BAB 2
TINJAUAN PUSTAKA

2.1 *Game*

Game atau dalam bahas Indonesia permainan adalah sesuatu yang ditempatkan dalam dunia buatan yang sudah diatur melalui ketentuan – ketentuan (*rules*) yang sudah ditetapkan. Ketentuan tersebut digunakan untuk menentukan semua aksi dan tindakan yang harus dilakukan dan yang tidak oleh pemain dalam permainan. *Game* termasuk dalam bentuk interaktif, partisifatif dan hiburan. Maksudnya orang yang bermain *game* akan terhibur dengan ikut berpartisipasi kedalam *game*nya secara aktif. *Game* dikelompokan kedalam beberapa macam, yaitu berdasarkan dimensi, *genre*, dan *platform* yang digunakan.

1. *Game* berdasarkan dimensi

Dimensi adalah ukuran yang terdiri dari panjang dan sebagainya atau suatu angka-angka yang berhubungan sifat metrik dari suatu objek. Berdasarkan objek dari suatu dimensi game terbagi kedalam beberapa macam, yaitu :

a. *Game* 2D

Game dua dimensi merupakan game yang objeknya hanya pada satu bidang datar saja atau dua sisi. Pergerakan game 2D hanya dibatasi dengan menggunakan koordinat pada sumbu X dan Y. baik itu bergerak secara vertikal, horizontal ataupun diagonal. Pergerakan kamera yang digunakan adalah kamera statis dan *side scrolling*. Pergerakan kamera statis yaitu gambar lartar untuk tempat game tidak bergerak. Dan *side scrolling* yaitu memiliki kamera yang bergerak mengikut karakter yang dimainkan.

b. *Game* 2.5D

Game 2.5D disebut juga 3D datar(3D *plane*) yaitu *game* dengan tampilan yang sudah termasuk 3D tapi tidak sepenuhnya 3D. *Gameplay* ini pergerakannya sama dengan 2D yaitu vertikal dan horizontal tapi terdapat

beberapa objek yangs sudah memakai teknik pembangunan gambarnya atau *rendering* 3D.

c. *Game* 3D

Game tiga dimensi adalah *game* ruang objeknya memiliki tiga sisi yaitu sumbu X, Y dan Z. pada *game* 3D dapat melihat objek dari semua sisi atau dilihat dari sudut 360 derajat.

2. *Game* berdasarkan *genre* yang dimainkan.

Game berdasarkan *genre* adalah *game* yang dikelompokan bedasarkan interaksi pada *game*nya,atau berdasarkan perbedaan model tantangan dan aturan dari *game*nya. Secara umum *genre game* dibagi menjadi beberapa macam, yaitu :

a. *Strategy Games*

Game strategi yang asal usulnya berasal dari permainan papan seperti othello dan catur. Dalam memainkanya terdapat strategi yang harus dipikir untuk menghadap tantangan pada *game*nya. Biasanya karakter yang dimainkan bisa lebih dari satu karakter. *Game* strategi juga terbagi dalam dua macam yaitu *real-time strategy* dan *classical turn-based strategy*.

b. *Puzzle Games*

Game teka-teki adalah *game* yang dimainkan dengan tujuan untuk memecahkan permasalahan atau menemukan jawaban. Pada game teka-teki biasanya diberi aturan atau petunjuk-petunjuk yang dapat memudahkan dalam mendapat solusi dari permasalahan pada *game*nya.

c. *Adventure Games*

Game petualangan adalah *game* yang memiliki cerita interaktif mengenai karakter yang dikendalikan pemain. Biasanya pada *game* ini terdapat tempat yang harus dilewati oleh pemain. Dari tempat yang dilewatinya pemain melakukan navigasi ke suatu tempat, melawan musuh, mencari pesan tersembunyi dan lain-lain.

d. *Action Games*

Game aksi adalah *game* yang fitur utamanya banyak melakukan aksi, pemain harus mempunyai keterampilan reaksi dan respon yang cepat untuk melawan atau menghindari serangan lawan.

e. *Board Games*

Board games atau permainan papan merupakan salah satu jenis permainan yang dimainkan dengan menggunakan alas berbentuk persegi sebagai tempat untuk bermainnya. *Board games* adalah permainan yang menggunakan papan atau karton tebal sebagai komponen utamanya, dan terdapat komponen pendukungnya berupa koin, bidak, kertas, kartu dan sebagainya. Board games dapat dimainkan oleh lebih dari satu orang atau pemain dalam tempat yang sama (*Multiplayer*). Sehingga pemain dapat berinteraksi langsung dengan pemain lainya.

3. *Game* bedasarkan *platform* yang digunakan.

Platform adalah kombinasi dari perangkat lunak (*software*) dan perangkat keras (*hardware*) yang digunakan dalam mengoperasikan *game*. Berdasarkan *platform* yang digunakan *game* dibagi kedalam beberapa macam, yaitu:

a. *Arcade Games,* merupakan *game* yang mempunyai mesin khusus yang dirancang untuk *video games* tertentu. Terdapat alat atau fitur yang membuat bermain serasa lebih nyata seperti terdapat stir mobil, pistol, sensor gerakan dan kursi khusus.

b. *PC Games,* merupakan *game* yang dimainkan menggunakan perangkat komputer.

c. *Console Games,* merupakan *game* yang menggunakan konsol tertentu dalam memainkannya. Seperti playstation, Nintendo, Xbox dan sebagainya.

d. *Handheld Games,* merupakan *game* yang menggunakan konsol khusus *game* yang dapat dibawa kemana-mana atau disebut juga *portable* dalam memainkannya. Seperti *Sony* PSP.

