PERTEMUAN 18
GAME PLAYING

A. Tujuan Pembelajaran

Setelah menyelesaikan materi pada pertemuan ini, mahasiswa mampu menganalisis *game palying*. Sub materi pada pertemuan ini yaitu:

1. Definisi *Game playing*
2. Metode penggunaan game
3. Mode Game AI
4. Finite State Machine (FSM)
5. Sistem berbasis aturan (*Rule Based System)*
6. Algoritma AI
7. Algoritma *Dijkstra*
8. Kompleksitas Kesalahan
9. Aplikasi *Game playing*

B. Uraian Materi

1. Definisi Game Playing

Game pada sebuah AI adalah sebuah aplikasi yang menjadikan sebuah hiburan kepada seseorang atau para pemain yang memainkan game tersebut, didalam sebuah game banyak sekali mengandung sebuah objek dari sebuah pemodelan yang udah didesain pada software tertentu, dalam pembentukan objek didalam sebuah game menjadikan objek tersebut hidup dan mendapatkan karakternya, tidak kalah dari itu didalam sebuah game juga terdapat beberapa objek yang bisa disebut sebagai lawan atau musuh dari objek karakter utama game. Peranan sebuah kecerdasan buatan didalam game adalah sebagai pengendali atau sebuah pemikiran yang otomatis untuk memainkan game tersebut, dengan kecerdasan buatan juga akan terjadi interaksi alami yang dilakaukan lawan atau objek tokoh utama dalam mengikuti rule game tersebut. Contoh media interaksi ialah:

a. Penglihatan (*vision)*
b. Suara (*voice),* ucapan (*speech)*
c. Gerakan anggota badan (*gesture)*

Beberapa unsur didalam Artificial Intelligence adalah sebuah elemen, elemen – elemen didalam sebuah game di terapkan sealami mungkin, selayaknya sebuah kehidupan yang dilakukan manusia, selain itu dalam implementasikan Artificial Intelligence didalam sebuah game terdapat sebuah algoritma, algoritma ini nanti akan menggabungkan beberapa intruksi penyampaian yang akan di sambungkan kembali ke beberapa elemen bertujuan untuk menjalankan perintah atau tujuan tertentu, contoh beberapa algoritma yang dapat diimplementasikan ke aturan game adalah, algoritma tree, atau pohon keputusan, didalam sebuah pohon keputusan terdapat beberapa aturan yang menyerupai cabang keputusan, dimana setiap keputusan yang diambil akan membuat rule-nya tersendiri dan ceritanya sendiri, namun tree selain mudah dipahami, algoritma ini akan menjadi nilai representasi yang memiliki nilai kelemahan yang cukup besar, kelemahannya salah satunya adalah nilai representasi pohon keputusan akan menjadi sangat luas dan sangat banyak, semakin panjang cerita yang dibuat maka semakin panjang juga sebuah rule dari tree ini akan dibuat, dan akan menjadikan sebuah percabangan yang banyak didalam percabangan.

Dalam mengimplementasikan sebuah objek dimana objek tersebut dibuat sebelumnya menggunakan beberapa software untuk membuat objek 3 dimensi, salah satu softwarenya adalah blender, blender adalah salah satu untuk membuah objek 3 dimensi, dan dapat membuat animasi didalamnya, setelah dibuat menggunakan software tersebut maka objek dapat di buka pada aplikasi pembuat game, salah satu aplikasi yang dapat mengimplementasikan game 3 dimensi adalah unity, unity adalah software untuk membuat game berbasis 2 dimensi atau 3 dimensi, yang dapat di terapkan ke permainan smartphone.

a. Beberapa karakteristik dan batasan *game*

Karakteristik adalah sebuah identitas yang digambarkan berdasarkan penyampaian atau bentuk ciri – ciri, didalam sebuah game beberapa karakteristik dan batasan game yang dimainkan oleh dua pemain atau dua player, baik manusia atau kecerdasan buatan itu sendiri, pergantian dalam permainan dalam melangkah dengan karakteristik sebagai berikut :

1) *Perfect Information Game* : Kedua pemain yang bermain game dapat menggunakan akses pada game atau sama – sama memiliki akses dalam

pengkontrolan game dan dapat melihat informasi yang lengkap terkait game tersebut, sehingga tidak ada sesuatu ketidak tahuan atau yang menutupi informasi bagi lawan mainnya.

