BAB II

LANDASAN TEORI

2.1. Game Design

Game merupakan suatu aktivitas yang didefinisikan menggunakan aturan dimana pemainnya mencoba meraih suatu *goal* (Galloway, 2006). Dalam memainkan suatu *game* pemainnya selalu diarahkan untuk menyelesaikan *game* tersebut, tetapi terdapat pula *game* yang tidak memiliki akhir seperti *Minecraft* minecraft merupakan *game* dengan *genre sandbox* dimana pemain dapat melakukan apapun tanpa batas di dalam *game* tersebut. Dalam pembentukan sebuah *game* terdapat *formal elements* yang membantu pembentukan struktur *game* (Nacke, 2014). Sebagai *game designer* terdapat beberapa *formal elements* yang perlu diperhatikan yaitu *players, objectives, procedures, rules, resources, conflict, boundaries, outcome.*

2.2. Procedural Coherent Noise

Coherent noise merupakan suatu *pseudo-random number* perbedaan antar nilainya sedikit yang dihasilkan melalui suatu *noise function* (Bevins, 2003). Semua jenis *coherent noise* dengan dimensi N selalu membutuhkan input sejumlah dengan N dimensi. Salah satu contoh dari *coherent noise* adalah *Perlin noise*. Selain *perlin noise* masih terdapat banyak jenis *noise* lain seperti *simplex, billow, ridged-multifractal* dan *voronoi*, masing-masing dari *noise* function tersebut menghasilkan nilai *pseudorandom* yang berebeda cukup signifikan antar jenis *noise*. Dalam

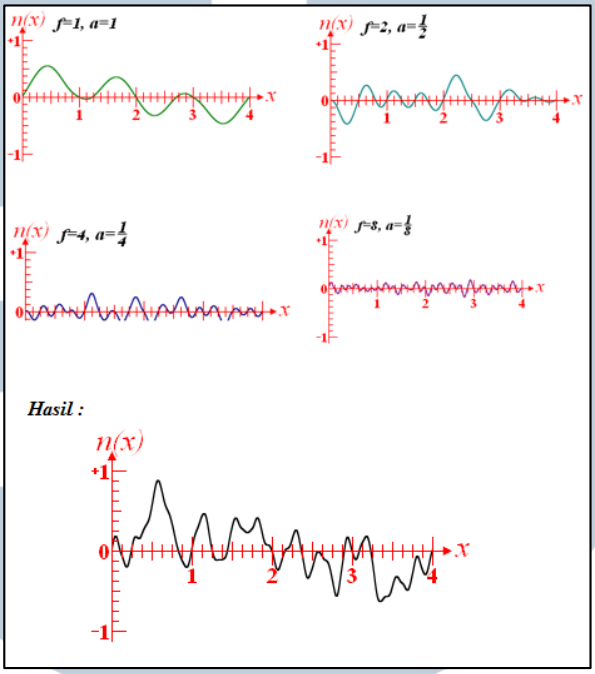
pembuatannya *noise* memiliki beberapa atribut dimana bila ada atribut yang berubah maka hasilnya berbeda, atribut tersebut yaitu (Bevins, 2003):

1. *Frequency*, yaitu jumlah gelombang yang dihasilkan setiap satu unit ukuran panjang.
2. *Octave*, merupakan kumpulan *coherent noise* yang memiliki tingkat detail berbeda dan digunakan untuk membentuk satu buah *noise*.
3. *Amplitude*, merupakan seberapa besar batas positif dan negative dari hasil *noise function* yang digunakan (-n & +n).
4. *Lacunarity*, merupakan seberapa cepat *frequency* bertambah setiap *octave*.
5. *Persistence*, merupakan seberapa cepat *amplitude* berkurang setiap *octave.*

2.2.1. Perlin Noise

Perlin noise merupakan suatu coherent noise yang dihasilkan dari hasil penambahan beberapa coherent noise dengan frekuensi yang meningkat dan amplitude yang menurun (Bevins, 2003). Perlin noise dibuat oleh Ken Perlin pada tahun 1982 dan memiliki atribut yang sama dengan *coherent noise* lainnya. pada tahun 1985 Ken Perlin melakukan *improvement* terhadap *perlin noise* pada *paper* yang berjudul *Improving Noise*. Perubahan yang dilakukan adalah penentuan nilai *gradient* dimana menjadi 12 pada *noise* 3D yaitu menjadi (1, 1, 0), (-1, 1, 0), (1, -1, 0), (-1, -1, 0), (1, 0, 1), (-1, 0, 1), (1, 0, -1), (-1, 0, -1), (0, 1, 1), (0, -1, 1), (0, 1, -1), (0, -1, -1) (Perlin, 2002). Nilai *gradient* akan ditentukan menggunakan nilai *P* modulus 12, dimana *P* merupakan *permutation* yang berisi *pseudo-random number*.

