BAB II
LANDASAN TEORI

2.1. Algoritma *Perlin-Noise*

Algoritma *Perlin-Noise* merupakan suatu algoritma yang digunakan dalam pembuatan *rendering* secara prosedural atau acak, yang pertama kali diperkenalkan oleh Ken Perlin pada tahun 1985. Algoritma ini umumnya digunakan dalam bidang *computer graphics* dan *computer vision* untuk menghasilkan tekstur atau pola secara acak. Keunggulan dari *Perlin-Noise* adalah efisiensi, fleksibilitas, kesederhanaan, serta kemampuannya untuk menghasilkan hasil yang benar-benar acak. Setiap variabel yang dimasukan kedalam *perlin-noise* dihitung secara acak berdasarkan *seed*. Dimana *seed* adalah satuan acak dari sistem.

Generasi *Perlin-noise* memiliki 4 tahap yaitu :

1. Grid Pseudo-Random
2. Dot Product
3. Interpolasi
4. Combining Octaves

Berdasarkan penelitian sebelumnya yang dilakukan oleh Lagae, et al. [1], *Perlin-Noise* telah terbukti menjadi algoritma yang efektif dalam menghasilkan pola acak dan berpola. Penelitian tersebut memberikan pemahaman yang mendalam tentang penggunaan metode *noise* secara prosedural dalam bidang grafika komputer, mulai dari definisi yang tepat hingga klasifikasi solusi *noise* secara sistematis. Secara matematis, *Perlin-Noise* bekerja dengan cara membangun sebuah grid tiga dimensi dari vektor gradien secara acak atau pseudo-random. Setiap titik grid ini memiliki vektor gradien yang menunjukkan arah "perubahan" nilai *noise*.

Ketika nilai *noise* diperlukan di suatu titik dalam grid, *Perlin-Noise* akan menghitung gradien di sekitar titik tersebut dan menginterpolasikan nilai-nilai *noise* di sekitarnya. Proses ini memungkinkan *Perlin-Noise* untuk menghasilkan *noise* yang berpola dan konsisten. Secara umum, tahapan dasar dalam menghasilkan nilai *Perlin-Noise* adalah sebagai berikut:

1. *Grid Pseudo-Random*

Grid pseudo-random adalah tahap pertama dalam generasi *perlin-noise.* Tujuan utama tahap ini yaitu untuk menetapkan vektor gradien secara *pseudorandom* di setiap *grid points.*

Grid points adalah pernyataan bilangan bulat (*integer)* dalam sebuah koordinat untuk titik sekitar yang berupa *input points* (x, y).

Pertama, menentukan *grid point* sekitarnya dengan pernyataan berikut :

$$(x_0, y_0), (x_1, y_0), (x_0, y_1), (x_1, y_1) \tag{2.1}$$

Dimana :

$x_0 = \lfloor x \rfloor (bilangan\ bulat\ terbesar\ yang\ kurang\ dari\ atau\ sama\ dengan\ x)$

$x_1 = x_0 + 1$

$y_0 = \lfloor y \rfloor (bilangan\ bulat\ terbesar\ yang\ kurang\ dari\ atau\ sama\ dengan\ y)$

$y_1 = y_0 + 1$

Setelah itu menentukan *gradient vector* dari *grid points* tersebut yang dinyatakan sebagai berikut :

$$g_{00}, g_{10}, g_{01}, g_{11} \tag{2.2}$$

Dimana :

$g_{00} = Gradient\ Vector\ pada\ titik\ (x_0, y_0)$

$g_{10} = Gradient\ Vector\ pada\ titik\ (x_1, y_0)$

$g_{01} = Gradient\ Vector\ pada\ titik\ (x_0, y_1)$

$g_{11} = Gradient\ Vector\ pada\ titik\ (x_1, y_1)$

2. *Dot Product*

Untuk setiap titik grid, *Perlin-Noise* menghitung *dot product* antara vektor gradien di titik grid tersebut dengan vektor yang menghubungkan titik grid dengan titik yang ingin diperoleh nilai *noise*-nya. *Dot-product* adalah proyeksi *distance vector* ke *gradient vector* untuk menghasilkan skalar. Perhitungan *dot-product* dihitung dengan *gradient vector* dengan *distance vector* dari *input point* menuju *grid point.*

Distance vector adalah kalkulasi vektor dari *input point* (x, y) untuk setiap titik *grid point*

Pertama lakukan perhitungan *distance vector* dari *input point* untuk setiap *grid point* dengan rumus berikut ini :

$$d_{00} = (x - x_0, y - y_0)$$
$$d_{10} = (x - x_1, y - y_0)$$
$$d_{01} = (x - x_0, y - y_1)$$
$$d_{11} = (x - x_1, y - y_1) \quad (2.3)$$

Dimana :

$d_{00} = adalah\ distance\ vector\ dari\ (x_0, y_0)\ menuju\ (x, y)$
$d_{10} = adalah\ distance\ vector\ dari\ (x_1, y_0) menuju\ (x, y)$
$d_{01} = adalah\ distance\ vector\ dari\ (x_0, y_1) menuju\ (x, y)$
$d_{11} = adalah\ distance\ vector\ dari\ (x_1, y_1)\ menuju\ (x, y)$

Setelah itu melakukan perhitungan *dot-product* dengan rumus berikut ini

$$dot_{00} = g_{00} \cdot d_{00} = g_{00x} \cdot (x - x_0) + g_{00y} \cdot (y - y_0)$$
$$dot_{10} = g_{10} \cdot d_{10} = g_{10x} \cdot (x - x_1) + g_{10y} \cdot (y - y_0)$$
$$dot_{01} = g_{01} \cdot d_{01} = g_{01x} \cdot (x - x_0) + g_{01y} \cdot (y - y_1)$$

$$dot_{11} = g_{11} \cdot d_{11} = g_{11x} \cdot (x - x_1) + g_{11y} \cdot (y - y_1) \quad (2.4)$$

Dimana :

$g_{00x}, g_{00y} = komponen\ gradient\ vector\ g_{00}$

$g_{10x}, g_{10y} = komponen\ gradient\ vector\ g_{10}$

$g_{01x}, g_{01y} = komponen\ gradient\ vector\ g_{01}$

$g_{11x}, g_{11y} = komponen\ gradient\ vector\ g_{11}$

3. Interpolasi

Setelah *dot product* dihitung, nilai *noise* di setiap titik grid diinterpolasi berdasarkan nilai dot product di sekitarnya. Interpolasi adalah tahap untuk memperhalus antar *noise value* yang ada. Pada tahap ini penelitian akan menggunakan metode *linear interpolation* untuk interpolasi

Linear interpolation adalah interpolasi dengan estimasi *value* yang baru dengan mengambil rata-rata tertimbang dari dua nilai yang sudah diketahui sebelumnya berdasarkan titik-titik *grid point* dan *input point*

Pertama, lakukan kalkulasi untuk *interpolation weight* :

$$u = x - x_0\ , \qquad v\ =\ y - y_0 \quad (2.5)$$

Dimana :

$u\ dan\ v\ adalah\ bagian\ fraksional\ dari\ x\ dan\ y$

Lalu lakukan interpolasi pada sumbu-x dengan rumus berikut :

$$interp_{x0}\ = lerp(u, dot_{00}, dot_{10})$$
$$interp_{x1}\ = lerp(u, dot_{01}, dot_{11}) \quad (2.6)$$

Dimana

$lerp(t, a, b) =\ fungsi\ dari\ linear\ interpolasi\ yang\ berisi\ (1 - t) \cdot a + t \cdot b$

$dot_{00}, dot_{10}, dot_{01}, dot_{11} = nilai\ dot\ product$

Setelah interpolasi sumbu x, lakukan interpolasi pada sumbu-y dengan rumus berikut :

$$final\_interp = lerp(v, interp_{x0}, interp_{x1}) \tag{2.7}$$

Dimana

$lerp(t, a, b) =\ fungsi\ dari\ linear\ interpolasi\ yang\ berisi\ (1-t)\cdot a + t$

$binterp_{x0}$ & $interp_{x1}\ =\ value\ interpolasi\ dari\ sumbu\ x$

4. *Combining Octaves*

Dalam prakteknya, *Perlin-Noise* dikombinasikan dari beberapa *octaves* atau frekuensi yang berbeda. Hal ini dilakukan dengan menggabungkan nilai-nilai *noise* dari beberapa *octaves* dengan bobot tertentu untuk menciptakan hasil yang lebih kompleks dan detail.