2) *No Determined by Chances* : Tidak menggunaka objek yang melibatkan probabilitas, contonya adalah dengan menggunaka sebuah dadu
3) *No Phsychological Factors* : Tidak menggunakan emosional yang melibatkan psikologi, seperti sebuah gertakan.

4) *No Oversight Errors. Smart Opponent* : Mempunyai lawan yang setara dengan pemahamannya dalam sebuah game sehingga menjadikan seimbang dalam permainannya, dan menghindari salah langkah.

2. Metode penggunaan game

a. Prosedur *Minimax*

The Game of Nim: Sejumlah dari sekumpulan token ditempatkan pada sebuah meja di antara lawan yang terdiri dua lawan. Pada masing-masing gerakan pemain harus membagi tumpukan *token* menjadi dua tumpukan tak kosong dari berbagai ukuran. Jadi, 6 token dapat dibagi menjadi 5 dan 1, 4 dan 2, tetapi tidak 3 dan 3. Pemain pertama yang mampu bergerak kehilangan permainan. Untuk sejumlah kecil token ruang pencarian dapat dicari secara mendalam. Dalam permainan dua-orang, Anda harus mengasumsikan bahwa lawan Anda memiliki pengetahuan yang sama yang Anda lakukan dan berlaku sebaik yang Anda lakukan. Jadi pada setiap tahap permainan Anda harus menganggap lawan membuat langkah terbaik yang tersedia. Ini adalah dasar dari prosedur minimax.

Dalam *minimax*, para pemain yang disebut sebagai *MAX* (pemain) dan *MIN* (lawan). Keduanya mencoba untuk memaksimalkan gerakan mereka. *MAX* pemain, mencoba untuk memaksimalkan nilainya. Dan *MIN* adalah lawan mencoba untuk meminimalkan skor *MAX.*

Prosedur Minimax pada Pencarian Ruang Lengkap

1) Label setiap tingkat dari ruang pencarian sesuai dengan yang bergerak itu di tingkat itu.

2) Mulai di node daun, setiap label simpul daun dengan 1 atau 0 tergantung pada apakah itu adalah kemenangan bagi *MAX* (1) atau MIN (0).
3) Merambat ke atas: jika negara induk *MAX*, memberikan *MAX* anak-anaknya.
4) Merambat ke atas: jika negara induk *MIN*, *MIN* memberikan anak-anaknya.

Pencarian *minimax* merupakan pencarian nilai terbaik dari nilai-nilai evaluasi yang didapat dari berbagaimacam cara untuk menghitung nilai evaluasi tersebut. Pencarian ini bekerja dengan cara menelusuri segala kemungkinan yang terjadi pada papan dengan melakukan pencarian untuk beberapa langkah kedepan (Aske Plaat, 1994). Berbagai macam metode yang dapat digunakan dalam pencarian minimax, beberapa di antaranya adalah *Negascout* dan MTDF.

