BAB 2

LANDASAN TEORI

2.1 Pengertian *Game*

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." [3]

Game merupakan permainan komputer yang dibuat dengan teknik dan metode animasi [4]. *Game* merupakan bidang AI yang sangat populer berupa permainan antara manusia melawan mesin yang mempunyai intelektual untuk berpikir. Komputer dapat bereaksi dan menjawab tindakan-tindakan yang diberikan oleh lawan mainnya.

Salah satu komputer yang ditanamkan AI untuk *game* bernama *Deep Blue*. *Deep Blue* adalah sebuah komputer catur buatan IBM pertama yang memenangkan sebuah permainan catur melawan seorang juara dunia (Garry Kasparov) dalam waktu standar sebuah turnamen catur. Kemenangan pertamanya

(dalam pertandingan atau babak pertama) terjadi pada 10 Februari 1996, dan merupakan permainan yang sangat terkenal.

Kini telah banyak berkembang *game* AI yang semakin menarik, interaktif, dan dengan grafis yang sangat bagus. Ditambah dengan kemajuan teknologi jaringan komputer yang semakin cepat, sudah banyak terdapat *game-game* AI yang berbasiskan *online*. Tidak sedikit orang yang tertarik dengan *game* saat ini. Mereka memainkan *game* untuk mengisi kekosongan waktu mereka atau pun melatih *skill* mereka dalam berpikir.

Ada banyak sekali jenis/*genre game*. *Genre* bisa berarti format atau gaya dari sebuah *game*. Fromat sebuah *game* bisa murni sebuah *genre* atau bisa merupakan camputan (*hybrid*) dari beberapa *genre* lain. Berikut ini adalah *genre game* [5]:

a. *Maze game*, jenis *game* ini adalah jenis *game* yang paling awal muncul. Contoh yang paling popiler di indonesia adalah Pacman dan Digger.
b. *Board game*, jenis *game* ini sama dengan *game board* tradisional seperti Monopoly.
c. *Card game*, *game* ini juga sama dengan versi tradisionalnya seperti Solitaire dan Hearts.
d. *Beattle card game*, *game* ini jarang masuk di Indonesia, contoh populernya adalah Battle Card Pokemon.
e. *Quiz game*, jenis ini juga agak jarang di indonesia. Salah satu yang umum dikenal adalah kuis Who Wants to Be a Millionere, sebuah *game* dengan nama yang sama dari acara kuis televisi. Cara bermain *game* ini sangat

sederhana yaitu dengan memilih jawaban yang benar dari beberapa pilihan jawaban.

f. *Puzzle game*, *game* jenis ini memberikan tantangan kepad pemainnya dengan cara menjatuhkan sesuatu dari sisi sebelah atas ke bawah. Contoh *game* ini adalah Tetris.

g. *Shoot then up*, jenis ini banyak diminati karena mudah dimainkan. Misi *game* ini biasanya hanya menembaki lawan secepat dan sebanyak-banyaknya yang biasanya datang dari arak atas.

h. *Sice scroller game*, *game* ini didasarkan pada 2D dimana pemain bergerak sepanjang jalur permainan ke satu arah dan menyelesaikan tugasnya. Ada yang meloncat, berlari, mengendap, dan menghindari halangan. Contoh *game* yang terkenal adalah Duke Nukem Asli, Commander Ken, Prince or Persia, Sonic the Hedgehog, dan sebagainya.

i. *Fighting game*, sesuai dengan namanya *game* ini merupakan *game* pertarungan yang bisa mengkombinasikan berbagai geraakan dalam pertarungancontoh *game* yang terkenal adalah Street Fighter 2, Samurai Showdown, Virtual Fighter, dan sebagainya.

j. *Racing game*, *game* balapan yang memberikan lomba kecepatan dari kendaraan yang dimainkan oleh pemain. Beberapa contoh *game* yang terkenal adalah Need for Speed Underground dan Toca Race Driver.

k. *Flight Sim*, *game* ini berfokus pada simulasi penerbangan. Simulasi yang diberikan meniru kondisi dari penerbangan sebenarnya, baik kondisi pesawat dan peralatannya maupun kondisi pemandangannya. Contoh *game* yang

terkenal dari *genre* ini adalah Microsoft Flight Simulator, IL-2 Sturmovik, dan sebagainya.

l. *Turn-base strategy* (TBS) *game*, *game* ini memerlukan strategi dari pemain untuk memenagkan permainan. Pemain melakukan gerakan setelah pemain lain melakukannya, jadi saling bergantian. Contoh *game* yang terkenal adalah Empire dan Civilization.

m. *Real-time strategy* (RTS) *game*, hampir sama dengan TBS namun tidak perlu menunggu. Malah yang tercepatlah yang akan menang, seperti pada *game* Warcraft.

n. *Sim*, *genre* ini mengetengahkan permainan simulasi yang berbeda denga flight simulator. Pada *game* ini pemain membangun secara simulasi kota, negara, atau koloni. Pemain berperan sebagai "Dewa" yang mengatur berbagai sumber daya dan mengambil keputusan.

o. *First person shooter*, contah *game* pada *genre* ini adalah Doom dan Counterstrike.

p. *Roll playing game*, di *genre* ini pemain berperan menjadi sebuah karakter. Pemain akan menjalankan perannya dengan berbagai atribut seperti kesehatan, intelegensi, kekuatan dan sebagainya seperti pada *game* Ultima.

q. *Adverture game*, merupakan *game* petualangan seperti *game* Beyond Good and Evil.

r. *Full motion video game* (FMV), pernah membaca buku "Pilih Sendiri Petualanganmu"? *genre game* ini adalah versi elektronik dari buku jenis itu.

Pemain bebas memilih langkah selanjutnya sampai bisa memecahkan misteri yang ada. Contoh *game* populer adalah Myst dan Riven.

s. *Educational* dan *edutaument*, *game* yang bertujuan untuk memancing minat belajar sambil bermain.
t. *Sports*, *game* yang mengetengahkan *genre* olahraga.

2.2 Edukasi

Edukasi adalah sesuatu yang bersifat menyeluruh. Edukasi tidak hanya berupa pelajaran ataupun diktat kuliah. Edukasi atau pendidikan memiliki makna yang lebih dalam daripada hanya sekedar mengajari. Mendidik berarti membentuk karakter dan pola pikir seseorang. Dalam hal ini tentunya ke arah yang baik. Ketika seseorang meniru perilaku sesuatu atau seseorang, ini juga merupakan bagian dari pendidikan [6].

Maka daripada itu, sebenarnya, implementasi pendidikan dalam *game* tidak hanya mengacu pada pelajaran semata. Ia harus mencakup semua tatanan pendidikan,meski dengan spesifikasi tertentu.

Penggunaan *game* sebagai sarana pendidikan sebetulnya bukan merupakan hal yang tabu ataupun salah. Karena *game* bersifat *entertaint* atau menghibur, yang disini adalah bermain. Psikologi manusia adalah lebih suka bermain daripada belajar serius. Dalam *game*, pendidikan diberikan lewat praktek atau pembelajaran dengan praktek (*learning by doing*). Dalam *game*, pemain seolah masuk ke dalam dunia baru tempat mereka bisa melakukan apa saja.

