtual Reality) hingga membantu developer untuk membuat beberapa hal di dalam game, mulai dari segi environment, UI (User Interface), gameplay, dan beberapa hal lainnya yang berkaitan dengan game.

Salah satu generator yang dapat digunakan pada bidang environment ialah generator yang berupa map generator untuk game, terutama game-game yang menggunakan tema rpg dan open world, namun plugin world itu sendiri belum benar-benar dirancang hingga ke dalam bagian yang dapat memudahkan developer tinggal mengatur suatu parameter agar dunia yang diinginkan terbentuk sesuai dengan harapannya, bahkan sebagian besar generator itu sendiri, sebenarnya hanya berupa dasaran yang nantinya akan dikembangkan oleh developer dalam penggunaan generatot yang akan dipakai ke dalam kode pemrograman serta harus dapat diimplementasikan sendiri, bahkan terkadang dapat memberikan kesulitan tersendiri bagi developer yang akan membuat game berbasis RPG (Role Playing Game) seperti Skyrim, Final Fantasy, dan game – game lainnya yang memanfaatkan pembuatan map di dalamnya. Salah satu contoh dari map generator yang ada di dalam unity ialah seperti kubikos 3D world yang dimana berupa map 3D namun berupa tiles dalam dunianya yang dimana tidak menggunakan lowpoly yang mengakibatkan beratnya saat dijalankan. Pada plugin lain seperti open world nature kit merupakan dunia 3D yang dimana pengguna dapat melakukan pembuatan dunia dengan sesuka hati namun pada plugin ini pula memiliki yang sama yaitu kurangnya lowpoly yang mengakibatkan beratnya plugin saat dijalankan.

Maka dari itu, penulis memberikan solusi berupa plugin berbasis map generator untuk mempermudah pembuatan dunia dengan memasukkan parameter-parameter yang sesuai dengan kebutuhan. Dengan bantuan plugin berupa generator ini bisa mempermudah developer untuk membuat map dalam game khusunya game dengan tema RPG sesuai dengan harapan developer dan perkembangan selanjutnya dapat membantu developer untuk mengembangkan game maupun aplikasi yang akan dibuat.

* 1. Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas dan permasalahan yang akan diteliti oleh penulis, maka dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana memanfaatkan generator 3D yang hanya diterapkan di aplikasi unity ini ?
2. Bagimana cara untuk membuat generator 3D ini menjadi lebih ringan saat dijalankan di suatu device?
   1. Batasan Masalah

Dengan banyaknya masalah yang timbul dalam pembuatan aplikasi ini, maka penulis mencoba membatasi ruang lingkup permasalahan pada dua bagian, yaitu:

1. Plugin ini hanya diterapkan di unity dan tidak dikembangkan di aplikasi lain seperti unreal engine dan semacamnya.
2. Plugin ini tidak meliputi pembuatan makhluk hidup dan bangunan, melainkan geografis dari dunia 3D secara mendasar
   1. **Tujuan**

Tujuan proyek akhir " Plugin 3D Map Generator Untuk Pembuatan Dunia Dalam Game” antara lain:

* + 1. Membantu mempermudah developer untuk membuat dunia game sendiri tanpa membuatnya dari awal.
    2. Membantu menghasilkan model map 3D yang mendetail dengan menggunakan cara yang lebih mudah.
    3. Membantu Pengguna membuat dunia 3D yang lebih fleksibel
  1. Manfaat

Manfaat proyek akhir " Plugin 3D Map Generator Untuk Pembuatan Dunia Dalam Game” antara lain:

1. Memberikan pemecahan masalah terkait pembuatan dunia 3D di dalam aplikasi unity.
2. Memaksimalkan dan mempermudah pembuatan dunia 3D menggunakan aplikasi unity.
   1. Metodologi Penelitian

Dalam penyusunan laporan kerja praktik, dan metode yang akan digunakan agar memperoleh data yang akurat, maka penulis menggunakan beberapa metode, diantaranya :

1. Study Literatur

Mempelajari bahasa pemrograman C# dan penggunaan aplikasi unity sebagai langkah awal untuk membangun plugin ini.

1. Pengumpulan Data

Mempelajari dan mengumpulkan data untuk disusun menjadi sebuah struktur data sesuai dengan sistem yang akan dibuat

1. Analisa Dan Perancangan Desain Sistem

Melakukan perancangan Desain sistem yang akan dibuat untuk mengintegrasikan data-data yang telah terkumpul.

1. Implementasi Sistem dan Pengujian

Dari data – data yang sudah dianalisa tersebut akan dibentuk untuk suatu sistem yang dapat membantu dalam pengolahan data dan setelah sistem selesai maka dilakukan pengujian plugin untuk mengetahui kelemahan- kelemahan yang ada serta melakukan perbaikan.

1. Pembuatan Laporan

Dari hasil program yang telah dirancang kemudian dilakukan pembuatan laporan akhir.

* 1. Sistematika Penulisan

Sistematika Penulisan adalah sebagai berikut :

Bab I Pendahuluan

Pada tahap ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan pengerjaan Tugas Akhir, manfaat pengerjaan Tugas Akhir, metodologi yang digunakan serta sistematika penulisan dalam penyusunan Tugas Akhir ini.

Bab II Landasan Teori

Dasar teori, yang berisi teori – teori yang menunjang pembuatan Tugas Akhir seperti noise mapping, curve, level of detail, dan topik – topik lainnya.

Bab III Metodologi

Perancangan dan pembuatan sistem, berisi diagram – diagram perancangan sistem, jalannya sistem algoritma, perancangan antarmuka (*User Interface*), serta langkah – langkah dan proses pembuatan program.

Bab IV Hasil Pengujian dan Analisa

Pada bab ini akan dibahas hasil komparasi dari pembuatan dunia 3D yang dijalankan oleh program. Komparasi dilakukan dengan melakukan perbandingan dari parameter – parameter pada antarmuka yang telah disediakan dan analisa diambil berdasarkan hasil komparasi.

Bab V Kesimpulan

Pada bab ini akan membahas kesimpulan dari pembahasan, perencanaan, pengujian, dan analisa berdasarkan data hasil pengujian sistem informasi.

**BAB II**

**TINJAUAN PUSTAKA**

* 1. Penelitian yang pernah dilakukan

Dunia game untuk video game modern semakin meningkat dalam skala dan detail, membutuhkan semakin banyak sumber daya untuk membuat konten ini. Kemampuan untuk menghasilkan dunia game secara prosedural yang memungkinkan pembuatan dunia game dengan detail serta kualitas tekstur, model, karakter, dan konten lain yang berkualitas, yang dibuat secara manual sehingga sesuai dengan keinginan pengguna.