b. Negascout

Negascout adalah sebuah metode atau teknik dimana melakukan pencarian minimax dengan pemiliran dimana bahwa langkah pertama yang diambil adalah sebuah langkah yang diasumsikan menjadi langkah yang baik dalam permainannya, dan langkah selanjutnya adalah merupakan langkah yang buruk.(Aske Plaat, 1994). Dan bila terdapat sebuah langkah yang baik dari langkah pertama maka akan terjadi sebuah langkah yang dapat menimbulkan proses *research* atau dapat diartikan pencarian ulang, Berikut penjelasan algoritma Negascout yang dijabarkan pada Gambar 18.1.

```
Function Negamax(u,Alpha,Beta)
1. [Cek apakah node u adalah leaf]
   IF node u = leaf THEN
   1.1 [Panggil fungsi evaluasi]
       RETURN eval(u)
   ELSE
   1.2 B ← Beta
   1.3 REPEAT FOR I = 1,2,3,...,jumlah anak
       1.3.1 t ← -Negascout(u[i],-B,-Alpha)
       1.3.2 IF (t>B) and (t<Beta) and
             (i>1) and (depth>1) THEN
             1.3.2.1 t ← -Negascout(u[i],-Beta,-t)
       1.3.3 Alpha ← max(Alpha, t)
       1.3.4 IF Alpha >= Beta THEN
             1.3.4.1 RETURN Alpha
       1.3.5 B = Alpha + 1
   1.4 RETURN Alpha
```

Gambar 18.1 Algoritma Negascout

Pada implementasi didalam gambar terseubt, pertama syarat dalam melakukan sebuah pencarian ulang atau *research* adalah dengan jika suatu nilai dari *t* lebih besar dari *B,* lebih kecil dari beta dan buakan suatu anak pertama dari node yang diproses (i>1), dan node itu harus memiliki sebuah anak dimana pengertian logikanya adalah sebagai berikut (depth>1). Syarat *t* lebih kecil dari beta diperlukan karena jika *t* lebih besar dari beta, maka beta pruning akan diproses sehingga proses *research* tidak akan terjadi.

c. Memory-enhanced *Test Driver value f (*MTDF)

Pencarian pada MTDF menggunakan *bound* sebagai tempat penyimpanan nilai minimax, dimana *bound* tersebut terbagi menjadi 2 macam, yaitu *upperbound dan lowerbound* yang menunjukkan rentang dimana nilai *minimax* berada. Cara kerja dari algoritma mtdf adalah dengan cara melakukan serangkaian pemanggilan algoritma alpha beta secara berulang-ulang. Berikut penjelasan algoritma MTDF yang dijabarkan pada Gambar 18.2. Pada algoritma Gambar 18.2, parameter *f* merupakan nilai perkiraan. Jika nilai dari parameter tersebut mendekati nilai minimax maka proses pemanggilan terhadap algoritma alpha beta akan semakin sedikit. Minimal dilakukan dua kali proses pemanggilan,

dimana untuk menentukan nilai dari upperbound dan lowerbound . Proses pemanggilan algoritma alpha beta akan berhenti jika nilai dari lowerbound lebih besar dari upperbound. Algoritma alpha beta yang digunakan sedikit berbeda dengan algoritma alpha beta konvensional.

```
Function MTD(u,f)
1. [Inisialisasi variabel]
   1.1 g ← f
   1.2 Upperbound ← ∞
   1.3 Lowerbound ← -∞
2. [Melakukan iterasi MTD(f)]
   2.1 REPEAT WHILE Upperbound > Lowerbound
       2.1.1 IF g = Lowerbound THEN
             2.1.1.1  Beta ← g +1
             ELSE
             2.1.1.2  Beta ← g
       2.1.2 g ← AlphabetaWithMemory(u,Beta-1,Beta)
       2.1.3 IF g < Beta THEN
             2.1.3.1  Upperbound ← g
             ELSE
             2.1.3.2  Lowerbound ← g
```

Gambar 18.2 Algoritma MTDF

Dimana pada algoritma *alpha beta with memory* menyimpan nilai upperbound dan lowerbound tiap-tiap node, yang nantinya digunakan untuk perbandingan dengan nilai alpha dan beta yang didapat.

d. Prosedur Alpha-Beta

Alpha-beta pruning adalah prosedur untuk mengurangi jumlah perhitungan dan mencari selama *minimax*. *Minimax* adalah pencarian dua-pass, satu lulus digunakan untuk menetapkan nilai-nilai heuristik ke node pada kedalaman dan yang kedua digunakan untuk menyebarkan nilai-nilai sampai pohon.

