# BAB 2
# LANDASAN TEORI

## 2.1 Game Development

*Game development* atau pengembangan *game* adalah seni membuat *game* dan menggambarkan desain, pengembangan hingga rilisnya sebuah *game*. Pengembangan *game* ini memiliki beberapa proses di antaranya melibatkan konsep, desain, pembuatan *game*, pengujian serta publikasi dari *game* itu sendiri. Pengembangan *game* ini sendiri dapat dilakukan oleh studio-studio besar atau bahkan hanya seorang saja. Pada umumnya, pengembangan suatu *game* dibutuhkan beberapa peran lainnya selain seorang developer, seperti desainer, artis, dan beberapa peran lainnya yang dibutuhkan untuk membuat *game* tersebut [12].

Menurut Devin, pada proses pengembangan *game* terdapat 7 tahapan yang akan dilalui di antaranya: [13]

1. Planning
   Melakukan perencanaan terhadap *game* yang ingin dibuat, pada tahapan ini pada umumnya akan ditentukan tipe *game* apa yang akan dibuat, apakah *game* tersebut akan menjadi *game* 2D atau 3D, fitur yang akan diberikan dari *game* tersebut, karakter, cerita yang akan diberikan, target pemain yang diharapkan, pada platform apa *game* ini akan dapat dimainkan dan kemudian didukung dengan adanya perencanaan finansial yang akan dibutuhkan baik sekarang atau kedepannya dari *game* yang akan dibuat.

2. Pre-production
   Pada tahapan ini, semua peran akan saling berdiskusi untuk melakukan *brainstorming* terhadap perencanaan yang sudah dibuat sebelumnya.

3. Production
   Pada tahapan inilah *game* yang direncanakan tersebut akan mulai dibuat. Segala aspek seperti model dari karakter yang ada, audio yang akan digunakan, desain *level*, serta kode dari *game* tersebut akan mulai dibuat.

4. Testing
   Sesuai dengan namanya, *testing* melibatkan seorang tester yang akan memainkan *game* setelah dari tahapan *production*. Pada tahapan inilah dimana *game* yang telah dibuat tersebut diuji, sehingga segala kemungkinan

masalah baik itu *bug*, adanya ketidaksesuaian terhadap cerita, dan masalah lainnya yang dapat menurunkan kepuasan pemain, akan dapat dilaporkan oleh banyak tester yang kemudian dapat ditangani dan diperbaiki.

5. Pre-launch

   Tahapan ini merupakan tahapan yang paling penting terhadap suksesnya *game* yang telah dibuat. Para studio besar pada umumnya akan mendapatkan banyak tekanan pada tahapan ini. Hal seperti bagaimana pemain akan memberikan reaksi terhadap *game* yang telah dibuat, apakah akan ada masalah seperti *bug* yang tidak ditangkap sebelumnya ditemukan oleh pemain, dan bagaimana cara menarik perhatian pemain. Pada saat inilah juga biasanya *publisher* dari *game* tersebut akan mulai menjadwalkan tempat untuk mengumumkan *game* tersebut. Hal tersebut mungkin tidak dapat dilakukan oleh seorang yang independen dikarenakan besarnya anggaran yang dibutuhkan, sehingga terkadang untuk *game developer* yang independen,memasarkan *game* tersebut kepada beberapa situs khusus, atau dengan mengirimkan *early-access beta* dari game tersebut ke beberapa orang terkenal untuk mendapatkan perhatian.

6. Launch

   Tahapan *launch* ini adalah saat dimana *game* yang telah dibuat dari semua tahapan sebelumnya, mulai dibuka ke publik. Pada saat inilah juga informasi seperti *bug* atau laporan lainnya yang diberikan oleh pemain baik secara otomatis atau manual, akan diurutkan sesuai dengan prioritasnya masing-masing.

7. Post-launch

   Pada tahapan ini *game studio* atau seorang developer dapat melihat hasil akhir dari kerja keras yang dilakukan. Namun, semua tidak berhenti disitu saja, walaupun mungkin penghasil bertambah, *game* yang telah dibuat tersebut masih harus dipantau dan diperbaharui kembali. Terutama, sudah lazim jika setelah rilisnya sebuah *game* akan didapatkan beberapa *bug* kecil yang nantinya akan dibetulkan dan ditambal di versi *game* berikutnya. Pada tahapan inilah juga *game* yang sudah jadi tersebut dapat dikembangkan kembali baik dengan menambah fitur atau konten baru lainnya.