Generator medan prosedural digunakan untuk membuat bentuklahan untuk aplikasi seperti game komputer dan simulator penerbangan. Sementara sebagian besar pekerjaan yang ada berpusat pada algoritma yang menghasilkan medan dengan bantuan masukan nilai dari pengguna. Ini memungkinkan perancang untuk membuat medan prosedural yang memiliki bentuk spesifik sesuai dengan keinginan

* 1. Teori panunjang yang digunakan dalam penelitian
     1. Noise Mapping

Noise mapping atau pemetaan kebisingan adalah suatu sketsa yang sangat teliti yang menggambarkan letak relatif dari semua titik sampling kebisingan. Ke dalam sketsa ini ditambahkan data tingkat kebisingan di sekitar titik sampling kebisingan. Adanya garis yang menghubungkan titik-titik di area kerja yang mempunyai tingkat kebisingan yang sama.

Peta kebisingan adalah dokumen tentang kebisingan di site plant yang disajikan dalam bentuk peta, berisi plot-plant area yang digambarkan dengan square grid lines yang berjarak 5 meter antara grid lines. Tingkat kebisingan tertera pada setiap grid lines.



**Gambar 2.1.** Contoh Noise Mapping

Sumber : ([www.google.co.id](http://www.google.co.id))

Pada dasarnya nilai-nilai tingkat kebisingan dipengaruhi oleh berbagai faktor, diantaranya:

1. Frekuensi

Frekuensi adalah jumlah satuan getaran yang dihasilkan dalam satuan waktu (detik), dengan satuan hertz (Hz). Frekuensi suara yang dapat didengar oleh manusia mulai dari 20 Hz sampai dengan 20.000 Hz.

1. Intensitas Suara

Intensitas suara didefinisikan sebagai energi suara rata-rata yang ditransmisikan melalui gelombang suara menuju arah perambatan dalam media (udara, air, benda, dan sebagainya)..

1. Amplitudo

Amplitudo adalah satuan kuantitas suara yang dihasilkan oleh sumber suara pada arah tertentu.

1. Kecepatan Suara

Kecepatan suara adalah satuan kecepatan perpindahan perambatan udara per satuan waktu.

1. Panjang Gelombang

Panjang gelombang adalah jarak yang ditempuh oleh perambatan suara untuk satu siklus.

1. Periode

Periode adalah waktu yang dibutuhkan untuk satu siklus amplitudo dengan satuan detik

1. Octave Band

Oktave band merupakan kelompok-kelompok frekuensi tertentu dari suara yang dapat didengar dengan baik oleh manusia.

1. Frekuensi Bandwith

Frekuensi bandwidth dipergunakan untuk pengukuran suara industri.

1. Puretone

Puretone adalah gelombang suara yang terdiri hanya dari satu jenis amplitudo dan satu jenis frekuensi.

1. Loudness

Loudness adalah persepsi pendengaran terhadap suara pada amplitudo tertentu. Satuannya adalah phon, 1phon setara dengan 4 dB pada frekuensi 1000 Hz.

1. Kekuatan Suara

Kekuatan suara adalah satuan dari total energi yang dipancarkan oleh suara per satuan waktu.

1. Tekanan Suara

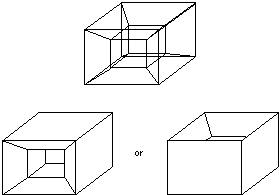
Tekanan suara adalah satuan daya tekan suara per satuan luas.

* + 1. Curves

Kurva memainkan peran yang sangat signifikan dalam pemodelan CAD (Computer Aided Design), terutama untuk menghasilkan model rangka gambar, yang merupakan bentuk paling sederhana untuk mewakili model. Kita dapat menampilkan objek pada layar monitor dalam tiga bentuk model komputer yang berbeda, yaitu :

1. Wireframe Model (Model Rangka Gambar)
2. Surface Model (Model Permukaan)
3. Solid Model (Model Padat)
   * + 1. Wireframe Model

Wireframe Model (Model Rangka Gambar) terdiri dari titik dan kurva saja, dan tampak seolah dibuat dengan sekelompok kabel. Ini adalah modelx CAD objek yang paling sederhana. Kelebihan dari model jenis ini ialah pembuatannya yang mudah dan dapat dilakukan dengan perangkat keras dan perangkat lunak tingkat rendah, serta tidak memerlukan penyimpanan data yang banyak. Kelemahan utama dari model rangka gambar adalah pengguna atau pembuat harus dapat memvisualisaikan apa yang ingin dibuat.

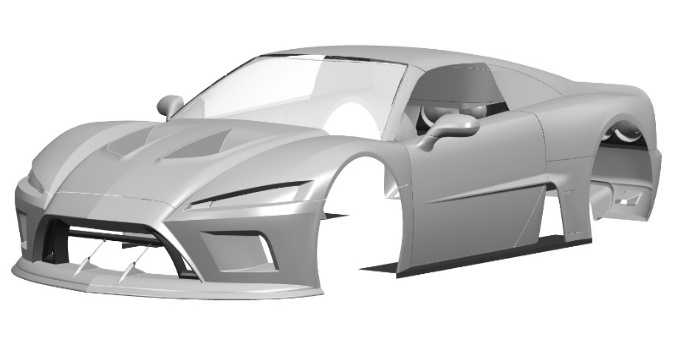


**Gambar 2.2**. Contoh Model Rangka Gambar

Sumber : ([www.google.co.id](http://www.google.co.id))

Terlepas dari ambiguitasnya, model rangka gambar masih merupakan bentuk yang paling disukai, karena dapat dibuat dengan cepat dan mudah untuk memverifikasi konsep suatu objek. Pembuatan model kerangka gambar agak mirip dengan menggambar sketsa dengan tangan untuk berkomunikasi atau membuat konsep suatu objek. Seperti yang dinyatakan sebelumnya, model gambar rangka dibuat hanya menggunakan titik dan kurva.

* + - 1. Surface Model



**Gambar 2.3.** Contoh Model Permukaan

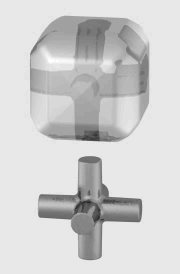
Sumber : ([www.google.co.id](http://www.google.co.id))

Tampilan atau resolusi model permukaan tergantung pada keadaan yang kita miliki untuk menjalankan model yang ingin dibuat. Untuk model yang tampak realistis, kita perlu memenuhi beberapa kriteria seperti, kita membutuhkan memori komputer yang besar, memiliki perangkat lunak dan perangkat keras yang terbilang cukup untuk menjalankan model tersebut dan beberapa faktor lainnya. Namun, kita juga bisa memilih model yang tidak terlalu realistis, yang dimana tidak memerlukan spesifikasi sistem perangkat lunak dan keras yang terlalu besar dan hanya memerlukan memori yang tidak terlalu besar.