Game secara tidak langsung mendidik manusia lewat apa yang mereka kerjakan dalam *game* tersebut. Apa yang merek akerjakan dalam *game* tersebut mempengaruhi pola pikir dan perilaku mereka.

2.3 Kecerdasan Buatan

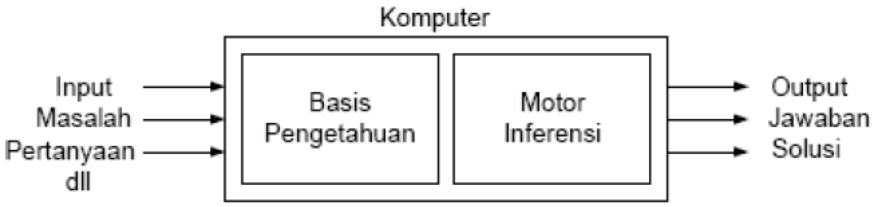
Banyak cara untuk mendefinisikan kecerdasan buatan atau *Artificial Intelegence* (AI), diantaranya adalah [7]:

1. Suatu studi yang mengupayakan bagaimana agar komputer berlaku cerdas.
2. Studi yang membuat komputer dapat menyelesaikan persoalan yang sulit.
3. Teknologi yang mensimulasikan kecerdasan manusia, yaitu bagaimana mendefinisikan dan mencoba menyelesaikan persoalan menggunakan komputer dengan meniru bagaimana manusia menyelesaikan dengan cepat.

Dengan beberapa definisi diatas, kecerdasan buatan menawarkan baik media maupun uji teori kecerdasan. Teori-teori ini dapat dinyatakan dalam bahasa pemograman komputer dan dibuktikan eksekusinya pada komputer nyata.

Kecerdasan Buatan Mempunyai 2 bagan utama, yaitu :

1. Basis Pengetahuan (*Knowledge Base*) : berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
2. Motor Inferensi (*Inference Engine*) : kemampuan untuk menarik kesimpulan berdasarkan pengalaman.



Gambar 2.1 Penerapan konsep kecerdasan buatan di komputer [8]

Kecerdasan buatan (AI) dapat dipandang dalam berbagai perspektif [9].

1. Dari perspektif kecerdasan (*intelligence*)

   AI adalah bagaimana membuat mesin yang "cerdas" dan dapat melakukan hal-hal yang sebelumnya dapat dilakukan oleh manusia

2. Dari perspektif bisnis

   AI adalah sekelompok alat bantu (*tools*) yang berdaya guna, dan metodologi yang menggunakan *tool-tool* tersebut guna menyelesaikan masalah-masalah bisnis.

3. Dari perspektif pemrograman (*programming*)

   AI termasuk didalamnya adalah studi tentang pemrograman simbolik, pemecahan masalah proses pencarian (*search*). Umumnya program AI lebih fokus pada simbol-simbol daripada pemrosesan numerik (huruf, kata, angka untuk merepresentasikan obyek, proses dan hubungannya).

   Bahasa pemrograman AI :

   - LISP, dikembangkan awal tahun 1950-an, bahasa pemrograman pertama yang diasosiasikan dengan AI.
   - PROLOG, dikembangkan pada tahun 1970-an.

- Bahasa pemrograman berorientasi obyek (Object Oriented Programming (Objective C, C++, Smalltalk, Java)

4. Dari perspektif penelitian (*research*)

   Riset tentang AI dimulai pada awal tahun 1960-an, percobaan pertama adalah membuat program permainan (*game*) catur, membuktikan teori, dan *general problem solving* (untuk tugas-tugas sederhana).

Kecerdasan buatan dapat dibagi ke dalam tiga domain, yaitu [10]:

1. *Mundane task*

   Secara harfiah, arti *mundane* adalah keduniaan. Di sini, AI digunakan untuk melakukan hal-hal yang sifatnya duniawi atau melakukan kegiatan yang dapat membantu manusia. Contohnya :

   - Persepsi (*vision and speech*).
   - Bahasa alami (*understanding, generation and translation*).
   - Pemikiran yang bersifat *commonsense*.
   - *Robot control.*

2. *Formal task*

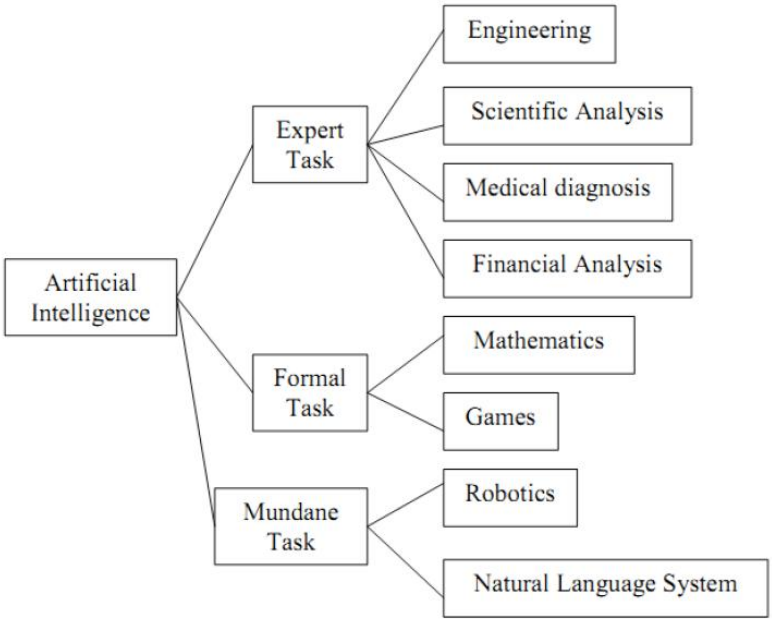
   AI digunakan untuk melakukan tugas-tugas formal yang selama ini manusia biasa lakukan dengan lebih baik. Contohnya :

   - Permainan/*games*.
   - Matematika (geometri, logika, kalkulus, integral, pembuktian).

3. *Expert task*

AI dibentuk berdasarkan pengalaman dan pengetahuan yang dimiliki oleh para ahli. Penggunaan ini dapat membantu para ahli untuk menyampaikan ilmu-ilmu yang mereka miliki. Contohnya :

- Analisis finansial
- Analisis medikal
- Analisis ilmu pengetahuan
- Rekayasa (desain, pencarian, kegagalan, perencanaan, manufaktur)



Gambar 2.2 Bidang-bidang tugas (*task domains*) dari AI [11]

2.4 Kecerdasan Buatan dalam *Game*

Salah satu unsur yang berperan penting dalam sebuah *game* adalah kecerdasan buatan. Dengan kecerdasan buatan, elemen-elemen dalam *game* dapat berperilaku sealami mungkin layaknya manusia.

