BAB 2
LANDASAN TEORI

2.1. Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan adalah suatu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia. Menurut beberapa ahli, kecerdasan buatan didefinisikan sebagai berikut :

1. Menurut H.A.Simon [1987] : ”Kecerdasan buatan (*artificial intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman computer untuk melakukan hal yang dalam pandangan manusia adalah cerdas”.
2. Menurut Rich and knight [1991] : “Kecerdasan buatan (*artificial intelligence*) merupakan sebuah studi tentang bagaimana membuat computer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia”.
3. Menurut Encyclopedia Britannica : “Kecerdasan buatan (*artificial intelligence*) merupakan cabang ilmu computer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi kecerdasan berdasarkan metode heuristic atau berdasarkan sejumlah aturan”.
4. Menurut John McCarthy, 1956, *AI* : “Untuk mengetahui dan memodelkan proses–proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia” [5].

Kecerdasan buatan dilihat dari berbagai sudut pandang adalah sebagai berikut :

1. Sudut pandang kecerdasan

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

2. Sudut pandang penelitian

   Studi bagaimana membuat agar komputer dapat melakukan sesuatu sebaik yang dilakukan oleh manusia.

3. Sudut pandang bisnis

   Kumpulan peralatan yang sangat powerful dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4. Sudut pandang pemograman

   Kecerdasan buatan termasuk didalamnya adalah studi tentang pemrograman simbolik, pemecahan masalah, proses pencarian.

Berikut adalah tujuan dari kecerdasan buatan [5]:

1. Membuat mesin menjadi lebih pintar (tujuan utama)
2. Memahami apa itu kecerdasan (tujuan ilmiah)
3. Membuat mesin lebih bermanfaat (tujuan *entrepreneurial*)

Berdasarkan definisi ini, maka kecerdasan buatan menawarkan media ataupun uji teori tentang kecerdasan. Teori-teori ini nantinya dapat dinyatakan dalam bahasa pemrograman dan eksekusinya dapat dibuktikan pada komputer nyata.

2.1.1. Subdisiplin Ilmu dalam Kecerdasan Buatan

Kecerdasan buatan dibagi kedalam beberapa subdisiplin sebagai berikut [5]:

1. Sistem Pakar (*Expert System*)

   Komputer sebagai sarana untuk menyimpan pengetahuan para pakar sehingga komputer memiliki keahlian menyelesaikan permasalahan dengan meniru keahlian yang dimiliki pakar.

2. Pengolahan Bahasa Alami (*Natural Language Processing*)

   Pemrosesan bahasa alami adalah suatu bidang kecerdasan buatan yang mempelajari tata bahasa manusia. Untuk itu komputer dibuat agar mengerti bahasa manusia, dengan demikian komputer dapat memberikan tanggapan terhadap masukan dari manusia dengan jawaban yang sesuai dengan struktur bahasa yang baik.

3. Pengenalan Ucapan (*Speech Recognition*)

   Pengenalan ucapan, atau yang sering disebut dengan *Automatic Speech Recognition* (*ASR*), adalah suatu pengembangan teknik dan sistem yang memungkinkan komputer untuk menerima masukan berupa kata yang diucapkan.

4. Robotika dan Sistem Sensor (*Robotics and Sensory Systems*)

   Robotika adalah suatu bidang rekayasa yang mencurahkan perhatiannya ke bidang duplikasi kemampuan fisik manusia, yaitu suatu komplemen alami dalam bidang *artificial intelligence* yang berusaha menirukan kemampuan mental manusia. Robot adalah mesin atau manipulator yang mampu melaksanakan fungsi fisik manusia secara terbatas.

5. *Computer vision*

   Menginterpretasikan gambar atau objek-objek tampak melalui komputer.

6. *Intelligent Computer-Aided Instruction*

   Komputer dapat digunakan sebagai tutor yang dapat melatih & mengajar.

7. *Game Playing*

   Kebanyakan permainan dilakukan dengan menggunakan sekumpulan aturan. Dalam permainan digunakan apa yang disebut dengan pencarian ruang. Teknik untuk menentukan alternatif dalam menyimak problema ruang merupakan sesuatu yang rumit. Teknik tersebut disebut dengan HEURISTIC. Permainan merupakan bidang yang menarik dalam studi heuristic.

