BAB 2

LANDASAN TEORI

2.1 Tinjauan Teori

2.1.1 Video Game

Video game atau *game* adalah sebuah hiburan interaktif berbentuk digital yang dimainkan oleh pemain melalui komputer, konsol *game*, *smartphone,* atau *tablet* [10]. Sementara pengertian lain yang lebih lengkap, *video game* atau game adalah sebuah alat dengan desain tujuan, seperti harus mengalahkan musuh atau memenangkan sesuatu, yang memerlukan interaksi dengan lingkungan, baik *virtual* maupun *real*, dan dapat memasukkan simulasi dari unsur-unsur yang dapat ditemui dalam dunia nyata, seperti gravitasi dan momentum, tetapi tidak dibatasi oleh parameter yang berlaku dalam dunia nyata [11].

2.1.2 Dungeon Game

Dungeon-crawling game atau yang lebih dikenal dengan *dungeon game* adalah sebuah *genre* turunan yang berasal dari *genre role-playing game* (RPG). *Dungeon game* berbeda dari RPG karena agak mengesampingkan narasi serta modifikasi karakter untuk lebih fokus pada eksplorasi dan pertarungan yang serba cepat, serta mengumpulkan berbagai benda dalam lingkungan tertutup yang berbentuk seperti labirin [3]. Tujuan akhir dari *game* dengan *genre* seperti ini di antaranya untuk mengalahkan semua musuh, mengambil berbagai benda yang tersebar, menyelesaikan teka-teki, mencari jalan keluar, dan lain sebagainya. Perkembangan *dungeon game* banyak dipengaruhi oleh permainan *Dungeons and Dragons* (DND) yang diciptakan oleh Gary Gygax dan Dave Arneson pada tahun 1974, khususnya dalam segi pertarungan dan perkembangan karakter [12].

2.1.3 Non-player Character (NPC)

Mackay pada tahun 2001 menyatakan bahwa *non-player character* (NPC) adalah karakter yang dikendalikan oleh komputer, bukan oleh pemain. Istilah ini sebenarnya sudah ada sejak sebelum *game* komputer ada, tepatnya sering ditemukan pada *tabletop* RPG, dimana karakter tersebut dikendalikan oleh *Dungeon Master* (DM). Istilah NPC ini tidak hanya ditemukan dalam *genre* RPG saja, tetapi di banyak *genre* lainnya pula, dengan membawa banyak konsep dari *tabletop* RPG yang bersifat analog ke dalam bentuk digital [13].

2.1.4 Algoritma A*

Algoritma A* adalah algoritma untuk mencari jalan tersingkat. Algoritma ini mengalisis *input*, mengevaluasi jalur mana saja yang bisa dilewati, dan menghasilkan solusi dari pertimbangan tersebut [14]. Algoritma A* pertama kali dikembangkan pada tahun 1968 oleh Peter Hart, Nils Nilsson, dan Bertram Raphael yang dinamakan "Algoritma A". Setelah itu dilakukan optimisasi heuristik dan nama algoritma ini berubah menjadi "Algoritma A*" [15]. Algoritma A* menggabungkan informasi yang digunakkan oleh algoritma Djikstra, yaitu untuk memilih *vertices/nodes* yang paling dekat dengan simpul awal, dan informasi yang digunakkan oleh *Greedy Best First Search*, yaitu memilih *vertices/nodes* yang paling dekat dengan simpul akhir [4]. Perhitungan pada algoritma A* dapat ditentukan sebagai berikut [15].

$$F(x) = G(x) + H(x) \quad (1)$$

Dimana:

a) G(x) adalah nilai pada pergerakan simpul awal menuju simpul berikutnya.
b) H(x) adalah perkiraan nilai pergerakan simpul awal menuju tujuan akhir simpul. Fungsi ini biasanya disebut fungsi heuristik karena perhitungan tersebut berdasarkan perkiraan.
c) F(x) adalah jumlah dari fungsi G(x) dan H(x) dengan nilai terkecil F(x) adalah jalur terpendek menuju tujuan akhir.

Dengan menggunakan rumus beserta fungsi heuristik dalam implementasinya, *pseudocode* dari algoritma A* dapat dilihat pada Gambar 2.1 [16].

```
Initialise
        open_list = {start}
        closed_list = { }

while open_list is not empty
loop
    m = node in open_list with lowest f_cost
    remove m from open_list
    add m to closed_list

    if m is the target node
        return

    n = neighbour
    foreach n of m
        if n is not traversable or in closed_list
            skip to next n

        if path to n is shorter or n is not in open_list
            set f_cost of n
            set parent of n to m
            if n is not in open_list
                add n to open_list
```

Gambar 2.1 *Pseudocode* Algoritma A*

Untuk menentukan nilai heuristik, diperlukan suatu cara atau metode. Ada beberapa cara atau metode yang tersedia, salah satunya yang umum digunakan adalah *Manhattan distance*. *Manhattan distance* dipilih karena cocok pada pergerakan objek pada *game* 2D yang mengaplikasikan *grid* berbentuk persegi empat sebagai dasar pembuatan peta [17]. *Pseudocode* dari *Manhattan distance* dapat dilihat pada Gambar 2.2.

```
function heuristic(node) =
        dx = abs(node.x – goal.x)
        dy = abs(node.y – goal.y)
        return (dx + dy)
```