Model permukaan berguna untuk mewakili permukaan seperti botol minuman ringan, penyok mobil, sayap pesawat, dan secara umum, permukaan melengkung yang rumit. Salah satu keterbatasan model permukaan adalah bahwa tidak ada definisi geometris dari titik-titik yang terletak di dalam atau di luar permukaan.

* + - 1. Solid Model

Representasi suatu objek dengan model padat adalah relatif konsep baru. Hanya ada beberapa program CAD pemodelan padat yang tersedia di akhir 1980-an, dan mereka membutuhkan komputer mainframe untuk dapat berjalan. Namun, pada tahun 1990-an, karena biaya rendah dan produktivitas yang dituntut cepat, Personal Computer menjadi platform pemodelan padat yang paling populer, mendorong hampir semua vendor CAD untuk memperkenalkan perangkat lunak pemodelan 3D solid yang akan dijalankan pada PC.



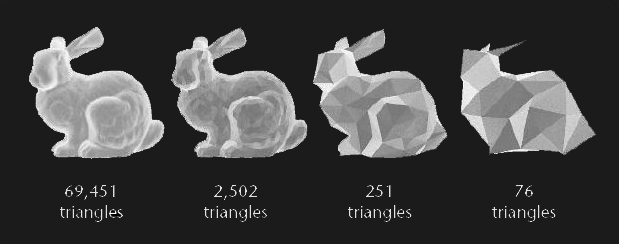
**Gambar 2.4.** Contoh Model Padat

Sumber : ([www.google.co.id](http://www.google.co.id))

Model padat mewakili objek dalam bentuk yang sangat realistis dan tidak ambigu. Namun, model - model padat tersebut membutuhkan sejumlah besar memori penyimpanan dan perangkat keras komputer kelas atas yang dapat menjalankan model padat tersebut. Model padat yang telah dibuat dapat diarsir dan ditampilkan dalam warna yang diinginkan untuk memberikan penampilan yang lebih realistis.

* + 1. Level Of Detail

Level Of Detail (LOD) merupakan suatu tingkat detail dari sebuah objek atau kumpulan polygon. LOD berfungsi untuk mengurangi banyaknya polygon yang akan digambar dengan menggunakan parameter tertentu. Fungsi lain yang mungkin dapat digunakan dari LOD adalah kompresi pada penyimpanan objek dengan memperkecil tingkat detail dari suatu objek.



**Gambar 2.5.** Contoh Level Of Detail

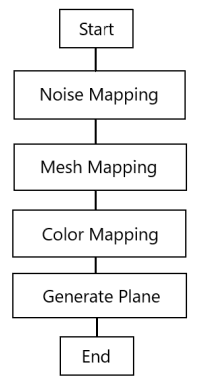
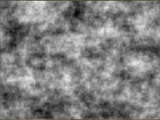
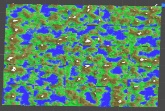
Sumber : ([www.google.co.id](http://www.google.co.id))

View-dependent LOD merupakan salah satu jenis LOD yang menggunakan jarak antara suatu kumpulan polygon dari suatu objek ke kamera sebagai acuan untuk menentukan tingkat detail. View-dependent LOD banyak digunakan pada penggambaran objek yang besar seperti terrain, model biometrik (seperti human map), atau pada model arsitektur (seperti the digital michelangelo project dari stanford university). Metode view-dependent LOD banyak diimplementasikan dengan menggunakan algoritma binary tree vertex sebagai struktur data yang digunakan. Algoritma tersebut cukup kompleks dan rumit untuk dipelajari. Pada tulisan ini akan diimplementasikan metode yang jauh lebih sederhana serta struktur data yang tidak terlalu rumit.

**BAB III**

**METODOLOGI**

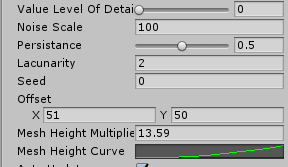
* 1. Rancangan Diagram Sistem



**Gambar 3.1.** Diagram Perancangan Sistem

Plugin ini memiliki kinerja sebagaimana yang terdapat pada diagram perancangan sistem, yaitu developer (pengguna) harus terlebih dahulu memasukkan beberapa nilai/elemen penting untuk membuat dunia 3D. Nilai-nilai tersebut mencakup noise yang diperlukan untuk melakukan pembacaan titik pada dunia 3D yang ingin dibuat, nilai/elemen opsional lainnya seperti peletakkan titik fokus terhadap dunia 3D, dan pengaturan longitude untuk mengatur bentuk/titik-titik dari gunung yang akan dihasilkan nantinya. Dari seluruh nilai yang telah dimasukkan oleh developer (pengguna), nilai-nilai tersebut nantinya akan dijalankan oleh sistem ke media plane pada unity 3D dengan mengeksekusi tombol generate untuk mendapatkan hasil dunia yang diinginkan.

Sedangkan secara garis besar cara kerja sistem ialah pertama – tama sistem akan melakukan pembacaan noise dari plane 3D yang telah disediakan oleh pengguna untuk dapat mempermudah pembuatan daratan dan lautan nantinya



**Gambar 3.2.** *User Interface Plugin 3D World*

Pada gambar 3.2 merupakan nilai – nilai pada plugin ini yang memiliki ketentuan sebagai berikut, persistance yang digunakan untuk mengatur pengurangan amplitudo dari oktaf, lacunarity yang digunakan untuk mengatur penambahan frequency dari oktaf, noise scale yang digunakan untuk memberikan scale pada noise dunia yang ingin dibuat, semakin besar noise scale yang diberikan maka semakin kecil / semakin sederhana pula pemetaan x dan y pada plane yang digunakan, sedangkan untuk seed digunakan untuk melakukan penampilan hasil offset 3D world yang berbeda – beda dengan bantuan system.random pada unity, dan untuk value level of detail digunakan untuk menjadikan dunia 3D yang dibuat menjadi lebih sederhama, semakin besar value of level detail, maka semakin sederhana dunia 3D yang dibuat.

Sedangkan untuk offset sendiri digunakan untuk mengatur bentuk dari 3D world yang ingin difokuskan dengan bantuan noise mapping untuk mendapatkan bentuk oktaf dengan bantuan persistance dan lacunarity, untuk mesh height multiplier digunakan untuk meningkatkan tinggi dari kurva dunia 3D yang ingin dibuat, sedangkan untuk mesh height curve digunakan untuk memberikan bentuk kurva dari dunia 3D dengan cara mendapatkan jumlah dari *vertices per line* (titik per garis) yang nantinya akan dipadukan dengan *curve animation* pada unity dan mendapatkan hasil titik untuk *mesh data* dalam pembuatan mesh pada dunia 3D yang ingin dibuat.