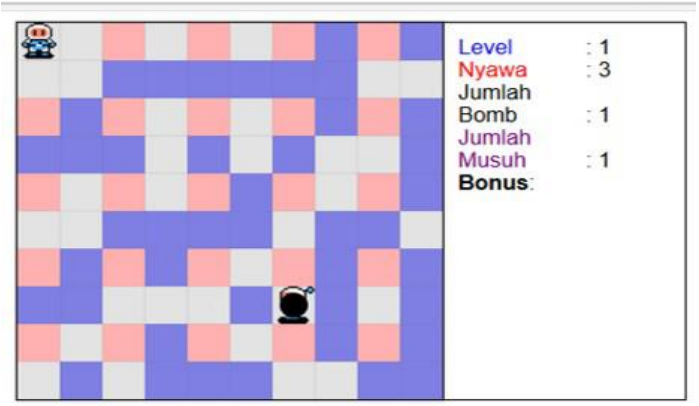
e. *Mobile Games*, merupakan *game* dimainkan pada *mobile phone*, *smartphone* atau PDA [6].

2.2 Bomberman

Game Bomberman diproduksi tahun 1985. Game bomberman merupakan game strategi berbasis labirin yang pertama kali diproduksi oleh Hudson Soft untuk Nintendo Entertainment System (NES) pada game 8-Bit yang diperuntukan bagi console Nintendo.

Game ini memiliki cara bermain yaitu bergerak ke empat arah (atas, bawah, kanan, kiri) dan mengeluarkan bom yang kemudian meledak dalam ledakan berbentuk silang (menciptakan aliran api vertikal dan horizontal) setelah waktu yang ditentukan (waktu meledaknya bom selama kurang lebih 3 detik) untuk membunuh atau mengalahkan NPC.

Bom dapat menghancurkan musuh serta meledakkan tembok penghalang untuk membuka jalur baru dan temukan power-up. Power-up menambah kemampuan tertentu untuk Bomberman, misalnya karena meningkatkan ukuran ledakan, meningkatkan jumlah bom yang bisa dijatuhkan Bomberman sekaligus, meningkatkan kecepatan Bomberman atau berjalan dari satu tempat ke tempat lain. Setiap jalan akan terbuka ketika meledakkan bata. Namun, selain menghancurkan bata pemain pun harus membunuh NPC untuk mendapat kemenangan. [1]. Untuk lebih jelas berikut adalah tampilan pada permainan Bomberman yang sering dimainkan pada gambar 2.1.



Gambar 2. 1 Tampilan Game Bomberman

2.3 *Artificial Intelligence*

Artificial Intelligence atau dalam bahasa indonesia disebut kecerdasan buatan didefinisikan berbeda-beda oleh para ahli tergantung pada sudut pandangnya masing-masing. Ada yang didefinisikan logika berpikir manusia saja, ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Menurut Stuart Russel dan Peter Norvig, kecerdasan buatan merupakan perangkat komputer yang bisa memahami lingkungannya dan bisa mengambil tindakan yang memaksimalkan keberhasilan di lingkungan tersebut untuk beberapa tujuan. pendekatan AI dikelompokan kedalam empat kategori yaitu :

1. *Thinking humanly*

Pendifinisian AI merupakan cara berpikir yang menyerupai berpikir manusia. Pendekatan ini dilakukan dengan dua cara yaitu melalui introspeksi maksudnya mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir dan melalui eksperimen-eksperimen psikologi.

2. Acting humanly

Acting humancly melakukan pendekatan dengan menirukan prilaku seperti manusia yang diperkenalkan pada tahun 1950 oleh Alan turing dalam merancang cara kerja pengujian melalui teletype yaitu jika penguji (integrator) tidak dapat membedakan yang mengintrogasi antara manusia dan komputer maka komputer tersebut dikatakan lolosdari turning test.

3. Thinking rationally

Ada dua masalah dalam pendekatan *thinking rationally*, yaitu pertama sulit dalam membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam formal term yang diperlukan oleh notasi logika. dan kedua terdapat perbedaan besar antara dapat memecahkan masalah dalam prinsip dan memecahkannya dalam dunia nyata.

4. *Acting rationally*

Pendekatan *acting rationally* memiliki tujuan untuk mengembangkan agen yang rasional. Dimana agen adalah sistem komputer yang beroperasi secara otomatis, yang mampu mempersepsikan lingkungan disekitarnya, kemudian dapat

beradaptasi terhadap perubahan, dan mampu membuat tujuan sekaligus berusaha untuk mencapainya [8].

