BAB 2
LANDASAN TEORI

2.1. Game

2.1.1. Pengertian Game

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan refreshing. Suatu cara belajar yang digunakan dalam menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukkan strategi-strategi yang rasional.

Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh John von Neumann and Oskar Morgenstern yang berisi: "Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi [1].

2.1.2. Beberapa pengertian tentang game

Menurut [1] terdapat beberapa pengertian tentang game oleh beberapa ahli yaitu:

1. Menurut Agustinus Nilwan dalam bukunya Pemrograman Animasi dan *Game* Profesional terbitan Elex Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan *game*. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

2. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Bernard Suits *Game* adalah upaya sukarela untuk mengatasi rintangan yang tidak perlu .
4. Menurut Greg *Cost*ikyan, *Game* adalah sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan.

2.1.3. Sejarah Game

Pada tahun 1952, seorang mahasiswa Universitas Cambridge bernama A.S. Gouglas membuat permainan OXO (tic-tac-toe) dalam versi grafik. Permainan ini ia kembangkan ketika hendak mendemonstrasikan tesisnya tentang interaksi antara manusia dan komputer. Pada tahun 1958 William Higin Botham mendesain sebuah *game* dengan judul Tennis For Two yang dimainkan dalam oscilloscope, dan kemudian ada pula Steve Russel pada tahun 1961 dengan *game* berjudul Spacewar yang dibuat dalam komputer mainframe DEC PDP-1 saat mereka menjalani studi di MIT. Sejarah *video game* tidak hanya mengenai orang-orang yang berperan di dalamnya, tetapi juga mengenai berbagai perusahaan *game* yang mempunyai kasus ironis. Atari adalah perusahaan Amerika dengan nama Jepang, dan perusahaan Jepang bernama SEGA didirikan oleh orang Amerika. Magnavox yang memulai peran perusahaan ini telah berumur 1 abad, dan Nintendo sebagai perusahaan yang mempopulerkan kembali *video game* juga sama tuanya, serta tidak ada yang pernah berpikir bahwa Sony yang merupakan perusahaan penemu banyak barang-barang elektronik mulai dari transistor radio hingga alat perekam *video*, akan membuat sebuah *console* yang menjadi produk dengan tingkat penjualan yang tinggi hingga saat ini [1].

2.1.4. Jenis-Jenis Game

Menurut [8], terdapat beberapa jenis game yang dapat dikelompokan sebagai berikut:

1. Simulasi

   Contoh permainan yang termasuk dalam game simulasi adalah simulasi konstruksi dan manajemen, simulasi kendaraan seperti yang diterapkan pada permainan balapan, perang, luar angkasa, dan mecha.

2. Edukasi

   Contohnya adalah edugames yang dibuat dengan tujuan spesifik sebagai alat pendidikan, baik itu untuk belajar mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. Developer yang membuatnya, harus memperhitungkan berbagai hal agar game ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya.

3. Entertainment

   a. Aksi – *Shooting*, (tembak-tembakan , atau hajar-hajaran bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya). Game jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga waktu. inti dari game jenis ini adalah tembak-tembakan.

   b. *Fighting* (pertarungan), ada yang mengelompokan game fighting di bagian Aksi, namun penulis berpendapat berbeda, jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari game ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya), pengenalan karakter dan waktu sangatlah penting. Dan berbeda seperti game Aksi pada umumnya yang umumnya hanya melawan *Artificial Intellegence* atau istilah umumnya melawan komputer saja, pemain jenis fighting game ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya.

   c. Petualangan, game murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan

rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.

d. *Role Playing*, game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya *Level*.

e. *Casual games*, sesuai namanya, game yang casual itu tidak kompleks, mainnya rileks dan sangat mudah untuk dipelajari. Jenis ini biasanya memerlukan spesifikasi komputer yang standar pada jamannya dan ukurannya tidak lebih dari 100 MB karena biasanya dapat di download versi demo-nya di website resminya. *Genre* permainannya biasanya puzzle atau action sederhana dan umumnya dapat dimainkan hanya menggunakan.

f. *Multiplayer Online*, game yang dapat dimainkan secara bersamaan oleh lebih dari 2 orang (bahkan dapat mencapai puluhan ribu orang dalam satu waktu) membuat pemain dapat bermain bersama dalam satu dunia virtual dari sekedar chatting hingga membunuh naga bersama teman yang entah bermain di mana. Umumnya permainan tipe ini dimainkan di PC dan bertema RPG, walau ada juga yang bertema musik atau *action*.