Terdapat pula perubahan rumus *interpolation* dari $3t^2 - 2t^3$ menjadi $6t^5 - 15t^4 + 10t^3$ (Perlin, 2002).



Gambar 2.1 Kumpulan Noise dan Hasil Akumulasi (Bevins, 2003)

Dalam pembuatannya *perlin noise* memiliki beberapa tahapan yang diperlukan untuk dilakukan yaitu (Scratchpixel, 2016):

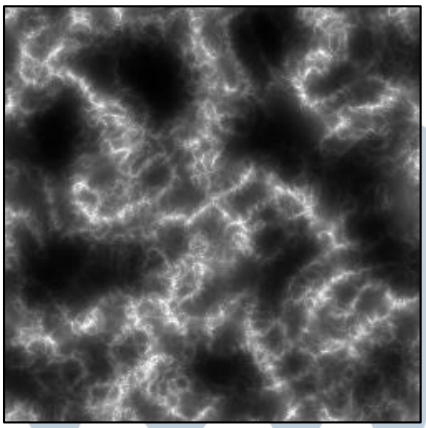
1. Menentukan 8 titik membentuk kubus.
2. Menentukan *gradient* setiap titik menggunakan *P* yang di modulus 12.
3. Mencari *dot product* dari setiap titik dengan mengalikan *gradient* dengan koordinat posisi yang ingin dicari.
4. Melakukan *smoothing* menggunakan $6t^5 - 15t^4 + 10t^3$.
5. *trilinear interpolation* menggunakan nilai setelah *smoothing*.

2.2.2. Ridged Multifractal Noise

Ridged-multifractal noise merupakan suatu *coherent noise* yang dibentuk memanfaatkan *perlin noise*, walaupun memanfaatkan *perlin noise* hasil yang dikeluarkan bukan merupakan hasil penggabungan dengan *perlin noise* sendiri. Hasil dari *ridged-multifractal noise* merupakan bentuk pegunungan yang panjang bukan melipat atau tumpul (Carpentier, 2011). *Ridged noise* memanfaatkan pengambilan nilai absolut dalam pembuatan *perlin noise* dan memiliki rumus (Carpentier, 2011):

$$ridgedmultifractal = 1 - |perlinnoise(p, seed)| \quad ...(2.1)$$

Dimana p merupakan koordinat posisi yang ingin dicari dan *seed* adalah *offset* dari posisi.



Gambar 2.2 Contoh *Ridged Noise* 2 Dimensi (Blevins, 2015)