## 2.2 Game Design Document

Situs Nuclino menjelaskan bahwa *Game Design Document* merupakan *blueprint* dari sebuah permainan atau *game* yang akan dibuat. Dokumen tersebut akan membantu mendefinisikan permainan yang akan dibuat dan memberikan ruang lingkup untuk proyek tersebut supaya seluruh tim berada pada halaman yang sama [14].

*Game Design Document* ini dapat mempunyai struktur yang bermacam-macam, namun pada dokumen tersebut selalu mempunyai beberapa poin utama di antaranya seperti:

1. Ringkasan secara umum
   Berisi konsep dari permainan yang akan dibuat, target para pemain dan juga cakupan dari proyek pembuatan permainan tersebut.

2. Gameplay
   Berisi objektif dari permainan tersebut, inti dari permainan yang akan dibuat, bagaimana cara memainkannya dan juga UI yang digunakan dalam permainan tersebut.

3. Mekanik
   Berisi cara kerja permainan tersebut dari peraturan yang akan ditetapkan dan cara bertempur.

4. Elemen permainan
   Berisi elemen-elemen yang ada dalam permainan tersebut, cerita dari permainan, lokasi dan juga desain level.

5. Aset permainan
   Berisi aset-aset yang akan digunakan dalam permainan itu, baik itu musik, efek suara, model dua dimensi atau tiga dimensi.

## 2.3 Rhythm Game

*Rhythm Game* merupakan genre permainan yang bertemakan musik dimana pemainnya bermain dengan melakukan tindakan sesuai dengan ritme dari musik tersebut. Permainan ini biasanya memiliki tingkat kesulitan mulai dari yang mudah

hingga ke sulit, konten dari permainan ini biasanya dibuat secara manual oleh seseorang dengan menyesuaikan aksi dan juga waktu terhadap kesesuaian ritme yang ada pada musik tersebut [15].

## 2.4 Unity Game Engine

Unity merupakan *cross-platform game engine* yang dikembangkan oleh Unity Technologies berbasis bahasa pemograman C#. *Game engine* Unity didesain khusus untuk memberikan kemudahan kepada pengembangan sebuah permainan baik itu dalam komputer, seluler dan juga sistem operasi yang berbeda-beda [16].

## 2.5 SQLite

SQLite merupakan *library* yang dapat digunakan untuk mengimplementasikan sistem *database Structured Query Language* (SQL) dan digunakan untuk menyimpan suatu data. Sesuai dengan namanya ”Lite”, SQLite ini memiliki fitur yang lebih sedikit dibanding dengan *database* SQL pada umumnya, salah satunya yaitu tidak diperlukan proses terpisah seperti adanya server. *Source Code* dari SQLite ini berada pada *public domain* dan dapat digunakan secara gratis untuk penggunaan pribadi atau komersial. SQLite ini juga memiliki beberapa *binding* untuk beberapa bahasa pemrograman lainnya seperti C, C++, BASIC, C#, Python, Java, dan Delphi sehingga dapat digunakan di banyak tempat [17].

SQLite membaca dan menulis data secara langsung ke dalam satu file khusus yang berada dalam tempat penyimpanan pemain. Format *database* yang digunakan juga cocok untuk berbagai macam platform. Sehingga sistem *database* SQLite ini cocok digunakan pada aplikasi yang tidak memiliki akses ke internet, atau game untuk menyimpan data yang mereka butuhkan. Dengan ukuran *library* dan penggunaan memori yang sangat kecil membuat SQLite ini mudah untuk digunakan pada perangkat-perangkat dengan memori yang kecil seperti *smartphone*, *mp3 player* atau *PDAs* [17].

## 2.6 Hamming Distance

*Edit Distance* merupakan cara yang digunakan untuk mengukur perbedaan di antara dua buah *string* dengan cara menghitung jumlah operasi minimum yang dilakukan untuk mengubah satu *string* ke *string* lainnya. Semakin sedikit jumlah operasi yang dibutuhkan, maka semakin mirip kedua *string* tersebut. Maka dari

itu, *Edit Distance* ini sering diaplikasikan dalam pemrosesan data teks khususnya seperti pada mesin bahasa atau penerjemah. Terdapat berbagai jenis *Edit Distance* yang dapat digunakan sesuai dengan rangkaian operasi *string* yang diinginkan di antaranya yaitu *Hamming Distance*, *Levenshtein Distance* dan *Jaro-Winkler* [11].

