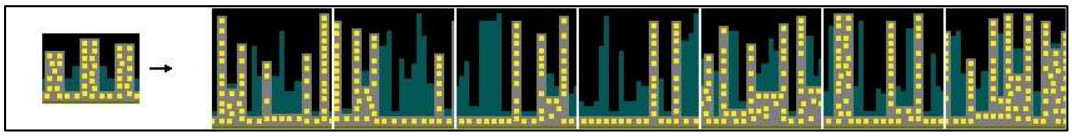
BAB II

LANDASAN TEORI

2.1. Wave Function Collapse

Wave Function Collapse (WFC) awalnya merupakan algoritma procedural generation yang bekerja pada bitmap. Algoritma ini membagi bitmap menjadi pattern dengan ukuran NxN. Kemudian *array output* diinisialisasi dengan setiap elemen menampung variabel *state* yang menentukan *pattern* apa saja yang dapat dipakai. Algoritma ini kemudian masuk ke fase observasi-propagasi. Dalam tahap observasi, elemen dengan jumlah *pattern* paling sedikit dipilih. Elemen tersebut kemudian memilih *pattern* yang tersedia secara acak. Dalam tahap propoagasi, terjadi pengurangan *pattern* yang dapat dipakai pada elemen-elemen di sekitar elemen yang diobservasi. Saat seluruh elemen telah diobservasi, *array* tersebut digunakan untuk membentuk bitmap baru yang mempunyai ciri khas yang sama dengan bitmap aslinya. (Gumin, 2016)



Gambar 2.1. Proses generasi bitmap menggunakan WFC. (Gumin, 2016)

2.2. Simple Tiled Model

Simple tiled model adalah WFC yang lebih sederhana dengan menggunakan *tilemap* sebagai input. Setiap *tile* memiliki *identifier* pada setiap sisinya, sehingga *tile* yang mungkin digunakan pada satu kotak *grid* dibatasi oleh empat *tile*

non-diagonal di sekitarnya. Algoritma ini juga mempertimbangkan *tile* yang simetri atau diputar. (Heaton, 2018)

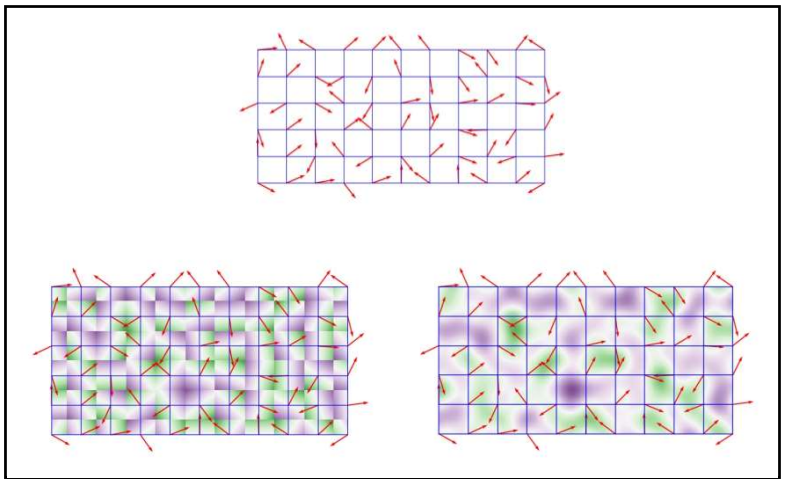
Gambar 2.2. Proses generasi *level* menggunakan WFC *simple tiled model*.

(Stalberg, 2017)

2.3. Perlin Noise

Noise pada dasarnya merupakan sebuah fungsi yang menghasilkan nilai *random* yang telah diinterpolasikan dalam sebuah grid. Fungsi *noise* yang ideal bersifat *pseudo-random*, dimana menggunakan input yang sama akan menghasilkan output yang sama. (scratchapixel.com)

Perlin noise merupakan algoritma *noise* yang dibuat oleh Ken Perlin. Noise ini dibentuk menggunakan suatu *grid* dimana setiap ujung kotak dari *grid* memiliki vektor gradien *random*. Setiap titik yang terdapat dalam grid ini terletak pada satu kotak grid. Pada setiap titik, *dot product* dari jarak titik ke ujung kotak *grid* dengan gradien dari setiap ujung kotak dihitung. Hasil *dot product* yang sejumlah dengan ujung kotak *grid* kemudian diinterpolasikan menggunakan *smoothstep interpolation*. (Biagioli, 2014)

