BAB 2
LANDASAN TEORI

2.1 Game

Game merupakan salah satu aplikasi yang diminati oleh berbagai macam kalangan pengguna karena mempunyai daya tarik yang tinggi. *Game* merupakan aktivitas menyenangkan berupa kolaborasi dan kompetisi yang dimainkan dengan sejumlah rintangan dan aturan bermain agar dapat memenuhi kondisi kemenangan [9]. Aturan dan rintangan yang dibuat ditujukan agar memberikan kesan membatasi pergerakan pemain untuk mencapai kemenangan. Aturan dan rintangan membuat rasa tantangan saat dalam proses menyelesaikan *game*.

Alur pembuatan *game* diawali dengan tahap perancangan atau desain. *Game* memiliki *element* yang dapat membuat *game* menjadi lebih menarik yakni *formal elements*. *Formal elements* merupakan elemen pembentuk struktur *game*. Tanpa adanya unsur *formal element* ini, maka *game* akan menjadi *dull* dan kurang menarik. Berikut merupakan penjabaran dari formal elements menurut Gibson [10].

1. *Player*

   Player merupakan pengguna yang akan langsung berinteraksi dengan *game*. *Player* akan diberikan aturan atau *rules* dan *resource* atau sumber daya untuk mencapai objektif. *Player* juga harus ditentukan sebagai apa atau siapa dan berapa banyak *player* di dalam *game* tersebut.

2. *Objective*

   Objective dibuat dengan tujuan agar menggambarkan arahan yang perlu dicapai oleh *player* untuk menyelesaikan *game*. *Objective* harus dibuat secara jelas sehingga tidak membingungkan pemain. *Objective* juga harus memiliki tingkat kesulitan yang memungkinkan untuk diselesaikan dan tidak terlalu memberatkan *player* saat bermain.

3. *Rules*

   Rules atau biasa disebut aturan ini merupakan pembatas yang ada di dalam *game*. Aturan ini memberikan suatu batasan kepada *player* tentang apa saja yang boleh dilakukan dan tidak boleh dilakukan. Pembuatan *rules* perlu dipikirkan dan diatur sedemikian sehingga game terlihat adil dan responsif

terhadap berbagai situasi yang terdapat di dalam *game*. Pembuatan dari *rules* juga perlu memerhatikan konsep dari kemudahan bagi *player* untuk memahaminya.

4. *Procedure*

   Procedure merupakan *step-step* yang dilakukan dan dilalui oleh *player* agar dapat mencapai *objective* dari *game*. *Procedure* memungkinkan game tetap berada dalam batasan *environment* yang terdapat dalam game. Dalam proses *game design*, kemampuan membuat *procedure* yang mudah dimengerti oleh pemain sangat diperlukan.

5. *Conflict*

   Conflict merupakan unsur dalam *game* yang memberikan rintangan dan tantangan kepada *player* agar objective dari suatu *game* tidak dapat dicapai dengan begitu mudah. Dalam *game design*, terdapat beberapa macam konflik dalam game. Konflik-konflik tersebut dapat berupa *opponents, obstacles* dan *dillema*.

6. *Boundaries*

   Boundaries merupakan unsur dalam *game* yang membedakan hal-hal game dengan hal-hal di luar *game*. *Boundaries* dapat berupa batasan fisik, ataupun unsur batasan lainnya dalam *game*. Unsur batasan dapat berupa bagaimana game dapat dimainkan, kapan *rules* dapat aktif berfungsi ataupun kapan *player* ingin memulai dan keluar dari *game*. Boundaries meningkatkan *user experience* dari pengguna.

7. *Resource*

   Resource atau biasa disebut dengan sumber daya ini merupakan unsur yang dapat digunakan oleh *player* agar dapat membantu dirinya mencapai objective dari *game*. *Resource* perlu dikelola dengan baik sehingga memiliki nilai usefulness di dalamnya bagi pemain. Contoh *resource* yang sering dijumpai yaitu nyawa, HP, *points*, dan lainnya.