Alpha-beta hasil pencarian secara mendalam-pertama. Sebuah nilai alpha adalah nilai awal atau sementara terkait dengan node *MAX*. Karena *MAX* node diberi nilai maksimum antara anak-anak mereka, nilai alpha tidak dapat menurunkan, hanya bisa naik. Sebuah nilai beta adalah nilai awal atau sementara terkait dengan node MIN. Karena node MIN diberi nilai minimum antara anak-anak mereka, nilai beta tidak pernah dapat meningkatkan, hanya bisa turun.

Misalnya, alpha node *MAX* = 6. Kemudian cari tidak perlu mempertimbangkan setiap cabang yang berasal dari keturunan MIN yang memiliki nilai beta yang kurang-dari-atau-sama dengan 6. Jadi, jika Anda tahu bahwa node MAX memiliki alpha 6, dan Anda tahu bahwa salah satu keturunan MIN yang memiliki beta yang kurang dari atau sama dengan 6, Anda tidak perlu mencari lebih jauh di bawah simpul MIN. Ini disebut pemangkasan alpha.

Alasannya adalah bahwa tidak peduli apa yang terjadi di bawah simpul MIN, tidak dapat mengambil nilai yang lebih besar dari 6. Jadi nilainya tidak dapat diperbanyak sampai dengan (alpha) orangtua *MAX* nya.

Demikian pula, jika nilai beta node MIN itu = 6, anda tidak perlu mencari lebih jauh di bawah MAX keturunan yang telah memperoleh nilai alpha dari 6 atau lebih. Ini disebut pemangkasan beta.

Alasannya lagi adalah bahwa apa pun yang terjadi di bawah simpul *MAX*, tidak dapat mengambil nilai yang kurang dari 6. Jadi nilainya tidak dapat diperbanyak sampai dengan (*beta) MIN* orangtua nya.

Aturan untuk *Alpha-beta* Pemangkasan

1) Pemangkasan Alpha: pencarian dapat dihentikan di bawah setiap simpul *MIN* memiliki nilai beta kurang dari atau sama dengan nilai alpha dari setiap leluhur *MAX* nya.
2) Pemangkasan Beta: Pencarian bisa dihentikan di bawah setiap simpul *MAX* memiliki nilai alpha lebih besar dari atau sama dengan nilai beta dari setiap leluhur *MIN* nya.

e. Fungsi Evaluasi Dalam Permainan *Othello*

1) Table

Edge Table merupakan wadah untuk menyimpan nilai-nilai mobility dari papan permainan. Cara kerja edge table adalah dengan hanya mengevaluasi satu sisi papan, yang mana selanjutnya hanya perlu mencerminkannya kesemua sisi-sisi pada papan permainan.

2) Mobility

Mobility merupakan jumlah langkah yang dapat dimainkan oleh pemain

pada tiap kali kesempatan. Jumlah langkah ini didapat dengan mengevaluasi pola pada satu sisi papan dan selanjutnya disimpan kedalam edge table.

3) Liberties

Liberties digunakan untuk menampung jumlah petak kosong yang berada disekeliling tiap-tiap petak pada papan. Setelah proses penyimpanan nilai-nilai liberties dilakukan, selanjutnya nilai-nilai yang disimpan tersebut akan diproses oleh frontier untuk mengetahui jumlah disc yang berbatasan dengan petak kosong.

4) Potensial Mobility atau Frontier

Frontier merupakan jumlah disc yang berbatasan dengan petak kosong. Jika terjadi kondisi dimana semakin banyak frontier yang didapat, maka akan semakin jelek pula posisinya, karena semakin banyak frontier memungkinkan lawan untuk mendapatkan semakin banyak mobility tambahan pada beberapa langkah ke depan dan juga mengurangi mobility-nya sendiri.

5) Penguasaan *Corner*

Penguasaan *corner* merupakan penguasaan terhadap posisi-posisi pojok dari papan, karena disc yang diletakkan pada posisi tersebut tidak dapat dirubah atau dibalik.