Octaves adalah sekumpulan *octave* atau layer dari *perlin-noise* dengan frekuensi dan amplitudo yang berbeda

Frekuensi dan amplitudo dalam *perlin noise* adalah variabel yang mengontrol kontribusi dari setiap *octave*

Rumus umum untuk menggabungkan nilai-nilai noise dari beberapa oktaf adalah :

$N(x, y)\ = \sum_{i=0}^{n-1} Perlin(x \cdot f_i y \cdot f_i)\ \cdot A_i$ (2.8)

Dimana

$N(x, y) =\ hasil\ akhir\ dari\ value\ pada\ noise$

$n = jumlah\ bilangan\ pada\ octave$

$Perlin(x \cdot f_i y \cdot f_i) =\ fungsi\ perlin\ noise\ yang\ telah\ dievaluasi$

$A_i = amplitudo\ untuk\ ke-i\ dari\ octave$

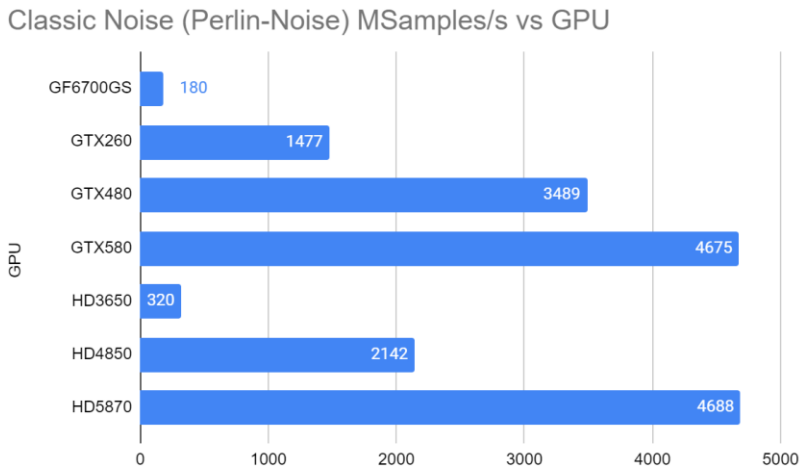
Gambar 2.1
Texture *perlin-noise*

Gambar 2.1 merupakan hasil dari generasi *perlin-noise.* Studi yang dilakukan oleh Eastman, et al. [2] lebih lanjut mengungkapkan implementasi konkret dari *Perlin-Noise* dalam berbagai konteks, terutama dalam generasi terrain. *Perlin-Noise* digunakan sebagai alat untuk menghasilkan peta kedalaman (depth-map) atau elevasi terrain. Titik tertinggi diberi nilai putih (1), sementara titik terendah diberi nilai hitam (0), menciptakan terrain secara prosedural dengan tingkat detail yang tinggi.

Penelitian yang dilakukan oleh McEwan, et al. [3] menunjukkan bahwa penggunaan prosedural untuk *noise* memberikan kinerja yang lebih baik daripada pendekatan sepenuhnya acak. Dengan menggunakan *noise* yang di *generate* secara prosedural untuk *pre-calculated* dalam merender tekstur, penelitian ini menggunakan GLSL 1.20 untuk menciptakan tekstur, Hasil penelitian menunjukkan bahwa beberapa perangkat keras mampu menjalankan sampel dengan efisiensi yang tinggi memiliki rata-rata diatas 100 Msamples/detik pada generasi 2D, *noise* tersebut menunjukkan performa yang tinggi. Berikut ini disajikan tabel performa hasil pengujian yang dilakukan pada penelitian tersebut.

Tabel 2.1 Performa *Perlin* dalam Msamples/detik

Vendor	GPU	Const Color	2D Classic Noise (Perlin-Noise) Msamples/s
Nvidia	GF6700GS	3,399	180
	GTX260	8,438	1,477
	GTX480	8,841	3,489
	GTX580	13,863	4,675
AMD	HD3650	1,974	320
	HD4850	9,416	2,142
	HD5870	18,061	4,688



Gambar 2.2
Performa Perlin dalam Msamples/detik

Dalam studi yang dilakukan oleh Korn et al. [7], diperinci bahwa *video game* yang mengandalkan generasi konten secara prosedural memerlukan pola yang konsisten pada setiap iterasi generasi. Temuan yang diperoleh dalam penelitian ini menunjukkan bahwa algoritma *Perlin-noise* memenuhi kriteria ini. Algoritma *Perlin-noise* terbukti menampilkan pola yang konsisten meskipun prosesnya menghasilkan nilai-nilai secara acak, sehingga menjadi opsi yang tepat untuk diterapkan dalam konteks generasi konten dalam *video game*.

Kesimpulan ini didukung oleh observasi pengalaman pengguna selama penelitian, yang menunjukkan bahwa sekitar 60% dari responden merespons positif terhadap pola yang konsisten dan terstruktur dalam pengalaman bermain video game yang menggunakan PCG, dibandingkan dengan sekitar 40% yang lebih memilih generasi konten secara manual.

Pembuatan algoritma *Perlin-Noise* bertujuan utama untuk mempercepat proses generasi hasil secara acak dan memberikan pola tertentu. Proses generasi yang sepenuhnya acak membutuhkan biaya komputasi yang tinggi, namun dengan menggunakan algoritma *Perlin-Noise* sebagai dasar untuk generasi secara acak, proses tersebut dapat dilakukan dengan cepat dan efisien, proses ini sering disebut juga dengan *Pre-calculated generation* atau generasi yang telah dikalkulasi terlebih dahulu. *Perlin-Noise* sering digunakan dalam simulasi berbagai jenis rendering seperti awan, *terrain*, air, tekstur, dan hal-hal lain yang memerlukan unsur keacakan.

Dalam *Unity Engine*, Implementasi *perlin-noise* dapat memanfaatkan pemanggilan *Mathf.PerlinNoise()* pada suatu kelas. *Mathf.PerlinNoise()* sendiri sudah terdiri dari kalkulasi dasar seperti *Grid Pseudo-Random, dot product,* interpolasi, *Combining Octaves,* dan dapat menentukan scaling, seed, resolusi, dan level.

2.2. *Procedural Content Generation (PCG)*

Procedural Content Generation (PCG) merujuk pada penciptaan konten *video game* secara algoritmik dengan keterlibatan manusia yang terbatas atau tidak ada sama sekali. Ini dapat mencakup generasi secara acak untuk berbagai aspek yang ada pada dunia *video game* seperti *terrain*, *object*, karakter, misi, item, dan lainnya menggunakan algoritma daripada merancang setiap elemen secara manual [5][6]. Sebagai contoh, game yang

mengimplementasikan PCG yaitu “Minecraft” seperti yang diperlihatkan pada Gambar 2.3

Gambar 2.3
Implementasi PCG pada minecraft dalam generasi terrain

2.2.1. Procedural Object Generation (POG)

Procedural Object Generation (POG) secara khusus berfokus pada generasi objek individual dalam dunia *video game*, seperti model, makhluk, senjata, atau item, menggunakan teknik procedural seperti yang dinyatakan J. Togelius, et al [5][6]. Salah satu contoh game yang mengimplementasikan ini yaitu “Left 4 Dead” seperti yang diperlihatkan pada gambar 2.4

Gambar 2.4
Implementasi POG pada Left 4 Dead 2 dalam generasi props/object

Menurut penelitian yang dilakukan oleh Korn et al. [7] Dengan *procedural generation* dalam pengembangan konten *video game*, pengembang dapat mencapai tingkat variasi yang lebih tinggi, mengurangi waktu dan biaya pengembangan, serta meningkatkan pengalaman bermain yang adaptif.

2.3. Layer-Based

Dalam penelitian yang dilakukan oleh J. Togelius, et al. [5][6], dalam konteks *Procedural Content Generation* (PCG), salah satu masalah yang dihadapi adalah generasi konten yang mendukung multi-level/multi-content. Aspek ini mencakup generasi berbagai *obstacle* atau objek yang berbeda dan generasi tempat-tempat yang spesifik, yang bertujuan untuk menambah nilai tambah dalam keunikan dan variasi pola generasi. Namun, riset ini menyoroti kendala-kendala yang terdapat pada aspek tersebut, khususnya dalam kaitannya dengan keterbatasan yang ada pada *Game Engine* yang digunakan pada masanya.

Saat ini, *Unity Engine* telah dilengkapi dengan fitur *Layer-Based*. *Layer-Based* adalah sebuah tipe data yang digunakan untuk menentukan *layer* mana yang akan dipertimbangkan atau diabaikan dalam proses generasi

konten. Dengan menggunakan *Layer-Based,* maka dapat mengontrol interaksi antara objek-objek dalam scene, memungkinkan pengaturan yang lebih spesifik dalam proses generasi. Hal ini bisa dipanggil dalam unity menggunakan *LayerMask().*

Meskipun *Unity Engine* menawarkan generasi *multi-level* dengan menggunakan *Layer-Based*, namun terdapat beberapa keterbatasan yang perlu diperhatikan. Salah satunya adalah batasan jumlah *layer* yang dapat ditangani, yang terbatas pada 32 *layer*. Selain itu, setiap *layer* dalam *Layer-Based* dapat didefinisikan sebagai "True" atau "False" sesuai dengan kebutuhan yang diperlukan, cara kerjanya sama seperti konsep himpunan pada matematika.