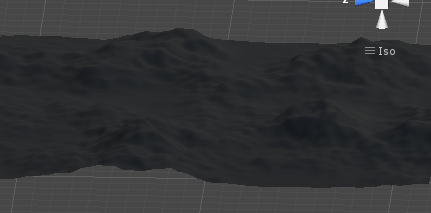
* + 1. Noise Mapping

Pada tahapan noise mapping ini pengguna perlu memasukkan beberapa nilai yang terdapat plugin yang diantaranya ialah noise scale, seed, oktaf, dan offset yang menjadi parameter dasar untuk membuat noise mapping plane yang telah dibuat oleh pengguna. Pada tahapan noise ini program akan melakukan pemetaan noise pada plane yang telah dimiliki pemain dengan cara menggunakan *perlin noise*, yang dimana pertama – tama sistem akan memeriksa terlebih dahulu panjang dan lebar dari plane yang telah diberikan ukuran.

Setelah mendapatkan ukuran dari plane yang dimiliki maka sistem akan melakukan peletakkan acak terhadap nilai seed yang dimasukkan namun dapat dikontrol, setelah seed didapatkan langkah selanjutnya ialah mendapatkan oktaf offset dengan cara mendapatkan offset x dan y dari seed yang dimasukkan sebelumnya dengan menggunakan system random next. Octave offset itu nantinya digunakan untuk mengidentifikasi nantinya bagian yang berwarna hitam, putih, hitam pekat, dan lain sebagainya untuk nantinya dapat dijadikan tanda apakah itu gunung, lautan dan sebagainya.

Setelah didapatkan octave offset maka nantinya pengguna bisa mendapatkan hasil perlin noise terkait dengan warna hitam dan putih tadinya yang digunakan untuk memberikan tanda terkait geografis 3D nantinya untuk dapat dijadikan lautan, daratan, pegunungan, salju, dan lain sebagainya untuk mempermudah saat proses pemberian mesh maupun pemberian warna.

* + 1. Mesh Mapping



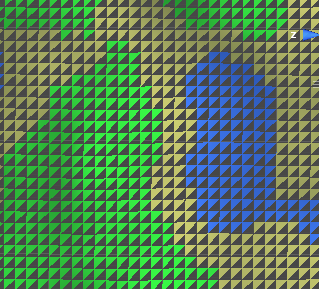
**Gambar 3.3.** *Mesh Mapping 3D World*

Setelah mendapatkan hasil dari noise mapping karena bantuan perlin noise, maka langkah selanjutnya ialah melakukan mesh mapping yang dimana pada tahapan ini kita membutuhkan 3 hal penting yang dimana mendapatkan vertex (sudut), uv, dan triangle (segitiga), yang dimana tiap hal penting tersebut memiliki fungsinya masing – masing namun berkaitan.

Untuk vertices disini cara mendapatkannya ialah dengan menggunakan bantuan animation curve miliki unity dan height curve multiplier yang dimana nantinya akan dikalikan dengan nilai yang memberikan identifikasi titik tertinggi dari plane yang dimiliki, dikarenakan dikalikan dengan height curve maka hasil vertices tersebut dapat di berikan *heightcurve.evaluate* untuk bisa mendapatkan titik tertingginya tersebut yang nantinya dapat dijadikan pedoman yang mempermudah pembuatan titik tertinggi dari plane yang telah dibuat oleh pengguna nantinya. Sedangkan untuk membuat titik terendahnya pun sama, yang dimana hanya perlu menggunakan parameter baru yang memberikan indikasi bahwa ia merupakan titik terendah dari plane yang dimiliki.

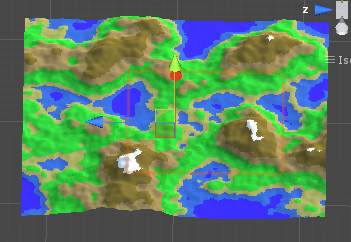
Untuk uv sendiri memiliki fungsi yang dimana nantinya digunakan untuk melakukan pemetaan dari plane itu sendiri untuk nantinya dapat diberikan mesh, yang dimana uvs ialah bentuk awal dari dunia 2D yang nantinya akan dijadikan 3D oleh bantuan dari vertices dan level of detail sendiri. Apabila semakin sederhana nilai dari level of detail yang dimiliki, maka semakin sederhana pula bentuk 2D dari plane yang dimiliki itu sendiri.

Sedangkan untuk triangle yang diberikan pada program mesh ini bertujuan membuat dunia 3D yang memiliki karakteristik low poly, yang dimana terlebih dahulu harus mendapatkan uvs dan vertices dari plane yang dimiliki, setelah itu plane yang dimiliki dapat dipecah menjadi 2 segitiga yang nantinya dapat mengikuti bentuk plane yang dibuat (agar dapat lebih fleksibel dan dapat disederhanakan) dengan memanfaatkan titik titik pada plane.



**Gambar 3.4.** Dunia Terbentuk Dari *Triangle* (Segitiga) *Plane*

* + 1. Color Mapping

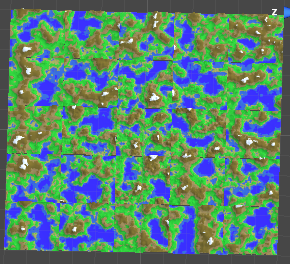


**Gambar 3.5.** *Color Mapping 3D World*

Pada bagian color mapping ini cukup sederhana yang dimana kita hanya perlu mendapatkan hasil mesh dan noise sebelumnya yang nantinya akan dipadukan dengan struct warna yang nantinya warna tersebut akan berada pada tinggi dan lebar dari dunia 3d yang dibuat, yang dimana nantinya untuk pewarnaan menggunakan mesh yang dilambangkan sebagai angka 1 (titik tertinggi) dan 0 (titik terendah) untuk mendapatkan letak pewarnaan.

Sedangkan untuk region sendiri pemberian warna dilakukan dengan memberikan warna, ketinggian, dan nama region yang ingin dimasukkan. Dari peletkkan tersebut pengguna bisa mendapatkan warna atas mesh yang telah dibuat dikarenakan bantuan dari height yang melakukan pemetaan ketinggian dari dunia 3D yang dibuat.

* + 1. Generating Plane



**Gambar 3.6.** *Generate**3D World Using Plane*

Pada generating plane ini, sistem akan menyimpan status dunia yang dibuat, warna dunia yang diberikan, noise yang dimiliki dalam satu paket. Hasil yang didapatkan ini akan disimpan menjadi queue atau antrian yang dimana nantinya akan dilakukan dequeue saat plugin dijalankan dikarenakan pada queue memiliki ketentuan dimana yang masuk terlebih dahulu maka akan keluar terlebih dahulu dikarenakan pada generate ini menggunakan struct, yang dimana akan lebih mudah untuk melakukan penyimpanan dan pemanggilan data dunia 3D yang dibuat.