2.2. Algoritma

Algoritma merupakan salah satu cabang ilmu komputer yang membahas prosedur penyelesaian suatu permasalahan. Algoritma adalah urutan langkah-langkah dalam menentukan suatu masalah. Algoritma juga dapat didefinisikan dengan deretan langkah komputasi yang mentransformasikan masukan (*input*) menjadi keluaran (*output*). Jadi algortima adalah serangkaian urutan langkah-langkah atau prosedur untuk menyelesaikan suatu masalah dengan memproses nilai masukan menjadi nilai keluaran.

Algoritma dapat ditulis dalam bentuk *pseudocode* ataupun *flowchart*. *Pseudocode* berasal dari kata *pseudo* yang artinya mirip atau menyerupai dan *code* berarti kode sehingga *pseudocode* adalah kode yang menyerupai kode program sebenarnya. Sedangkan *flowchart* merupakan bentuk gambar atau diagram yang mempunyai aliran satu atau dua arah secara sekuensial.

Algoritma dalam mempelajarinya, kita diperlukan untuk dapat menganalis algoritma dalam menentukan performansinya. Suatu Algoritma tidak saja harus benar, tetapi juga harus efisien dengan menghitung nilai dari performansinya. Setiap algortima memiliki nilai performansi yang berbeda-beda [5].

2.2.1. *Rule Based Systems*

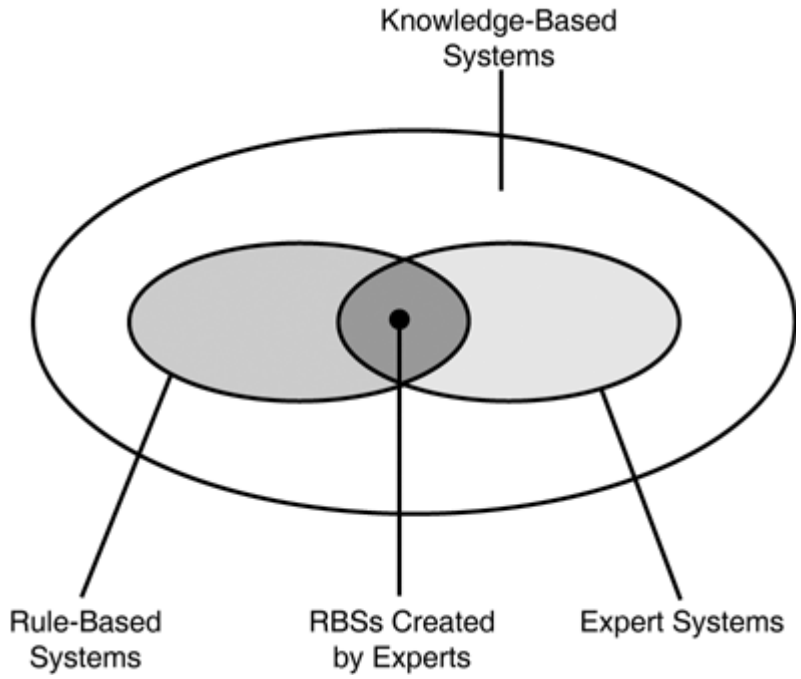
Rule Based Systems (RBSs) pada dasarnya terdiri dalam koleksi pernyataan "*IF ... THEN*". Pernyataan tersebut sangat cocok untuk *encoding rules-of-thumb*, mirip dengan keahlian manusia. Para ahli umumnya menerapkan pengetahuan tersebut secara mekanis untuk menyelesaikan masalah. Dengan mentransfer pengalaman mereka ke dalam komputer, proses yang sama dapat dilakukan dengan keandalan yang sama, tetapi jauh lebih efisien dalam mode otomatis.