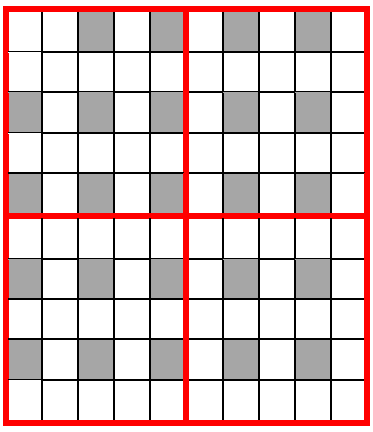
2.4 Algoritma *Hierarcichal Pathfinding A*(HPA*)*

Hierarchical Path-Finding A*(HPA*) merupakan perbaikan dari metode A*. Dalam permasalahan path-finding secara realtime untuk suatu permainan, komputer akan menggunakan banyak resource CPU dan memori, terutama jika ukuran daerah cukup besar. Dengan metode HPA*, sebuah peta daerah dimodelkan menjadi local cluster yang terhubung. Pada level local, jarak optimal untuk menyebrangi setiap cluster sudah dihitung terlebih dulu dan disimpan pada cache. Pada level global, semua cluster ditelusuri secara lengkap dan ditelusuri secara menyeluruh.

Pada HPA* agen menentukan langkah berikutnya dengan cara melihat titik *entrance* terdekat pada posisi agen saat itu. Saat agen berhenti di titik *entrance*, akan dicari titik *entrance* berikutnya yang terdekat dengan tujuan. Jika jarak agen dengan tujuan sudah dekat, agen langsung menuju ke tujuan, mengabaikan *entrance* untuk sementara [3].

Berikut ini merupakan langkah untuk pencarian yang dilakukan oleh algoritma *Hierarchical Pathfinding A** :

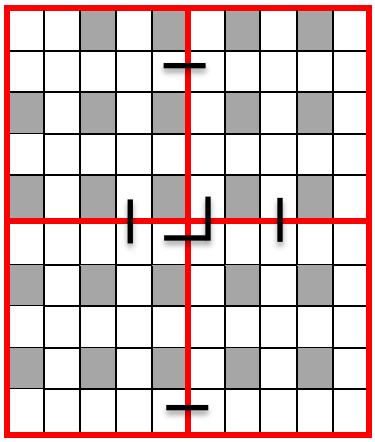
1. Lakukan clustering terhadap map atau daerah path yang akan ditelusuri seperti gambar 2.2 .



Gambar 2. 2 Clustering map game Bomberman

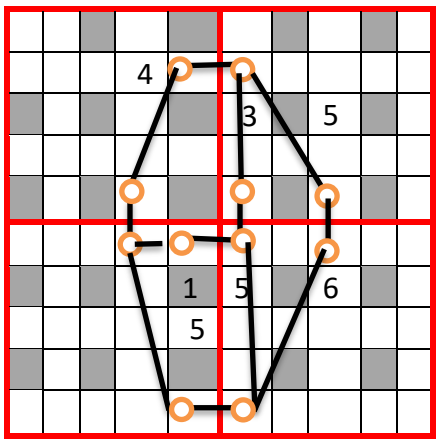
Pada gambar dijelakan bahwa map bomberman yang berukuran 10x10 telah diperkecil menjadi 2x2 melalui proses clustering dan setiap cluster memiliki ukuran map sebesar 5x5 [1].

2. Lakukan pemberian node terhadap setiap cluster seperti gambar 2.3 .



Gambar 2. 3 Pemberian node pada setiap Cluster

Selanjutnya setelah pembentukan cluster maka dibuatlah node untuk menghubungkan setiap cluster. Nilai jarak dari setiap node atau graph yang ada pada cluster dilihat dari berapa kotak yang dilewati dari node awal ke node tujuan seperti gambar 2.4 .



Gambar 2. 4 Posisi node pada setiap Cluster dengan nilai graph

3. Mencari jarak yang terdekat dari start menuju goal

   Untuk mencari jalan atau rute terpendek maka dicari dengan menelusuri semua kemungkinan graph yang bisa di lalui lalu kemudian memilih jalur mana yang paling dekat menuju ke target atau goal. Cara mencari jalur yaitu dengan melihat nilai graph yang paling kecil dari node awal ke node tujuan dari satu cluster ke cluster lain sampai menemukan node target lalu menjumlahkannya. Setelah didapatkan nilai jarak, maka jarak terkecilah yang akan menjadi langkah dari NPC.

   Ataupun untuk pencarian rute bisa menggunakan ***Manhattan Distance***. Perhitungan nilai heuristik untuk simpul ke-*n* dalam *HPA** dapat menggunakan perhitungan *Manhattan Distance* yang rumusnya adalah sebagai berikut :

$$h = \sum_{i=1}^{n} |(xi - xj)| + |(yi - yj)| \quad \text{.....(2.1)}$$

   Dengan :

   h = perkiraan jarak terdekat atau terkecil dari simpul NPC ke simpul Player

   xi = posisi X simpul posisi awal

   xj = posisi X simpul posisi tujuan

   yi = posisi Y simpul posisi awal

   yj = posisi Y simpul posisi tujuan

Kemudian dimasukan nilai perhitungan A* yaitu :

$$f = g + h \quad \text{.....(2.2)}$$

dimana :

f = jarak rute terpendek NPC ke player

g = nilai awal dari langkah yaitu 0

h = nilai heuristic rute hasil *Manhattan Distance*

Untuk nilai g pada saat pencarian rute dilihat dari nilai pada array ke tujuan node selanjutnya yaitu nilai 1 dan 2 (jika nilai 0 maka diabaikan). Untuk menghitung nilai g maka di cek jumlah nilai dari node awal ke tujuan :

$$g = jumlah\ nilai\ node * 20$$

Maka nilai penelusuran node menjadi :

$$f = (g)\ jumlah\ nilai\ node + (h)\ jarak\ hasil\ manhattan\ ke\ tujuan$$

setelah didapat nilai f maka selanjutnya dilakukan penelusuran node untuk mencari rute terdekat mana yang memiliki nilai terkecil atau sama dengan f dan memilih nilai jumlah node dengan nilai f sebagai tumpuan nilai jarak terpendek.