Game AI adalah aplikasi untuk memodelkan karakter yang terlibat dalam permainan baik sebagai lawan, ataupun karakter pendukung yang merupakan bagian dari permainan tetapi tidak ikut bermain (NPC = *Non Playable Character*). Peranan kecerdasan buatan dalam hal interaksi pemain dengan permainan adalah pada penggunaan interaksi yang bersifat alami yaitu yang biasa digunakan menusia untuk berinteraksi dengan sesama manusia. Contoh media interaksi ialah:

- Penglihatan (*vision*)
- Suara (*voice*), ucapan (*speech*)
- Gerakan anggota badan (*gesture*)

Untuk pembentukan kecerdasan buatan pada *game* ternyata digunakan pula algoritma, yaitu jenis pohon aray untuk suatu struktur. Implementasi pohon (*tree*) ini biasa disebut *game tree*. Berdasarkan *game tree* inilah sebuah *game* disusun algoritma kecerdasan buatannya. Kecerdasan buatan yang disematkan dalam sebuah *game* yang membentuk analisis *game tree* biasanya merepresentasikan kondisi atau posisi permainan dari *game* sebagai suatu *node*, dan merepresentasikan langkah yang mungkin dilakukan sebagai sisi berarah yang menghubungkan *node* kondisi tersebut ke anak (*child*) sebagaimana representasi suatu pohon (*tree*).

Namun, biasanya representasi langsung tersebut mempunyai kelemahan, yaitu representasi data pohon akan menjadi sangat lebar dan banyak. Mungkin bagi sebuah mesin komputer mampu melakukan kalkulasi sebanyak apapun masalah, namun *game tree* yang lebar dan besar memberikan beberapa masalah, antara lain konsumsi proses memori, kapasitas penyimpanan yang cukup besar dan kinerja

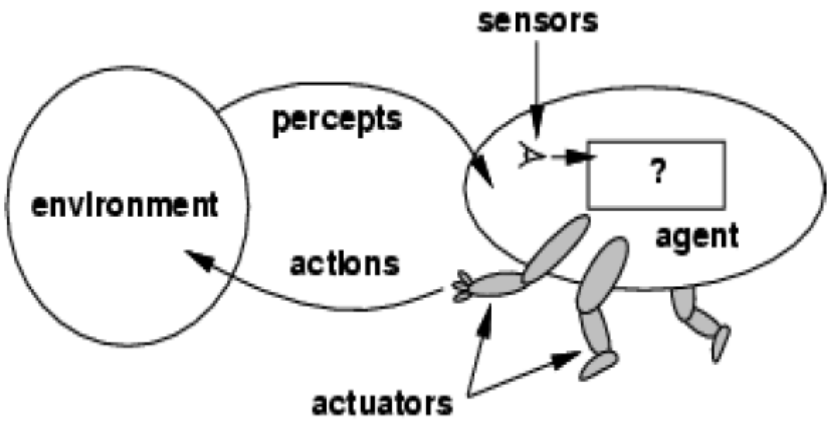
yang kurang pada konsol *game* berspesifikasi rendah. Karena itu dibentuklah beberapa algoritma dan penyederhanaan bagi sebuah *game tree*.

2.5 Agen Cerdas (*Intelligent Agent*)

Desiani dan Arhami memuat kutipan dari Okamoto dan Takaoka (1997) menyatakan bahwa agen dapat dipandang sebagai sebuah objek yang mempunyai tujuan dan bersifat autonomous (memberdayakan *resource*-nya sendiri) untuk memecahkan suatu permasalahan melalui interaksi, suatu kolaborasi, kompetisi, negosiasi dan sebagainya.

Agen adalah sesuatu yang seolah-olah merasakan sesuatu dari lingkungannya melalui sensor dan memberikan aksi balasan kepada lingkungannya tersebut melalui *effector* (*actuator*). Dan kumpulan dari beberapa agen yang berada pada lingkungan kerja yang sama disebut *multi-agent* [12].

Sebuah agen dibuat menyerupai manusia (human agent) memiliki sensor berupa mata, telinga, dan organ lain serta effector yang berupa kaki, mulut dan lainnya. Berbeda dengan agen robot yang menggunakan kamera dan sinar infrared dalam jangkauan tertentu sebagai sensor dan berbagai mesin (motor) sebagai *effector*. Berikut ini adalah ilustrasi diagram agen.



Gambar 2.3 Ilustrasi Diagram Agen [13]

Kecerdasan buatan berperan penting dalam mendesain program bagi agen. Program ini berfungsi mengimplementasikan pemetaan percepts ke agen. Program yang dibangun harus menyatu dengan *computing device* yang disebut arsitektur untuk menerima dan menjalankan agen. Arsitektur dapat berupa *hardware* seperti kamera *image, filtering audio input* dan sebagainya.

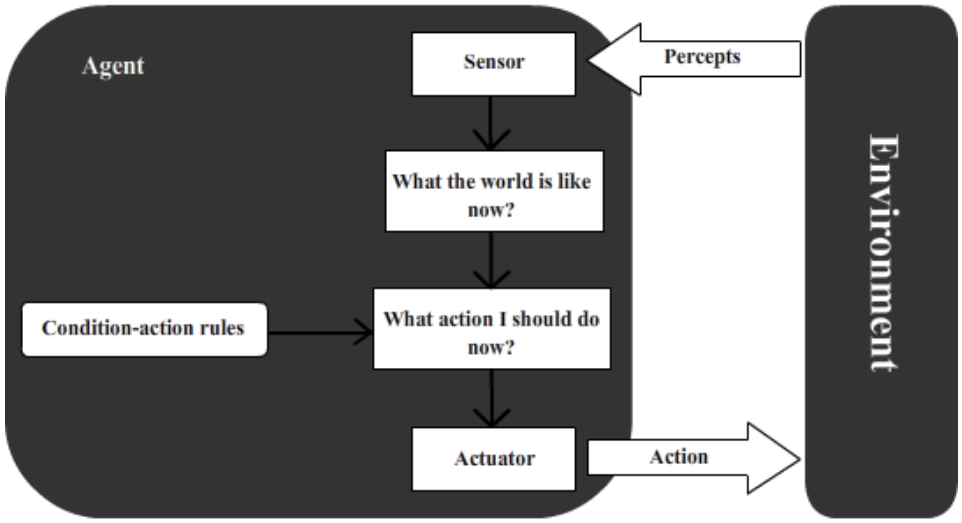
Hubungan antara agen, arsitektur, dan program dapat disimpulkan sebagai berikut : Agen = arsitektur + program

Sebelum mendesain agen, harus dideskripsikan terlebih dahulu mengenai tipe agen, *percepts* dan *action*, tujuan atau *performance measure* yang akan dicapai serta lingkungan tempat agen beroperasi.