2.2. Kecerdasan Buatan / Artificial Intelligence (AI)

Pada penelitiannya menurut [9], kecerdasan buatan atau artificial intelligence merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi digunakan

sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Agar komputer bisa bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekal pengetahuan, dan mempunyai kemampuan untuk menalar. Untuk itu pada AI, akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin yang pintar.

Pengertian kecerdasan buatan dapat dipandang dari berbagai sudut pandang, antara lain:

1. Sudut pandang kecerdasan.

   Kecerdasan buatan akan membuat mesin menjadi cerdas (mampu berbuat seperti apa yang dilakukan oleh manusia).

2. Sudut pandang penelitian.

   Kecerdasan buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu sebaik yang dikerjakan oleh manusia.

   Domain yang sering dibahas oleh para peneliti meliputi :

   a. Mundane task
      1. Persepsi (*vision & speech*)
      2. Bahasa alami (*understanding, generation, & translation*)
      3. Pemikiran yang bersifat *commonsense*.
      4. Robot control.
   b. Formal task
      1. Permainan/games.
      2. Matematika (geometri, logika, kalkulus integral, pembuktian)
   c. Expert task
      1. Analisis finansial
      2. Analisis medikal
      3. Analisis ilmu pengetahuan

4. Rekayasa (desain, pencarian kegagalan, perencanaan manufaktur)

3. Sudut pandang bisnis.

   Kecerdasan buatan adalah kumpulan peralatan yang sangat powerful dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4. Sudut pandang pemograman.

   Kecerdasan buatan meliputi studi tentang pemograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*).

   Untuk melakukan aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan, yaitu :

   a. Basis pengetahuan (*knowledge base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
   b. Motor inferensi (*inference engine*), yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.

2.2.1 Sejarah Kecerdasan Buatan

Menurut [9], Kecerdasan buatan termasuk bidang ilmu yang relatif muda. Pada tahun 1950-an para ilmuwan dan peneliti mulai memikirkan bagaimana caranya agar mesin dapat melakukan pekerjaannya seperti yang bisa dikerjakan oleh manusia. Alan turing, seorang matematikawan dari Inggris pertama kali mengusulkan adanya tes untuk bias melihat bisa tidaknya sebuah mesin dikatakan cerdas. Hasil tes tersebut kemudian dikenal dengan Turing test, dimana si mesin tersebut menyamar seolah-olah sebagai seseorang di dalam suatu permainan yang mampu memberikan respon terhadap serangkaian pertanyaan yang diajukan. Turing beranggapan bahwa, jika mesin dapat membuat seseorang percaya bahwa dirinya mampu berkomunikasi dengan orang lain, maka dapat dikatakan bahwa mesin tersebut cerdas (seperti layaknya manusia).

Kecerdasan buatan atau artificial intelligence itu sendiri dimunculkan oleh seorang professor dari *Massachusetts Institute of Technology* yang bernama John McCarthy pada tahun 1956 pada Dartmouth Conference yang dihadiri oleh para peneliti AI. Pada konferensi tersebut juga didefinisikan tujuan utama dari

kecerdasan buatan, yaitu: mengetahui dan memodelkan proses-proses berfikir manusia dan mendesain mesin agar dapat menirukan kelakuan manusia tersebut.
Beberapa program AI yang mulai dibuat pada tahun 1956-1966, antara lain :

1. Logic Theorist, diperkenalkan pada Darmouth Conference, program ini dapat membuktikan teorema-teorema matematika.
2. Sad Sam, diprogram oleh Robert K. Lindsay (1960). Program ini dapat mengetahui kalimat-kalimat sederhana yang ditulis dalam bahasa inggris dan mampu memberikan jawaban dari fakta-fakta yang didengar dalam sebuah percakapan.
3. ELIZA, diprogram oleh Joseph Weizenbaum (1967). Program ini mampu melakukan terapi terhadap pasien dengan memberikan beberapa pertanyaan.