*Hamming distance* itu sendiri, merupakan jumlah perbedaan simbol di antara kedua simbol *string* dengan panjang yang sama. Simbol tersebut dapat bermacam-macam seperti sebuah huruf, bits, atau angka. *Hamming distance* ini pada awalnya digunakan sebagai salah satu cara untuk mengukur jumlah kebisingan yang ada pada saat melakukan transmisi pesan melalui suatu saluran, dari hasil ukurannya tersebut digunakan untuk mengukur jumlah error yang terjadi pada saat proses transmisi dilakukan. Hal tersebut dikarenakan pada cara tradisional yang digunakan dulu, yang dikhawatirkan adalah apakah bit tersebut pada kedua string yang ada, sesuai atau tidak [18, 19].

*Hamming distance* dapat dicontohkan dengan kedua *bitstring* yang berbeda, misalnya 1011101 dan 1001001. Dari kedua *bitstring* sebelumnya didapatkan *Hamming Distance* yaitu 2, dikarenakan ada dua bit yang berbeda di antara mereka berdua. Contoh lainnya yaitu pada kedua *string* berbeda dengan panjang yang sama dapat dilihat pada Gambar 2.1.

| | [0] | [1] | [2] | [3] |
|---|---|---|---|---|
| String 1 | h | a | l | o |
| String 2 | h | e | e | o |
| Hamming Distance | - | a | l | - |
| Nilai Jarak | 0 | 1 | 1 | 0 |

Gambar 2.1. Ilustrasi *Hamming Distance* pada kedua *string* berbeda

```
int hammingDist(char str1[], char str2[])
{
    int i = 0, count = 0;
    while(str1[i]!='\0')
    {
        if (str1[i] != str2[i])
            count++;
        i++;
    }
    return count;
}
```

Gambar 2.2. Algoritma *Hamming Distance* pada bahasa C

Contoh algoritma dari *Hamming Distance* dalam bahasa C dapat dilihat pada Gambar 2.2. Dapat dilihat bahwa fungsi *hammingDist* akan menerima dua buah *string* dengan panjang yang sama, kemudian akan dilakukan proses iterasi sebanyak jumlah panjang dari kedua *string* yang diberikan. Kemudian pada setiap iterasi tersebut, sesuai dengan nilai pengulangan iterasi, akan dilakukan pengecekan karakter pada posisi yang sama di antara kedua *string* yang diberikan. Jika kedua karakter tersebut berbeda, maka variabel *count* akan ditambah. Setelah proses iterasi selesai, maka hasil akhir dari variabel *count* merupakan nilai *Hamming Distance* yang didapatkan dari *string* yang diberikan. Dari algoritma tersebut dapat disimpulkan bahwa algoritma *Hamming Distance* ini memiliki kompleksitas waktu terburuk $O(n)$ dan kompleksitas waktu terbaik $O(1)$ [20].

## 2.7 Game User Experience Satisfaction Scale 18 (GUESS-18)

*Game User Experience Satisfaction Scale 18* atau dapat disebut sebagai GUESS-18, merupakan suatu alat ukur yang dapat digunakan untuk mengukur kepuasan pemain terhadap 9 aspek hal yang terdapat dalam suatu *game*. GUESS-18 ini merupakan GUESS yang sebelumnya sering digunakan dengan maksud dan tujuan yang sama, telah dikembangkan lagi sehingga yang sebelumnya membutuhkan 55 pertanyaan untuk mendapatkan tujuan yang sama, di potong menjadi hanya 18 pertanyaan. Hal ini dilakukan dikarenakan berdasarkan penelitian yang dilakukan oleh Joseph dan kawan-kawan, mereka melihat bahwa jika pengguna menjawab 55 buah pertanyaan, dimana hal tersebut akan dilakukan berulang-ulang, maka akan menjadi tidak praktis. Maka dari itu mereka membuat sistem penilaian yang bernama GUESS-18, yang memiliki tujuan untuk menyelesaikan permasalahan sebelumnya [21].

GUESS dan GUESS-18 memiliki 9 subskala berbeda yang mengukur berbagai macam aspek dalam suatu *game*, di antaranya adalah [22]:

1. Usability/Playability
   Seberapa mudah *game* dapat dimainkan untuk menyelesaikan objektif yang diberikan tanpa adanya gangguan atau halangan dari antarmuka atau kontrol *game* yang ada.