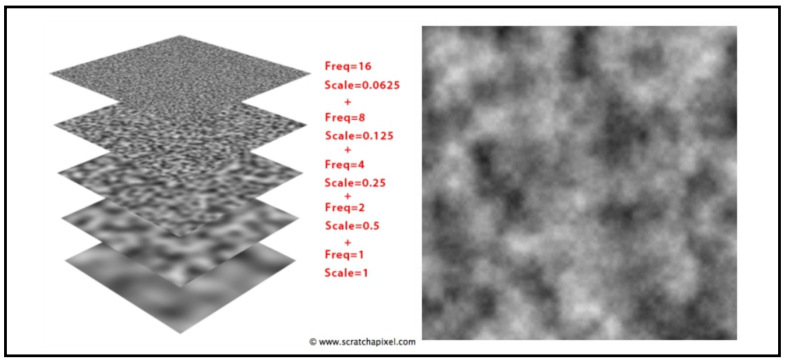


Gambar 2.3. Proses pembentukan *perlin noise*. (Matthewslf, 2019)

2.4. Fractal Noise

Fractal noise adalah hasil dari teknik akumulasi pada *noise* apapun seperti *perlin noise*, *simplex noise*, atau *value noise* dengan frekuensi dan amplitudo yang berbeda yang dikenal sebagai *fractal sum*. *Noise* ini dikontrol oleh tiga parameter yaitu *octave*, *lacunarity*, dan *persistence* (disebut sebagai *gain* dari sumber). *Octave* menentukan jumlah *noise* yang diakumulasi, *lacunarity* menentukan peningkatan frekuensi dalam setiap *octave*, dan *persistence* menentukan penurunan amplitudo dalam setiap *octave*. (scratchapixel.com, 2016) Rumus dari fractal noise ditampilkan pada Rumus 2.1.

$$\text{fractalnoise} = \sum_{n=0}^{o-1} noise(xy * lacunarity^n) \times persistence^n \quad (2.1.)$$



Gambar 2.4. Proses pembentukan *fractal noise* (scratchapixel.com, 2016)

2.5. Infinite Terrain Generation

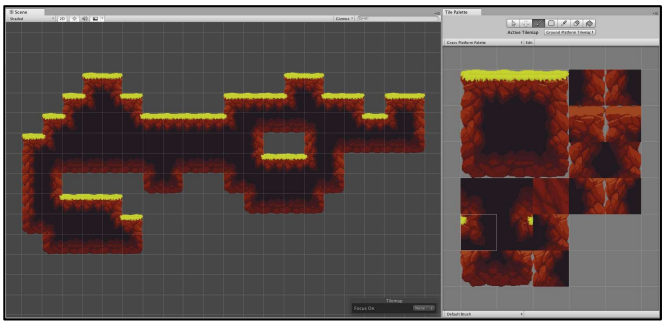
Infinite terrain generation adalah proses pembentukan *terrain* dengan ukuran tak terbatas. Proses ini menggunakan *seeded pseoudo-randomness*, dimana nilai *seed* digunakan untuk menentukan rangkaian angka yang dihasilkan *pseudo-random number generator*. S*eeded pseoudo-randomness* berguna saat dibutuhkan terrain yang sama misalnya dalam *game multiplayer*, dimana setiap pemain diberikan nilai seed yang sama oleh server. Penggunaan nilai *seed* yang sama akan menghasilkan terrain yang sama. *Infinite terrain* dibagi menjadi wilayah-wilayah persegi yang dikenal sebagai *chunk*. Setiap *chunk* memiliki ukuran yang sama. Saat sebuah titik acuan (umumnya posisi pemain atau kamera) bergerak, *chunk* baru dibentuk mengikuti arah gerakan sedangkan *chunk* dengan jarak yang lebih jauh dari jarak yang telah ditentukan dinonaktifkan. (Scholz, 2019) . *Biome* adalah sebuah cara untuk mengklasifikasikan terrain yang dibentuk, seperti hutan, padang pasir, atau laut. *Biome* dapat ditentukan melalui berbagai faktor seperti suhu, kelembapan, dan/atau ketinggian yang didapatkan melalui

noise map. (Gallant, 2016) Gambar 2.5. menampilkan contoh *infinite terrain* dengan visualisasi *chunk* dimana tepi *chunk* digambar sebagai garis merah.

Gambar 2.5 *Infinite terrain* 2 dimensi (spin.atomicobject.com, 2015)

2.6. Tilemap

Tilemap merupakan teknik yang sangat populer dalam *game development*, dimana dunia/level dibentuk melalui pecahan gambar / model berukuran kecil yang disebut *tile*. *Tile* tersebut dapat disusun dalam *grid* untuk membentuk struktur yang kompleks dan bervariasi. Kumpulan *tile* gambar dapat dikelompokkan menjadi satu gambar yang disebut *tile atlas*. *Tilemap* sering dikombinasikan *grid* non-visual untuk mengimplementasi logika *game* seperti *pathfinding* dan *collision handling*. (MDN Web Docs, 2019)



Gambar 2.6. Implementasi *tilemap* (Unity Blog, 2018)