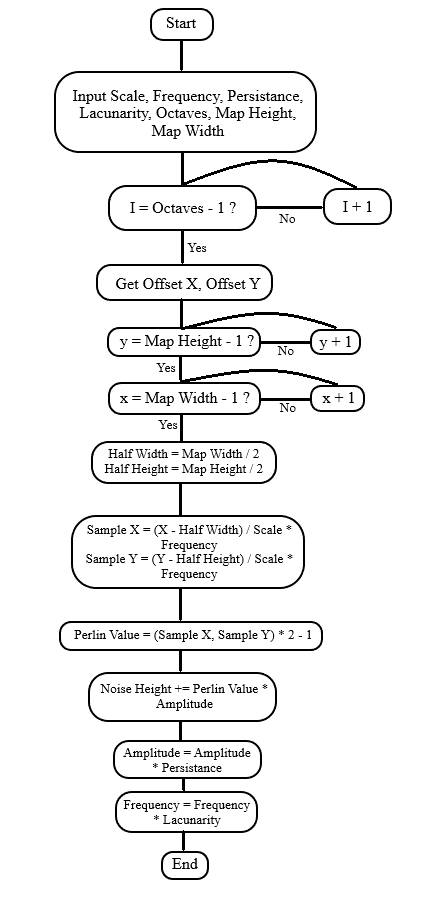
Saat dijalankan data struct dunia 3D yang disimpan tadinya dapat dipanggil menggunakan dequeue dari antrian yang telah dilakukan queue, untuk mencegah terjadinya pengeluaran dan penyimpanan struct disini saya menggunakan thread untuk melakukan penyimpanan dan pengeluaran data dunia 3D agar dapat melakukan pengecekkan apakah struct yang berhasil tersimpan ataupun berhasil dikeluarkan. Apabila gagal tersimpan maupun dikeluarkan maka sistem akan melakukan callback yang dimana dapat digunakan untuk melakukan pemanggilan ulang terhadap status dunia 3D tersebut.

* 1. Algoritma dan Implementasi Sistem
     1. Implementasi Noise Mapping

Sebagaimana yang sudah di uraikan pada bab 2.2.1 mengenai *noise* beserta ilustrasi dari algoritma *noise* pada *map*, yang dimana pada noise memiliki dua nilai penting yaitu lacunarity dan persistence untuk membuat skema *map* 3D, diimplementasikan konsep dari algoritma ini ke dalam bentuk *pseudocode* di bawah :

|  |
| --- |
| Input scale, frequency, persistence, lacunarity, octaves, mapHeight, mapWidth  // get offset x and offset y to get octave offset  for (i = 0; i < octaves; i++){  offsetX = (minValue, maxValue) + offset x;  offsetY = (minValue, maxValue) + offset y;  octaveOffset[i] = (offsetX, offsetY);  }  // check the given mapheight and mapwidth  for (y = 0; y < mapHeight; y++) {  for (x = 0; x < mapWidth; x++) {  // set basic value for noise mapping formula  amplitude = 1;  frequency = 1;  noiseHeight = 0;  // check half width and half height to easily divide  between octaves  halfWidth = mapWidth / 2;  halfHeight = mapHeight / 2;  // check the given octaves  for (i = 0; i < octaves; i++) {  // check sample x and sample y value to get perlin  value  samplex = (x - halfWidth) / scale \* frequency +  octaveOffsets[i]X;  sampley = (y - halfHeight) / scale \* frequency +  octaveOffsets[i]Y;    // help to get maximum and minimum noise height  perlinValue = mapping plane using value (samplex,  sampley) \* 2 - 1;  // help to make various noise height and noise width  across the plane  noiseHeight += perlinValue \* amplitude;  amplitude = amplitude \* persistance;  frequency = frequency \* lacunarity;  }  // to prevent noise height from exceed max noise  height  if (noiseHeight > maxNoiseHeight) {  maxNoiseHeight = noiseHeight;  }  // to prevent noise height from exceed min noise height  else if (noiseHeight < minNoiseHeight) {  minNoiseHeight = noiseHeight;  }  // to read noise map height and width  noiseMap[x, y] = noiseHeight;  }  } |

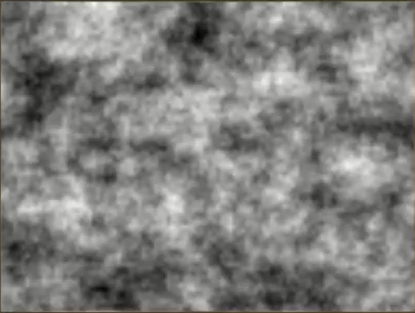
*Pseudocode 3.1 Mesh Mapping Menggunakan Perlin Noise*

**

**Gambar 3.7.** Flow Mesh Mapping

Berdasarkan *pseudocode 3.1* dan *gambar 3.7* di atas, noise memiliki fungsi untuk mengatur bentuk utama 3D dari *map* yang dibuat, yang dimana pada satu pemetaan *map* akan terdapat sejumlah oktaf yang ingin di berikan, lalu tiap oktaf yang ada akan digunakan untuk mendapatkan nilai panjang, lebar, dan tinggi dari pemetaan 3D *map* yang ada.

Untuk mendapatkan nilai panjang, lebar dan tinggi dari pemetaan 3D *map* ini diperlukan ketentuan sebagai berikut : lacunarity digunakan untuk mengkontrol penambahan nilai frekuensi (X) oktaf, sedangkan persistence digunakan untuk mengkontrol pengurangan nilai amplitudo (Y) oktaf, yang dimana oktaf digunakan untuk melakukan pembagian tiap – tiap pemetaan dalam *map*.



**Gambar 3.8.** *Noise In 3D World*

Pada gambar diatas merupakan contoh dari noise plane yang digunakan untuk membentuk dunia 3D yang akan dibuat. Yang dimana noise tersebut digunakan untuk melakukan pembacaan daratan dan lautan yang ada pada dunia 3D yang akan dibuat, dengan cara melakukan peletakkan seed pada plane yang ada dengan menggunakan system.random untuk melakukan peletakkan seed secara acak, walaupun menggunakan system.random bukan berarti ketika dijalankan seed akan diacak ulang dikarenakan pengacakan seed masih terkait dengan bentuk dan ukuran dari plane yang telah diberikan serta nilai – nilai yang terkait didalamnya seperti amplitudo dan frekuensi yang nantinya akan dibantu dengan mathf.perlinnoise unity untuk melakukan pemetaan x dan y pada plane yang ada.