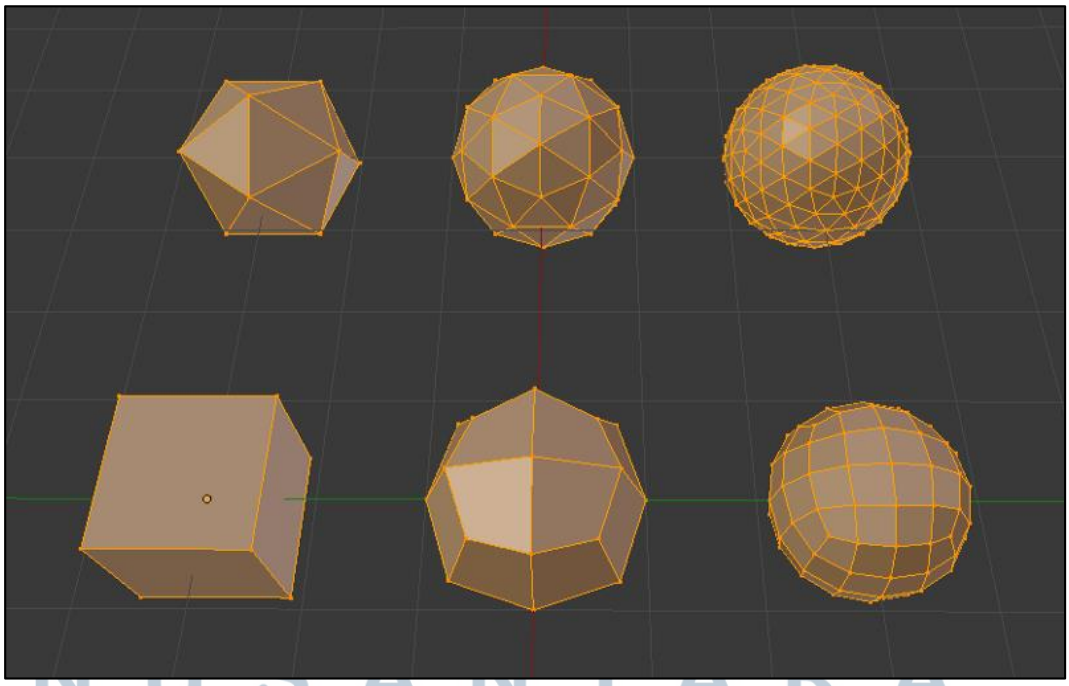
2.2.3. Libnoise Library

Libnoise merupakan sebuah *open source library* yang digunakan untuk membuat *coherent noise*, suatu noise yang berubah dengan halus (Bevins, 2003).

Libnoise memiliki banyak jenis *coherent noise* modul untuk *noise* yang tersedia merupakan 3D *noise*.

2.3. Cube Sphere

Cube Sphere merupakan metode dimana pembuatan bentuk *sphere* memanfaatkan bentuk dasar *cube*. Metode ini memiliki cara dengan membangun sebuah cube dengan melakukan render bentuk *primitive* (segitiga) untuk membangun sebuah bentuk persegi dengan resolusi yang ditentukan untuk dijadikan satu *face* untuk *cube*. Setiap *face* dari *cube* di *scaling* sesuai dengan titik tengahnya, proses ini dilakukan berulang sampai semua *face* selesai dibuat. Terdapat pula metode lain untuk pembuatan bentuk *Sphere*, metode ini yaitu *Icosahedron*. Metode yang digunakan adalah *Cube Sphere* dikarenakan dengan penggunaan metode tersebut hasilnya lebih mudah diberikan *tiling texture* dibandingkan dengan *icosahederon* (Winslow, 2017).



Gambar 2.3 *Icosahedron* dan *Cube Sphere* (Winslow, 2017)

Pada gambar 2.3 dapat dilihat bahwa bentuk *icosahedron* murni dibuat menggunakan segitiga yang disusun sedemikian rupa sehingga mendekati bentuk bola. Sedangkan pada gambar bagian bawahnya yaitu *cube sphere* memanfaatkan pembentukan *cube* menggunakan kumpulan segitiga dan melakukan kalkulasi ulang untuk membentuk *sphere*.

2.4. Faux Gravity

Faux Gravity merupakan metode untuk membuat sebuah simulasi gravitasi yang didapatkan dengan menerapkan prinsip *gravitational attraction*. Metode ini diperlukan sebagai salah satu elemen penting dimana agar pemain dapat berpetualang pada planet yang berbentuk *spherical*. Ini juga dipengaruhi oleh kemampuan simulasi gravitasi dari *Unity Engine* yang terbatas dan hanya mensimulasikan gravitasi selalu jatuh ke bawah. Konsep dari *gravitational attraction* adalah seluruh obyek memiliki gaya tarik-menarik dimana gaya tersebut proporsional dengan massa dari kedua objek tersebut dan berbanding terbalik dengan jarak antar benda kuadrat (Physicsclassroom, 2019). Dikarenakan gaya yang dihasilkan proporsional dengan massa dari benda maka, semakin besar massanya maka semakin kuat gayanya, dan bila jarak antar benda semakin jauh maka semakin kecil gaya yang dihasilkan (Physicsclassroom, 2019). Berikut ini adalah rumus dari *gravitational attraction* (Khanacademy, 2014):

$$F = \frac{GM_1M_2}{r^2} . \overrightarrow{r} \quad \text{... (2.2)}$$

Dimana:

1. G : merupakan gaya gravitasi dengan nilai 6.67 x 10^{-11} N(m/kg)^2

2. M1 & M2 : merupakan massa dari obyek 1 dan 2
3. r : merupakan jarak antar benda
4. $\underset{r}{\rightarrow}$: merupakan vector dari obyek 1 ke obyek 2

Dengan menggunakan rumus ini dan diletakan pada seluruh obyek yang ada, maka dengan penggunaan massa yang sesuai, dapat diimulasikan gravitasi yang terjadi pada bumi.