Ada 4 (empat) tipe dasar pada program agen yang mewujudkan sistem cerdas, yakni:

- **Agen Refleks Sederhana (*Simple Reflex Agent*)**

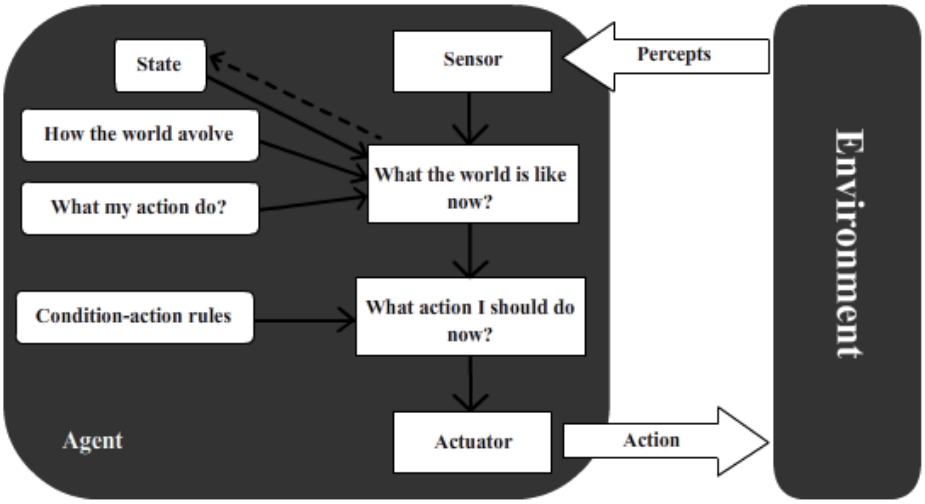
  Agen refleks sederhana ini merupakan tipe agen yang paling sederhana. Agen ini memilih tindakan (action) atas dasar persepsi (percept) yang diterimanya. Tipe ini dapat pula disebut *a condition-action rule*



Gambar 2.4 Skema Agen Refleks Sederhana [13]

- **Agen Refleks Berbasis Model**

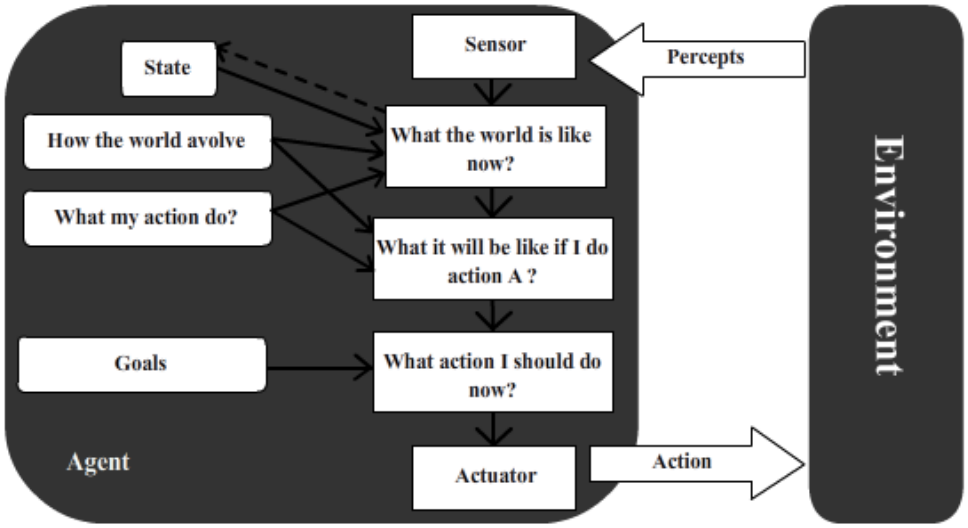
  Agen ini harus menjaga keadaan internal yang bergantung pada persepsi lalu untuk merefleksikan setidaknya beberapa aspek keadaan sekarang yang tidak terobservasi. Pembaharuan (*update*) informasi dari internal state terus berjalan, dengan demikian dibutuhkan dua jenis pengetahuan yang harus dikodekan ke program. Pertama, informasi mengenai bagaimana lingkungan mempengaruhi kebebasan agen, dan kedua informasi mengenai bagaimana agen melakukan aksinya.



Gambar 2.5 Skema Agen Refleks Berbasis Model [13]

- **Agen Berbasis Tujuan (Goal Based Agent)**

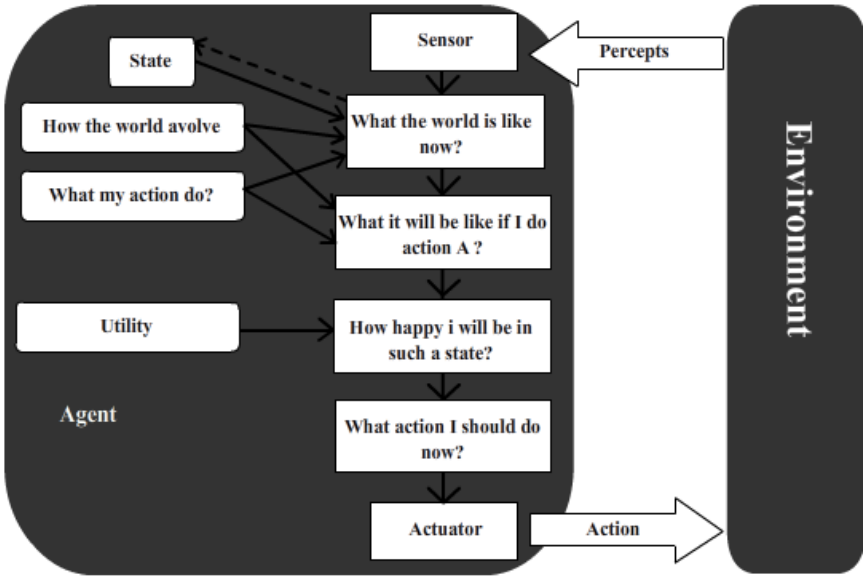
  Program agen dapat dikombinasikan dengan informasi mengenai hasil dari aksi yang mungkin dilakukan. Tipe ini turut mempertimbangkan akibat yang diberikan serta hasil yang dicapai atas aksi yang dilakukan. Pada dasarnya pengetahuan agen akan keadaan lingkungannya tidak selalu cukup untuk memutuskan aksi yang akan dilakukan. Dengan kata lain, selain keadaan sekarang, agen juga memerlukan beberapa informasi tujuan yang menerangkan tentang tujuan kondisi yang dikehendaki. Pencarian dan perencanaan adalah dua hal yang dilakukan untuk mencapai tujuan agen. Meskipun tipe ini terlihat kurang efisien, namun sangat fleksibel. Agen secara otomatis melakukan aksi yang relevan apabila terjadi perubahan kondisi, begitu pula bila tujuannya diperbaharui maka agen akan membangkitkan aksi yang baru pula.



Gambar 2.6 Skema Diagram Agen Berbasis Tujuan [13]

- **Agen Berbasis Kegunaan (Utility Based Agent)**

  Tipe ini merupakan pengembangan dari tipe berbasis tujuan. Tujuan dianggap tidak cukup untuk membangkitkan perilaku agen berkualitas tinggi. *Utility* memiliki fungsi yang dapat memetakan suatu keadaan ke dalam bilangan riil, yang menerangkan derajat pencapaian keberhasilan. Spesifikasi lengkap fungsi *utility* mengizinkan keputusan rasional dalam dua jenis kasus dimana agen menghadapi masalah, sehingga tujuannya tidak tercapai. Pertama, ketika terjadi konflik tujuan dimana hanya beberapa saja yang terpenuhi, fungsi *utility* menspesifikasikan tukar tambah yang sesuai. Kedua, ketika terdapat beberapa tujuan yang dapat dilakukan agen, namun tidak dapat ditentukan mana tujuan yang berhasil dicapai, dalam hal ini fungsi *utility* menyediakan kemungkinan bobot kesuksesan dari masing-masing tujuan.