2.2.2 Lingkup Kecerdasan Buatan pada Aplikasi Komersial

Dewasa ini, kecerdasan buatan juga memberikan kontribusi yang cukup besar di bidang manajemen. Adanya system pendukung keputusan, dan Sistem Informasi Manajemen juga tidak terlepas dari andil kecerdasan buatan. Adanya irisan penggunaan kecerdasan buatan di berbagai disiplin ilmu tersebut menyebabkan cukup rumitnya untuk mengklasifikasikan kecerdasan buatan menurut disiplin ilmu yang menggunakannya. Untuk memudahkan hal tersebut, maka pengklasifikasian lingkup kecerdasan buatan didasarkan pada output yang diberikan yaitu pada aplikasi komersial (meskipun sebenarnya kecerdasan buatan itu sendiri bukan merupakan medan komersial).
Lingkup utama dalam kecerdasan buatan adalah:

1. Sistem Pakar (*Expert System*). Disini komputer digunakan sebagai sarana untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah permasalahan dengan meniru keahlian yang dimiliki oleh pakar.
2. Pengolahan Bahasa alami (*Nature Language Processing*). Dengan pengolahan bahasa alami ini diharapkan user dapat berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.

3. Pengenalan Ucapan (*Speech Recognition*). Melalui pengenalan ucapan diharapkan manusia dapat berkomnikasi dengan komputer dengan menggunakan suara.
4. Robotika dan Sistem Sensor (*Robotics and Sensor Systems*).
5. Computer Vision, mencoba untuk dapat mengintepretasikan gambar atau obyek-obyek tampak melalui komputer.
6. *Intelligent Computer-aided Instruction*. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.
7. Game Playing.

Beberapa karakteristik yang ada pada system yang menggunakan artificial intelligence adalah pemogramannya yang cenderung bersifat simbolik ketimbang algoritmik, bisa mengakomodasi input yang tidak lengkap, bisa melakukan inferensi, dan adanya pemisahan antara kontrol dengan pengetahuan [9].

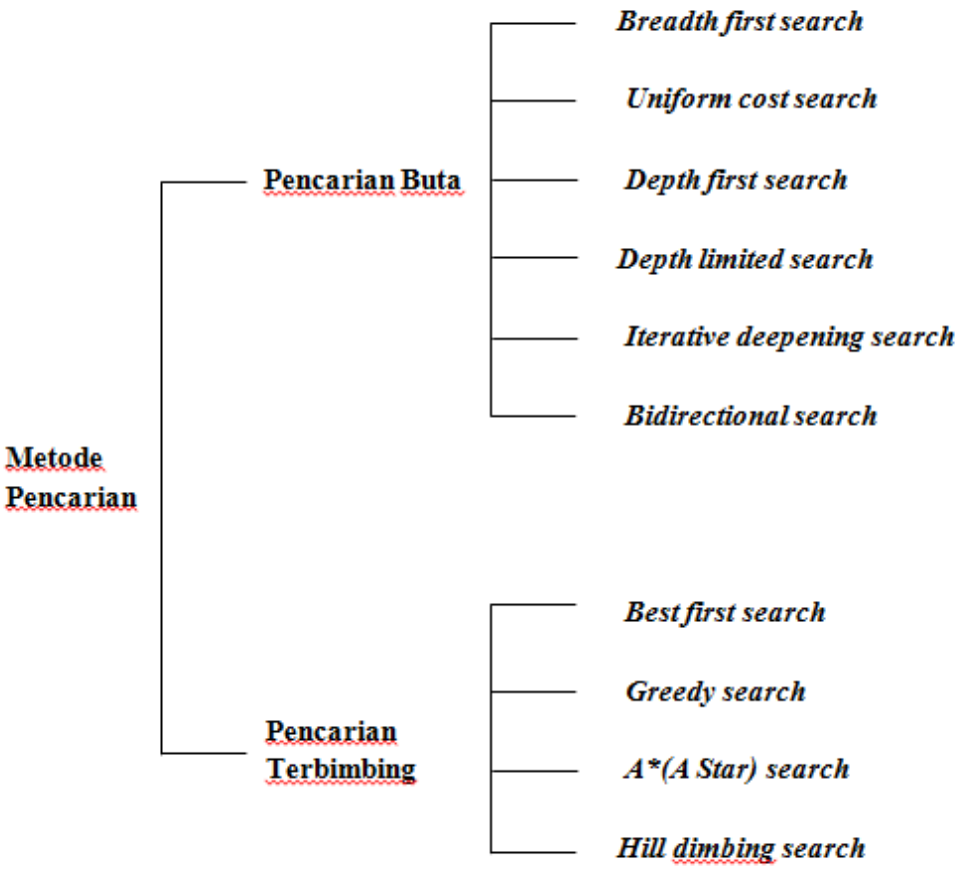
2.2.3 Algoritma Pencarian

Permasalahan pencarian dapat diselesaikan dengan beberapa metode yaitu:

1. Metode pencarian yang pertama adalah metode yang sederhana yang hanya berusaha mencari kemungkinan penyelesaian yang disebut juga pencarian buta.
2. Metode yang lebih kompleks yang akan mencari jarak terpendek. Metode ini adalah *British Museum Procedure*, *Branch* and *Bound*, *Dynamic Programming*, *Best First Search*,*Greedy Search*, *A** (A Star) *Search*, dan *Hill Climbing Search*. Metode-metode ini digunakan pada saat harga perjalanan untuk mencari kemungkinan menjadi perhitungan. Beberapa *procedure*/metode yang kita terapkan saat berhadapan dengan musuh. Prosedur ini adalah *minimax search* dan *alpha-beta pruning*. Metode ini banyak digunakan pada program-program seperti catur dan sebagainya.

Metode pencarian dikatakan penting untuk meyelesaikan permasalahan karena setiap *state* (keadaan) menggambarkan langkah-langkah untuk menyelesaikan permasalahan. Metode pencarian dikatakan penting untuk

perencanaan karena dalam sebuah permainan akan menentukan apa yang harus dilakukan, dimana setiap *state* menggambarkan kemungkinan posisi pada suatu saat. Metode pencarian adalah bagian dari kesimpulan, dimana setiap *state* menggambarkan hipotesis dalam sebuah rangkaian deduktif [10].



Gambar 2. 4 Bagan metode pencarian [10]

2.2.4 Algoritma A*

Algoritma A* merupakan perbaikan dari metode BFS dengan memodifikasi fungsi heuristicnya. A* (A Star) akan meminimumkan total biaya lintasan. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal.

Pada pencarian rute kasus sederhana, dimana tidak terdapat halangan pada peta, A* bekerja secepat dan seefisien BFS. Pada kasus peta dengan halangan, A* dapat menemukan solusi rute tanpa „terjebak" oleh halangan yang ada.

Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS, hanya saja dengan dua faktor tambahan.

1. Setiap sisi mempunyai "*Cost*" yang berbeda-beda, seberapa besar *Cost* untuk pergi dari satu simpul ke simpul yang lain.
2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian, sehingga lebih kecil kemungkinan kita mencari ke arah yang salah.

Cost untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila kita ingin mencari jalan dengan waktu tercepat untuk dilalui. Sebagai contoh, bila kita berkendaraan melewati jalan biasa bisa saja merupakan jarak terdekat, tetapi melewati jalan tol biasanya memakan waktu lebih sedikit.

Algoritma A* bekerja dengan prinsip yang hampir sama dengan BFS, kecuali dengan dua perbedaan, yaitu :

1. Simpul-simpul di list "terbuka" diurutkan oleh *Cost* keseluruhan dari simpul awal ke simpul tujuan, dari *Cost* terkecil sampai *Cost* terbesar. Dengan kata lain, menggunakan priority queue (antrian prioritas). *Cost* keseluruhan dihitung dari *Cost* dari simpul awal ke simpul sekarang (current *node*) ditambah *Cost* perkiraan menuju simpul tujuan.
2. Simpul di list "tertutup" bisa dimasukkan ke list "terbuka" bila jalan terpendek (*Cost* lebih kecil) menuju simpul tersebut ditemukan.

Karena list "terbuka" diurutkan berdasarkan perkiraan *Cost* keseluruhan, algoritma mengecek simpul-simpul yang mempunyai perkiraan *Cost* yang paling kecil terlebih dahulu, jadi algoritmanya mencari simpul-simpul yang kemungkinan mengarah ke simpul tujuan. Karena itu, lebih baik perkiraan *Cost*-nya, lebih cepat pencariannya. *Cost* dan perkiraannya ditentukan oleh kita sendiri. Bila *Cost*-nya adalah jarak, akan menjadi mudah.