Dalam *Unity Engine*, *game object* dapat ditempatkan pada salah satu dari 32 *layer* yang berbeda. *Layer-Based* memberikan cara untuk mengelompokkan beberapa *layer* ini menjadi satu unit data yang mudah digunakan. Setiap bit dalam *Layer-Based* mewakili satu *layer*, dan dengan mengatur bit-bit ini, dapat menentukan *layer* mana yang ingin pertimbangkan. Misalnya, jika ingin melakukan deteksi tabrakan *collision* hanya dengan objek-objek yang berada di *layer* tertentu (misalnya, "*Wall*" atau "*Player*"), maka dapat digunakan tipe data *Layer-Based* untuk menentukan sekumpulan *layer* tersebut [14].

2.4. Algoritma *Euclidean Distance*

J.Tabak [13] dalam bukunya menjelaskan teori mengenai *Euclidean Distance*. *Euclidean Distance* sendiri adalah formula atau rumus matematis yang digunakan untuk mengukur jarak antara dua titik dalam ruang Euclidean. Ruang Euclidean adalah ruang geometris yang mematuhi prinsip-prinsip geometri Euclidean, yang didefinisikan oleh matematikawan Yunani kuno, Euclid.

Sebagai definisi, Diberikan dua titik P dan Q dalam ruang Euclidean dengan koordinat (x1, y1) dan (x2, y2) dalam dimensi dua, atau (x1, y1, z1) dan (x2, y2, z2) dalam dimensi ketiga Euclidean distance antara titik P dan Q dihitung sebagai akar kuadrat dari jumlah kuadrat perbedaan koordinat antara titik-titik tersebut.

Berikut ini rumus *euclidean-distance* dalam berbagai dimensi :

$$1D\ Distance = \sqrt{(x_2 - x_1)^2}$$
$$2D\ Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$
$$3D\ Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (2.9)$$

Dimana :

$Distance = Jarak\ akhir\ dari\ euclidean x_n, y_n, z_n$

$x, y, z = Koordinat\ dari\ sebuah\ titik$

Secara intuitif, Euclidean Distance mewakili panjang garis lurus yang menghubungkan dua titik dalam ruang tiga dimensi. Metrik ini merupakan ukuran paling sederhana dan paling umum untuk jarak antara dua titik dalam geometri Euclidean. Dalam konteks PCG, Euclidean Distance berguna untuk memeriksa dan memberikan jarak antara dua titik melebihi batas tertentu, yang menandakan bahwa posisi yang ingin ditempatkan terlalu dekat dengan posisi-posisi obstacle atau objek yang sudah ada.

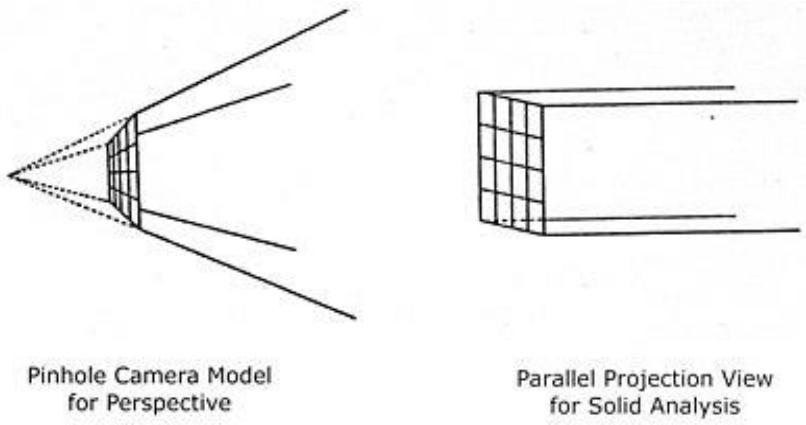
Euclidean Distance digunakan dalam ruang tiga dimensi, meskipun dasar generasi menggunakan Perlin-Noise dalam ruang dua dimensi. Obstacle atau objek yang dihasilkan akan memiliki ruang dimensi tiga, sehingga Euclidean Distance berbasis tiga dimensi digunakan untuk memastikan bahwa tidak ada *overlapping* antara objek yang dihasilkan, dalam unity engine, formula ini dapat dipanggil dengan menggunakan vector3.distance

Euclidean distance adalah salah satu metrik jarak yang paling umum digunakan dalam berbagai bidang, termasuk matematika, ilmu komputer, dan statistik.

2.5. *Raycasting*

Raycasting adalah teknik yang digunakan untuk menentukan apakah suatu garis lurus (*ray*) bersinggungan dengan objek tertentu dalam sebuah *scene*, serta menentukan titik di mana garis tersebut bersinggungan dengan objek tersebut [15].

Dalam konteks *Unity*, *raycasting* biasanya dilakukan dengan menggunakan fungsi-fungsi seperti Raycast(), Physics.Raycast() atau Physics.RaycastAll(), yang memungkinkan untuk mengirimkan ray ke dalam scene dan mendapatkan informasi tentang objek apa yang tersentuh oleh ray tersebut, serta di mana titik sentuhannya, Gambar 2.5 memperlihatkan cara kerja dari raycasting dengan metode perspektif dan *orthographic/projection*



Gambar 2.5
Cara kerja Raycasting

2.6. C# (Bahasa Pemrograman)

C# (*C Sharp*) adalah bahasa pemrograman yang diterbitkan oleh Microsoft dan digunakan untuk membangun aplikasi berbasis .NET Framework. Bahasa ini dirancang untuk mempermudah pengembangan aplikasi yang

berbeda, termasuk video game, aplikasi desktop, aplikasi web, dan aplikasi lainnya. C# dapat digunakan untuk membuat aplikasi yang berjalan di berbagai platform, termasuk macOS, Linux, dan perangkat lunak mobile [16].

2.6.1. *Pseudocode C-Style*

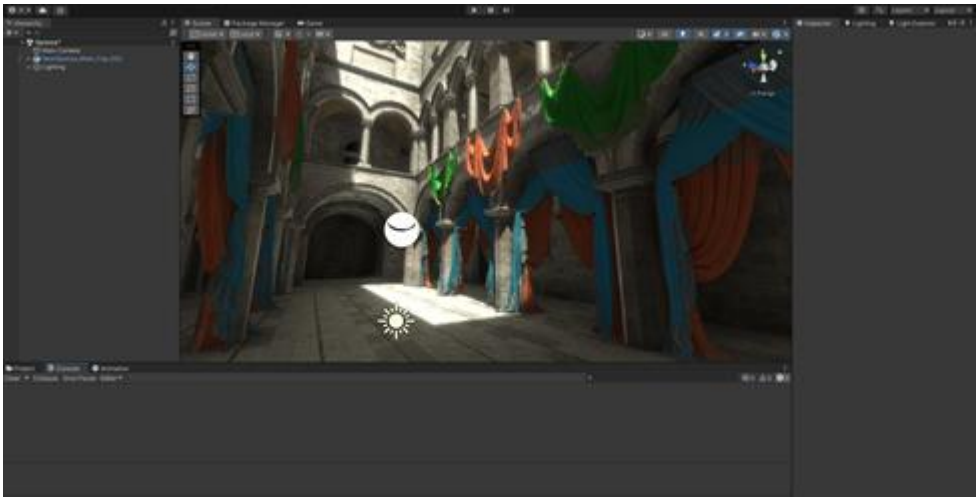
Pseudocode (C-Style) [16] adalah representasi informal dari algoritma atau program komputer yang menggunakan sintaks mirip dengan bahasa pemrograman C. Pemrograman C# dan C++ dapat digunakan karena memiliki prinsip yang sama, Pseudocode Ini memberikan panduan tentang bagaimana sebuah program harus beroperasi tanpa memperhatikan detail sintaks dari bahasa pemrograman tertentu.

2.7. *Video Game Engine*

Video Game Engine adalah perangkat lunak yang digunakan untuk membangun dan mengembangkan *video game*. *Engine* biasanya menangani rendering dan teknologi penting lainnya, tetapi juga memungkinkan dalam menangani fungsi tambahan seperti kecerdasan buatan, simulasi fisika, dan hal lainnya.