Gambar 2.7 Skema Diagram Agen Berbasis Kegunaan [13]

2.6 Teknik-Teknik Dasar Pencarian

Pencarian atau pelacakan merupakan salah satu teknik untuk menyelesaikan permasalahan kecerdasan buatan. Keberhasilan suatu sistem salah satunya ditentukan oleh kesuksesan dalam pencarian dan pencocokan. Teknik dasar pencarian memberikan suatu kunci bagi banyak sejarah penyelesaian yang penting dalam bidang kecerdasan buatan. Ada beberapa aplikasi yang menggunakan teknik pencarian ini, yaitu :

1. Papan *game* dan *puzzle* (*tic-tac-toe*, catur, menara hanoi).
2. Penjadwalan dan masalah *routing* (*travelling salesman problem*).
3. *Parsing* bahasa dan inteprestasinya (pencarian struktur dan arti).
4. Logika pemrograman (pencarian fakta dan implikasinya).
5. *Computer vision* dan pengenalan pola.
6. Sistem pakar bebasis kaidah *(rule based expert system)*.

Pencarian adalah proses mencari solusi dari suatu permasalahan melalui sekumpulan kemungkinan ruang keadaan *(state space)*. Ruang keadaan merupakan suatu ruang yang berisi semua keadaan yang mungkin. Kondisi suatu pencarian meliputi :

1. Keadaan sekarang atau awal.
2. Keadaan tujuan-solusi yang dijangkau dan perlu diperiksa apakah telah mencapai sasaran.
3. Biaya atau nilai yang diperoleh dari solusi.

Solusi merupakan suatu lintasan dari keadaan awal sampai keadaan tujuan. Secara umum, proses pencarian dapat dilakukan seperti berikut :

1. Memeriksa keadaan sekarang atau awal.
2. Mengeksekusi aksi yang dibolehkan untuk memindahkan ke keadaan berikutnya.
3. Memeriksa jika keadaan baru merupakan solusinya. Jika tidak, keadaan baru tersebut menjadi keadaan sekarang dan proses ini diulangi sampai solusi ditemukan atau ruang keadaan habis terpakai.

2.6.1 Masalah Pencarian

Masalah pencarian merupakan proses pencarian solusi yang direncanakan, yang mencari lintasan dari keadaan sekarang sampai keadaan tujuan. Suatu masalah pencarian direpresentasikan sebagai graf berarah. Keadaan direpresentasikan sebagai simpul *(node)*, sedangkan langkah yang dibolehkan atau

aksi direpresentasikan dengan busur *(arc).* Dengan demikian, secara khusus masalah pencarian didefinisikan sebagai berikut :

1 *State space* (ruang keadaan).

2 *Start node* (permukaan simpul).

3 Kondisi tujuan dan uji untuk mengecek apakah kondisi tujuan ditemukan atau tidak.

4 Kaidah yang memberikan bagaimana mengubah keadaan.

Terdapat beberapa cara untuk merepresentasikan ruang keadaan, antara lain pohon *AND/OR,* graph keadaan, dan pohon pelacakan.

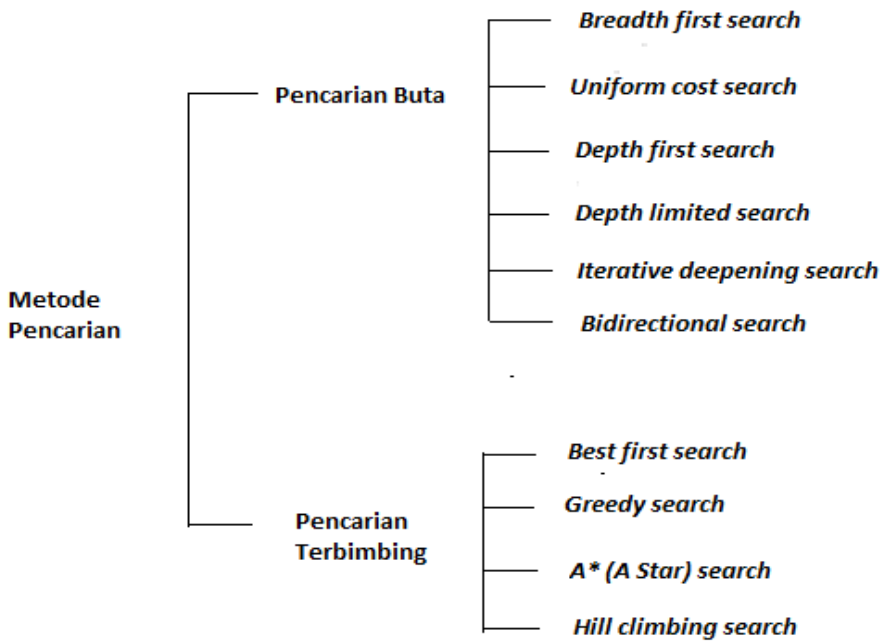
2.6.2 Algoritma Pencarian

Permasalahan pencarian adalah merupakan yang sering dijumpai oleh peneliti dibidang kecerdasan buatan. Permasalahan ini merupakan hal penting dalam menentukan keberhasilan sistem kecerdasan buatan. 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. Gambar 2.8 merupakan bagan metode pencarian.



Gambar 2.8 Bagan metode pencarian (*searching*) [14]

2.7 Algoritma A* (*A Star*)

Algoritma A* merupakan perbaikan dari metode BFS (*Best-First Search*) dengan memodifikasi fungsi *heuristic*-nya. 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.

$$f(n) = g(n) + h(n) \quad ...[13]$$

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.

Performansi algoritma A* dapat diketahui dengan melihat perbandingan metode dari tabel 2.1.

Tabel 2.1 Perbedaan ketiga algoritma

No.	Nama algoritma	Metode	Keterangan
1.	*Optimal Search (Djikstra's Algorithm)*	*g(ni)=g(n)+c(n,ni)*	*g(n)* adalah *cost* dari IS ke *node n* *c(n,ni)* adalah *cost* dari *node n* ke *ni*
2.	*Best First Search*	*h(n)*	*h(n)* adalah estimated cost dari jalur terpendek dari node n ke GS

3.	*A**	*f(n)=g(n)+h(n)*	*f(n)* adalah fungsi *heruistic* *g(n)* adalah *cost* dari IS ke *node n* *h(n)* adalah *estimated cost* dari jalur terpendek dari node n ke GS

Dari informasi pada tabel 2.1 di atas dapat dibuktikan bahwa bila :

a. $\boldsymbol{g}(\boldsymbol{n}) = \mathbf{0}$, maka $\boldsymbol{f}(\boldsymbol{n}) = \boldsymbol{h}(\boldsymbol{n})$ sehingga algoritma A* akan bertingkah laku sebagaimana *Best First Search*.

b. $\boldsymbol{h}(\boldsymbol{n}) = \mathbf{0}$, maka $\boldsymbol{f}(\boldsymbol{n}) = \boldsymbol{g}(\boldsymbol{n})$ sehingga algoritma A* akan bertingkah laku sebagaimana *Optimal Search (Dijkstra's Algorithm)*.