2.5 Algoritma *Fuzzy Sugeno*

Logika *fuzzy Sugeno* merupakan Algoritma yang biasa digunakan untuk pengambilan keputusan. Algoritma ini pertama kali diperkenalkan oleh Michio Sugeno dan memiliki derajat keanggotaan dalam rentang 0 (nol) hingga 1(satu) [4].

Proses *fuzzy inference* dalam Fuzzy Sugeno dapat dibagi dalam 4 bagian, yaitu :

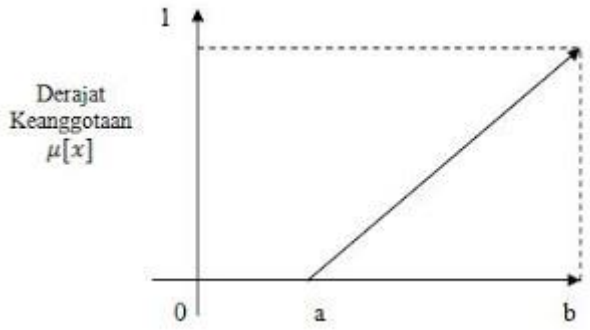
2.5.1 Fuzzyfikasi Input

Yaitu mengambil masukan-masukan dan menentukan derajat keanggotaannya dalam semua *fuzzy* set. Pada proses Fuzzyfikasi diperlukan representasi linear. Dan pada penelitian kali ini digunakan tiga representasi linear yaitu : representasi linear naik, representasi linear turun, dan representasi linear segitiga [4].

1. Representasi linear naik

Kenaikan himpunan dimulai pada nilai *domain* yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju ke nilai *domain* yang memiliki derajat keanggotaan lebih tinggi yang disebut dengan representasi fungsi linear naik [4].

Representasi fungsi keanggotaan untuk linear naik adalah seperti gambar 2.5 berikut :



Gambar 2. 5 Representasi linear naik

Representasi linear naik memiliki rumus sebagai berikut :

$$\mu[x,a,b] = \begin{cases} 0; & x \leq a \\ \frac{(x-a)}{(b-a)}; & a \leq x \leq b \\ 1; & x \geq b \end{cases} \quad \text{.....(2.3)}$$

Dan untuk derajat keanggotaan memiliki rumus sebagai berikut :

$$\mu[x] = \frac{(x-a)}{(b-a)} \quad \text{.....(2.4)}$$

Dimana :

a = nilai domain yang mempunyai derajat keanggotaan nol

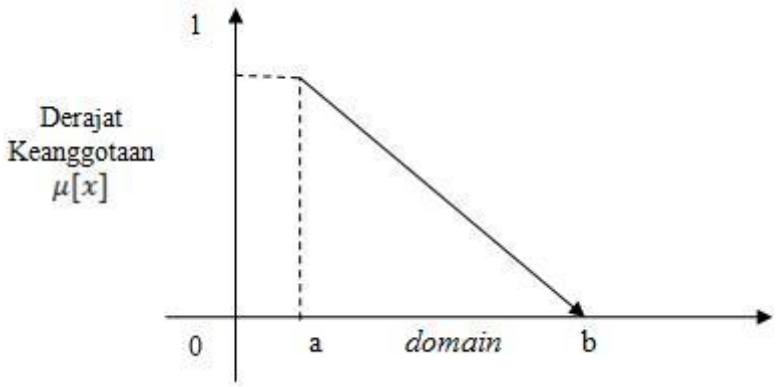
b = nilai domain yang mempunyai derajat keanggotaan satu

x = nilai input yang akan di ubah ke dalam bilangan *fuzzy*

2. Representasi linear turun

Kenaikan himpunan dimulai pada nilai *domain* yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju ke nilai *domain* yang memiliki derajat keanggotaan lebih tinggi yang disebut dengan representasi fungsi linear naik [4].

Representasi fungsi keanggotaan untuk linear turun adalah seperti gambar 2.6 berikut :



Gambar 2. 6 Representasi linear turun

Representasi linear turun memiliki rumus sebagai berikut :

$$\mu[x,a,b] = \begin{cases} \frac{(b-x)}{(b-a)}; & a \leq x \leq b \\ 0; & x \geq b \end{cases} \quad \text{.....(2.5)}$$