8. *Outcome*

   Outcome merupakan hasil yang didapatkan dari hasil permainan oleh *player* karena *objective* telah dicapai. *Outcome* dalam *game* biasanya tolak ukur pen-

capaian dari *gameplay player* yang dapat berupa penambahan skor ataupun yang lainnya.

2.2 2D Platformer Game

Platform game merupakan *game* yang memiliki tujuan untuk mencapai titik akhir dengan melewati rintangan dan musuh di setiap *stage* [11]. Rintangan dan musuh yang ada akan berusaha untuk menghambat *player* untuk mencapai titik akhir dari permainan. Pada platform *game* juga terdapat sistem darah bagi karakter pemain maupun musuh. Selain itu, terdapat juga *point system* atau *item goal* yang perlu dicapai oleh *pemain* agar dapat mencapai titik akhir. *Game platformer* merupakan *video game* yang mempunyai *gameplay* membimbing karakter agar melompat di antara *platform*, *obstacle* untuk melanjutkan permainan. Pemain mengendalikan loncatan agar dapat menghindari karakter jatuh ataupun mati dari musuh. Unsur pemersatu yang paling umum dari genre ini adalah lompatan.

2.3 Educational Game

Educational games atau biasa dikenal sebagai *game* edukasi menggunakan *interface* dan *scenario* yang mirip dengan *digital entertainment* untuk memotivasi murid-murid dalam meningkatkan minat belajar [12]. Beberapa contoh *game* edukasi untuk *programming* berfokus pada konsep *programming* yang spesifik. Contoh dari latihannya adalah bagaimana cara menyelesaikan segmen kode, mencocokkan potongan kode, dan bukan semata mengembangkan program komplit dari awal. *Game* edukasi yang telah dibuat ini diyakini dapat meningkatkan faktor intrinsik dalam pemainnya agar dapat memahami konsep *programming* dengan cara yang lebih menyenangkan dan efisien.

2.4 Object-oriented Programming

Object-oriented programming (OOP) merupakan bahasa pemrograman yang didesain disekitar objek dan informasi aktual [13]. *Object-oriented programming* merupakan sebuah pendekatan yang secara utama memfokuskan pada bagaimana objek berinteraksi untuk berkomunikasi dan berbagi informasi. Konsep ini berkebalikan dengan konsep prosedural *programming* dimana fokus pada prosedural adalah eksekusi proseduralnya.

Object oriented programming (OOP) merupakan mata kuliah yang secara fundamental dibutuhkan dan wajib dikuasai oleh industri IT terutama bagi mahasiswa jurusan Informatika (IT) [14]. Contoh bahasa pemrograman yang menganut konsep OOP salah satunya adalah *Java*. Pemrograman *Java* tidak cukup sebatas memahami teknik pemrograman namun dari segi konsep pilar dasar dari OOP juga perlu diperhatikan. OOP terdiri atas *object*, *class*, *method* dan 4 pilar yaitu *Inheritance*, *Encapsulation*, *Abstraction*, dan *Polymorphism*. Berikut merupakan penjelasan dari masing-masing pilar tersebut:

1. *Inheritance*

   Istilah *Inheritance* mengacu pada adopsi semua properti non-privat dan metode dari *superclass* oleh *subclass*. *Inheritance* merupakan cara membuat salinan kelas yang sudah ada sebagai titik awal bagi kelas yang lain. *Subclass* dalam *inheritance* juga disebut sebagai kelas turunan. *Inheritance* mengacu pada konsep *reusability code*. Idealnya, superclass ditulis pada *level* paling umum. Subclass dapat dibuat dari superclass dengan tujuan yang lebih spesifik. *Inheritance* juga diartikan sebagai mekanisme dimana objek memperoleh semua properti dan perilaku dari *superclass*. Tujuan utama dari *inheritance* ini adalah membangun kelas baru diatas kelas yang sudah ada [15].