2.5. Game Design Document

Dalam industri *game* tidak terdapat standar dalam penulisan dokumentasi, ada baiknya bila terdapat sebuah formula atau *style* yang dapat digunakan sebagai acuan. Setiap orang setuju bahwa sebuah *design document* yang bagus memiliki detail yang dibutuhkan untuk membangun *game* (Fullerton, 2014). Menurut Leonardo Gonzalez penulisan *Game Design Document* dapat dibagi menjadi sembilan bagian yaitu (Gonzalez, 2016):

1. Characters : merupakan pengenalan karakter-karakter yang digunakan dalam *game.*
2. Story : menceritakan sebuah kumpulan kejadian yang terjadi selama *game* berlangsung.
3. Story Progression : merupakan penjelasan perkembangan alur cerita secara detail, Contohnya ketika *player* menyelesaikan misi pertama maka kelanjutan *story* dijelaskan kepada *player.*
4. Gameplay : merupakan bagian terpenting dalam *Game Design Document* dimana menjelaskan:

a. Goals : alasan *player* memainkan *game*, misalnya membunuh semua monster.
b. User skills : merupakan kemampuan *player* dalam melakukan *input* pada *game.*
c. Game mechanics : menjelaskan bagaimana *game* seharusnya bekerja.
d. Item & power-ups : menjelaskan alat bantu *player* untuk mencapai *goals*, seperti *health potion*.
e. Losing : menjelaskan kondisi apa saja yang menyebabkan *player* kalah.

5. Art Style : menjelaskan penggunaan *art* pada keseluruhan *game*, contohnya 3D menggunakan *low polygon assets*.
6. Music & Sound : memberikan kumpulan musik dan efek yang digunakan dalam *game*.
7. Technical Description : menjelaskan pada *platform* apa saja *game* dirilis.
8. Marketing & Funding : menjelaskan bagaimana cara memasarkan *game* dan meraih pemasukan dari *game* tersebut. Bagian ini tidaklah wajib.
9. Other Ideas : merupakan kumpulan-kumpulan ide yang dimiliki tetapi tidak yakin untuk diimplementasikan pada *game.*

Dalam penulisan *Game Design Document* tidak seluruh bagian harus ditulis. Bila terdapat bagian yang tidak ada dalam *game* maka bagian tersebut dapat dihilangkan, misalnya bagian *story*, maka bagian dengan unsur *story* tidak ditulis dan lebih mendetail pada bagian yang dimiliki oleh *game* (Gonzalez, 2016).

2.6. Game User Experience Statisfaction Scale (GUESS)

Game User Experience Satisfaction Scale merupakan salah satu metode untuk mengukur *video game satisfaction* yang memanfaatkan 55 pertanyaan yang menggunakan *seven point likert scale*. Dari 55 pertanyaan tersebut dapat dibagi menjadi Sembilan buah *subscale* yaitu (Phan, 2016), *usability / playability*, *narratives, play engrossment, enjoyment, creative freedom, audio aesthetics, personal gratification, social connectivity,* dan *visual aesthetics.*

Dalam penilaian menggunakan metode GUESS bila dalam *game* tidak terdapat elemen seperti *narratives* dan *social connectivity*, maka *subscale* tersebut dapat dihilangkan dari penilaian (Phan, 2016). Cara penilaian dalam metode GUESS yaitu dengan mencari rata-rata setiap *subscale* lalu mencari rata-rata dari gabungan rata-rata *subscale* yang telah diperoleh untuk dijadikan sebagai *video game satisfaction* (Phan, 2016). Berikut ini adalah tabel *interval* dengan 7 poin yang didapat dengan membagi *range* (100) dengan jumlah *interval* (7) untuk mendapatkan intervalnya (Idtesis, 2011).

Tabel 2.1 Tabel *Interval* 7 Poin

Range nilai	Keterangan
0% - 14.285	Sangat Tidak Puas (*Very Dissatisfied*)
14.286% - 28.571%	Tidak Puas (*Not Satisfied*)
28.572% - 42.857%	Agak Tidak Puas (*Quite Dissatisfied*)
42.858% - 57.142%	Biasa Saja (*Ordinary*)
57.143% - 71.428%	Cukup Puas (*Quite Satisfied*)
71.429% - 85.714%	Puas (*Satisfied*)
85.715% - 100%	Sangat Puas (*Very Satisfied*)