Dan untuk derajat keanggotaan memiliki rumus sebagai berikut :

$$\mu[x] = \frac{(b-x)}{(b-a)} \quad \text{.....(2.6)}$$

Dimana :

a = nilai domain yang mempunyai derajat keanggotaan satu

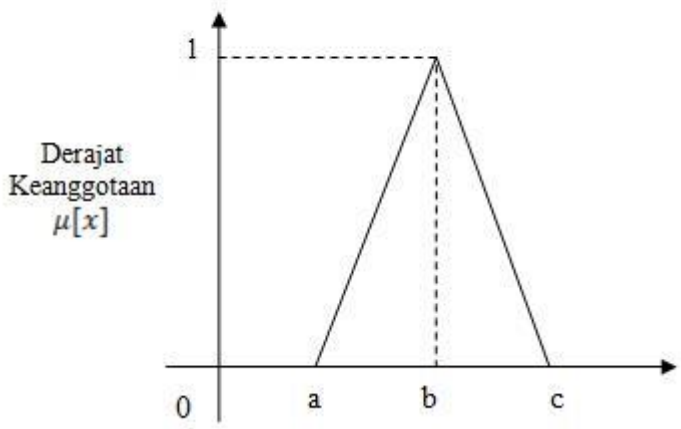
b = nilai domain yang mempunyai derajat keanggotaan nol

x = nilai input yang akan di ubah ke dalam bilangan *fuzzy*

3. Representasi linear segitiga

Represetasi linear Segitiga, pemetaan *input* ke derajat keanggotaannya digambarkan dengan bentuk segitiga dimana pada dasarnya bentuk segitiga tersebut gabungan antara 2 garis (linear). Nilai-nilai di sekitar b memiliki derajat keanggotaan turun yang cukup tajam (menjahui 1) [4].

Representasi fungsi keanggotaan untuk linear segitiga adalah seperti gambar 2.7 berikut:



Gambar 2. 7 Representasi segitiga

Representasi linear segitiga memiliki rumus sebagai berikut :

$$\mu[x,a,b] = \begin{cases} 0; & x \leq a \; atau \; x \geq c \\ \frac{(x-a)}{(b-a)}; & a \leq x \leq b \\ \frac{(c-x)}{(c-b)}; & b \leq x \leq c \end{cases} \quad \text{.....(2.7)}$$

Dan untuk derajat keanggotaan memiliki rumus sebagai berikut :

$$\mu[x] = \frac{(x-a)}{(b-a)} \text{ .....(2.8)}$$

Dimana :

a = nilai domain terkecil yang mempunyai derajat keanggotaan nol

b = nilai domain yang mempunyai derajat keanggotaan satu

c = nilai domain terbesar yang mempunyai derajat keanggotaan nol

2.5.2 Operasi logika Fuzzy (rule Fuzzy)

Hasil akhir dari operasi ini adalah derajat kebenaran *antecedent* yang berupa bilangan tunggal yang bertujuan untuk pemberian rule.

Proses pembentukan aturan memiliki rumus sebagai berikut :

IF *x1 is A1* ***AND*** *..Xn is An* ***THEN*** *y=f (X1,X2,..Xn)* .....(2.9)

Dimana x merupakan parameter input, A merupakan nilai dari parameter, f merupakan sembarang fungsi dari variabel-variabel masukan yang nilainya berada dalam interval variabel keluaran [4].

2.5.3 Mesin Inferensi

Merupakan proses mendapatkan *consequent* atau keluaran sebuah IF-THEN *rule* berdasarkan derajat kebenaran *antacedent*. Proses ini menggunakan mengambil nilai MIN/terkecil dari dua bilangan [4].

2.5.4 Defuzzyfikasi

Keluaran dari de*fuzzy*fikasi adalah sebuah bilangan tunggal [4]. Untuk menghitung nilai defuzyfikasi menggunakan rumus seperti berikut :

$$z = \frac{\Sigma ai\ Zi}{ai} \text{ .....(2.10)}$$

Dimana :

z = nilai tunggal hasil defuzyfikasi

α = nilai antecedent prediksi hasil mesin inferensi

Z = nilai dari perilaku yang dilakukan oleh NPC

2.6 Object Oriented Programming (OOP)

Pemrograman berorientasi objek merupakan suatu pengembangan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi di dalamnya. Pengembangan berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Pengembangan ini terdiri dari rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pemrograman berorientasi objek memiliki keuntungan sebagai berikut:

1. Pendekatan berorientasi objek dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut atau disebut penggunaan ulang (*reuse*) sehingga dapat meningkatkan produktivitas.
2. Pendekatan berorientasi objek mudah dalam pemeliharaan karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dengan pola-pola yang mungkin sering berubah-ubah.
3. Relatif cepat dalam pengembangannya karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean [8].

Pemrograman berorientasi objek dibagi menjadi beberapa ciri utama, berikut adalah ciri-cirinya :

1. Kelas

Kelas (*class*) adalah contoh abstrak dari sebuah objek yang telah terbentuk dari proses penyederhanaan, dengan kata lain kelas (class) adalah berasal dari objek (*object*), contoh nyata dari sebuah objek dinamakan *instance*. Maka apabila kita mempunyai sebuah kelas manusia, maka beberapa *instances* (wujud nyata) dari kelas manusia adalah Ary, Erik, Rangga dan masih banyak yang lainnya [8]. Perbedaan antara kelas (*class*) dengan objek (object) dalam OOP dibagi menjadi dua, yaitu :

a. *Class* adalah rancangan dan *object* adalah perwujudan dari suatu *class*.
b. *Class* bersifat abstrak sementara object bersifat konkrit (atau nyata).