* + 1. Implementasi Curves

Sesuai dengan apa yang telah disampaikan pada bab 2.2.2 mengenai *curves* beserta jenis dan fungsinya masing – masing pada suatu obyek, *curves* bekerja dengan melakukan pembacaan detail dari hasil yang diperoleh dari kerja *noise*, yang dimana ketika telah mendapatkan detail map tersebut *curves* akan mengimplementasikan bentuk *curves* yang diinginkan pada *map* yang telah di *generate*.

|  |
| --- |
| // to help decide what mode to apply in the world  // to make world has a look like noise  if (drawMode == DrawMode.NoiseWorld) {  displayDrawTexture(TextureFromHeightMap(worldHeightWorld));  }  // to apply color function for the world  else if (drawMode == DrawMode.ColourWorld) {  displayDrawTexture(TextureFromColourMap(worldColourWorld, worldPartSize, worldPartSize));  }  // to show the mesh of the world using combination of noise, color, and mesh itself  else if (drawMode == DrawMode.Mesh) {  displayDrawMesh(GenerateTerrainMesh(worldHeightWorld, meshHeightMultiplier, meshHeightCurve, ValueLevelOfDetail), TextureFromColourMap(worldColourWorld, worldPartSize, worldPartSize)); } |

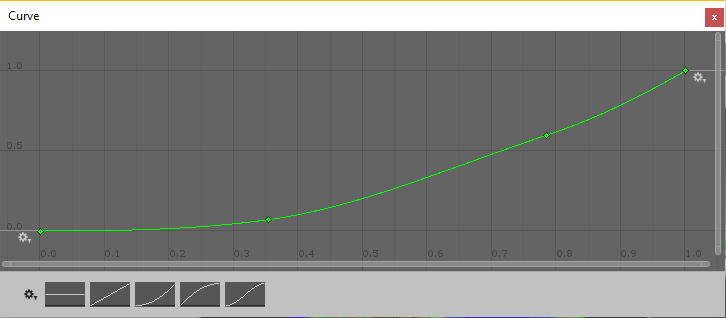
*Pseudocode 3.2 Pemilihan Tipe Mapping*

Berdasarkan *pseudocode 3.2* di atas menunjukkan salah satu fungsi dari curve yang dimana digunakan sebagai pemetaan *map* 3D mode *mesh* yang dimana pemetaan yang dilakukan merupakan pemetaan dari nilai – nilai yang telah didapatkan sebelumnya pada saat melakukan *generate,* yang dimana nilai – nilai yang didapatkan tadi akan digunakan untuk membentuk *map* 3D dan nantinya akan dipetakan atau digambarkan dengan menjalankan program di atas.

|  |
| --- |
| // to determine the mesh data that will store later  MeshData = GenerateTerrainMesh(worldHeightWorld, meshHeightMultiplier, meshHeightCurve, LevelOfDetails);  lock (meshStatsThreadQueue) {  meshStatsThreadQueueEnqueue(new WorldThreadingStats<MeshData>(callback, meshData));  } |

*Pseudocode 3.3 Pembuatan Mesh Data*

Berdasarkan *pseudocode 3.3* di atas menunjukkan salah satu fungsi dari curve yang dimana digunakan sebagai pemetaan *map* 3D yang dimana digunakan untuk mendapatkan nilai – nilai yang nantinya digunakan untuk membantu pembuatan bentuk atau fisik dari *map* 3D yang diinginkan yang disimpan dalam satu paket, yang dimana nilai – nilai tersebut ialah tinggi, penggandaan tinggi, kurva ketinggian, dan nilai level detail.



**Gambar 3.9.** *Animation Curve in Unity*

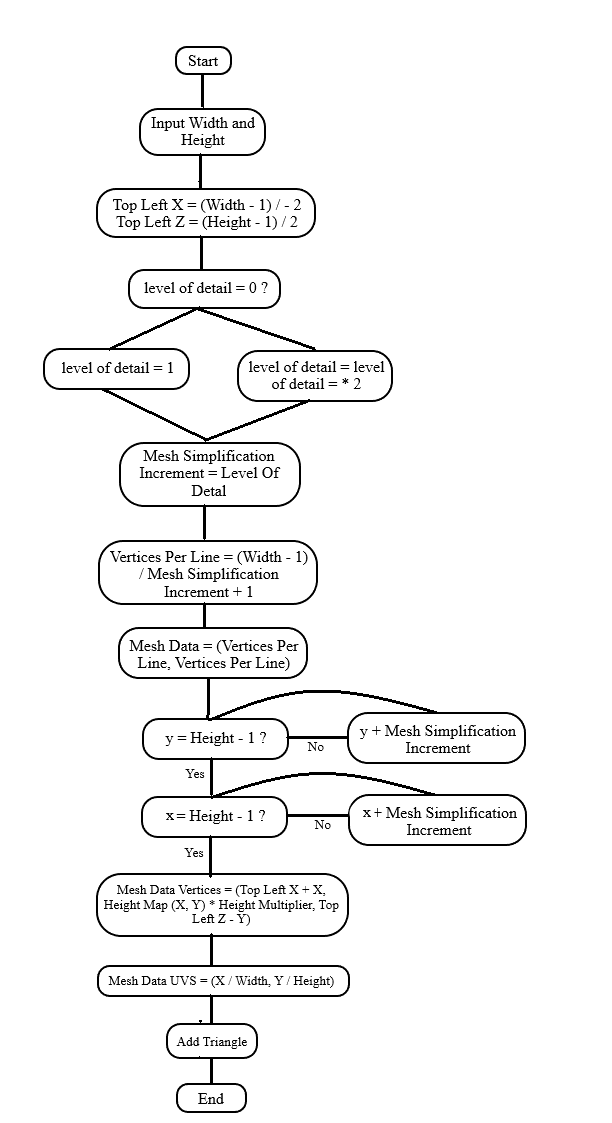
Gambar diatas merupakan *animation curve* yang terdapat pada unity yang digunakan untuk membentuk kurva dari dunia 3D yang dibuat dengan mengimplementasikanya ke dalam titik / vertex pada mesh data untuk nantinya dikalikan dengan nilai pengkali kurva untuk mendapatkan tinggi gunung dan dalamnya air pada dunia 3D yang dibuat, dari *animation curve* yang dibuat ini pengguna dapat mengatur sendiri bentuk kurva yang diinginkan entah itu landai, berliku, dan lain sebagainya.