Dalam permainan, RBSs dapat digunakan pada *Non Player Character* (NPC) untuk dua tujuan yang berbeda: pemecahan masalah (mencapai kesimpulan tunggal dari fakta-fakta awal) atau kontrol (menghasilkan urutan tindakan dalam simulasi). Dalam pemecahan masalah, aturan menjelaskan hubungan antara

kebenaran yang berbeda (misalnya, "*IF* item tidak ada, *THEN* seseorang mengambilnya") sehingga sistem dapat mencapai jawabannya. Adapun kontrol, aturan menjelaskan tindakan yang harus diambil dalam setiap situasi (misalnya, "*IF* ada dinding di depan, *THEN* berhenti").

Dalam prakteknya, *RBSs* memiliki keuntungan memisahkan perilaku dari kode. Dari semua teknik yang memungkinkan ini pendekatan data-driven, *RBSs* yang paling sederhana untuk mengembangkan, sangat cepat pada saat runtime, sangat fleksibel, dan mudah diperluas.

Dalam prakteknya, *RBSs* sangat luas. Banyak domain manfaat dari *RBSs*, mulai dari obat untuk industri manufaktur, termasuk dukungan teknis. Banyak bidang keuntungan dari sistem deduktif memanipulasi data sesuai dengan aturan sederhana. Dengan demikian, *RBSs* merupakan salah satu keberhasilan utama *AI*. Ada puluhan ribu dari *RBSs*, jatuh ke dalam berbagai subkategori sistem berbasis pengetahuan, seperti yang ditunjukkan pada Gambar 2.



Gambar 2.1 Diagram Venn KBS

2.3. Metode *Forward Chaining*

Metode *forward chaining* adalah suatu rantai yang dicari atau dilewati/dilintasi dari suatu permasalahan untuk memperoleh solusinya disebut dengan *forward chaining.* Cara lain menggambarkan *forward chaining* adalah penalaran dari fakta menuju konklusi terdapat dari fakta.

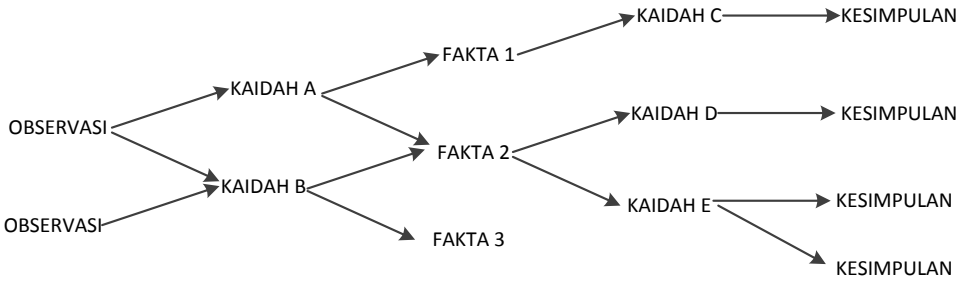
Forward chaining merupakan perunutan yang dimulai dengan menampilkan kumpulan data atau fakta yang menyakinkan menuju konklusi akhir. *Forward chaining* dimulai dari premis – premis atau informasi masukan (IF) dahulu kemudian menuju konklusi atau derived information (THEN) atau dapat dimodelkan sebagai berikut [1]:

IF (informasi masukan)

THEN (konklusi)

Informasi masukan dapat berupa data, bukti, temuan atau pengamatan. Sedangkan konklusi dapat berupa tujuan, hipotesa, penjelasan atau diagnosa. Sehingga jalannya *forward chaining* maju dapat dimulai dari data menuju tujuan dan bukti menuju hipotesa, dari temuan menuju penjelasan, atau dari pengamatan menuju diagnosa[4].

Strategi dari sistem ini adalah dimulai dari inputan beberapa fakta, kemudian menurunkan beberapa fakta dari aturan-aturan yang cocok pada *knowledge base* dan melanjutkan prosesnya sampai jawaban sesuai. *Forward chaining* dapat dikatakan sebagai penelusuran deduktif[1]. Berikut alur *forward chaining* dapat dilihat pada gambar 2.2.