Gambar 2.2 *Pseudocode Manhattan Distance*

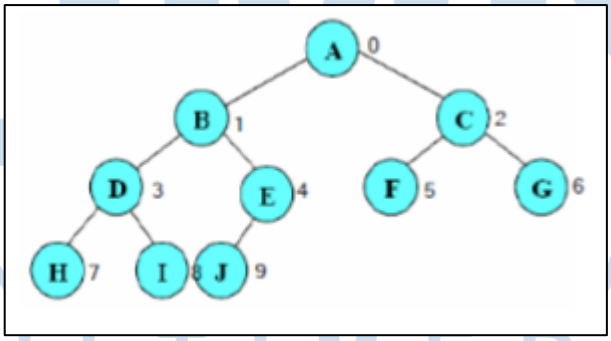
Dimana:

a) node.x adalah koordinat x target
b) node.y adalah koordinat y target
c) goal.x adalah koordinat x tujuan
d) goal.y adalah koordinat y tujuan

2.1.5 Heap Sort

Heap sort adalah sebuah metode *sorting* atau pengurutan yang mempunyai memiliki kompleksitas waktu paling baik. *Heap sort* mengaplikasikan teknik unik dalam menyelesaikan masalah pengurutan, yaitu dengan menggunakan *heaptree* [8]. *Heaptree* adalah sebuah pohon biner yang mempunyai beberapa persyaratan sebagai berikut [17].

a. Struktur pohon harus lengkap kecuali untuk level paling bawah, dengan syarat semua *node* pada level paling bawah harus berada di sebelah kiri.
b. Struktur pohon harus memenuhi properti *heap* sebagai berikut.
   - *Node* akar (*root node*) memiliki data terbesar atau terkecil yang terdapat dalam *tree.*
   - *Subtree* pada sisi kiri dan sisi kanan dari *node* akar harus memenuhi persyaratan, yaitu *node* induk (*parent*) memiliki data yang paling besar atau paling kecil dibandingkan dengan *node* anak (*child*) pada kedua sisi.



Gambar 2.3 *Heaptree*

Cara kerja dari *heap sort* adalah dengan mengurutkan sekumpulan data pada sebuah *heaptree* seperti pada Gambar 2.3. *Heap sort* akan mengambil data pada *node* akar (*index array* = 1) dan menukarnya dengan data pada *node* paling

akhir (*index array* = *index* maksimum dari *heaptree*). Setelah itu *node* terakhir akan dihapus dan *heap sort* memanggil fungsi untuk membangun ulang *heap*.

2.1.6 Game User Experience Satisfaction Scale-18 (GUESS-18)

Dalam pembuatan suatu *video game* diperlukan sebuah alat ukur untuk menilai kepuasan pemain. Salah satu alat ukur tersebut adalah *Game User Experience Satisfaction Scale* (GUESS). GUESS adalah sebuah alat ukur yang menilai sembilan pilar utama yang menggambarkan kepuasan terhadap suatu *video game* [9]. Sembilan pilar utama tersebut adalah *usability/playability, narratives, play engrossment, enjoyment, creative freedom, audio aesthetics, personal gratification, social connectivity,* dan *visual aesthetics* [19]. Penjelasan setiap pilar dapat dilihat dalam Tabel 2.1.

Tabel 2.1 Sembilan Pilar yang Menggambarkan Kepuasan Terhadap *Video Game*

Pilar	Deskripsi
Usability/Playability	Kemudahan memainkan *game* dengan tujuan akhir yang jelas tanpa gangguan dari *user interface* dan kontrol permainan.
Narratives	Aspek cerita dalam *game* serta kemampuan untuk menangkap minat dan membentuk emosi pemain.
Play Engrossment	Sejauh apa *game* dapat menarik perhatian dan minat pemain.
Enjoyment	Kesenangan dan kegembiraan yang dirasakan oleh pemain setelah bermain.
Creative Freedom	Sejauh mana *game* dapat menumbuhkan kreativitas dan rasa ingin tahu pemain dan memungkinkan pemain untuk secara bebas mengekspresikan individualitas mereka saat bermain.
Audio Aesthetics	Aspek suara dalam *game* dan seberapa berpengaruhnya dalam meningkatkan pengalaman bermain.

Tabel 2.1 Sembilan Pilar yang Menggambarkan Kepuasan Terhadap *Video Game* (Lanjutan)

Personal Gratification	Aspek motivasional dalam *game* yang meningkatkan rasa pencapaian dan keinginan untuk berhasil dari pemain dan melanjutkan permainan.
Social Connectivity	Sejauh mana *game* memfasilitasi hubungan sosial di antara pemain melalui fitur yang ada.
Visual Aesthetics	Grafik dari *game* dan seberapa menariknya mereka bagi pemain.

GUESS secara utuh memiliki 55 pertanyaan yang harus dijawab dan rata-rata menghabiskan waktu 10-15 menit untuk diselesaikan. Hal ini dapat dipercepat dengan menggunakan versi lebih singkat yang diberi nama GUESS-18. GUESS-18 hanya berisi 18 pertanyaan yang harus dijawab dan rata-rata hanya menghabiskan waktu 3-5 menit untuk diselesaikan [9]. GUESS-18 cocok untuk penilaian *game* yang harus diselesaikan secara cepat dengan tetap mencakup 9 pilar utama dari GUESS.

Pertanyaan yang telah dijawab akan dihitung nilai rata-ratanya berdasarkan nilai tiap jawaban yang didasarkan pada skala likert 7 poin, dimana nilai 1 berarti sangat tidak setuju dan nilai 7 berarti sangat setuju [9]. Perhitungan dapat ditentukan sebagai berikut.

$$Rata-rata(\%) = \big((Sangat\ Tidak\ Setuju * 1) + (Tidak\ Setuju * 2) + (Cukup\ Tidak\ Setuju * 3) + (Netral * 4) + (Cukup\ Setuju * 5) + (Setuju * 6) + (Sangat\ Setuju * 7)\big) / (Jumlah\ Responden * Skala\ Tertinggi) * 100\% \quad (2)$$