Cost antara simpul adalah jaraknya, dan perkiraan *Cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan. Atau agar lebih mudahnya bisa ditunjukkan seperti berikut ini.

$$f(n) = g(n) + h(n)$$

dengan :

f(n) = fungsi evaluasi

g(n) = biaya (*Cost*) yang sudah dikeluarkan dari keadaan sampai keadaan n

h(n) = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Perhatikan bahwa algoritma ini hanya bekerja bila *Cost* perkiraan tidak lebih besar dari *Cost* yang sebenarnya. Bila *Cost* perkiraan lebih besar, bisa jadi jalan yang ditemukan bukanlah yang terpendek. *Node* dengan nilai terendah merupakan solusi terbaik untuk diperiksa pertama kali pada g(n) + h(n). Dengan fungsi *heuristic* yang memenuhi kondisi tersebut, maka pencarian dengan algoritma A* dapat optimal.

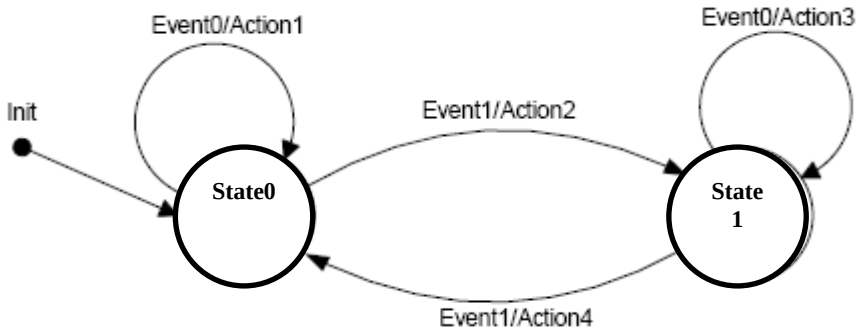
Keoptimalan dari A* ini cukup langsung untuk dianalisis apabila digunakan dengan tree search. Pada kasus ini, A* dinilai optimal jika h(n) adalah sebuah *admissible heuristic* yaitu nilai h(n) tidak akan memberikan penilaian lebih pada *Cost* untuk mencapai tujuan. Salah satu contoh dari *admissible heuristic* adalah jarak dengan menarik garis lurus karena jarak terdekat dari dua titik adalah dengan menarik garis lurus [10].

2.3. Finite State Machine (FSM)

Bahasa formal dapat dipandang sebagai entitas abstrak, yaitu sekumpulan string-string simbol alphabet tertentu. Namun bahasa juga dapat dipandang sebagai entitas-entitas abtrak yang dapat dikenali atau dibangkitkan melalui suatu mesin komputasi. Mesin yang dapat mengenali bahasa kelas ini adalah *finite state machine*.

Ada beberapa definisi mengenai *Finite State Machine* (FSM) atau sering juga disebut dengan *Finite State Automata* (FSA).

1. FSM didefenisikan sebagai perangkat komputasi yang memiliki input berupa string dan output yang merupakan satu dari dua nilai yang dapat di-*accept* dan *reject*.
2. *Finite Automata* adalah model matematika sistem dengan masukan dan keluaran diskrit. Sistem dapat berada di salah satu dari sejumlah berhingga konfigurasi internal disebut *state*.
3. FSM adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: *State* (Keadaan), *Event* (kejadian) dan *action* (aksi). Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada pada salah satu *state* yang aktif. Sistem dapat beralih atau bertransisi menuju *state* lain jika mendapatkan masukan atau event tertentu, baik yang berasal dari perangkat luar atau komponen dalam sistemnya itu sendiri. Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relatif kompleks.



Gambar 2. 5 Contoh diagram state sederhana [6]

Diagram tersebut memperlihatkan FSM dengan dua buah *state* dan dua buah *input* serta empat buah aksi *output* yang berbeda : seperti terlihat pada gambar, ketika sistem mulai dihidupkan, sistem akan bertransisi menuju *state0*, pada keadaan ini

sistem akan menghasilkan Action1 jika terjadi masukan *Event0*, sedangkan jika terjadi *Event1* maka *Action2* akan dieksekusi kemudian sistem selanjutnya bertransisi ke keadaan *State1* dan seterusnya.