Gambar 2.2 Alur *Forward Chaining*

Implementasi *forward chaining* sangat membantu *developer* aplikasi dalam membangun suatu sistem, karena dengan menggunakan metode ini jika *developer* ingin menambah beberapa kondisi dan aturan, *developer* tidak perlu membongkar lagi kode program dari awal. Metode *forward chaining* juga diartikan sebagai pendekatan yang dimotori data. Pendekatan metode *forward chaining* pelacakannya dimulai dari informasi masukan, dan selanjutnya mencoba menggambarkan kesimpulan, sehingga metode ini juga sering disebut "*Data driven*" [3].

Tabel 2.1 menunjukan karakteristik *forward chaining* :

Tabel 2.1 Karakteristik *Forward Chaining*

Forward Chaining
Perencanaan,Monitoring, Control
Disajikan untuk masa depan
Data memandu, penalaran dari bawah ke atas
Bekerja kedepan untuk mendapatkan solusi apa yang mengikuti fakta
Breadth first search dimudahkan

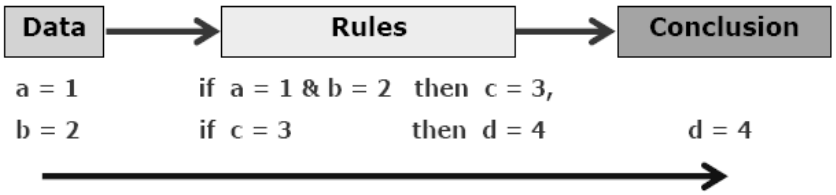
Operasi dari sistem *forward chaining* dimulai dengan memasukan sekumpulan fakta yang diketahui ke dalam memori kerja *(working memory),* kemudian menurunkan fakta baru berdasarkan aturan yang premisnya cocok dengan fakta yang diketahui. Proses ini dilanjutkan sampai dengan mencapai *goal* atau tidak ada lagi aturan yang premisnya cocok dengan fakta yang diketahui. Langkah-langkah yang harus dilakukan dalam membuat sistem *forward chaining* berbasis aturan, yaitu :

1. Pendefinisian Masalah
   Tahap ini meliputi pemilihan domain masalah dan akuisisi pengetahuan
2. Pendefinisian Data Input
   Sistem *forward chaining* memerlukan data awal untuk memulai inferensi
3. Pendefinisian Struktur Pengendalian Data

Aplikasi yang kompleks memerlukan premis tambahan untuk membantu mengendalikan.*Forward chaining* merupakan suatu teknik untuk mengkalkulasi setiap *game state,* yang mana sistem mengkalkulasi *state* yang sedang berjalan dijadikan sebagai tolak ukur untuk melanjutkan ke *state* berikutnya [3].

2.3.1. Mesin Inferensi *Forward Chaining*

Mesin inferensi merupakan suatu mesin yang menggerakan suatu baris instruksi yang digunakan untuk mencapai suatu solusi, mesin ini menyambungkan antara *knowledge base* yang dibuat dengan aturan *IF-THEN* dan mengacu terhadap fakta – fakta yang ada pada *database.* Mesin inferensi adalah mekanisme control untuk navigasi yang memanipulasi *knowledge* dan memberikan solusi terbaik yang tersusun secara terorganisir, seperti yang diperlihatkan pada Gambar 2.3.



Gambar 2.3 Skema *Data Driven*

2.4. Pemograman Terstruktur

Pemrograman Terstruktur adalah suatu proses untuk mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dalam bentuk program [4]. Selain pengertian diatas Pemrograman Terstruktur adalah suatu aktifitas pemrograman dengan memperhatikan urutan langkah-langkah perintah secara sistematis, logis, dan tersusun berdasarkan algoritma yang sederhana dan mudah dipahami. Prinsip dari pemrograman terstruktur adalah Jika suatu proses telah sampai pada suatu titik/langkah tertentu, maka proses selanjutnya tidak boleh mengeksekusi langkah sebelumnya/kembali lagi ke baris sebelumnya, kecuali pada langkah – langkah untuk proses berulang (Loop).

2.4.1. DFD (Data Flow Diagram)

DFD adalah merupakan model dari system untuk menggambarkan pembagian system ke modul yang lebih kecil [4]. Salah satu keuntungan menggunakan diagram aliran data adalah memudahkan pemakai atau user yang kurang menguasai bidang komputer untuk mengerti system yang dikerjakan.