2. *Encapsulation*

   Istiliah *Encapsulation* pada OOP merupakan konsep tentang pengikatan data atau *data binding* yang disatukan atau dikapsulkan menjadi satu unit data. *Encapsulation* mempermudahkan dalam *code readability* karena informasi yang diberikan tidak perlu dibaca secara rinci karena sudah satu kesatuan. Dijuluki dengan fitur *information-hiding mechanism*, fitur ini dapat menghilangkan akses publik ke atribut yang terdapat di dalam kapsul tersebut. Metode ini dapat mempermudah dalam mendefinisikan atribut yang dibaca dan diperbarui [16].

3. *Abstraction*

   Istilah *Abstraction* merupakan proses untuk menyembunyikan detail implementasi dan hanya sisi fungsionalitas (gambaran umum) saja yang disajikan. *Abstraction* memungkinkan agar melihat suatu object dalam bentuk yang lebih sederhana. Dengan *abstraction*, suatu sistem yang kompleks dapat dipandang sebagai kumpulan subsistem-subsistem yang lebih sederhana. Contohnya adalah seperti mobil yang terdiri atas komponen elektronik, sistem

mekanisme, sistem mengemudi dan lain-lainnya. Dengan *abstraction*, subsistem tersebut dapat dibuat lebih sederhana seperti subsistem kemudi, komponen, dan lain sebagainya [17].

4. *Polymorphism*
   Istilah *Polymorphism* merupakan konsep dimana suatu objek dapat memiliki bentuk yang bermacam-macam. Konsep ini memungkinkan agar suatu objek melakukan aksi atau tindakan yang secara prinsip sama namun proses yang berbeda. *Polymorphism* merupakan kemampuan *method* untuk bekerja dengan lebih dari satu tipe argumen. *Polymorphism* secara singkatnya merupakan istilah yang menyatakan objek dapat memiliki berbagai bentuk, sebagai *object* dari *class* sendiri atau *object* dari *superclassnya* [18].

2.5 Algoritma Fisher Yates

Algoritma *Fisher Yates* merupakan salah satu algoritma yang digunakan untuk melakukan proses pengacakan pada suatu himpunan terhingga. Sering disebut dengan *Knuth Shuffle* juga, algoritma *Fisher Yates* merupakan sebuah algoritma yang digunakan untuk memunculkan sebuah permutasi acak dari urutan tertentu. Hasil dari permutasi tersebut memiliki tingkat probabilitas yang sama. Algoritma *Fisher Yates* ini dibuat oleh dua orang yang bernama *Ronald Fisher* dan *Frank Yates*. Algoritma *Fisher Yates* juga memiliki kemampuan metode pengacakan kongruensi linear sebanyak 11,76% [19]. *Pseudocode* algoritma *Fisher Yates* yang diberikan untuk menghasilkan permutasi acak dari angka 1 – *N* terdapat pada Gambar 2.1:

```
To shuffle an array a of n elements (indices 0..n-1):
  for i from n - 1 downto 1 do
       j = random integer with 0 <= j <= i
       exchange a[j] and a[i]
```

Gambar 2.1. *Pseudocode* Algoritma *Fisher Yates*

Algoritma *Fisher Yates* menggunakan istilah *range*, *roll* dan *scratch*. *Range* adalah jumlah angka yang belum terpilih. *Roll* merupakan angka *random* yang terpilih. *Scratch* merupakan *list* angka yang belum terpilih. Tabel 2.1 merupakan contoh dari suatu kemungkinan penggunaan *Fisher Yates Shuffle* mengikuti *step* di atas.

Tabel 2.1. Simulasi Algoritma Fisher Yates

Range	Roll	Scratch	Result
		1 2 3 4 5	
1-5	4	1 2 3 5	4
1-4	2	1 5 3	2 4
1-3	1	3 5	1 2 4
1-2	1	5	3 1 2 4
			5 3 1 2 4

2.6 Game User Experience Satisfaction Scale (GUESS)

Game User Experience Satisfaction Scale (GUESS) merupakan sebuah *tool* yang digunakan dalam melakukan penilaian kepuasan terhadap *video game*. GUESS ini memiliki *tools* yang komprehensif untuk melakukan penilaian terhadap pengalaman bermain *video game* [20]. GUESS sudah digunakan secara luas pada berbagai ranah *game* seperti simulasi *healthcare*, *social interaction* dan lainnya. Model pertanyaan yang ditawarkan pada GUESS terdiri atas beberapa bagian yaitu:

1. *Usability/Playability* yang mengacu pada kemudahan *gameplay* dan *control*.
2. *Narratives* yang mengacu pada alur permainan berupa latar suasana dan cerita.
3. *Play Engrossment* yang mengacu pada sensasi pemain ketika bermain
4. *Enjoyment* yang mengacu pada tingkat kesenangan pemain.
5. *Creative Freedom* yang mengacu pada *game* dapat menambahkan ilmu dan kreativitas pada pemain.
6. *Audio Aesthetics* yang mengacu pada kepuasan pengguna terhadap suara *in-game*.
7. *Personal Gratification* yang mengacu pada tingkat keseriusan pemain dalam bermain *game*.
8. *Social Connectivity* yang mengacu pada keinginan untuk bermain dengan teman lainnya (hanya berlaku pada *game multiplayer*).

9. *Visual Aesthetics* yang mengacu pada tingkat kepuasan pengguna terhadap grafik yang disajikan dalam *game*.

Penilaian GUESS setiap kategori dihitung dengan menggunakan *rating scale* yang diperoleh dengan menghitung total point, kemudian dibagi dengan total maksimal dari jumlah responden dikalikan dengan point tertinggi. Nilai yang didapatkan tersebut dibagi menjadi beberapa tingkat kepuasan, yaitu:

1. Sangat tidak puas dengan nilai 0% sampai 14%.
2. Tidak puas dengan nilai 15% sampai 28%.
3. Agak tidak puas dengan nilai 29% sampai 42%.
4. Biasa saja / Netral dengan nilai 43% sampai 56%.
5. Agak puas dengan nilai 57% sampai 70%.
6. Puas dengan nilai 71% sampai 84%.
7. Sangat puas dengan nilai 85% sampai 100%.

GUESS menggunakan 55 pertanyaan yang digunakan untuk menguji tingkat kepuasan video *game*. Karena jumlah pertanyaan yang sangat banyak, maka GUESS-18 diciptakan dan digunakan untuk mengurangi jumlah soal yang dipertanyakan yaitu menjadi 18 total soal. Hal ini ditujukan untuk mengurangi waktu yang diperlukan responden dalam menjawab pertanyaan yang diberikan [20]. Berikut merupakan Tabel 2.2 model pertanyaan yang dicantumkan di dalam GUESS-18.

Tabel 2.2. Model Pertanyaan GUESS-18 Keebler

Abv	***Subscale Item***	***Statement***
U1	*Usability* 1	*I find the game controls to be straightforward*
U2	*Usability* 2	*I find the game's interface to be easy to navigate*
N1	*Narratives* 1	*I am captivated by the game's story from the beginning.*
N2	*Narratives* 2	*I enjoy the fantasy or story provided by the game.*
PE1	*Play Engrossment* 1	*I feel detached from the outside world while playing the game.*
PE2	*Play Engrossment* 2	*I do not care to check events that are happening in the real world during the game.*
E1	*Enjoyment* 1	*I think the game is fun.*
E2	*Enjoyment* 2	*I feel bored while playing the game (REVERSE CODE)*
CF1	*Creative Freedom* 1	*I feel the game allows me to be imaginative.*
CF2	*Creative Freedom* 2	*I feel creative while playing the game.*
AA1	*Audio Aesthetics* 1	*I enjoy the sound effects in the game.*
AA2	*Audio Aesthetics* 2	*I feel the game's audio (e.g., sound effects, music) enhances my gaming experience.*
PG1	*Personal Gratification* 1	*I am very focused on my own performance while playing the game.*
PG2	*Personal Gratification* 2	*I want to do as well as possible during the game.*
SC1	*Social Connectivity* 1	*I find the game supports social interaction (e.g., chat) between players.*
SC2	*Social Connectivity* 2	*I like to play this game with other players.*
VA1	*Visual Aesthetics* 1	*I enjoy the game's graphics.*
VA2	*Visual Aesthetics* 2	*I think the game is visually appealing.*