Dengan demikian dapat diambil kesimpulan bahwa algoritma A* mengkombinasikan kelebihan dari algoritma *Optimal Search* dan *Best First Search*.

2.7.1. Kompleksitas Algoritma A* (*A Star*)

Kompleksitas waktu dari algoritma A* tergantung dari *heuristic*-nya. Dalam kasus terburuk (*worst case*), jumlah simpul yang diekspansi bisa eskponensial dalam solusinya (jalan tependek). Akan tetapi, kompleksitasnya bisa berupa polinomial bila fungsi heuristik *h* bertemu kondisi berikut:

$$| h(x) - h^*(x) | = O(\log h^*(x)) \text{ ...[13]}$$

Dimana *h** adalah heuristik optimal, atau *cost* pasti untuk menuju tujuan dari *x*. Dengan kata lain, kesalahan (*error*) dari *h* tidak boleh tumbuh lebih cepat dari

algoritma "*perfect heuristic*" *h** yang mengembalikan jarak sebenarnya dari *x* menuju tujuan.

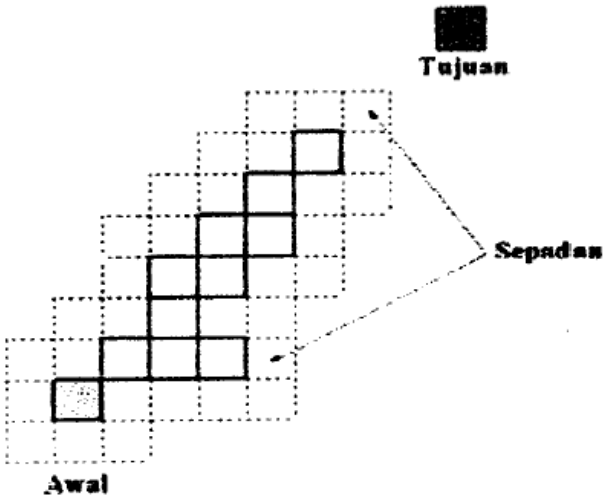
2.7.2. Efisien Waktu Algoritma A* (*A Star*)

Dengan digunakannya fungsi *heuristic* H(n), algoritma A* dapat memfokuskan pencarian pada *node-node* yang berada pada arah yang mendekati *node* tujuan. Kemudian pencarian diterminasikan pada waktu *node* tujuan diperiksa. Hal ini dapat meminimalisasikan jumlah *node* yang harus diperiksa dan arena waktu yang diperlukan untuk mendapatkan jalur berbanding lurus dengan jumlah *node* yang diperiksa, maka waktu pencarian dapat diminimalisasikan.

Walaupun jumlah *node* yang diperiksa dapat diminimalisasikan, algoritma A* mempunyai kasus terburuk. Pada kasus ini, sebagian besar ataupun keseluruhan *node* pada jalan diperiksa, sehingga algoritma A* bekerja seperti algoritma dijkstra atau BFS *(Best-First-Search).* Ada dua hal yang dapat menyebabkan keadaan terburuk ini, yaitu keadaan sepadan dan jika jalur yang dicari tidak ditemukan.

2.7.3. Keadaan Sepadan pada Algoritma A* (*A Star*)

Jika dua atau lebih *node* yang diperiksa mempunyai harga *f(n)* yang sama, maka keadaan sepadan *(tie)* terjadi. Hal ini dimungkinkan karena *f(n)* bergantung pada dua fungsi, yaitu fungsi *g(n)* dan *h(n).* Hal ini sangat mungkin terjadi antara *node-node* yang letaknya berjauhan, dan kemungkinan besar *node* yang satu terletak dekat *node* tujuan sedangkan yang lainnya terletak jauh dari *node* tujuan.



Gambar 2.9 Keadaan sepadan

Karena algoritma A* memberikan prioritas berdasarkan harga *f(n)*, maka jika keadaan sepadan terjadi, terdapat lebih dari satu *node* dengan prioritas sama. Akibatnya adalah *node-node* tersebut akan diperiksa lebih dulu, yang mungkin *node* tersebut terletak berjauhan dengan *node* tujuan. Hal ini berakibat turunnya kinerja algoritma A*.

2.7.4. Fungsi *Heuristic*

A* merupakan algoritma pencarian yang menggunakan fungsi *heuristic* untuk "menuntun" pencarian rute, khususnya dalam hal pengembangan dan pemeriksaan *node-node* pada peta. Dalam aplikasi ini, fungsi *heuristic* dipakai untuk pencarian rute mengisi nilai/notasi h pada algoritma A*. Ada beberapa fungsi *heuristic* umum yang bisa dipakai untuk algoritma A* ini. Salah satunya adalah yang dikenal dengan istilah "*Manhattan Distance*". Fungsi *heuristic* ini digunakan untuk kasus dimana pergerakan pada peta hanya lurus (horizontal atau vertikal), tidak diperbolehkan pergerakan diagonal.



Gambar 2.10 Rute dengan langkah diagonal tidak diperbolehkan

Perhitungan nilai *heuristic* untuk *node* ke-n menggunakan *Manhattan Distance* adalah sebagai berikut :

$$h(n) = (abs(n.x - goal.x) + abs(n.y - goal.y)) \quad ...[15]$$

Dimana h(n) adalah nilai *heuristic* untuk *node* n, dan *goal* adalah *node* tujuan.

2.8 Alat Pemodelan Sistem

Alat-alat pemodelan sistem informasi sangat dibutuhkan dalam proses analisis dan perancangan sistem. Alat-alat pemodelan sistem informasi terdiri dari:

1. Bagan Alir Dokumen (*Document Flowmap*)

   Bagan alir dokumen atau disebut juga bagan alir formulir merupakan bagan alir yang menunjukan arus dari laporan dan formulir termasuk tembusantembusannya.

2. *Entity-Relasionship Diagram* (ERD)

   ERD adalah diagram yang memperlihatkan entitas-entitas yang terlibat dalam suatu sistem serta hubungan-hubungan (relasi antar entitas.

Komponen-komponen pembentuk model ERD yaitu:

a. Entitas (*Entity*)

   Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Entitas dapat berupa orang, tempat, benda, peristiwa atau konsep yang bias memberikan atau mengandung informasi.

b. Atribut (*Attributes/Properties*)

   Setiap entitas pasti memiliki atribut yang mendeskripsikan karakteristik dari entitas tersebut.

c. Relasi (*Relationship*)

   Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

d. Kardinalitas/derajat

   Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpnan entitas lain.