3. Mode Game AI

a. *Pathfinding*

Pathfinding paling mudah ditemui pada *game-game* bertipe strategi dimana kita menunjuk satu tokoh untuk digerakkan ke lokasi tertentu dengan mengklik lokasi yang hendak dituju. Si tokoh akan segera bergerak ke arah yang ditentukan, dan secara “cerdas” dapat menemukan jalur terpendek ataupun menghindari dari rintangan-rintangan yang ada. Salah satu algoritma pathfindin yang cukup umum dan yang paling banyak digunakan utnuk mencari jarak terpendek secara efisien adalah algoritma A*.

Secara umum, algoritma A* adalah mendefinisikan area pencarian menjadi sekumpulan *node-node (tiles).* Titik awal dan titik akhir ditentukan terlebih dulu untuk mulai penelusuran pada tiap-tiap *node* yang memungkinkan untuk ditelusuri. Dari sini, akan diperoleh skor yang menunjukkan besarnya biaya untuk menempuh jalur yang ditemukan, ditambah dengan nilai heuristik yang merupakan nilai biaya estimasi dari node yang ada menuju tujuan akhir. Iterasi akan dilakukan hingga akhirnya mencapai target yang dituju.

b. Jaringan Saraf Tiruan (Neural Network)

Neural network cukup baik ketika diterapkan pada kasus-kasus yang sifatnya non-linier atau mengambil keputusan yang tidak dapat dilakukan dengan metode tradisional. Penerapannya seringkali pada *game-game* yang memerlukan kemampuan adaptif atau belajar dari pengalaman. Sebagai contoh, jika suatau ketika terjadi pertempuran antar player dengan unit komputer, dan unit komputer mengalami kekalahan, maka pada kesempatan lain yang serupa, komputer akan memilih untuk tidak bertempur. Semakin banyak pengalaman yang dialami komputer, maka komputer menjadi semakin cerdas. Prinsip dasar dari jaringan saraf tiruan ini adalah perbaikan bobot secara terus menerus aga*r output* yang dihasilkan menjadi semakin akurat (semakin cerdas).

c. Algoritma Genetis (*Genetic Algorithm)*

Algoritma genetis sedikit banyak dipengaruhi oleh teori evolusi yang dicetuskan Darwin, yaitu bahwa spesies akan terus menerus beradaptasi dengan lingkungannya dan ciri khasnya yang terletak pada kromosom, akan diturunkan pada generasi berikutnya. Generasi turunan ini menerima gabungan kromosom dari kedua induknya, yang disebut dengan *crossover*. Pada algoritma genetis, akan diterapkan langkah ranking fitness untuk melakukan seleksi terhadap langkah *ranking fitness* untuk melakukan seleksi terhadap generasi turunan yang terbaik. Pada game berbasis algorima genetis, turunan terbaik inilah yang dilibatkan ke dalam game, dimana akan digunakan oleh komputer untuk merespons perubahan-perubahan tingkah laku *user.*

d. Algoritma, Struktur Data, dan Representasi

1) Decision tree

Decision Tree adalah salah satu metode klasifikasi yang paling banyak digunakan dala membuat sebuah pemodelan dimana mudah dipahami, dalam sebuah bentuk struktur decision tree ini dibentuk seperti struktur percabangan sebuah pohon yang membentuk hirarkii. Konsep dari decion tree ini adalah mengubah sebuah data menjadi bentuk pohon keputusan dan menuangkan beberapa kejadian didalam setiap percbangan tersebut, kelebihan dari sebuah pohon keputusan atau algoritma decision tree ini adalah dapat mem-break down sebuah proses dalam pengambilan keputusan yang begitu kompleks dan menjadikan dalam bentuk yang lebih simpel sehingga dalam pengambilan sebuah keputusannya dapat lebih menginterpretasikan solusi dari permasalahan.