2. Objek

Sebuah objek pada pemrograman berorientasi objek dapat didefinisikan sebagai berikut :

a. Setiap objek dimiliki oleh kelas objek, sehingga sebuah objek tidak bisa hadir tanpa sebuah kelas yang mendefinisikannya.
b. Sebuah objek adalah sebuah pengkapsulan (*encapsulation*) yang memasukkan data dan operasi untuk pemrosesannya.
c. Atribut-atribut objek membantu untuk menyimpan dan menjaga status objek. Atribut-atribut ini menentukan apa yang berkaitan mengenai objek. *Method* objek adalah satu-satunya cara untuk mengakses data dan

memodifikasi statusnya. Cara pengaksesan dan pemodifikasian data dilakukan dengan mengirimkan sebuah pesan ke objek tersebut[8].

Pemrograman berorientasi objek memiliki beberapa karakteristik utama, yaitu :

1. Abstrack

Abstraksi dapat diartikan sebagai suatu proses melakukan *desain class* dan menentukan data dan *method* yang akan dimiliki oleh sebuah kelas. Sebuah *method* abstrak mendefinisikan sebuah antarmuka dalam kelas dasar dan meninggalkan implementasi pada kelas turunan. Kelas abstrak adalah sebuah kelas yang berisi satu atau beberapa *method* abstrak [9].

2. *Encapsulation*

Dalam bahasa indonesia disebut pengkapsulan yaitu dasar untuk membatasi ruang linkup program pada data yang diproses. Data dan prosedure dikemas dalam satuan objek sehingga dapat terlindung dari objek luar [9].

3. *Inheritance*

Inheritance (pewarisan) diartikan konsep sebuah objek dapat diturunkan data dan prosedur yang dimilikinya ke anak dari objek tersebut. Pewarisan dapat dilakukan dari kelas induk ke kelas anaknya jika terdapat hubungan antar keduanya. Konsep ini dapat menyederhanakan dan mempermudah dalam program java [9].

4. *Polymorphism*

Polimorphism merupakan konsep dari sesuatu yang sama memiliki bentuk dan perilaku yang beda. Dapat diartikan juga penggunaan lebih dari satu metode dengan nama sama. Dengan konsep ini dapat menghindari duplikasi objek [9].

2.7 *Unified Modeling Language* (UML)

Unified modeling language (UML) merupakan model berorientasi objek yang digunakan untuk menggambarkan, menspesifikasi, mendokumentasi dari kontruksi sistem dan pembangunan sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukungAwal pengembangan UML yaitu

pada tahun 1994 dengan melakukan kerja sama antara Grady Booch dan James Rumbaugh dalam mengkobinasi dua metode booch dan OMT. Dan pada tahun 1997, UML di usulkan kepada OMG (*Object Management Group*) yaitu standarisasi teknologi objek supaya dijadikan notasi dan bahasa pemodelan.

Model UML bertujuan untuk :

1. Penyedia bahasa pemodelan grafis yang ekpresif yang siap pakai untuk pengembangan sistem.
2. Penyedia mekanisme dan spesifikasi mendapat konsep - konsep pada sistem
3. Penyedia basis formal untuk pemahaman pemodelan sistem.
4. Mendukung konsep-konsep pengembangan derajat yang lebih tinggi seperti framework, pattern, kolaborasi dan komponen.

UML mengekspresikan pemodelan yang berorientasi objek dalam bentuk diagram yang terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Berikut kategori dan macam-macam diagram tersebut :

1. Kategori *structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Diagram yang termasuk dalam kategori ini adalah *class diagram, objek diagram, component diagram, compenent struture diagram, package diagram* dan *deployment diagram*.
2. Kategori *behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Diagram yang termasuk dalam kategori ini adalah *use case diagram, activity diagram* dan *state machine diagram*.
3. Kategori *interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. Diagram yang termasuk kategori ini adalah *sequence diagram,communication digram, timing diagram* dan *interaction overview diagram* [9].

2.7.1 *Use case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. *use case* merepresentasikan hubungan antara aktor dengan sistem. *Use case* sebagai berbagai pekerjaan pada sistem sedangkan aktor sebagai sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu menyusun kebutuhan sebuah sistem, menjelaskan rancangan dengan klien.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. *Use case* dapat di-*include* oleh lebih dari satu use case lain, sehingga dapat terhindar dari duplikasi fungsionalitas. *Use case* juga dapat meng-*extend use case* lain dengan behaviournya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain [9].

2.7.2 *Activity Diagram*

Activity diagram menggambarkan alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, proses apa saja yang dilakukan, dan bagaimana alurnya berakhir. *Activity diagram* menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum [9].

2.7.3 *Class Diagram*

Class Diagram menggambarkan struktur dan deskripsi pada setiap *class*, *package* dan objek serta hubungan satu sama lain seperti agregasi, pewarisan, asosiasi, dan lain-lain [9].

2.7.4 *Sequence Diagram*

Sequence diagram menggambarkan hubungan antar objek yang berada di dalam sistem. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* digunakan untuk menggambarkan rangkaian langkah-langkah yang dilakukan sebagai respons dari

sebuah *event* untuk menghasilkan keluaran tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan keluaran apa yang dihasilkan [9].

2.8 Web Programming

HTML

Hypertext Markup Language (HTML) adalah bahasa markup yang umum digunakan untuk membuat halaman web. Sebenarnya HTML bukanlah sebuah bahasa pemrograman. Apabila di tinjau dari namanya, HTML merupakan bahasa markup atau penandaan terhadap sebuah dokumen teks. Tanda tersebut di gunakan untuk menentukan format atau style dari teks yang di tandai.

