BAB 2
LANDASAN TEORI

2.1 Video Game Design Elements

Video Game merupakan bentuk dari hiburan interaktif yang dapat dimainkan oleh pemain melalui *computer*, *gaming console*, hingga *mobile device*. Dalam mendesain sebuah *video game*, ada beberapa poin yang harus diperhatikan oleh *game developer*. *Context*, terdiri dari ruang, objek, cerita, dan perilaku yang akan ditemukan pemain dalam permainan. *Participants*, hal ini merujuk pada para pemain *game* dan perilaku yang mereka miliki saat mereka bermain. Biasanya perilaku ini merupakan bentuk balasan atas *context* dari *game* yang disiapkan, dan pemain diberikan pilihan untuk melakukan aksi berdasarkan *context* tersebut. *Meaning*, merupakan konsep yang menjadi pilihan sebagai bentuk respon dan interaksi pemain dengan semua komponen dalam *game*.

Dalam mengembangkan sebuah *video game*, terdapat beberapa elemen yang dinilai sebagai elemen yang ada dalam sebuah permainan. Elemen-elemen ini akan menjadi struktur utama dari *video game* yang dibangun, dan terdiri dari *Formal Elements* dan *Dramatic Elements*.

Formal Elements merupakan komponen yang menjadi pembentuk dari *video game*. Berikut merupakan bagian dari *Formal Elements* [5].

1. *Player*, pemain yang berinteraksi baik antar *player* ataupun dengan *environment* dalam sebuah *game*.

2. *Objectives*, merupakan tujuan yang menjadi motivasi *player* untuk masuk pada *gameplay*.

3. *Procedures*, merupakan rangkaian aksi dan metode yang sesuai dengan aturan dalam *video game* untuk memungkinkan tercapainya *objective* pada game tersebut.

4. *Rules*, batasan yang ditentukan sehingga *gameplay* memiliki aturan yang boleh dan tidak boleh diikuti oleh *player* untuk menjadi patokan tindakan yang dapat diambil.

5. *Resources*, merupakan *game object* hingga *game elements* yang membantu *player* untuk mencapai *objective*.

6. *Conflict*, adalah ringtangan yang muncul dari rangkaian *procedures* dan *rules* yang ditetapkan, yang dapat menghalangi *player* untuk mencapai *objective* dari *game*.

7. *Boundaries*, merupakan batasan; baik dalam aturan maupun dari *game elements*, yang biasanya berkaitan dengan ruang *gameplay*.

8. *Outcome*, adalah hasil dari permainan, hasil ini biasanya akan muncul saat *objective* telah dicapai ataupun gagal dicapai (*endings*).

Dramatic Elements merupakan elemen yang digunakan untuk membuat *game* menjadi lebih imersif pada *player*. Berikut merupakan bagian dari *dramatic elements* [6].

1. *Story*, merupakan bagian yang menjelaskan cerita atau *plot* pada *environment* dalam *game* tersebut.

2. *Challenge*, merupakan rangkaian tantangan untuk mendapatkan pencapaian dan tujuan kecil untuk meningkatkan kepuasan dan kemampuan saat bermain.

3. *Play*, merupakan pendekatan untuk terjadinya suatu kejadian (*event*) antar dan interaksi ini berdasarkan *player* dan *game object* yang berbeda pula.

4. *Premise*, merupakan inti dari berjalannya suatu kejadian berdasarkan apa saja yang sudah dan akan terjadi.

5. *Characters*, adalah sejumlah karakter yang dapat berinteraksi dengan pemain dan berperan dalam alur cerita dari permainan.

2.2 Procedural Content Generation

Procedural Content Generation (PCG) adalah bentuk pengaplikasian untuk menghasilkan konten *game* yang unik tiap kali instansi tersebut dijalankan oleh komputer, sehingga hasil nya akan menjadi unik dan berbeda untuk tiap pemain [7]. *Procedural Content Generation* merujuk pada algoritma yang digunakan untuk menciptakan suatu konten. Dimana konten tersebut dibuat secara otomatis, dan dapat mengurangi beban kerja dari *game design* dan *development*. Beberapa metode yang diciptakan telah menjadi metode yang awam digunakan dalam industri *gaming* meskipun masih diterapkan pada konteks dan *game element* yang spesifik pada

suatu *game* [8]. Dalam pembuatan *game element* seperti *level design*, atau pada contoh penelitian ini merupakan *maze*, *Procedural Content Generation* akan menggunakan *maze generation algorithms*, dimana salah satu algoritma yang dapat diimplementasikan dalam metode ini adalah *Prim's Algorithm*.

Dengan *Prim's Algorithm*, implementasi *Procedural Content Generation* dalam membuat sebuah *maze* akan dimulai dengan menciptakan sebuah *grid* yang akan menjadi 'lantai' dari *maze* (*maze cell*). Kemudian akan dilakukan fungsi untuk melakukan inisialisasi bagian-bagian yang akan dijadikan bagian dari *maze*, baik sebagai dinding dari *maze* (*maze wall*) maupun sebagai jalan dari *maze* (*maze passage*).

2.3 Prim's Algoritm

Maze generation algorithm memiliki beberapa algoritma yang dapat digunakan dalam penerapan metodenya. Perbedaan yang dimiliki algoritma-algoritma ini adalah panjang, bentuk, dan jumlah koridor dari hasil yang dibuat saat implementasi tingkat kesulitan, efisiensi dan tingkat komprehensif dari algoritma tersebut. Pada game yang akan dibuat ini, algoritma yang dipilih adalah Prim's Algorithm [9].