2) *Kelebihan*

a) Pengambilan keputusan didalam algoritma yang sebelumnya kompleks menjadi mudah dalam pemahaman dan spesifik dan simpel.
b) Tidak ada perhitungan yang mengeliminasi, dikarnakan hanya menggunakan sebuah struktur hierarki pohon keputusan.
c) Memiliki feature dan internal yang tidak sama dalam pemilihannya dikarnakan memiliki bentuk yang fleksibel.
d) Dapat menghindari sebuah permasalahan dikarnakan menggunakan sebuah kriteria yang memiliki jumlah sedikit didalam sebuah node internal dan tanpa mengurangi sebuah kualitas didalam keputusannya, hasil tetap memiliki kualitasnya.

3) *Kekurangan*

a) Terjadi percabangan yang sangat banyak bila kasus dan pembahasan memiliki bentuk yang sangat panjang, dimana kelas – kelas dan kriteria yang digunakan dalam hal tersebut sangat banyak, dan dapat menyebabkan waktu yang dibutuhkannya menjadi lama dalam keputusan yang dihasilkan, serta memory yang diperlukan cukup besar.

b) Memiliki tingkat eror yang besar bila kasus eror terjadi pada cabang sebelumnya, hal tersebut akan mengakibatkan eror pada cabang dibawahnya.
c) Sulitnya menentukan desain yang memiliki bentuk optimal, dimana kualitas sebuah pohon keputusan dari algoritma ini sangat mengutamakan optimal pada saat desain pohon keputusan.

4. Finite *State Machines (FSM)*

FSM atau *Finite State Machines* adalah sebuah metode perancangan dari sebuah pengendali dari sebuah prinsip kerja sistem, dimana prinsip kerja sistem menggunakan tiga hal, *State* (Keadaan*), Event* (kejadian) dan *Action* (aksi). Dari tiga hal tersebut dapat dipahami bahwa pengendali atau pengontrol memiliki sebuah kondisi. Pada saat kondisi dimana periode untuk waktu yang signifikan, sistem memulai dengan kondisi keadaan aktif atau state on, metode ini sering digunakan untuk sebuah perancangan sebuah perangkat lunak atau software aplikasi, Penggunaan pada metode ini pada kenyataannya diimplementasikan menjadi sebuah bentuk modeling sebagai basis untuk perancangan protokol-protokol komunikasi, perancangan perangkat lunak game, aplikasi WEB dan sebagainya. *Finite State Machine(FSM)* ini memiliki kelebihan dan kekurangan,antara lain :

a. Kelebihan

1) Dalam melakukan implementasi dapat dilakukan dengan mudah dan cepat
2) Dapat memudahkan proses *debugging*.
3) Proses komputasi yg minimal, karena sejatinya FSM hanyalah conditional statement yang dikemas dalam bentuk yang lebih elegan.
4) Fleksibel, dikarnakan dapat digabungkan dengan penggunaan metode dan algoritma lainnya.

b. Kekurangan

1) Behaviour mudah dalam prediksinya, tidak ada searching didalamnya.
2) Karena mudah diimplementasi, kadang programmer langsung tembak di eksekusi tanpa melakukan desain FSM terlbih dahulu. Biasanya akan terjadi FSM yang terfragmentasi
3) Ketika terjadi sebuah batasan yang menipis, akan timbul State Oscillation.

5. Sistem berbasis aturan *(Rule Based System)*

adalah suatu program komputer yang memproses informasi yang terdapat di dalam *working memory* dengan sekumpulan aturan yang terdapat di dalam basis pengetahuan menggunakan mesin inferensi untuk menghasilkan informasi baru.

Sebuah *Rule-Based System* dapat dibentuk dengan menggunakan sebuah assertions set, yang secara kolektif membentuk *working memory*, dan sebuah rule set yang menentukan aksi pada *assertions set. RBS* secara relatif adalah model sederhana yang bisa diadaptasi ke banyak masalah. Namun, jika ada terlalu banyak peraturan, pemeliharaan sistem akan rumit dan terdapat banyak failure dalam kerjanya.

a. Kelebihan

1) *Availability*-bertambah, *intelligent tutor, intelligent dB, danger-reduced, performance*
2) *multiple expertise, reability*-bertambah, *explanation steady, unemotional and complete response*

b. Kekurangan

Jika terlalu banyak aturan, sistem menjadi sulit dalam me-maintain performance dan Keterbatasan dalam memutuskan teknik yang digunakan untuk suatu masalah.