Dibawah ini terdapat istilah-istilah untuk DFD yaitu:

1. *Entity*: *Terminator* atau *Source* atau *Destination* atau dikenal juga dengan External Entity, berupa orang, organisasi atau sistem lain yang berada diluar batas sistem yang berinteraksi dengan sistem yang sedang dikembangkan.
2. Proses: Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, prosedur atau alat yang digunakan untuk mentrasformasikan data.
3. *Data Flow*: (Arus Data), data yang mengalir dengan arah tertentu dari asal ke tujuan. Data yang mengalir dapat berupa dokumen, surat atau bentuk lainnya.
4. Data *Store*: (Penyimpanan Data), digunakan untuk menyimpan dan mengambil data oleh proses. Data yang disimpan dapat berupa data yang terkomputerisasi maupun tidak terkomputerisasi.

2.4.2. Komponen Terminator/Entitas Luar

Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Biasanya terminator dikenal dengan nama entitas luar (*external entity*). Sesuatu yang berada diluar system, tapi memberikan data kedalam sistem atau memberikan data dari sistem. Disimbolkan dengan notasi kotak [4].

2.4.3. Komponen Proses

Komponen proses merupakan apa yang dikerjakan oleh system, proses dapat mengolah data atau aliran data masuk menjadi aliran data keluar. Proses berfungsi mentransformasikan satu atau beberapa data masukan menjadi satu atau beberapa data keluaran sesuai dengan spesifikasi data yang dinginkan [4].

2.4.4. Komponen Data *Store*

Komponen ini digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak, misalnya Mahasiswa. *Data store* ini biasanya berkaitan dengan penyimpanan-penyimpanan, seperti file atau database yang berkaitan dengan penyimpanan secara komputerisasi, misalnya file disket, file harddisk, file pita magnetik. *Data store* juga berkaitan dengan penyimpanan secara manual seperti buku alamat, file, folder, dan agenda.

Suatu *Data Store* dihubungkan dengan alur data hanya pada komponen proses, tidak dengan komponen DFD lainnya. Alur data yang menghubungkan *Data Store* dengan suatu proses mempunyai pengertian sebagai berikut :

a. Alur data dari *Data Store* yang berarti sebagai pembacaan atau pengaksesan satu paket tunggal data, lebih dari satu paket data, sebagian dari satu paket tunggal data, atau sebagian dari lebih dari satu paket data untuk suatu proses.
b. Alur data ke *Data Store* yang berarti sebagai peng-update-an data, seperti menambah satu paket data baru atau lebih, menghapus satu paket atau lebih, atau mengubah/memodifikasi satu paket data atau lebih.

2.4.5. Komponen *Data Flow*/Alur Data

Arus data merupakan tempat mengalirnya informasi dan digambarkan dengan garis yang menghubungkan komponen dari sistem [4]. Arus data ditunjukan dengan arus panah dan garis diberi nama atas arus data yang mengalir. Arus data ini mengalir diantara proses, *data store* dan menunjukan arus data dari data yang berupa masukan untuk sistem atau hasil proses sistem.

2.5. Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan pengkodean[6].

2.5.1. Pengujian *Black box*

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa

perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* bukan merupakan alternatif dari teknik *white box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kesalahan-kesalahan daripada metode *white box*[6].

Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan dalam *interface.*
3. Kesalahan dalam struktur data atau akses *database eksternal.*
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

2.6. Game

Game 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.

"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 ”.

2.6.1. Jenis-jenis *Game*

Ada beberapa jenis platform di dunia game yang selalu dipilih oleh pengguna games , yaitu [7] :

1. *Arcade games,* yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang

khusus di design untuk jenis *video games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa masuk dan menikmati, seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya).

2. *PC Games* , yaitu *video game* yang dimainkan menggunakan *Personal Computers*.
3. *Console games*, yaitu *video games* yang dimainkan menggunakan *console* tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.
4. *Handheld games*, yaitu yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.
5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau *PDA*.