2.7.1. *Unity Engine*

Unity Engine adalah sebuah *game engine* yang digunakan untuk membangun dan mengembangkan proyek 3D *real-time*, termasuk *video game*, animasi, film, arsitektur, simulasi fisika, dan perancangan. Unity Engine memiliki kerangka kerja yang lengkap untuk pengembangan game dan menggunakan bahasa pemrograman seperti C# dan JavaScript. Interface Unity Engine diperlihatkan pada Gambar 2.6 [18]



Gambar 2.6
Interface Unity Engine

2.7.2. *System.Diagnostics (Unity)*

System.Diagnostics adalah namespace dalam C# yang menyediakan berbagai kelas dan metode untuk membantu debugging, pelacakan, dan pembacaan *code* secara berulang [19], *System.Diagnostics* memberikan beberapa manfaat seperti :

1. Melacak kinerja aplikasi dengan menggunakan kelas *Stopwatch()* untuk mengukur waktu yang dibutuhkan untuk menjalankan kode tertentu.
2. Menganalisis penggunaan memori dengan menggunakan kelas Process untuk mendapatkan informasi tentang memori yang digunakan oleh proses *Unity*.
3. Mencatat informasi dengan menggunakan kelas *Trace* dan *Debug* untuk mencatat informasi ke konsol atau *file log*.
4. Menguji aplikasi dengan menggunakan kelas *Assert* untuk memverifikasi bahwa kondisi tertentu terpenuhi.
5. Memberikan Log penggunaan daya pada CPU dan RAM yang dipakai dengan Profiler

2.8. Pengujian Akurasi

Pengujian akurasi bertujuan untuk mengevaluasi seberapa tepat algoritma dalam menghasilkan objek sesuai dengan target tanpa adanya kesalahan seperti *overlapping*. Akurasi mengukur seberapa dekat hasil yang dihasilkan dengan kondisi yang diinginkan, dinyatakan sebagai persentase dari objek yang ditempatkan dengan benar dibandingkan dengan jumlah total objek yang direncanakan. Secara umum, rumus akurasi dapat dinyatakan sebagai berikut ini :

$$CR = \frac{C}{A} \cdot 100\% \tag{2.10}$$

Dimana :

$CR = Rate\ Akurasi, mengindikasikan\ persentase\ akurasi.$

$C = adalah\ jumlah\ sampel\ yang\ benar.$

$A = adalah\ jumlah\ total\ sampel, total\ objek\ yang\ direncanakan.$

Dalam konteks penelitian ini, rumus akurasi diadaptasi menjadi:

$$Akurasi = \frac{Jumlah\ objek\ yang\ telah\ di\ generasi}{Total\ objek\ tujuan} \cdot 100\% \tag{2.11}$$

2.9. Rendering (Computer Graphics)

Dalam konteks *computer graphics*, *rendering* merujuk pada proses menghasilkan untuk sesuatu sebagai hasil akhir dari sebuah *scene*, kalkulasi rumus, atau model 3D. Rendering melibatkan simulasi interaksi cahaya, kalkulasi fisika, dan hal lainnya dengan objek dalam adegan untuk menghasilkan hasil akhir yang akan ditampilkan [20].

2.10. Studi Literatur

Tabel 2.2 Literature Review [1]

Judul Journal/Thesis/Buku/Dll	*State of the Art in Procedural Noise Functions*
Penulis	Ares Lagae, S. Lefebvre, R. L. Cook, T. DeRose, G. Drettakis, D. Ebert, J. P. Lewis, K. Perlin, & Matthias Zwicker
Tahun Publikasi	2010
Latar Belakang Masalah	Latar belakang permasalahan yang dibahas pada penelitian ini adalah tantangan dalam menghasilkan detail visual dalam grafika komputer secara efisien. Fungsi *noise* pada generasi prosedural, dengan *Perlin noise* sebagai contoh utama, telah menjadi aspek utama dalam mencapai hal ini. Fungsi-fungsi ini digunakan secara luas untuk mensimulasikan elemen alam seperti *terrain*, awan dan gelombang, dan hal tersebut juga dapat digunakan untuk meningkatkan efek visual dalam film dan video game. Seiring dengan kemajuan teknologi, terutama dengan peningkatan daya komputasi perangkat keras (*hardware)*, akan dibutuhkan penyempurnaan dan perluasan teknik *noise* secara prosedural untuk memenuhi tuntutan yang semakin meningkat akan tekstur dan pola rumit dalam grafika komputer. penelitian ini memberikan gambaran menyeluruh tentang fungsi noise pada prosedural, baik yang bersifat lampau maupun terbaru, untuk mengatasi tantangan berkelanjutan dalam menghasilkan detail visual yang rumit dengan efisien di bidang ini.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma Prosedural Generasi dengan *Noise* seperti *perlin noise, wavelet noise, anisotropic noise, sparse convolution noise, Gabor noise, spot noise, surface noise, solid noise*

Metode Penelitian	Metode Penelitian : Metode Kuantitatif penelitian terutama menggunakan pendekatan penelitian secara kuantitatif untuk mendefinisikan dan menganalisis fungsi *noise* secara prosedural dalam grafik komputer. Ini dimulai dengan memberikan definisi *noise* secara intuitif dan formal, menekankan representasinya melalui spektrum daya dan fungsi autokorelasi. penelitian ini berfokus pada karakteristik statistik, seperti momen pada *noise* dan Gaussianitas, untuk mendeskripsikan fungsi *noise* secara komprehensif. Pendekatan kuantitatif ini memungkinkan evaluasi dan perbandingan konstruksi jenis-jenis *noise* yang berbeda berdasarkan sifat statistiknya. Secara keseluruhan, metode penelitian berpusat pada analisis kuantitatif dan pemodelan matematika untuk mengkarakterisasi dan memahami gangguan prosedural secara terstruktur.
Hasil dan Kesimpulan Penelitian	penelitian ini menyajikan gambaran komprehensif tentang metode *noise* secara prosedural dalam grafik komputer, yang mencakup kemajuan terkini dari delapan tahun terakhir (terhitung saat penelitian ini dibuat). Hal ini dimulai dengan menyatakan definisi *noise* yang tepat, yang berakar pada proses stokastik, yang meletakkan dasar bagi klasifikasi sistematis solusi *noise* secara prosedural. Tiga kategori utama, yaitu *lattice gradient*, *explicit*, and *sparse convolution noises*, hal tersebut telah dirinci, menekankan pentingnya kontrol spektral dalam pola pemodelan. Penerapan *noise* pada permukaan dieksplorasi, mengatasi masalah terkait pemfilteran dan

	tekstur prosedural. Analisis ini menggunakan spektrum daya dan distribusi amplitudo untuk menjelaskan perbedaan antara jenis *noise* dan efek visual yang dihasilkannya. Penilaian komparatif mempertimbangkan faktor-faktor seperti persyaratan penyimpanan, kontrol spektrum daya, anisotropi, aplikasi permukaan, metode penyaringan, dan kecepatan komputasi. Kesimpulannya menyoroti bahwa setiap solusi *noise* menawarkan *trade-off* yang berbeda, sehingga memerlukan pemilihan yang disesuaikan berdasarkan kebutuhan aplikasi tertentu.
Saran Penelitian Lanjutan	penelitian ini menguraikan beberapa arah yang menjanjikan untuk penelitian masa depan di bidang metode *noise* secara prosedural. Hal ini menunjukkan perlunya kontrol menyeluruh terhadap spektrum daya *noise*, yang dapat menjembatani kesenjangan antara *noise* dan tekstur stokastik, khususnya yang berguna untuk metode *noise-by-example*. Menjelajahi proses stokastik *non-Gaussian* dan *non-stasioner* diusulkan untuk memperluas kemampuan pengembangan *noise*, menawarkan pola yang lebih kaya dan beragam. penelitian ini juga menggarisbawahi potensi pemahaman dan pengendalian fase *noise* sebagai sarana untuk membedakan stokastik dari tekstur terstruktur.

Tabel 2.3 Literature Review [2]

Judul Journal/Thesis/Buku/Dll	*Procedural Generation: 2D perlin noise*
Penulis	Dave Mount & Roger Eastman
Tahun Publikasi	2018

Latar Belakang Masalah	Masalah utama yang dibahas dalam penelitian ini adalah Generasi pada *perlin noise* dibidang 2D, yang merupakan fungsi acak terstruktur yang digunakan dalam grafik komputer dan pembuatan konten prosedural. *perlin noise* dalam satu dimensi (1D) dikenal karena kemampuannya menghasilkan pola *Noise* yang menarik secara alami. Namun, memperluas konsep ini ke dua dimensi (2D) menghadirkan sebuah tantangan, karena memerlukan penciptaan metode yang menghasilkan noise 2D yang kompleks, tampak alami, dan *grid-free*. Solusinya melibatkan pengenalan gradien acak di setiap titik *vertex*, memadukan gradien ini dengan mulus, menerapkan fungsi *fading* untuk mengontrol pengaruhnya, dan menggabungkan beberapa oktaf fungsi *noise* dengan penskalaan. *perlin noise* 2D yang dihasilkan mampu menghasilkan pola medan (*terrain* dan tekstur yang rumit dan realistis untuk berbagai aplikasi dalam grafik komputer dan pembuatan prosedural.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *perlin noise* untuk bidan 2D (atau disebut juga tekstur)
Metode Penelitian	Metode Penelitian : Metode Kuantitatif Metode penelitian dalam penelitian ini sebagian besar bersifat kuantitatif, dengan fokus utama pada teknik matematika dan algoritma yang digunakan untuk membuat *perlin noise* secara 2D. Hal ini melibatkan operasi matematika terperinci, persamaan, dan parameter numerik untuk secara tepat mengontrol pembentukan pola *noise* yang kompleks. *perlin noise* pada dasarnya adalah

	model matematika, dan penelitian tersebut secara kuantitatif mendefinisikan komponen-komponennya, termasuk nilai acak, vektor gradien, interpolasi, fungsi fading, dan faktor penskalaan, semuanya dinyatakan melalui rumus dan perhitungan matematika yang tepat.
Hasil dan Kesimpulan Penelitian	Hasil penelitian penelitian ini menunjukkan algoritma prosedural *perlin noise* 2D, yang secara efektif memperluas konsep ke dua dimensi. Metode ini menggabungkan gradien acak, interpolasi, fungsi fading, dan penskalaan untuk menciptakan pola *noise* yang kompleks untuk pembuatan medan (*terrain)* dan tekstur. Meskipun *noise* yang ditimbulkan cukup rumit, namun *noise* tersebut mungkin tidak sepenuhnya meniru kondisi *terrain* secara alami, karena faktor-faktor lain seperti erosi dan kekuatan geologi tidak dipertimbangkan. Kesimpulannya, penelitian ini menawarkan landasan algoritmik kuantitatif untuk *perlin noise* dibidang 2D, yang berguna dalam grafik komputer dan pembuatan konten prosedural.
Saran Penelitian Lanjutan	penelitian ini menyarankan upaya di masa depan dalam meningkatkan realisme medan (*terrain)* dan tekstur yang dihasilkan dengan memasukkan faktor tambahan seperti erosi dan proses geologi. Hal ini juga mendorong eksplorasi berbagai fungsi fade dan teknik penskalaan untuk variasi dan aplikasi *noise* lebih lanjut. Selain itu, penelitian ini menyoroti kesederhanaan implementasi *perlin noise*, menyarankan potensi optimasi untuk efisiensi dalam aplikasi real-time. Penelitian di lanjutan juga dapat terfokus pada peningkatan kinerja algoritma

	dan kemampuan beradaptasi terhadap beragam konteks grafis dan game.