Prim's Algorithm merupakan algoritma yang biasanya digunakan untuk menemukan *Minimum Spanning Tree* (MST) pada sebuah grafik. Algoritma ini akan mencari *subset* dari *edge* pada tiap *vertex* dan akan mencari jumlah kombinasi yang terkecil. Langkah yang diambil pada *prim's algorithm* ini adalah:

1. Mencari titik awal
2. Memilih titik *edge* yang menghubungkan ke titik berikutnya, dan akan mencari *edge* dengan jarak paling kecil untuk mencapai titik berikut tersebut.
3. Setelah *edge* dengan jarak terkecil ditemukan, *edge* tersebut akan dimasukkan ke MST
4. Langkah 2 dan 3 diulang hingga semua titik berhasil dicari *edge* terkecil.
5. Algoritma berhenti, dan *list* dengan edge MST telah selesai.

Sedangkan untuk implementasi *maze generation*, algoritma ini memiliki sedikit modifikasi dalam langkah yang ada sehingga langkah yang ada adalah sebagai berikut:

1. Mulai dengan membuat *two-dimensional array* yang akan diisi *maze cell* yang disimpan dalam bentuk *grid* (x, y).
2. Pilih salah satu *maze cell* dari *array*, tandai sebagai bagian dari *maze* dan masukan bagian tersebut ke dalam sebuah list.
3. Dengan daftar *maze cell* di dalam *list*, akan dilakukan langkah iterasi di bawah pada semua anggota di dalam list tersebut:
    - Pilih anggota (*maze cell*) yang ada di dalam list, jika anggota tersebut telah diproses pada segala arah dalam *maze cell* (baik inisialisasi *maze wall* ataupun *maze passage*), maka anggota tersebut akan dihapus dari *list* dan iterasi kembali diulang dengan anggota lain yang dipilih.
    - Pilih anggota (*maze cell*) yang ada di dalam list, jika anggota tersebut belum diproses pada segala arah dalam *maze cell*, akan dilakukan inisialisasi komponen *maze* pada *maze cell* tersebut, baik untuk *maze wall* atau *maze passage*.

2.4 GUESS

The *Game User Experience Satisfaction Scale* (GUESS) merupakan sebuah skala yang dirancang untuk mengevaluasi *video game satisfaction* atau tingkat kepuasan pemain akan *game* secara psikometri dalam bentuk skala Likert yang terdiri dari 9 *subscales*. *Subscales* ini terdiri dari *Usability, Narratives, Play Engrossment, Enjoyment, Creative Freedom, Audio Aesthetics, Personal Gratification, Social Connectivity*, dan *Visual Aesthetics*. Setiap *subscales* ini memiliki sejumlah pertanyaan yang dijawab dengan menggunakan ukuran skala Likert (poin 1 s/d 7, dimana 1 adalah nilai untuk '*Strongly Disagree*', dan secara progresif akan naik hingga poin ke 7 untuk '*Strongly Agree*') yang jika ditotal adalah 55 pertanyaan. Adapun ketentuan yang dimiliki adalah, setiap *subscale* yang ada tidak wajib dimasukkan dalam penilaian jika komponen *subscale* tersebut memang tidak ada dalam *game* yang dinilai [10]s.

Setiap pertanyaan yang dijawab dengan skala Likert kemudian akan dikumpulkan sebagai nilai yang kemudian akan dihitung rata-ratanya dengan rumus sebagai berikut.

$$Rata - rata\ (\%) = \frac{\sum_{i=1}^{7}(n_i * i)}{n_{total} * 7}$$

Gambar 2.1. Perhitungan Rata-rata hasil GUESS

Maksud dari n dalam rumus pada Gambar 2.1 di atas adalah jumlah responden. n_i merupakan jumlah dari responden yang memilih jawaban pilihan i yang berkisar dari skala 1 sampai dengan 7 pada pertanyaan GUESS yang diberikan. n_i kemudian akan dikali dengan skala jawaban mereka masing-masing, dimana n_1 akan dikali 1, n_2 dikali 2, dan seterusnya hingga n_7 dikali 7. Setelah dikali, angka tersebut akan dijumlahkan dan dibagi dengan notasi total responden keseluruhan dikali dengan skala tertinggi dalam pilihan jawaban, yaitu 7.

$$Rata - rata\ subscale\ (\%) = \frac{\sum_{i=1}^{n}(rata - rata\ pertanyaan\ i)}{n} x\ 100\%$$

Gambar 2.2. Perhitungan Rata-rata subscale GUESS

0% - 14,825%	Sangat Buruk
14,826% - 28,571%	Buruk
28,572% - 42,857%	Cukup Buruk
42,858% - 57,142%	Cukup
57,143% - 71,428%	Cukup Baik
71,429% - 85,714%	Baik
85,715% - 100%	Sangat Baik

Tabel 2.1. Tabel Interval Penilaian GUESS

Setelah hasil penilaian rata-rata pada setiap pertanyaan ditemukan, nilai tersebut akan dikelompokan berdasarkan *subscales* yang ada dan akan dicari rata-ratanya untuk menentukan hasil penilaian berdasarkan subscale tersebut (Gambar

2.2). Hasil rata-rata *subscales* kemudian akan dikali 100% untuk mencari persentase interval dalam hasil penilaian. Hasil pencarian rata-rata tiap *subscale* tersebut akan kembali dicari rata-ratanya untuk menentukan hasil penilaian keseluruhan GUESS dari MazeGame, beserta persentasenya. Skala penilaian menggunakan persentase rata-rata dan hasil penilaian dapat diamati hasilnya berdasarkan dengan Tabel 2.1 di atas.