2.6.2. *Genre Game*

Game dibagi atas beberapa *genre*, diantaranya yaitu [7] :

1. *Action Shooting* (tembak – menembak). Menembak , memukul , bisa juga tusuktusukan, tergantung cerita dan tokoh di dalamnya, *video game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari game jenis ini adalah tembak, tembak dan tembak.

   Contoh : *CS* (*Counter Strike*) dan *Crysis*.
2. *Fighting* (pertarungan). Ada yang mengelompokan *video 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 timing sangatlah penting, iya, combo-pun menjadi esensial untuk mengalahkan lawan secepat mungkin.

   Contoh : *Mortal Kombat* dan *Tekken*.
3. *Adventure* (Petualangan). Memasuki gua bawah tanah, melompati bebatuan di antara lahar, bergelayutan dari pohon satu ke pohon lain, bergulat dengan ular sambil mencari kunci untuk membuka pintu kuil legendaris, atau sekedar mencari telepon umum untuk mendapatkan misi berikutnya, itulah beberapa

dari banyak hal yang karakter pemain harus lakukan dan lalui dalam *video game* jenis ini.

Contoh : *Kings Quest*, dan *Space Quest.*

4. *Strategy* (strategi). *Video game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusal pelatihan tempur, tergantung dari tema ceritanya. Kebanyakan *game* stategi adalah *game* perang. Contoh : *Warcraft.*
5. Simulation (Simulasi). *Video game* jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium cloning. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas.
6. *Puzzle* (teka-teki). *Video game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula *game* jenis ini adalah juga unsur *game* dalam *video game* petualangan maupun *game* edukasi.
7. *Sport game* (Olahraga). *Game* ini merupakan adaptasi dari kenyataan, membutuhkan kelincahan dan juga strategi dalam memainkannya. *Game* berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau tim. Contoh *game* tipe ini antara lain sepakbola, bola basket, tenis, dan *bilyard*.
8. *RPG* (*Role Playing Game*). *Video game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran

perwakilan pemain di dalam *game*, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman maupun mahluk peliharaan.

9. *Education* (edukasi). *Game* edukasi merupakan paket *software* yang menciptakan kemampuan pada lingkungan *game* yang diberikan sebagai alat bantu untuk memotivasi atau membantu siswa untuk melalui prosedur *game* secara teliti untuk mengembangkan kemampuannya. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar game ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan keterampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan *design visual* ataupun animasinya.

2.7. Blender 3D

Blender, merupakan salah satu *software* animasi *3D* yang mumpuni. Selain untuk animasi *3D*, Blender juga dapat digunakan untuk mengerjakan konten-konten kreatif di bidang multimedia seperti spesial efek, koreks warna, pembuatan *game* iteraktif dan penyuntingan (editing) *video*. Salah satu kelebihan lainnya, Blender dapat berjalan di berbagai sistem operasi.

Fitur utama pada *Blender 3d* adalah sebagai berikut :

1. Penciptaan penuh (suite) yg terintegrasi, menawarkan berbagai alat-alat penting untuk penciptaan konten 3D, termasuk model, uv-pemetaan, texturing, rigging, skinning, animasi, partikel dan simulasi lainnya, script, render, composite, pasca produksi, dan penciptaan permainan.
2. Cross platform, dengan keseragaman OpenGL GUI pada semua platform (dapat disesuaikan dengan skrip python), siap pakai untuk semua versi Windows (98, NT, 2000, XP, Vista, 7), Linux, OS X, FreeBSD, Irix, Sun dan berbagai sistem operasi lain.

3. Kualitas tinggi arsitektur 3D yang memungkinkan kececepatan dan efisiensi penciptaan.
4. Lebih dari 200.000 download dari rilis masing-masing (pengguna) di seluruh dunia.
5. Dukungan masyarakat pengguna (komunitas) melalui forum untuk pertanyaan, jawaban, dan kritik pada http://BlenderArtists.org dan layanan berita di http://BlenderNation.com; http://blenderindonesia.org/forum;
6. Ukuran executable kecil, distribusi yang mudah.