Tabel 2.4 Literature Review [3]

Judul Journal/Thesis/Buku/Dll	*Efficient computational noise in GLSL*
Penulis	Ian McEwan, David Sheets, Stefan Gustavson, & Mark Richardson
Tahun Publikasi	2012
Latar Belakang Masalah	Penelitian ini membahas implementasi *Perlin noise* dan *Perlin simplex noise* pada GLSL untuk rendering grafis secara langsung (*real-time)* pada perangkat keras GPU Generasi *perlin noise* merupakan hal yang penting untuk sistem prosedural dalam grafik komputer, memungkinkan terciptanya material dan lingkungan alami. Mempelajari dan menganalisis performa GPU dan kendala-kendala yang dihadapi secara langsung (*real-time*), serta mengadaptasi metode prosedural perlin noise yang belum pernah diadopsi sebelumnya. Pengujian performa diuji dari berbagai aspek noise, mulai dari noise pada bidang 2D, 3D, dan 4D di berbagai GPU baik dari Nvidia maupun Radeon Menunjukkan keterbatasan penelitian sebelumnya dan membandingkan penelitian yang dilakukan penulis dengan menyajikan implementasi GLSL yang efisien pada fungsi *noise* yang murni dikomputasi dan tidak bergantung atau membahas pada tekstur atau tabel pencarian.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *perlin noise* dan *Perlin Simplex Noise* dengan GLSL (Perbandingan)

	Perlin noise algorithm and Perlin simplex noise algorithm in GLSL (Comparison)
Metode Penelitian	Metode Penelitian : Metode Kuantitatif Metode penelitian ini melibatkan perancangan algoritma komputasi yang bersifat kuantitatif untuk menghasilkan *noise* pada GLSL tanpa bergantung atau membahas tekstur atau tabel pencarian. Penulis menjelaskan penggunaan polinomial permutasi untuk membuat indeks gradien pseudo-acak, yang penting untuk menghasilkan *noise*. Mereka juga membahas konsep gradien pada *N-cross-polytopes*, yaitu metode untuk memetakan titik-titik yang terdistribusi secara merata dari kubus berdimensi N ke batas oktahedron ekuivalen berdimensi N. Hubungan geometris ini digunakan untuk menghitung nilai *noise* secara efisien.
Hasil dan Kesimpulan Penelitian	penelitian tersebut melaporkan bahwa algoritma yang disajikan efisien dan praktis untuk perangkat keras GPU saat ini (saat penelitian ini dibuat). Performa *noise* 2-D dinyatakan berjalan pada kecepatan beberapa miliar sampel per detik pada perangkat keras GPU terkini (saat penelitian ini dibuat), dan noise 3-D dan 4-D juga mencapai performa yang baik. Tekstur prosedural disorot sebagai pendekatan yang menguntungkan, terutama untuk GPU dengan paralelisme yang semakin meningkat. Penulis menyediakan kode sumber GLSL untuk generasi simpleks 2D, 3D, dan 4D serta generasi *Perlin* klasik. Mereka mencatat bahwa pendekatan komputasi mereka

	pada dasarnya berbeda dari implementasi sebelumnya, karena tidak bergantung pada tabel pencarian. Meskipun versi lama lebih cepat, versi komputasi mengejar ketertinggalan dengan munculnya GPU yang lebih bertenaga. Tidak adanya tabel pencarian membuat implementasi ini cocok untuk berbagai aplikasi, termasuk implementasi perangkat keras VLSI dan lingkungan di mana pencarian tekstur tidak dijamin.
Saran Penelitian Lanjutan	Penulis merekomendasikan pekerjaan lanjutan untuk menilai komputasi *noise* murni terhadap metode alternatif atau metode lainnya, mengeksplorasi algoritma secara *custom* untuk *noise* yang disesuaikan dan diimplementasikan untuk aplikasi tertentu, dan mengoptimalkan algoritma komputasi *noise* murni untuk arsitektur GPU modern.

Tabel 2.5 Literature Review [4]

Judul Journal/Thesis/Buku/Dll	*Procedural textures using tilings with perlin noise*
Penulis	David Maung, Yinxuan Shi, R. Crawfis
Tahun Publikasi	2012
Latar Belakang Masalah	Pembuatan konten prosedural (PCG) banyak digunakan dalam industri *video game* dan film untuk meningkatkan realisme dan mengurangi biaya pengembangan. Makalah ini mengeksplorasi kombinasi tekstur prosedural dan *tiling*, dengan fokus pada menghasilkan *tiling* aperiodik dari *perlin noise* untuk mengatasi tantangan dalam menghindari repetisi (*repetition)* dan meningkatkan variasi dari tekstur.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	*perlin noise* untuk generasi prosedural dan menghindari repetisi (*repetition*)
Metode Penelitian	Metode Penelitian: Metode Kuantitatif Penelitian ini mengutamakan adopsi pendekatan komputasi atau kuantitatif, memanfaatkan implementasi GPU untuk menghasilkan tekstur yang efisien dan variatif. Penulis memperkenalkan konsep *tiling noise* dan merinci konstruksi set *tiling* untuk aperiodik pada *tiling*. Fokusnya adalah mengurangi penggunaan memori dan meningkatkan fleksibilitas melalui realisasi GPU. Metodologi ini melibatkan perluasan implementasi GPU untuk menggabungkan *mipmaps* untuk menghasilkan turbulensi dan memvariasikan frekuensi *noise* di seluruh *tiling*.

Hasil dan Kesimpulan Penelitian	Makalah ini mendemonstrasikan keberhasilan generasi *noise* aperiodik menggunakan *perlin noise*. Implementasi GPU menonjol karena efisiensinya dalam hal penggunaan memori dan fleksibilitas. Penggunaan baru *mipmaps* untuk generasi turbulensi dan variasi frekuensi disajikan. Studi ini menyimpulkan bahwa teknik ini memungkinkan terciptanya suara aperiodik yang kaya dan menarik, cocok untuk pembuatan tekstur dalam game dan film.
Saran Penelitian Lanjutan	Makalah ini menyarankan bahwa teknik yang disajikan membuka jalan untuk menciptakan pola kompleks seperti kayu dan marmer menggunakan *Tiling-noise* dengan memanfaatkan *perlin noise*. Para penulis mengusulkan penelitian lanjutan untuk mengeksplorasi aplikasi tambahan dan kemajuan lebih lanjut dalam penggunaan tekstur prosedural dan *tiling*

Tabel 2.6 Literature Review [5]