   Kardinalitas relasi yang terjadi di antara dua himpunan entitas dapat berupa :

   a) Satu ke satu (*one to one*)
   b) Satu ke banyak (*one to many*)
   c) Banyak ke satu (*many to one*)
   d) Banyak ke banyak (*many to many*)

3. Skema Relasi
4. Skema relasi adalah untuk pressentasi atribut-atribut dari entity yang terdapat dalam sistem dan hubungan antar *entity* Ipada model ERD. Skema relasi merupakan turunan dari ERD.
5. Diagram Konteks (*Context Diagram*)

   Diagram konteks merupakan diagram alir pada tingkat palinng atas yang merupakan penggambaran yang berfungsi untuk memperlihatkan interkasi atau hubungan langsung antara sistem dengan lingkungannya. Diagram konteks menggambarkan sebuah sistem berupa sebuah proses yang berhubungan dengan satu atau beberapa entitas/entity.
6. *Data Flow Diagram* (DFD)

   DFD adalah suatu alat pemodelan yang digunakan untuk memodelkan fungsi dari sistem, menggambarkan secara rinci mengenai sistem sebagai jaringan kerja antar fungsi yang berhubungan satu sama lain dengan menunjukan dari dan kemana data mengalir serta penyimpanannya. Beberapa simbol digunakan di DFD:

   a. Kesatuan luar (*external entity*) atau batas sistem (*boundary*), merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, orhanisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan masi=ukan atau menerima keluaran dari sistem.
   b. Arus data (*data flow*) ini mengalir diantara proses (process), simpanan data (*data store*) dan kesatuan luar (*external entity*). Arus data ini menunjukkan arus dari data yang dapat berupa masukan untuk system atau hasil dari proses sistem

c. Proses (*process*) merupakan kegiatan yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk ke dalam proses untuk dihasilkam arus data yang akan keluar dari proses.
d. Simpanan data (data store) merupakan simpanan dari data yang dapat berupa suatu file atau database di sistem komputer, suatu arsip atau catatan manual, suatu kotak tempat data di meja seseorang, suatu table acuan manual, dan suatu agenda atau buku.
e. Spesisikasi Proses (*Process Specification (PSPEC)*)
f. Spesifikasi proses digunakan untuk menggambarkan semua proses model aliran yang. Nampak pada tingkat akhir penyaringan. Kandungan dari spesifikasi proses dapat termasuk teks naratif, gambaran bahasa desain program (*Program Design Language*) dari algoritma proses, persaaan matematika, tabel, digram, atau bagan.

7. Kamus Data (*Data Dictionary*)

   Kamus data adalah katalog fakta tentang data dan kebutuhan kebutuhan informasi dari suatu sistem informasi.Dengan menggunakan kamus data, analisis sistem dapat mendefinisikan data yng mengalir di sistem dengan lengkap. Kamus data harus memuat hal-hal berikut ini :

   a. Nama arus data.
   b. Alias atau nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data yang samamempunyai nama yng berbeda untuk orang atau departemen satu dengan yang lainnya.

c. Bentuk data, dapat berupa dokumen dasar atau formulir. Dokumen hasil cetakan komputer, laporan cetak, tampilan di layar monitor, variabel, parameter dan *field.*
d. Arus data menunjukkan dari mana data mengalir dan kemana data akan menuju.
e. Penjelasan dapat diisi dengan keterangan-keterangan tentang arus data tersebut.
f. Periode menunjukan kapan terjadinya arus data.
g. Volume digunakan untuk mengidentifikasikan besarnya simpanan luar yang akan digunakan, kapasitas dan jumlah dari alat input, alat pemroses dan alat *output.*
h. Struktur data menunjukkan arus data yang dicatat di kamus data berisi item-item data apa saja.

2.9 Perangkat Pengembangan (*Tools*)

Dalam membangun dan mengembangkan permainan *shopping time* ini digunakan *tools* Adobe Flash CS 4. Flash bekerja dengan bahasa pemprograman yang dinamakan *Action Script*. Bahasa ini telah mengalami beberapa kali perkembangan sejak pertama kali diperkenalkan. Versi terakhir adalah *Action Script* 3.0 atau disingkat AS3, dengan struktur bahasa yang sangat menyerupai javascript. Dengan dukungan *Action Script*, maka aplikasi yang dibangun dengan Flash dapat ditampilkan secara online di internet. Hal ini kemudian membuat multimedia interaktif di internet. Di antaranya adalah aplikasi untuk *game online*, aplikasi web interaktif, *online* video interaktif.

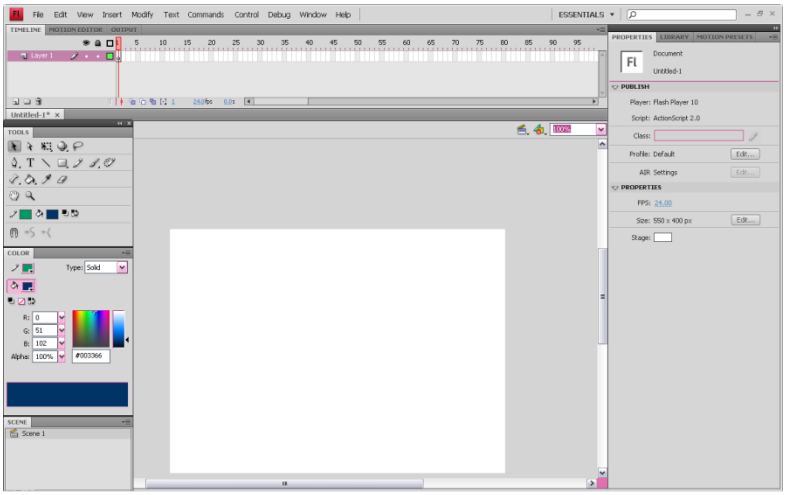
Flash menghasilkan file dalam format SWF yang ringan. Sebuah *game* yang ditulis dalam format Flash bisa menghasilkan file SWF berukuran beberapa kilobyte saja. SWF adalah kependekan dari "*Shock Wave Flash*", kata lain yang biasanya dipergunakan untuk menyebutkan file swf adalah *Flash Movie*. File *flash movie* memiliki ekstensi .swf, yang hanya dapat ditampilkan apabila komputer yang menjalankan aplikasi tersebut telah memiliki *plug-in* "Flash Player". File .swf yang ditampilkan secara *online* tidak dapat "dimainkan" pada komputer yang tidak terhubung ke internet (*offline*). Hal ini sangat menguntungkan bagi para web *developer*.

Flash tidak hanya menghasilkan file dalam format SWF saja. Sebuah aplikasi multimedia interaktif dapat disimpan dalam bentuk file *executable*, atau dinamakan sebagai Projector (sebuah file yang dapat dieksekusi oleh Microsoft Windows dengan ekstensi .exe, atau oleh Macintosh dengan ekstensi .hqx). Ada pula berupa file video dengan format QuickTime dengan ekstensi .mov.

Dengan demikian, Flash dapat dikembangkan untuk berbagai keperluan, mulai dari membuat animasi, sampai pada pemprograman yang rumit.

2.9.1 Perkembangan Flash

Ini merupakan tampilan desktop Adobe Flash CS 4.