6. Algoritma AI

Algoritma AI atau Artificial Intelligence adalah sebuah teknik yang terdapat disebuah sistem cerdas, dan sudah banyak sekali diterapkan pada beberapa bidang, salah satunya adalah bidang pembelajaran, kesehatan, game, dan pendidikan, kehidupan sehari – hari pun tak lepas dari sebuah sistem cerdas ini, salah satunya adalah penggunaan mesin cuci, pada penggunaan mesin cuci dimana didalamnya sudah diimplementasi sistem cerdas, maka alat tersebut akan mengoperasikannya sendiri, dari awal sampai akhir, solusi penggunaan sistem cerdas juga dimanfaatkan industri dalam meningkatkan sebuah produksi didalamnya, semakin tepat nilai yang dihasilkan kecerdasan tersebut maka semakin baik untuk kemajuan perusahaan tersebut.

Algoritma AI ini juga menjadikan pemodelan yang sebalumnya susah akan menjadi mudah bila diterapkan pada tempat yang sesuai, banyak sekali teknik didalam sebuah algoritma ini diantaranya sebuah metode fuzzy, didalam sebuah metode fuzzy terdapat beberapa algoritma, contonya algoritma tsukamoto, algoritma mamdani dan algoritma sugeno, implementasi algoritma tersebut dapat diterapkan dibeberapa masing – masing kasusnya, begitu juga dengan metode sistem pakar, metode sistem pakar adalah sebagian dari kecerdasan buatan dimana didalamnya terdapat algoritmanya, contohnya adalah algoritma forward chaininig dan algoritma certainty factor, dan banyak lagi pengkelompokan sebuah metode kecerdasan buatan.

Tujuan dari kecerdasan buatan dan algoritmanya adalah menjadi sebuah alat untuk mengatasi permasalahan yang terjadi dan menyelesaikannya dengan cara dan teknik tersebut, dengan cara tersebut dapat menghasilkan solusi dan pemanfaatan yang terasa bagi pemakai kecerdasan buatan tersebut, pengoperasian beberapa alat yang menanamkan algoritma AI pun akan menjadi sebuah teknologi yang mendukung perkembangan dalam kemajuan zaman.

7. Algoritma *Dijkstra*

Algoritma *Dijkstra* atau aturan yang membahas optimalisasi dari jarak dikenal pada umumnya sebagai algoritma rakus (greedy algorithm), penggunaan dari algoritma ini adalah untuk mengatasi permasalahan dan memecahkan suatu jarak dengan mencari sebuah jarak terpendek, dan digambarkan dengan bentuk graf berarah, dengan bobot yang bukan nilai negatif, sebagai contohnya adalah sebuah graf yang dilambangkan jarak antara suatu kota untuk menentukan sebuah jarak atau rute terpendek dari sebuah jalur. Tujuan dari algoritma ini adalah mencari serta menemukan sebuah rute terpendek yang dilihat berdasarkan nilai bobot terkecil dari suatu titik yang satu ke titik yang lain. Dan kelemahan dari algoritma ini adalah semakin banyak titik maka akan mengakibatkan waktu yang digunakan dalam prosesnya lama juga.

a. Urutan Logika Algoritma *Dijkstra*

1) Memberikan sebuah nilai dari jarak kepada sebuah titik awal ke titik yang lainnya, dan selanjutnya set nilai 0 pada sebuah node awal menyebabkan nilai tak hingga terhadap node lainnya

2) Membuat set pada semua node, dari node yang belum ditempati hingga node keberangkatan
3) Dimana node keberangkatan, membuat suatu pertimbangan kepada node yang belum ditempati dan menghitung sebuah jarak dari titik node keberangkatan
4) Setelah selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai “Node terjamah”. Node terjamah tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5) Set “Node belum terjamah” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan lanjutkan dengan kembali ke step 3.