Judul Journal/Thesis/Buku/Dll	*Compositional procedural content generation*
Penulis	J. Togelius, Tróndur Justinussen, Anders Hartzen
Tahun Publikasi	2012
Latar Belakang Masalah	Makalah ini membahas mengenai tantangan dan peluang dalam pembuatan konten prosedural (PCG) untuk *video game*, dengan fokus pada kebutuhan untuk menggabungkan berbagai metode PCG secara efektif. Para penulis memperkenalkan konsep komposisional PCG, menekankan perpaduan pendekatan generasi prosedural yang berbeda untuk memanfaatkan kelebihannya sekaligus mengurangi atau mengatasi kelemahannya.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	*Answer Set Programming (ASP), Diamond-square*, *Perlin noise* dan *cellular automata*
Metode Penelitian	Metode Penelitian : Metode Kuantitatif dan Kualitatif Penelitian ini menggabungkan aspek-aspek kuantitatif seperti data dan kualitatif berdasarkan studi-studi yang dilakukan oleh penelitian sebelumnya, terutama studi yang berhubungan dengan pembuatan konten prosedural (PCG) .
Hasil dan Kesimpulan Penelitian	Hasil penelitian menunjukkan kelayakan komposisi PCG, menampilkan eksperimen di mana ruang bawah tanah atau

	dungeon untuk game sebuah *roguelike* dibuat menggunakan pendekatan hybrid. ES mengembangkan parameter untuk program ASP, menghasilkan kumpulan jawaban yang ditafsirkan sebagai generasi level ruang bawah tanah atau *dungeon*. Penulis berpendapat bahwa pendekatan komposisi ini memungkinkan untuk memanfaatkan kekuatan algoritma yang berbeda-beda, memastikan properti tingkat rendah dijamin oleh algoritma dalam (*inner algorithm)*, sementara algoritma luar (*outer algorithm)* mengejar properti tingkat tinggi (*higher-level properties)*. Makalah ini menyimpulkan dengan menyarankan bahwa komposisi PCG merupakan hal penting untuk mengatasi tugas-tugas pembuatan generasi konten yang lebih menuntut atau *demanding*.
Saran Penelitian Lanjutan	Makalah ini menyarankan bahwa pembagian kerja antara algoritma yang berbeda dalam komposisi PCG adalah pendekatan yang benar dan menjanjikan. Diusulkan bahwa metode ini mungkin penting untuk solusi praktis dalam tugas pembuatan konten yang lebih kompleks. Penelitian lanjutan dapat mengeksplorasi dan menyempurnakan pendekatan komposisi PCG untuk berbagai jenis pembuatan konten seperti object atau level, yang berpotensi mengarah pada teknik pembuatan prosedural yang lebih efisien dan serbaguna.

Tabel 2.7 Literature Review [6]

Judul Journal/Thesis/Buku/Dll	*Procedural Content Generation: Goals, Challenges and Actionable Steps*
Penulis	Julian Togelius, Alex J. Champandard, Pier Luca Lanzi, Michael Mateas, Ana Paiva, Mike Preuss, & Kenneth O. Stanley
Tahun Publikasi	2013
Latar Belakang Masalah	penelitian ini membahas bidang Pembuatan konten prosedural (PCG) dalam pengembangan *video game*. PCG melibatkan pembuatan konten *video game* secara algoritmik, tidak termasuk *Engine Game* dan perilaku NPC, dan memiliki berbagai tujuan, seperti memperkenalkan variasi, mengurangi waktu pengembangan, dan memungkinkan adaptasi dalam game. Meskipun PCG telah digunakan dalam permainan sejak tahun 1980an, penelitian akademis di bidang ini muncul baru-baru ini, dengan pergeseran ke arah algoritma yang dapat beradaptasi dan dikontrol. penelitian ini menguraikan tujuan penelitian yang signifikan untuk PCG, mengakui bahwa hal tersebut memerlukan terobosan untuk dicapai, dan menyoroti evolusi yang sedang berlangsung dari bidang ini mulai dari penggunaan historisnya dalam permainan hingga pengembangan akademisnya, menyiapkan landasan bagi langkah-langkah yang dapat ditindaklanjuti untuk memajukan penelitian PCG.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	- *penelitian ini berfokus pada teori dan representasi dari

	PCG pada Video-Game
Metode Penelitian	Metode Penelitian : Metode Kualitatif penelitian ini menggunakan pendekatan penelitian kualitatif, yang terlihat dari penekanannya pada analisis tekstual mendalam, eksplorasi konseptual, dan pendapat para ahli. Daripada mengandalkan data kuantitatif atau analisis statistik, makalah ini menggali bidang pembuatan konten prosedural (PCG) melalui pemeriksaan menyeluruh terhadap literatur yang ada, perkembangan sejarah, dan wawasan kolektif para penulis. penelitian ini menawarkan pemahaman naratif dan konseptual tentang PCG, menguraikan tujuan besar, tantangan penelitian, dan langkah-langkah yang dapat ditindaklanjuti berdasarkan keahlian dan pengetahuan penulis, sehingga memandu penelitian masa depan di bidang ini. Pendekatan penelitian kualitatif ini memungkinkan makalah ini berfungsi sebagai sebuah pemikiran kepemimpinan, mendorong diskusi dan inovasi di bidang PCG tanpa bergantung pada eksperimen empiris atau data numerik.
Hasil dan Kesimpulan Penelitian	Hasil dan kesimpulan dari penelitian ini adalah penulis menyoroti tiga tujuan utama pembuatan konten prosedural (PCG) dalam pengembangan *video game* dan menekankan pentingnya mengatasi delapan tantangan penelitian untuk memajukan kecanggihan PCG secara signifikan. Tantangan-tantangan ini disajikan sebagai proyek jangka panjang dengan potensi untuk mengubah bidang ini. Para penulis menekankan bahwa keberhasilan memenuhi salah satu tantangan ini dapat memberikan kontribusi besar,

	menjadikannya topik yang cocok untuk penelitian mendalam, termasuk calon Ph.D. tesis. Selain itu, mereka menekankan bahwa meskipun tantangan-tantangan ini bersifat abstrak dan berjangka panjang, penelitian ini memberikan serangkaian langkah-langkah yang lebih konkrit yang dapat ditindaklanjuti—pertanyaan penelitian spesifik dengan prasyarat dan tolok ukur teknis yang ada. Langkah-langkah ini dirancang untuk memandu para peneliti dan praktisi dalam mencapai kemajuan dalam menghadapi tantangan, menawarkan titik masuk yang nyata bagi mereka yang tertarik untuk memajukan bidang PCG. Para penulis menyimpulkan dengan mencatat bahwa PCG mempunyai potensi besar baik bagi industri game maupun ilmu desain game, menghadirkan lahan subur untuk eksperimen dan inovasi, dan mereka mendorong penelitian lebih lanjut mengenai teknik-teknik baru untuk memajukan bidang ini.
Saran Penelitian Lanjutan	Penelitian lanjutan yang berasal dari penelitian ini mencakup beberapa aspek utama. Pertama, para peneliti didorong untuk menyelidiki tantangan penelitian yang teridentifikasi dalam pembuatan konten prosedural (PCG) untuk memajukan teknologi terkini secara signifikan. Tantangan-tantangan ini dapat menjadi landasan bagi penyelidikan komprehensif, yang bertujuan untuk mendorong batas-batas teknik PCG. Kedua, para akademisi dan praktisi harus mengambil langkah-langkah yang dapat ditindaklanjuti sebagai proyek nyata. Pertanyaan penelitian spesifik dengan prasyarat teknis yang ada ini mewakili titik masuk langsung untuk

	mencapai kemajuan dalam PCG, menawarkan peluang untuk mengatasi beberapa tantangan yang lebih luas. Selain itu, sangat penting untuk terus mengeksplorasi teknik dan metodologi baru di PCG untuk membuka potensi penuhnya, sehingga memberikan manfaat bagi industri *video game* dan ilmu desain *video game*. Secara keseluruhan, upaya berikutnya harus berpusat pada memperdalam pemahaman kita tentang PCG, mengatasi tantangannya, dan berinovasi di lapangan untuk mewujudkan tujuan besarnya.

Tabel 2.8 Literature Review [7]

Judul Journal/Thesis/Buku/Dll	*Procedural Content Generation for Game Props? A Study on the Effects on User Experience*
Penulis	Oliver Korn, M. Blatz, A. Rees, J. Schaal, V. Schwind, Daniel Görlich
Tahun Publikasi	2017
Latar Belakang Masalah	Menjelaskan mengenai meningkatnya minat terhadap pembuatan konten prosedural (PCG) untuk *video game* yang telah memicu perdebatan tentang kualitasnya dengan membandingkan konten tradisional buatan *traditional-artist* dan PCG. Studi ini menjelaskan kebutuhan dan perbandingan akan keberagaman grafis dan *gameplay* yang menyebabkan PCG dapat dieksplorasi pada berbagai elemen game, serta menjelaskan ketertarikan pemain terhadap PCG dibandingkan manual.
Algoritma yang dipakai dalam	Algoritma *perlin noise* untuk PCG

Journal/Thesis/Buku/Dll	
Metode Penelitian	Metode Penelitian : Metode Kuantitatif Penelitian ini menggunakan pendekatan komparatif sehingga bersifat kuantitatif, membandingkan antara permainan menggunakan PCG dan permainan menggunakan kreasi *traditional-artist.* Studi ini melibatkan 41 pemain yang memainkan keduanya, dan pengalaman mereka dievaluasi menggunakan skala Likert. dengan fokus pada metrik pengalaman pengguna.
Hasil dan Kesimpulan Penelitian	Hasilnya menunjukkan bahwa konten yang dibuat secara prosedural, khususnya pada kasus *'reefs'* dalam hal ini, dianggap lebih realistis dan estetis dibandingkan dengan konten yang dibuat secara manual. Pemain yang terlibat juga menyatakan preferensi lebih baik terhadap perubahan lingkungan yang dihasilkan secara prosedural dengan hasil 60% setuju dengan generasi PCG, dengan menekankan peningkatan variasi dalam gameplay.
Saran Penelitian Lanjutan	Penelitian lanjutan harus memvalidasi temuan ini melalui studi perbandingan serupa, dengan fokus pada berbagai aspek elemen grafis dan gameplay. Disarankan untuk memperluas cakupan untuk generasi *quest,* karakter, dan perilaku kecerdasan buatan yang dihasilkan secara prosedural.