HTML dibuat oleh Tim Berners-Lee ketika masih bekerja untuk CERN dan dipopulerkan pertama kali oleh browser Mosaic. Selama awal tahun 1990 HTML mengalami perkembangan yang sangat pesat. Setiap pengembangan HTML pasti akan menambahkan kemampuan dan fasilitas yang lebih baik dari versi sebelumnya.

Sebelum suatu HTML disahkan sebagai suatu dokumen HTML standar, ia harus disetujui dulu oleh W3C untuk dievaluasi secara ketat. Setiap terjadi perkembangan suatu versi HTML, maka mau tak mau browser pun harus memperbaiki diri agar bisa mendukung kode-kode HTML yang baru tersebut. Sebab jika tidak, browser tak akan bisa menampilkan HTML tersebut [10].

Hypertext Processor (PHP)

Hypertext Preprocessor (PHP) adalah bahasa *server-side* scripting yang menyatu dengan HTML untuk membuat halaman web yang dinamis. PHP banyak dipakai untuk pemrograman situs WEB dinamis. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di server kemudian hasilnya dikirim ke browser dalam format HTML. Dengan emikian kode program yang ditulis dalam PHP tidak akan terlihat oleh user sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membentuk sutu

tampilan berdasarkan permintaan terkini, seperti menampilkann isi basis data ke halaman web.

Beberapa kelebihan PHP dari bahasa pemrograman web, antara lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunanya.
2. PHP memiliki tigkat akses yang lebih cepat.
3. PHP memiliki tingkat *lifecycle* yang cepat sehingga selalu mengikuti perkemangan teknologi internet.
4. PHP juga mendukung akses ke beberapa database yang sudah ada baik yang bersifat *free/*gratis ataupun komersial. Database itu antara lain : MySQL, PostgresSQL, infomix, dan MicrosoftSQL Server. Web server yang mendukung PHP dapat ditemukan dimana mana dari mulai Apache, IIS, AOserver, phttp. Fhttp. PWS, Lighttpd hingga Xitami dengan konfigurasi yang relative mudah.

Dalam sisi pengembangan lebih mudah, karena banyaknya milis-milis dan *developer* yang siap membantu dalam pengembangan [10].

Java Script

JavaScript adalah bahasa yang berbentuk kumpulan skrip berjalan pada suatu dokumen HTML. Bahasa ini adalah bahasa pemrograman untuk memberikan kemampuan tambahan terhadap HTML dengan mengizinkan pengeksekusian perintah-perintah disisi user variabel atau fungsi dengan nama TEST berbeda dengan variabel dengan nama test dan setiap instruksi diakhiri dengan artinya disisi browser bukan disisi server web. JavaScript adalah bahasa yang “case sensitive” artinya memnedakan penamaan variabel dan fungsi yang menggunakan huruf besar dan huruf kecil, contoh karakter titik koma [11].

JQuery

JQuery adalah library Javascript yang dibuat untuk memudahkan pembuatan website dengan HTML yang berjalan di sisi Client. JQuery diluncurkan pada tanggal 26 Januari 2006 di Barcamp NYC oleh John Resig dan Rancang Bangun

E – Voting Berbasis Websiteberlisensi ganda di bawah MIT dan GPL. Script JQuery dibuat untuk memudahkan pengaturan document seperti menyeleksi object dengan element DOM dan membuat aplikasi dengan AJAX. Jquery juga menyediakan layanan atau support para developers untuk membuat plug-ins di dalam bahasa Javascript tentunya. Sehingga memungkinkan para developer website membuat website lebih interaktif dengan animasi, efek – efek, tema dan widget.

JQuery juga adalah kumpulan kode JavaScript siap pakai. Keunggulan menggunakan jQuery dibandingkan dengan JavaScript standar, yaitu menyederhanakan kode JavaScript dengan cara memanggil fungsi-fungsi yang disediakan oleh jQuery. JavaScript sendiri merupakan bahasa Scripting yang bekerja disisi Client/Browser sehingga website bisa lebih interaktif.

jQuery adalah salah satu javascript framework terbaik saat ini. jQuery dikembangkan oleh John Resig pada tahun 2006 di BarCamp NYC. Pada awal perkembangannya, jQuery pertama dibuat untuk meringkas penggunaan CSS Selector dalam suatu pustaka fungsi. jQuery memiliki ciri khas pada penggunaan perintahnya, prefix untuk jQuery dengan tanda $ kemudian dilanjutkan dengan fungsi atau perintah [11].

2.9 Studi Literatur

Review literatur yang dilakukan adalah dengan cara membaca, menelaah, dan memahami dari beberapa jurnal sebagai berikut :

1. Pada Jurnal yang berjudul *"Beating Bomberman with Artificial Intelligent"* yang ditulis oleh Juarez Monteiro, dkk. Disimpulkan bahwa pada game Bomberman terdapat dua algoritma yang biasa digunakan untuk menggerakan karakter AI yaitu A* dan BFS yang diujikan kedalam game. Dari hasil perbandingan dari A* dan BFS didapatkan bahwa A* memiliki kinerja yang lebih efisien dari pada BFS. [1]
2. Pada jurnal yang berjudul *"Hierercichal Pathfinding and AI-Based Learning Approach in Strategy Game Design"* yang ditulis oleh Le Minh