* + 1. Implementasi Level Of Detail

Sesuai apa yang telah disampaikan pada bab 2.2.3 mengenai LOD (Level Of Detail), salah satu cara untuk mendapatkan hasil dari LOD ialah dengan menggunakan view dependent LOD, yang dimana menggunakan jarak antara suatu kumpulan polygon dari suatu objek ke kamera, yang dimana diimplementasikan pada pseudocode berikut :

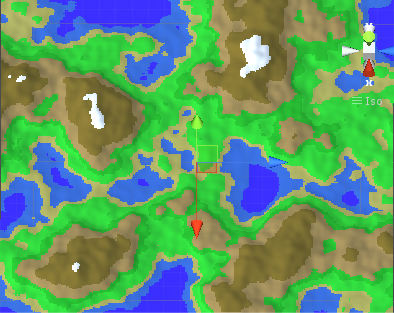
|  |
| --- |
| // get width and height of the noise using plane width and height  width = heightMap.GetLength(0);  height = heightMap.GetLength(1);  // to determine top left x and z value to help read noise that implemented in the plane  topLeftX = (width - 1) / -2;  topLeftZ = (height - 1) / 2;  // to determine level of detail  if (levelOfDetail == 0){  levelOfDetail = 1;  }  else if (levelOfDetail != 0){  levelOfDetail = levelOfDetail \* 2;  }  // to set value of simplification to make a simple world  meshSimplificationIncrement = levelOfDetail;  verticesPerLine = (width - 1) / meshSimplificationIncrement + 1;  // to get the mesh data from using two point vertices  meshData = new MeshData (verticesPerLine, verticesPerLine);  vertexIndex = 0;  // to determine which part that will have simplification on it  for (y = 0; y < height; y+= meshSimplificationIncrement)  {  for (x = 0; x < width; x+= meshSimplificationIncrement)  {  // to help mesh that has been generated to get evaluated by simplification  meshDataVertices[vertexIndex] = Vector3 (topLeftX + x, heightCurveEvaluate (heightMap[x, y]) \* heightMultiplier, topLeftZ - y);  meshDataUvs [vertexIndex] = Vector2 (x / width, y / height);  // to give triangle on a part of mesh for the world that has been generated  if (x < width - 1 && y < height - 1)  {  meshDataAddTriangle(vertexIndex, vertexIndex + verticesPerLine + 1, vertexIndex + verticesPerLine);  meshDataAddTriangle(vertexIndex + verticesPerLine + 1, vertexIndex, vertexIndex + 1);  }  vertexIndex++;  }  }  return meshData; |

*Pseudocode 3.4 Penyederhanan Mesh*

**

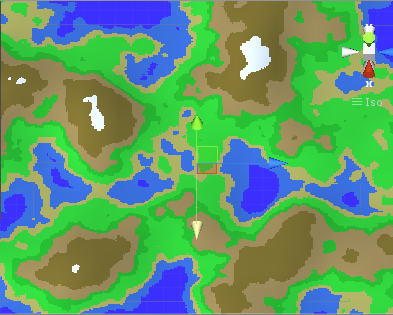
**Gambar 3.10.** Flow Level Of Detail

Berdasarkan *pseudocode 3.4* dan *gambar 3.10* diatas menunjukkan cara dari LOD bekerja, yang dimana pertama kali yang dilakukan ialah mendapatkan tinggi dan lebar dari dunia dengan membaca nilai yang telah dimasukkan sebelumnya, lalu mendapatkan nilai kenaikan dari dunia yang ada dengan membaca masukan level of detail seperti yang telah terterah pada pseudocode, yang dimana apabila nilai level of detail 0 maka level of detail akan bernilai 1, namun apabila nilai level of detail tidak 0 maka nilai level of detail akan dikalikan 2, setelah itu baru kita bisa mendapatkan nilai titik – titik 3D pada garis obyek yang kita miliki dengan mengurangi lebar obyek yang kita miliki sebanyak 1 dan membaginya dengan nilai kenaikan dari dunia yang ada yang telah ditambahkan 1, jika telah mendapatkan nilai titik pergaris maka nilai tersebut dapat dimasukkan sebagai tinggi dan lebar dari dunia yang dibuat.



**Gambar 3.11.** Contoh LOD (Level Of Detail) bernilai 0

Gambar diatas merupakan contoh dari bentuk dunia yang dibuat dengan memberikan nilai level of detail menjadi nol (0) yang dimana hasil akan nampak seperti yang terlihat diatas, yang dimana dunia yang dibuat akan terlihat lebih mendetail dibandingkan apabila nilai level of detail dibesarkan atau lebih dari nol (0) itu dikarenakan semakin besar nilai level of detail yang dimiliki, maka semakin sedikit pula titik yang terdapat ditiap barisnya seperti yang telah ditunjukkan pada pseudocode diatas.



**Gambar 3.12.** Contoh LOD (Level Of Detail) bernilai 1

Sedangkan untuk gambar diatas merupakan contoh dari bentuk dunia yang telah dibuat dengan memasukkan nilai level of detail menjadi nol yang dimana bentuk dari dunia yang dibuat akan menjadi lebih halus (tidak terlihat bergerigi seperti gambar pixel), itu semua dikarenakan apabila semakin tinggi nilai dari level of detail maka itu akan mengurangi vertices per line yang ada pada dunia yang telah dibuat, dikarenakan pada program nilai vertices per line merupakan nilai yang didapat dari lebar dunia yaitu 241 (sebagai default) dikurangi 1 yaitu titik tengah dan dibagi oleh *mesh simplification increment*, dimana nilai tersebut diambil dengan memeriksa nilai level of detail itu sendiri yang dimana apabila nilai level of detail ialah nol (0), maka nilai level of detail akan dengan sendirinya bernilai 1, namun apabila level of detail tidak bernilai nol (0), maka nilai level of detail itu sendiri akan dikalikan 2.

**BAB IV**

**PENGUJIAN**

* 1. Skenario Uji Coba

Skenario uji coba dari program ini dilakukan dengan cara melakukan *generate* map dengan memasukkan nilai – nilai yang diperlukan dengan menggunakan algoritma *noise*, *curves*, dan *level of detail* untuk mendapatkan hasil 3D *map* yang diinginkan.