Tabel 2.9 Literature Review [8]

Judul Journal/Thesis/Buku/Dll	*Procedural Generation in Games Focusing on Dungeons*
Penulis	Zhenyuan Shen
Tahun Publikasi	2022
Latar Belakang Masalah	penelitian ini membahas implementasi generasi secara prosedural, Generasi secara prosedural adalah algoritma utama dalam *video game digital*, khususnya untuk menghasilkan medan (*terrain*), lanskap (*landscape)*, konten, dan objek dalam dunia *video game*. Penelitian ini menyelidiki prinsip-prinsip pembuatan *dungeon generation*, menekankan tiga komponen utama: pembuatan medan (*terrain*), pembuatan konten, dan pembuatan objek. Tujuannya adalah untuk memahami cara kerja metode pembuatan generasi prosedural dan relevansinya dalam berbagai skenario permainan. Makalah ini menyoroti konteks historis pembuatan prosedural dalam game, menelusuri asal-usul kembali game RPG di era permulaan seperti '*Dungeons & Dragons*'.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *Fractal generation, bitmap generation,* dan *perlin noise* untuk generasi *procedural dungeon*
Metode Penelitian	Metode Penelitian: Metode Kualitatif Penelitian ini melibatkan analisis komprehensif metode pembuatan prosedural dan penerapannya dalam permainan *video game*. Studi ini dibagi menjadi tiga bagian: pembuatan medan (*terrain),* pembuatan konten,

	dan pembuatan objek. Beberapa algoritma dan contoh kode yang relevan digunakan untuk menjelajahi setiap bagian yang dijelaskan sebelumnya. Penelitian ini membahas pentingnya generasi secara prosedural dalam mengembangkan konten *video game* yang bervariasi dan dinamis, meningkatkan efisiensi pengembangan *video game*, dan memaksimalkan pengalaman pemain.
Hasil dan Kesimpulan Penelitian	Penelitian ini menyimpulkan generasi secara prosedural dalam pengembangan *video game,* dapat memungkinkan pengembang membuat beragam konten tanpa pekerjaan manual yang ekstensif. Metode pembuatan medan (*terrain)* yang berbeda akan dianalisis dengan berbagai cara, seperti Pembuatan Medan (*terrain)* dengan Fraktal yang menawarkan pembuatan medan acak yang cepat, Pembuatan Medan (*terrain)* Bitmap untuk medan (*terrain)* yang kaya fitur, dan *perlin noise* untuk generasi medan (*terrain)* yang lebih halus dan alami. Penelitian ini menekankan bahwa generasi secara prosedural berpengaruh signifikan mengurangi waktu dan biaya yang diperlukan dalam pengembangan *video game.* Penelitian ini juga menguraikan pekerjaan di masa depan, termasuk menghasilkan *procedural dungeon* dan medan (*terrain)* yang multi-dimensi.

Saran Penelitian Lanjutan	Penelitian lanjutan harus melibatkan eksplorasi mendalam tentang metode pembuatan medan (*terrain)* dengan bereksperimen dengan variasi parameter dan mengembangkan pendekatan hibrida untuk pembuatan medan (*terrain)* yang lebih serbaguna dan realistis. Selain itu, harus ada fokus pada optimalisasi proses pembuatan prosedural, dengan penekanan pada pengurangan overhead komputasi, peningkatan manajemen memori, dan memastikan kinerja yang konsisten di berbagai konfigurasi perangkat keras, khususnya di lingkungan game berskala besar.

Tabel 2.10 Literature Review [9]

Judul Journal/Thesis/Buku/Dll	Procedural city generation using Perlin noise
Penulis	*Niclas Olsson & Elias M. Frank*
Tahun Publikasi	2017
Latar Belakang Masalah	Latar belakang masalah dalam penelitian ini berkisar pada pemilihan perlin noise sebagai teknik pembuatan konten prosedural (PCG) untuk menghasilkan kota dalam konteks pengembangan *video game*. *perlin noise* sendiri dipilih karena kelayakan implementasinya dalam jangka waktu terbatas. Para penulis menyoroti bahwa banyak metode PCG untuk generasi kota yang rumit dan memakan waktu, sehingga kurang praktis untuk pengembangan *video game*. *perlin noise* lebih disarankan karena kemampuannya membuat konten yang tampak lebih alami karena sifat bawaannya.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *perlin noise* untuk pembuatan konten prosedural (PCG)
Metode Penelitian	Metode Penelitian : Metode Kualitatif dan Kuantitatif Metodologi penelitian dalam penelitian ini mencakup dua komponen utama: pertama, fase implementasi di mana penulis mengembangkan sistem generasi kota secara prosedural menggunakan *perlin noise* sebagai teknik

	utamanya, yang melibatkan pengembangan perangkat lunak, perhitungan kuantitatif, dan pembuatan konten algoritmik. Kedua, studi pengguna dilakukan untuk mengevaluasi efektivitas dan kesesuaian kota-kota yang dihasilkan secara prosedural untuk permainan *video game*. Pada studi kualitatif ini menggabungkan survei atau kuesioner yang diberikan kepada minimal 20 peserta yang berinteraksi dengan kota-kota yang dihasilkan dari generasi secara prosedural, hal bertujuan untuk mengumpulkan data kuantitatif dan kualitatif mengenai pengalaman pengguna, preferensi, dan umpan balik.
Hasil dan Kesimpulan Penelitian	Kesimpulan dari penelitian ini adalah *perlin noise* dinyatakan dapat dimanfaatkan secara efektif untuk menghasilkan kota-kota yang cocok untuk digunakan dalam permainan. Namun, kredibilitas kota yang dihasilkan bergantung pada parameter spesifik yang digunakan selama proses generasi. Intinya, keberhasilan pendekatan ini bergantung pada pemilihan parameter generasi yang cermat untuk mencapai hasil yang diinginkan.
Saran Penelitian Lanjutan	Untuk penelitian lanjutan, ada beberapa arah potensial yang dapat dieksplorasi. Pertama, memperluas jumlah parameter generasi, seperti mempertimbangkan kedekatan antar distrik, ukuran kota, dan menyempurnakan *perlin noise*, dapat memberikan pengguna kontrol yang lebih besar terhadap generasi kota. Kedua, berfokus pada pembuatan generasi *mesh* dari awal / dari nol dibandingkan mengandalkan *mesh* buatan tradisional atau manual yang terbatas untuk dapat

	menghasilkan lebih banyak variasi bangunan unik sekaligus menghemat waktu. Terakhir, menyelidiki cara untuk mengenerasi jalan yang melengkung untuk melepaskan diri dari batasan dua dimensi dan memperkenalkan tata ruang kota yang lebih organik dan menarik secara visual dapat meningkatkan kealamian kota yang dihasilkan. Pembuatan *mesh* secara prosedural direkomendasikan untuk memfasilitasi perbaikan ini.

Tabel 2.11 Literature Review [10]

Judul Journal/Thesis/Buku/Dll	*Procedural Content Generation pada Game World Exploration Sandbox Menggunakan Algoritma perlin noise*
Penulis	Desando Anugrah Ramadhan & Aries Dwi Indriyanti
Tahun Publikasi	2022
Latar Belakang Masalah	Artikel ini membahas latar belakang masalah pembuatan konten prosedural (PCG) pada *Sandbox Game World Exploration*. Artikel ini memperkenalkan pembuatan konten prosedural (PCG), sebuah metode untuk menghasilkan konten game secara otomatis, termasuk dunia pada *video game*. Algoritma yang berbeda diperlukan untuk menghasilkan berbagai jenis konten termasuk Algoritma *perlin noise*. Artikel ini menekankan penggunaan algoritma *perlin noise* untuk menciptakan medan (*terrain)* yang tampak alami, seperti bukit dan gua, dan menyoroti kemampuannya untuk menghasilkan angka acak terstruktur dengan pola gradien halus.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *perlin noise* untuk pembuatan konten prosedural (PCG)
Metode Penelitian	Metode Penelitian : Metode Kualitatif dan Kuantitatif Fokus utama penelitian ini adalah perancangan dan implementasi permainan *sandbox* 2D menggunakan algoritma *perlin noise* yang merupakan metode kualitatif. Sementara itu, tahap pengujian memang melibatkan