Duc, dkk. Didapatkan bahwa konsep AI dapat diterapkan untuk game baik itu konsep algoritma sederhana maupun Algoritma dengan konsep tingkat tinggi. Pada penelitian ini game menjadi alat bantu untuk membandingkan tingkat efisiensi dari berbagai macam algoritma untuk nantinya dikombinasikan untuk mendapatkan game yang lebih optimal. Algoritma kombinasi ini biasanya digunakan untuk game yang biasanya memiliki ruang lingkup yang besar namun harus optimal, maka dikombinasikan beberapa algoritma untuk membantu menguji kinerja game yang dibangun.[2]

3. Pada jurnal yang berjudul *"Near Optimal Hierarchical Path-Finding"* yang ditulis oleh Andy Botea, dkk. Dijelaskan bahwa secara sistematis meninjau beberapa algoritma dan teknik berbasis A * populer sesuai dengan optimasi A *. Ini menunjukkan peta relasional yang jelas antara algoritma A * dan variannya. Inti dari algoritma pathfinding hanyalah sebagian kecil dari teka-teki dalam AI game. Tantangan paling banyak adalah bagaimana menggunakan algoritma untuk menyelesaikan masalah rumit. Algoritma A * adalah algoritma paling populer dalam penelusuran jalan karena A * terbukti optimal. Banyak upaya telah dilakukan untuk mempercepatnya mengoptimalkannya dari perspektif yang berbeda. Cara untuk meningkatkan kinerja A* pencarian termasuk mengoptimalkan ruang pencarian yang mendasarinya, mengurangi penggunaan memori, meningkatkan fungsi heuristik dan memperkenalkan struktur data baru. Penelitian potensial adalah untuk terus mengoptimalkan A * algoritma dari perspektif ini atau untuk menggabungkan beberapa teknik optimasi menjadi satu solusi tunggal. Cara lain untuk memberikan kontribusi pada komunitas game AI adalah dengan menerapkan teknik-teknik yang dijelaskan di atas untuk game komputer karena tidak semua teknik yang dijelaskan dalam makalah ini telah banyak digunakan dalam industri game saat ini.[3]
4. Pada jurnal yang berjudul *"Path Planning with Modified A Star Algorithm for a Mobile Robot"* yang ditulis oleh Frantisek Duchon, dkk.

Dijelaskan bahwa dalam hal penemuan jalur yang cepat, JPS atau Jump Point Search (A *) adalah algoritma terbaik di berbagai lingkungan. Hanya saja kekurangannya adalah bahwa jika menemukan jalur pencarian atau titik yang lebih panjang, seringkali lebih lama dan masih memakan memori yang cukup besar. Oleh karena itu, algoritma ini cocok untuk digunakan dalam kasus-kasus menemukan jalan atau rute yang pendek. Jika waktu komputasi tidak signifikan dan panjang lintasan sangat penting maka sangat cocok untuk menggunakan algoritma Basic Theta *. Kemampuan Basic Theta * ini telah disorot terutama di lingkungan dengan simetri yang lebih rendah. Dan perlu adanya pengembangan dan variasi turunan agar algoritma A* dapat lebih efisien.[5]

5. Pada jurnal yang berjudul *"Penerapan Algoritma Logika Fuzzy untuk Dynamic Difficulty Scalling pada Game Labirin"* yang ditulis oleh T. Mahardika Abdi Prawira. Dijelaskkan bahwa Penerapan algoritme fuzzy dilakukan dengan cara mengambil data input dari pemain yang telah menyelesaikan suatu level dari game labirin, yang di ambil sebagai input yaitu nilai dari waktu penyelesaian permainan (time resolved), total score yang di dapat (score), dan sisa health point dari pemain (hp). Pedoman yang dipakai dalam menentukan kriteria fungsi keanggotaan adalah nilai dari total jalan atau path yang dapat diambil (pathCount), total dari coin yang bernilai positif (coinPlus), dan range area dari level labirin (range). Dari ketiga inputan tersebut dilakukan proses fuzzyfikasi untuk menggolongkan fungsi keanggotaan dari setiap variabel. Dari setiap fungsi keanggotaan yang diperoleh kemudian dimasukkan pada setiap rule base dari knowledge base untuk proses implikasi dengan menggunakan metode MIN atau penilaian terkecil, hasil tersebut kemudian dilakukan proses defuzzyfikasi untuk mendapatkan nilai keluaran dari perhitungan fuzzy, hasil dari fuzzy tersebut yang dijadikan sebagai tolok ukur sebagai penentu level pada level selanjutnya dari game labirin[6].

6. Pada Jurnal yang berjudul "*Implementasi Algoritma Simplified Memori Bounded A**" yang ditulis oleh A.J. Purnomo dan G. Hermawan dijelaskan bahwa Algoritma SMA* adalah algoritma turunan dari A* sama seperti HPA*. Algoritma ini mampu mengingat nilai f-cost dari setiap iterasi sampai sejumlah atau sebanyak simpul yang ada di dalam memori. Karena memori memiliki batasan kapasitas tertentu, maka jumlah iterasi dalam algoritma ini akan mengikuti atau akan memiliki batas iterasi sampai batas maksimum dari memori. Setelah itu, maka selanjutnya akan dicari nilai terbaik dari hasil pencarian tersebut. Misalkan memori hanya menampung 50 simpul maka pencarian memiliki batas iterasi sebanyak 49 simpul [7].