8. Kompleksitas Kesalahan

Sebuah konteks didalam sebuah kecerdasan buatan terutama pada sebuah video game, tindak kecurangan mendukung kepada programmer dimana mereka akan memberikan sebuah akses informasi kepada seorang pemain lainnya. Di contohkan pada sebuah game sederhana, jika seorang yang bermain game yang seharusnya didalam game tersebut membutuhkan usaha untuk mendapatkan sebuah nilai tertentu, seseorang tersebut tidak membutuhkan waku yang lama untuk mendapatkan nilai tersebut bahkan tanpa usaha pun mereka dapat dengan mudah mendapatkan nilai tersebut, hal ini akan mengandung ketidak harusan terhadap sebuah game dan akan menjadikan kompleksitas kesalahan, yang tidak seharusnya terjadi pada game tersebut.

Kesalahan pu n sering terjadi kapada alat yang menerapkan sebuah kecerdasan buatan didalamnya, contoh lainnya adalah sebuah mesin cuci otomatis, dimana seharusnya mecin cuci akan menentukan sebuah waktu untuk menentukan waktu mencuci yang dilihat berdasarkan berat cucian dan mesin cuci akan menentukan banyaknya air yang dibutuhkan, bila kesalahan kompleksitas terjadi maka akan menyebabkan kesalahan dalam menentukan waktu proses mencuci, hal ini akan menjadi kerusakan atau kesalahan pada sistem cerdasnya dan perlu di program ulang pada mesin cuci tersebut.

9. Aplikasi Game playing

Game Playing (permainan game) merupakan bidang AI yang sangat populer berupa permainan antara manusia melawan mesin yang memiliki intelektual untuk berpikir. Dalam game playing sendiri memiliki beberapa karakteristik dan batasan tertentu. Selain itu metode yang digunakan pun beragam tergantung game apa yang dibuat dan proses penyelesaiannya, karena, berbeda game, berbeda juga metode yang dipakai dalam menyelesaikannya. Banyak jenis game yang menggunakan AI atau Artificial Intelligence, mulai dari game jaman dahulu hingga game modern seperti sekarang ini, seperti *Catur, Othello, Tic Tac Toe, Counter Strike, Gran Turismo.* Permainan game laiinnya yaitu permainan catur, permainan dengan permasalahan ember dan air (Dahria, 2008). Game sendiri memiliki berbagai genre atau tipe permainan, seperti *racing, FPS, RPG, MMORPG,* dan lain sebagainya. Game sendiri sekarang sudah banyak tersedia di berbagai platform seperti *PC, Konsol dan Android.* Tidak sedikit pula game yang ternyata banyak yang memiliki manfaat dan nilai positif, seperti meningkatkan kerja otak, menghilangkan stress, meningkatkan semangat positif, menumbuhkan kreatifitas dan kemampuan dalam memecahkan masalah, dan lain sebagainya. Dengan demikian, game tidak hanya dipandang sebagai hal yang negative dan membuang-buang waktu, game juga dapat menjadi hal yang positif apabila kita dapat dengan bijak dalam menggunakannya.

C. Soal Latihan/Tugas

1. Dengan menggunakan tahapan metode game carilah 1 jenis game AI dan jelaskan metode dan serta tahapannya?
2. Buatlah sebuah game yang menerapkan kecerdasan buatan menggunakan unity 2d atau 3d
3. Buatlah implementasi metode kecerdasan buatan kedalam sebuah sistem aplikasi

D. Referensi

Dahria M. 2008. Kecerdasan buatan(artifial intelligence). Jurnal SAINTIKON. Vol 5 No.2

Sutojo, T. Edy ,Mulyanto dan Suhartono,Vincent. 2010. Kecerdasan Buatan. Andi Offset. Yogyakarta