Secara formal FSM dinyatakan oleh 5 tupel atau M=(Q, Σ, δ, S, F), dimana:

Q = himpunan *state*/kedudukan

Σ = himpunan *symbol input*/masukan/abjad

δ = fungsi transisi

S = *state* awal/ kedudukan awal (*initial state*), S Q

F = himpunan *state* akhir, F Q

FSM terdiri dari dua jenis, yaitu FSM ber-*output* dan FSM tidak ber-*output*. FSM tidak ber-*output* digunakan untuk pengenalan bahasa dalam komputer, dengan input yang dimasukkan akan diperoleh apakah *input* tersebut dikenal oleh bahasa komputer atau tidak. Salah satu penggunaan FSM tidak ber-*output* adalah program *compiler*, yaitu program untuk memeriksa apakah perintah yang digunakan pengguna benar atau salah. Sementara untuk FSM ber-*output* digunakan untuk merancang mesin atau sistem [6].

2.4. Budaya Indonesia

2.4.1. Kebudayaan Sumatra Utara

1. Bangsa Batak

Bangsa Batak, demikian mereka mengklaim diri sebagai sebuah bangsa, bukan suku. Karena Bangsa Batak terdiri dari berbagai suku, seperti Karo, Mandailing, Simalungun, dan lainnya.

satu kelompok rumah yang tergabung dalam sebuah huta, umumnya dibatasi oleh tanaman bambu. Menurut keyakinan mereka, bambu-bambu itu adalah benteng baik terhadap ancaman fisik maupun metafisik, seperti ilmu teluh dan sebagainya.

Jika ditilik lebih dalam lagi, rumah adat Batak juga memiliki keunikan, mulai dari arsitekturnya hingga motif yang menghiasi rumah tersebut. Menurut penuturan salah seorang narasumber dari warga setempat, dinding rumah adat

Batak dibuat miring menengadah sebagai penggambaran perahu. Karena nenek moyang Batak disinyalir datang menggunakan perahu hingga sampai di wilayah Sumatera.

Pintu rumah adat Batak dibuat kecil, sehingga orang tidak dapat langsung masuk ke rumah tanpa harus menundukan kepalanya. Makna yang dapat dipetik dari bentuk pintu ini adalah bagi setiap orang yang masuk rumah harus taat dan tunduk dengan peraturan yang berlaku di dalam rumah tersebut, siapapun orang itu.

Uniknya, rumah adat Batak atau pun Karo, selalu menggambarkan cicak di dinding rumah mereka, baik nampak seperti cicak sebenarnya ataupun bentuk yang menyerupainya. Dalam hal ini, orang Batak memiliki falsafah hidup hendaknya dapat meniru cicak, binatang yang dapat hidup di rumah mana saja. Artinya, orang Batak dapat beradaptasi dengan lingkungannya seperti hidup cicak. [11].



Gambar 2. 6 Rumah Adat Batak

2. Rumoh Aceh

Rumah Aceh yang dibangun menyerupai rumah tempat tinggal tradisional masyarakat Aceh, berbentuk rumah panggung. Lantai bangunan ini dirancang setinggi 9 kaki atau lebih dari permukaan tanah. Bersandar pada tiang-tiang penyangga dari kayu dengan ruang kolong di bawahnya.

Luas lantai bangunan ini lebih dari 200 m2 dengan tinggi atap pada bagian rabung lebih kurang 8 m. Keistimewaan "Rumah Aceh" dan sejenisnya terletak pada segi kekokohan bangunannya; walaupun bagian-bagian rumah hanya dipersatukan dengan ikatan tali ijuk, pasak serta baji sebagai pangganti paku dan sekrup.

Tiang-tiang rumah ini terbuat dari jenis kayu keras pilihan yang rata-rata berdiameter lebih kurang 20 cm, dan berjumlah 44 buah tegak berjajar dalam posisi 4 x 11 memanjang dari Timur ke Barat. Penempatan tangga dengan jumlah anak tangga genap masing-masing 14 buah, di ujung Timur bawah "seuramoe keue" dan di ujung Barat bawah "seuramoe likot", berkesan tidak biasa.

Dalam masyarakat Aceh tidak dikenal adanya istilah rumah adat. Fungsi masing-masing ruangan ini ditata agar sedapat mungkin menggambarkan fungsi pokok-pokok ruangan pada rumah tempat tinggal tradisional masyarakat Aceh. [12]



Gambar 2. 7 Rumoh Aceh

2.5. Tools yang digunakan

2.5.1. Object Oriented Programming (OOP)

Object Oriented Programming (OOP) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri [13].