	penilaian tampilan dunia game yang terstruktur dan alami. Pengujian melibatkan responden yang memberikan umpan balik kualitatif, seperti preferensi, kepuasan, dan masalah yang mereka temui saat bermain game.
Hasil dan Kesimpulan Penelitian	Hasil penelitian penelitian tersebut melibatkan pengujian game yang dibuat oleh 12 responden. Tujuan dari pengujian ini adalah untuk menilai kesesuaian nilai skala *perlin noise* yang berbeda untuk berbagai medan (*terrain*) dalam *video game* 2D dan untuk mengevaluasi kepuasan responden terhadap dunia game yang dihasilkan. Pengujian secara khusus berfokus pada tiga medan (*terrain)* yaitu : bukit, gua, dan awan. Hasil penelitian menunjukkan bahwa untuk formasi bukit, 7 dari 12 responden menyatakan puas. Untuk kasus gua, 9 responden merasa puas, dan 3 responden netral. Untuk formasi awan, 3 responden menyatakan tidak puas, 5 responden netral, dan 4 responden menyatakan puas. Tercatat, beberapa responden berpendapat bahwa formasi awan lebih cocok untuk dilihat dari sudut tertentu, baik dari atas maupun bawah. Pengujian ini juga melibatkan penentuan nilai rentang *noise* yang sesuai untuk medan (*terrain)* yang berbeda. Untuk formasi perbukitan, sebagian besar responden lebih menyukai nilai rentang *noise* sebesar 0,03, diikuti oleh 0,1. Untuk gua, responden cenderung menyukai kisaran *noise* 0,1, yang menunjukkan preferensi terhadap formasi gua yang lebih kecil dan banyak. Mengenai pembentukan awan, 7 responden memilih nilai rentang *noise* x=0.1,

	y=0.2, dan 4 responden memilih x=0.05, y=0.15, yang menunjukkan preferensi pada awan berukuran sedang. Beberapa responden mengalami masalah selama bermain game, seperti karakter tidak dapat memasuki gua satu blok, dan lebar layar dianggap terlalu sempit. Hasilnya juga menunjukkan bahwa pilihan nilai rentang *noise* dipengaruhi oleh faktor-faktor seperti ukuran dunia game, ukuran karakter, dan ukuran kamera, yang menunjukkan adanya ketergantungan pada parameter ini.
Saran Penelitian Lanjutan	Penelitian lanjutan diuraikan dalam penelitian ini berfokus pada beberapa aspek. Hal ini menunjukkan potensi integrasi algoritma pembuatan konten secara prosedural yang berbeda untuk mendiversifikasi konten dalam *video game*, termasuk elemen seperti pembuatan objek/model secara prosedural. Selain itu, penelitian lanjutan dapat mengoptimalkan pembentukan medan (*terrain)* untuk berbagai perspektif game, sehingga memastikan pengalaman visual yang menarik. Ada ruang untuk mengeksplorasi elemen *gameplay* tambahan dan fitur interaktif yang dapat meningkatkan pengalaman bermain *video game* secara keseluruhan. Masukan pengguna akan sangat berharga dalam meningkatkan aspek seperti ukuran karakter dan lebar layar untuk memastikan *gameplay* yang lebih lancar dan menyenangkan. Pada akhirnya, penelitian lanjutan memerlukan perluasan proses pengembangan game untuk menciptakan dunia game yang lebih kompleks dan rumit, menantang kemampuan metode pembuatan konten secara prosedural, dan mendorong batas-batas yang dapat dicapai dalam desain game dan pembuatan

	dunia.

Tabel 2.12 Literature Review [11]

Judul Journal/Thesis/Buku/Dll	*Island Generator pada Game Open world Menggunakan Algoritma Perlin noise*
Penulis	Naufal Azzmi, Lailatul Husniah, A. S. Kholimi
Tahun Publikasi	2020
Latar Belakang Masalah	Studi ini berfokus pada perkembangan pesat industri game, dengan penekanan khusus pada pembuatan konten prosedural (PCG) untuk desain sebuah dunia pada *video game*. Secara khusus, penelitian ini mengeksplorasi implementasi generasi pulau untuk dunia pada *video game* yang terbuka (*open-world)* menggunakan algoritma *perlin noise.* PCG semakin banyak digunakan untuk mengurangi beban pengembang dan menciptakan konten game yang unik dan luas secara dinamis.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *perlin noise* untuk generasi tekstur yang akan digunakan sebagai *height-map* pada pulau
Metode Penelitian	Metode Penelitian : Metode Kuantitatif Penelitian ini berfokus pada pengujian kemampuan bermain dan kinerja. Variabel seperti oktaf, persistensi, dan *lacunarity* dimanipulasi untuk mengontrol hasil tekstur. Studi ini menggabungkan filter, termasuk filter radial dan kotak (*box),* untuk membentuk medan (*terrain)* 3D dan membentuk pulau.
Hasil dan Kesimpulan Penelitian	Penelitian menyimpulkan bahwa algoritma *perlin noise* berhasil membentuk tekstur utama untuk peta *height-map* atau digunakan sebagai generasi medan (*terrain)* dalam

	pengembangan *video game*. Variabel *noise* yang koheren berdampak signifikan terhadap penciptaan tekstur, dengan oktaf, persistensi, dan *lacunarity* yang memainkan peran penting. Generasi sebuah pulau secara prosedural ketika diuji kemampuan bermainnya, terbukti cocok untuk berbagai jenis game, khususnya *video game* yang terbuka (*open-world)* dengan setting pulau. Analisis kinerja menggunakan notasi Big O menunjukkan kinerja linier, dengan kecepatan algoritma sebanding dengan jumlah pulau yang dihasilkan.
Saran Penelitian Lanjutan	Penelitian lanjutan dapat mengeksplorasi variabel dan parameter tambahan untuk lebih mendiversifikasi pulau-pulau yang dihasilkan dan meningkatkan fleksibilitas algoritma. Teknik optimasi untuk pembangkitan yang lebih cepat, serta eksplorasi algoritma *noise* alternatif, dapat berkontribusi untuk meningkatkan efisiensi dan realisme generasi pulau seperti menghasilkan objek untuk mengisi pulau.

Tabel 2.13 Literature Review [12]

Judul Journal/Thesis/Buku/Dll	*Implementasi Generate Map dan Pemunculan Objek secara Acak pada Game 3D menggunakan Bahasa C# dan Metode perlin noise di Unity: Studi Kasus pada Game Development*
Penulis	Renita Selviana, Dauw Bastha Fiastat Lugata, Alimin Alimin
Tahun Publikasi	2023
Latar Belakang Masalah	Studi ini berfokus pada bidang pengembangan game yang sedang berkembang saat ini, dimana menciptakan lingkungan yang menarik sangatlah penting. Pembuatan lingkungan secara manual dapat memakan waktu, sehingga mendorong pengembang untuk beralih ke teknik Pembuatan Konten Prosedural (PCG). Maka dari itu penelitian ini memilih algoritma *perlin noise*, yang dikenal menghasilkan elemen visual yang realistis, banyak digunakan dalam pengembangan game. Penelitian ini bertujuan untuk mengimplementasikan teknik PCG menggunakan C# dan *perlin noise* pada *Unity* untuk pengembangan *video game* 3D yang tujuan utama yaitu mengotomatiskan pembuatan peta dan generasi objek atau *obstacle* secara acak untuk meningkatkan pengalaman bermain game.
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Algoritma *perlin noise* untuk generasi map dan generasi objek atau *obstacle*
Metode Penelitian	Metode Penelitian : Metode Kualitatif

	Penelitian ini menggunakan pendekatan kualitatif untuk menilai dampak penerapan teknik PCG menggunakan C# dan *perlin noise* pada *Unity*. Analisis data kualitatif melibatkan pengamatan interaksi dan pengalaman pengguna dalam lingkungan game.
Hasil dan Kesimpulan Penelitian	Studi penelitian tersebut menyimpulkan bahwa penerapan teknik PCG meningkatkan kualitas *gameplay* dan memberikan pengalaman yang lebih mendalam bagi pengguna. Pembuatan peta otomatis dan kemunculan objek secara acak berkontribusi dalam menciptakan variasi dalam *gameplay*, meningkatkan keterlibatan pengguna. Selain itu, penelitian ini menyoroti potensi untuk mempercepat proses pengembangan game melalui pembuatan konten secara otomatis.
Saran Penelitian Lanjutan	Penelitian lanjutan disarankan untuk dapat mengeksplorasi parameter tambahan untuk pembuatan peta dan generasi objek untuk mendiversifikasi *gameplay* yang lebih lanjut seperti generasi dengan multi-level atau layering dan juga masalah pada *overlapping*. Penelitian ini masih memiliki kekurangan mengenai variasi dalam pembuatan objek karena objek dihasilkan sepenuhnya di-generasi secara acak. Selain itu, menyelidiki optimalisasi algoritma untuk efisiensi dan mengeksplorasi metode PCG alternatif/lainnya yang dapat bermanfaat untuk penelitian lanjutan

Tabel 2.14 Literature Review [13]

Judul Journal/Thesis/Buku/Dll	*Geometry: The Language of Space and Form*
Penulis	J. Tabak
Tahun Publikasi	2014
Latar Belakang Masalah	-
Algoritma yang dipakai dalam Journal/Thesis/Buku/Dll	Teori dasar Algoritma *euclidean-distance*
Metode Penelitian	Metode Kuantitatif Buku ini mengkaji secara kuantitatif mulai dari teori dasar, hingga contoh dalam mengimplementasikan *euclidean-distance* dalam aspek matematika
Hasil dan Kesimpulan Penelitian	Buku ini digunakan untuk referensi dalam perhitungan rumus *euclidean-distance* untuk solusi dari pencarian metode *overlapping-detection*
Saran Penelitian Lanjutan	-