2. Narratives
   Aspek cerita dari *game* baik itu peristiwa atau karakter yang ada, seberapa

besar kemampuan dari cerita tersebut menangkap perhatian dan membentuk emosi pemain.

3. Play Engrossment
   Seberapa besar kemampuan dari *game* untuk menahan perhatian dan minat dari pemain.

4. Enjoyment
   Besarnya kesenangan dan sukacita yang dirasakan oleh pemain dari hasil memainkan *game*

5. Creative Freedom
   Seberapa besar kemampuan dari *game* untuk dapat mengikuti kreativitas dan keingintahuan pemain.

6. Audio Aesthetics
   Seberapa besar pengaruh audio pada *game* memperkaya pengalaman bermain pemain.

7. Personal Gratification
   Aspek motivasi dari *game*, seperti tantangan yang diberikan untuk meningkatkan keinginan pemain menyelesaikan dan atau melanjutkan memainkan *game*.

8. Social Connectivity
   Sejauh mana *game* memfasilitasi koneksi sosial antar pemain melalui alat dan fitur yang diberikan.

9. Visual Aesthetics
   Seberapa menarik grafik dari *game* menurut pemain.

Kesembilan aspek diataslah yang nantinya akan dilihat melalui hasil dari 18 pertanyaan yang diberikan. Setiap pertanyaan tersebut dijawab dengan menggunakan skala likert dari 1 hingga 7 yang merepresentasikan:

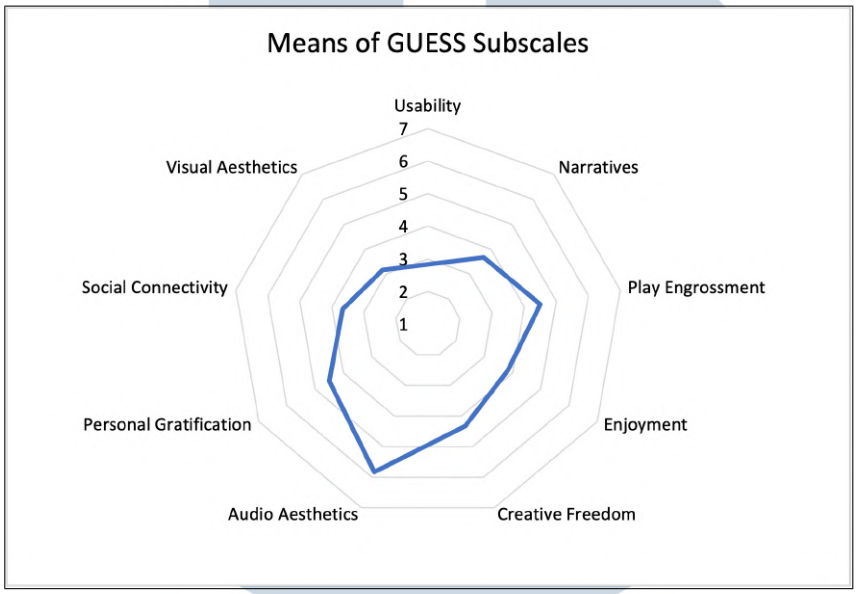
- Sangat tidak setuju (1)
- Tidak setuju (2)
- Cukup tidak setuju (3)

- Netral (4)
- Cukup setuju (5)
- Setuju (6)
- Sangat setuju (7)

Data yang didapatkan nantinya akan dilakukan perhitungan dengan mencari nilai rata-rata dari setiap pertanyaan pada aspek yang sama dari data semua responden. Perhitungan rata-rata tersebut dapat dilakukan dengan cara sebagai berikut:

$$\begin{aligned} Rata-Rata(\%) = &((\text{Sangat tidak setuju} * 1) + (\text{Tidak setuju} * 2) \\ &+ (\text{Cukup tidak setuju} * 3) + (\text{Netral} * 4) + (\text{Cukup setuju} * 5) \\ &+ (\text{Setuju} * 6) + (\text{Sangat setuju} * 7)) \\ &/(\text{Jumlah Nilai Responden} * \text{Nilai Skala Tertinggi}) * 100\% \end{aligned} \tag{2.1}$$

Dari hasil rata-rata tersebutlah yang akan menentukan nilai kepuasan pemain terhadap setiap aspek yang ada dan juga nilai kepuasan dari keseluruhan *game* yang diberikan. Hasil-hasil tersebut pada nantinya dapat direpresentasikan menggunakan *radar chart* sebagai contoh:




Gambar 2.3. *Radar Chart* dari GUESS-18