2.5.2. UML (Unified Modeling Language)

Unified Modelling Language (UML) menurut [14] adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Konsepsi dasar UML dapat dirangkum seperti dalam Tabel 1.1.

Tabel 2. 1 Konsep Dasar UML

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Abstraksi konsep dasar UML yang terdiri dari *structural classification, dynamic behavior*, dan *model management*, bisa kita pahami dengan mudah

apabila kita melihat gambar diatas dari Diagrams. Main concepts bisa kita pandang sebagai term yang akan muncul pada saat kita membuat diagram. Dan *view* adalah kategori dari diagaram tersebut.

Seperti juga tercantum pada gambar diatas UML mendefinisikan diagram-diagram sebagai berikut:

1. use case diagram
2. *class* diagram
3. *state*chart diagram
4. *Activity* diagram
5. *sequence* diagram
6. collaboration diagram
7. component diagram
8. deployment diagram.

2.5.3. Java

Java menurut [13]. adalah bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naugton, Chris Warth, Ed Frank, dan Mike Sheridan di suatu perusahaan perangkat lunak yang bernama Susn Microsystems, pada tahun 1991. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama "Oak", namun oada tahun 1995 diganti namanya menjadi "Java".

Alasan utama pembentukan bahasa Java adalah untuk membuat aplikasiaplikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control,* sehingga Java harus bersifat porTabel atau yang sering disebut dengan platform independent (tidak bergantung pada platform). Itula yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah *'write once, run everywhere*', yang berarti kode program hanya ditulis sekali, namun dapat dijalankan di bawah platform manapun, tanpa harus melakukan perubahan kode program.

1. Java 1

   Pada awalnya perilisannya, versi Java masih disebut dengan JDK (*Java Development Kit*). Dalam JDK, semua kebutuhan untuk pengembangan program dan eksekusi program masih tergabung jadi satu. Penamaan ini berlaku sampai dengan Java 1.1. Namun sekarang, setelah Java 1.2, Sun Mycrosystems menamainya dengan JSDK (*Java Software Development Kit*) dalam hal ini kebutuhan untuk pengembangan program dipisahkan dengan kebutuhan eksekusi. Bagian software yang digunakan untuk kebutuhan eksekusi program disebut dengan JRE (*Java-Runtime Environment*). Selanjutnya, Java 1.2 disederhanakan penamaannya menjadi "Java 2".

2. Java 2

   Sun Myrosystems telah mendefinisikan tiga buah edisi dari Java 2 yaitu sebagai berikut :

   a. *Java 2 Standard Edition* (J2SE), yang digunakan untuk mengembangkan aplikasi-aplikasi desktop dan applet (aplikasi untuk Java yang dapal dijalankan di dalam browser web).

   b. *Java 2 Enterprise Edition* (J2EE), merupakan superset dari J2SE yang memperbolehkan kita untuk mengembangkan aplikasi-aplikasi berskala besar(*enterprise*), yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi server dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi web dengan menggunakan Servlet dan JSP (*JavaServer Pages*) dan teknologi lainnya seperti CORBA (*Common Object Request Broker Architecture*) dan XML (*Extensible Markup Language*).

   c. *Java 2 Micro Edition* (J2ME), merupakan subset dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh seperti pada teknologi mobile.

2.5.4. Unity3D

Unity adalah game engine buatan Unity Technologies Inc. Setelah mengalami masa pengembangan selama 4 tahun, pada tahun 2005 Unity pertama kali diluncurkan. Tidak seperti saat ini pada waktu itu Unity sangat minim fitur jika dibandingjkan dengan unity 3. Game engine yang bisa berjalan native baik pada windows platform maupun Mac OS X, ini memungkinkan "seamless collaboration antara developer pengguna Windows dan OSX [15].



Gambar 2. 8 Logo Unity3D

2.5.5. Autodesk 3ds Max

3D Studio Max merupakan program standar animasi berbasis Windows yang telah diakui dan banyak dipergunakan untuk membuat animasi professional oleh perusahaan-perusahaan animasi tingkat internasional. 3D Studio Max telah digunakan secara luas dibanyak film 3 dimensi (3D), game, iklan dan sebagainya [16].



Gambar 2. 9 Logo Autodesk 3Ds Max