* + 1. Perangkat yang Digunakan

Dalam proses uji coba ini, akan dilakukan pada beberapa sistem perangkat. Sistem perangkat tersebut memiliki perbedaan spesifikasi komponen hardware di dalamnya, berikut detail dari perangkat yang digunakan :

**Tabel 1** Daftar Perangkat Untuk Alat Uji Coba

|  |  |  |
| --- | --- | --- |
| Spesifikasi Komponen Hardware | Nama Perangkat | |
| PC | Macbook Pro 2018 |
| Processor | Intel Core i5-3470 | Intel Core i5 Quad – Core 2,4 GHz |
| GPU Card | Nvidia 750 TI | Intel Iris Plus Graphics 655 |
| Memory (RAM) | 8 GB | 8 GB |

* + 1. Parameter Pengamatan Uji Coba

Dalam pengujian ini, parameter – parameter yang akan digunakan dalam proses komparasi diantaranya :

**Tabel 2** Daftar Parameter yang Akan Dijadikan Bahan Uji Pembuatan Dunia 3D Pada PC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Parameter | PC | | |
| Nilai Maksimal | CPU Usage | Memory Usage |
| 1 | Value Level Of Detail |  |  |  |
| 2 | Noise Scale |  |  |  |
| 3 | Persistence |  |  |  |
| 4 | Lacunarity |  |  |  |
| 5 | Seed |  |  |  |
| 6 | Mesh Height |  |  |  |

**Tabel 3** Daftar Parameter yang Akan Dijadikan Bahan Uji Pembuatan Dunia 3D Pada Macbook Pro 2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Parameter | Macbook Pro 2018 | | |
| Nilai Maksimal | CPU Usage | Memory Usage |
| 1 | Value Level Of Detail |  |  |  |
| 2 | Noise Scale |  |  |  |
| 3 | Persistence |  |  |  |
| 4 | Lacunarity |  |  |  |
| 5 | Seed |  |  |  |
| 6 | Mesh Height |  |  |  |

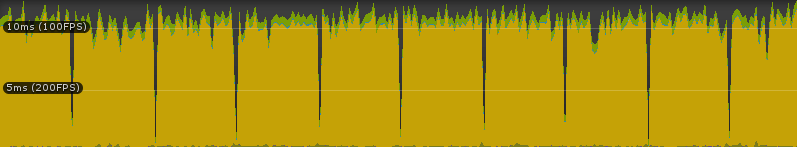
* 1. Hasil Pengujian
     1. Pengujian dengan Procedural Generation

Pada langkah ini dilakukan pengujian terhadap beberapa parameter yang terdapat pada plugin dengan memasukkan nilai – nilai yang diinginkan untuk membentuk dunia 3D yang diinginkan. Pengukuran yang digunakan ialah *CPU Usage* *dan Memory Usage* sebagai berikut :



**Gambar 4.1.** Grafik CPU Usage Diagnostic Session dengan Nilai Parameter Maksimal Pada Asus X550J

Pada saat nilai – nilai parameter yang dibutuhkan seperti value level of detail, noise scale, persistence, lacunarity, seed dan mesh height diberikan nilai tinggi secara berurutan 0, 1, 0, 60, 200, 100, maka cpu usage akan memiliki FPS (Frame Rate / Second) yang rendah yaitu ± 60 FPS bahkan FPS tersebut terkadang dapat lebih rendah lagi yaitu ± 30FPS.



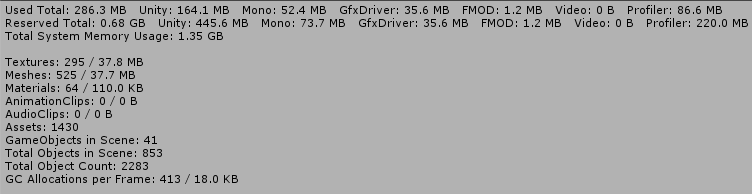
**Gambar 4.2.** Grafik CPU Usage Diagnostic Session dengan Nilai Parameter Minimal Pada Asus X550J

Sedangkan ketika nilai – nilai parameter yang dibutuhkan diubah menjadi nilai yang relatif rendah seperti nilai value level of detail bernilai 6, noise scale bernilai 100, persistence bernilai 0.5, lacunarity bernilai 2, seed bernilai 0 dan mesh height bernilai 30 maka FPS cpu usage akan menjadi ± 100 FPS yang dimana lebih cepat dibandingkan ketika parameter yang dibutuhkan, diatur ke angka yang relatif tinggi, sedangkan untuk parameter seed sendiri tidak mempengaruhi kinerja cpu secara signifikan.



**Gambar 4.3.** Detail Memory Usage Diagnostic Session dengan Nilai Parameter Maksimal Pada Asus X550J

Pada saat nilai – nilai parameter yang diberikan mencapai nilai yang relatif tinggi, maka memory yang digunakan terbilang cukup tinggi. Apabila *memory usage* terlalu tinggi seperti pada hasil *memory usage* diatas, maka hal tersebut dapat menyebabkan penurunan performa kinerja saat dijalankan yang dimana dapat memberatkan perangkat yang memiliki spesifikasi yang randah.



**Gambar 4.4.** Detail Memory Usage Diagnostic Session dengan Nilai Parameter Minimal Pada Asus X550J

Sedangkan untuk penggunaan parameter yang relatif rendah dapat memangkas penggunaan memori seperti yang terlihat pada hasil diatas, yang dimana dapat memperingan kinerja perangkat saat menjalankan plugin dikarenakan terdapat perubahan yang cukup signifikan dibandingkan ketika menjalankan plugin dengan parameter yang relatif tinggi.

**DAFTAR PUSTAKA**

1. Carpenter, Eoghan. 2011. Procedural Generation of Large Scale Gameworlds. <https://www.scss.tcd.ie/publications/theses/diss/2011/>

TCD-SCSS-DISSERTATION-2011-056.pdf. (15 Februari 2018).

1. Jatiningrum, Tri Astuti. 2010. Penilaian Resiko Kebisingan Berdasarkan Analisa Noise Mapping dan Noise Dose Di Unit Produksi HOT STRIP MILL P.T. Krakatau Steel Cilegon – Banten. <https://core.ac.uk/download/pdf/12348987.pdf>. (17 September 2019).
2. Juniasti, Susi., Mochammad Hariadi, Reza Fuad Rachmadi, 2008. View – Dependent Level of Detail (LOD) Untuk Penggambaran Terrain. <https://www.researchgate.net/publication/235950929_View>

-Dependent\_Level\_Of\_Detail\_LOD\_Untuk\_Penggambaran\_Terrain

. (13 September 2019).

1. Lakhwani, Kamlesh. 2016. Curves. <https://www.researchgate.net/>

publication/299562336\_Curves. (27 Agustsus 2019).

1. Parberry, Ian. 2014. Designer Worlds : Procedural Generation of Infinite Terrain from Real – World Elevation Data. https://pdfs.

semanticscholar.org/9d3a/526cb09b9f7cd4cd9ecad835011899a1f907.pdf. (27 Februari 2018).

1. Sepp, Andreas. 2016. Procedural Infinite Terrain Generation with Noise Algorithms. <https://courses.cs.ut.ee/student_projects/downloa>

d/98.pdf. (24 Februari 2018).

1. Togelius, Julian., Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelback, Georgios N. Yannakakis, Corrado Grappiolo. 2013. Controllable Procedural Map via Multiobjective Evolution. <http://julian.togelius.com/Togelius2013Controllable.pdf>. (17 Juli 2019).