Gambar 2.11 Tampilan Desktop Adobe Flash CS4

Berikut adalah sejarah perkembangan Flash:

1. Jonathan Gay, Charlie Jackson, dan Michelle Welsh mengembangkan SmartSketch pada tahun 1993. Aplikasi tersebut dibangun untuk menggambar dengan mode vektor.
2. Future Splash Animator sebagai versi awal Flash diperkenalkan pada bulan April tahun 1996 dengan kemampuan menggambar ditambah dengan timeline, sehingga dapat menghasilkan animasi.
3. Macromedia Flash 1 diperkenalkan pada bulan November 1996. Macromedia adalah *re-branded* dari Future Splash Animator.
4. Macromedia Flash 2 diperkenalkan pada bulan Juni 1997, dengan Flash Player 2, sudah terdapat object library.
5. Macromedia Flash 3 diperkenalkan pada bulan Mei 1998. dengan Flash Player 3, fitur baru meliputi Movie Clip, integrasi dengan plug-in

JavaScript, mengandung objek tranferens dan Flash Player yang berdiri sendiri.

6. Macromedia Flash 4 diperkenalkan pada bulan Jun 1999, dengan Flash Player 4, fitur baru meliputi variabel internal, kotak untuk input tulisan, pengembangan *Action Script*, dan *streaming* MP3.
7. Macromedia Flash 5 diperkenalkan pada bulan Agustus 2000, dengan Flash Player 5, dan telah memiliki *Action Script* 1.0.
8. Macromedia Flash MX diperkenalkan pada tahun 2002, dengan Flash Player 6.
9. Macromedia Flash MX 2004 diperkenalkan pada bulan September 2003, dengan Flash Player 7, dan telah mendukung *Action Script* 2.0.
10. Macromedia Flash 8 diperkenalkan pada bulan September 2005. Terdiri dari :
    - Macromedia Flash Basic 8, yang memiliki fitur yang terbatas untuk menggambar, animasi, dan interaktiviti.
    - Macromedia Flash 8 Professional dengan tambahan fitur yang difokuskan untuk tampilan yang lebih ekspresif, kualitas, video, dan multimedia.
11. Adobe Flash CS3 diperkenalkan pada bulan April 2007. Merupakan versi pertama Flash dibawah nama Adobe, Flash CS3 telah mendukung *Action Script* 3.0.
12. Adobe Flash CS4 diperkenalkan pada bulan Oktober 2008. Memiliki fitur baru Inverse Kinematik (bones), manipulasi objek 3D, animasi objek,

engine teks yang lebih maju, dan pengembangan lebih jauh pada ActionScript 3.0.

2.9.2 Fitur-fitur Kunci pada Flash

Adobe melakukan banyak perubahan pada Flash CS4. Perubahan yang dilakukan menyangkut inovasi pada *desaign* dan pengembangan pada fitur pemprograman *Action Script*. Pada versi ini kita akan menemukan Flash yang lebih menyatu dengan produk lainnya dari Adobe, khususnya Illustrator dan Photoshop. Lebih dari itu, Flash CS4 sekarang dapat dipergunakan untuk memanipulasi *video* dan *sound*. Flash CS4 juga dapat terhubung dengan *database* untuk mengembangkan aplikasi web.

Kita mengenal Flash sebagai aplikasi pengembang animasi grafik yang sangat baik, hal ini tidak terlepas dari fitur-fitur utama yang dimiliki oleh Flash.

2.9.2.1 Lingkungan Menggambar yang Lengkap

Flash adalah sebuah lingkungan yang lengkap untuk menggambar. Peralatan yang disediakan dapat dipergunakan untuk mengerjakan gambar dan tipografi yang rumit sekalipun. Seperti saudaranya Illustrator, Flash merupakan aplikasi untuk menggambar berbasis vektor. Bekerja dengan Flash berarti kita dapat menghasilkan gambar yang kaya, detail, dan dapat diperbesar atau diperkecil tanpa takut mengalami perubahan pada resolusi gambar. Flash sekarang telah dapat membaca file Illustrator dan Photoshop dalam bentuk aslinya, yaitu .ai dan

.psd membuat Flash semakin mempermudah penggunanya untuk dapat bekerja pada lingkungan yang disukainya.

2.9.2.2 Kemampuan Animasi yang Mudah

Flash dapat dipergunakan untuk menciptakan animasi yang ringan. Flash juga dapat mengakomodasi gambar bitmap, *sound*, dan *video*. Flash *movie* dapat dengan cepat di-*download* dan dapat diintegrasikan melalui website. Beberapa hal inilah yang menyebabkan Flash menjadi lingkungan yang disukai oleh *desaigner* dan *developer* web yang ingin mengembangkan kreativitasnya pada tingkat yang lebih tinggi. Animasi flash dapat diintegrasikan dengan website, *video*, dan *game*. Animasi flash juga sangat popular untuk dipergunakan pada multimedia interaktif, atau web *banner*. File Flash yang sangat ringan membuatnya sangat ideal untuk diaplikasikan dengan animasi, permainan, dan bahkan pada *mobile phone*.

Animasi yang dibangun dengan Flash dapat meliputi animasi tradisional *frame-by-frame*, atau animasi *tweening*. Berbeda dengan animasi *frame-by-frame*, animasi *tweening* dapat dipergunakan untuk membentuk animasi dengan hanya menentukan frame awal dan akhirnya saja. Flash dapat menciptakan *frame-frame* antara (*in between*) yang diperlukan untuk menciptakan gerakan, warna, dan transformasi bentuk ataupun efek transisi. Dengan demikian, kita dapat menciptakan animasi yang diperlukan dengan berbagai pendekatan.

2.9.2.3 *Layout* yang Tanpa Batas

Stage pada Flash memberikan fleksibiliti yang tinggi dalam menciptakan *layout* halaman-halaman website yang menarik tanpa batasan yang biasanya sangat menyulitkan apabila dikembangkan dengan struktur HTML biasa. Kita dapat menempatkan kandungan web di manapun pada Stage dengan fleksibiliti dan tingkat presisi yang tinggi, jauh meninggalkan website statik di belakangnya. Flash movie juga dapat menyertakan tipografi yang kita pilih sehingga dapat menghasilkan tampilan yang menarik dengan *font* yang kaya tanpa batasan yang biasanya sulit diaplikasikan di luar Flash.

2.9.2.4 Lingkungan Pemprograman yang Bertenaga

Keindahan tersembunyi pada Flash CS4 Professional terletak pada kemampuan pemprogramannya yang bertenaga. *Action Script* adalah sebuah bahasa pemprograman yang dapat mengembangkan kemampuan mendesain dan animasi menjadi sebuah aplikasi interaktif yang hidup dan menarik. Dengan *Action Script* kita dapat mengendalikan *movie*, membuat dan memberi berbagai fungsi pada tombol. Lebih dari itu, Flash merupakan lingkungan pemprograman *visual* yang lengkap dan luas. *Action Script* dapat menjadikan *movie flash* sebagai sebuah aplikasi online yang lengkap untuk berbagai keperluan, termasuk *shopping cart*, musik dan *video player*, *game*, bahkan dapat diaplikasikan untuk *mobile phone*.