BAB 2

TINJAUAN PUSTAKA

2.1 Permainan (*Game*)

Permainan merupakan sebuah aktivitas rekreasi dengan tujuan bersenang-senang ataupun untuk mengisi waktu luang. Permainan biasanya dilakukan secara sendiri-sendiri ataupun secara kelompok. Permainan sendiri terbagi menjadi 2 bagian yaitu:

1. Permainan Anak-Anak

   Banyak permainan yang sering dilakukan oleh anak-anak bersama teman-teman mereka di sekitar rumah mereka. Mereka biasanya berkelompok, senang berlarian, ataupun kadang duduk melingkar untuk memainkan salah satu permainan, dan dari situlah biasanya akan tercipta suatu keakraban diantara mereka. Beberapa permainan dari masa lalu disebut dengan permainan tradisional, sedangkan untuk permainan yang muncul di jaman sekarang ini disebut dengan permainan modern dan biasanya menggunakan peralatan canggih seperti halnya *video game*. Dan berikut beberapa jenis permainan yang bisa dimainkan oleh anak-anak :

   a. Bentengan.
   b. Boneka.
   c. Congklak.
   d. Goncang Kaleng.
   e. Kasti.

f. Layangan.

g. Main Hadang.

h. Petak Umpet.

i. Yoyo.

2. Permainan Dewasa

Untuk permainan orang dewasa biasanya memerlukan pemikiran untuk memainkannya. Dan berikut beberapa jenis permainan yang bisa dimainkan oleh orang dewasa :

a. Kartu Remi atau Bridge.

b. Catur.

c. Domino.

d. Kuis.

e. Sudoku [6]

2.1.1 Pengertian Game

Game bukanlah hal yang tabu di mata masyarakat. Hampir semua orang dari berbagai kalangan pernah memainkan game.

Mendefinisikan apakah yang dimaksud dengan game, tidak cukup dengan hanya melihat kamus bahasa. Makna sekilas dari game memberikan pengertian bahwa game merupakan suatu aktifitas yang tidak dilakukan dengan sungguh-sungguh. Untuk mengetahui apa yang sesungguhnya disebut dengan game, maka paling tidak kita dapat memahaminya dari adanya sejumlah pengertian game yang biasa kita alami dalam kehidupan [6].

Dalam hal paling tidak terdapat 5 katagori istilah game, yaitu :

1. *Board Games* (Permainan Papan) Game pada katagori ini membutuhkan suatu papan yang terbagi dalam sektor-sektor tertentu (dengan garis-garis) dan didalamnya terdapat sejumlah alat main yang dapat digerakkan. Termasuk dalam katagori ini adalah catur. Dua buah pemain akan berhadapan dan saling mengadu strategi sesuai dengan aturan untuk mencapai daerah lawan atau mempertahankan daerahnya sendiri, mengalahkan bidak musuh, mengumpulkan sesuatu. Pemain pada board games ini akan berusaha menganalisis hubungan-hubungan geometri yang ada pada papan dan bidak.
2. *Card Games* (Permainan Kartu) Games ini akan memanfaatkan simbol dari 52 kartu yang terbagi dalam dua faktor : suit (4 nilai) dan rank (13 nilai). Permainan akan dilakukan sekitar bagaimana membuat kombinasi dari 52 kartu tersebut. Sejumlah ketentuan dibuat untuk mengatur bagaimana cara-cara untuk membuat kombinasi tersebut. Permainan kartu bridge/truf termasuk kelompok games ini.
3. *Athletic Games* (Permainan Atletik) Permainan games jenis ini lebih cenderung pada penggunaan fisik daripada mental. Aturan game dibuat dengan keharusan pemain untuk melakukan sejumlah aksi tertentu. Hal yang terkait dengan kekuatan badan, kecepatan, ketepatan dan kerjasama menjadi bagian utama dari game atletik. Dalam hal ini harus dibedakan antara game dengan kompetisi. Kompetisi tidak mengharuskan adanya kerjasama dan pemain bekerja secara individu. Dua orang yang beradu lari bukanlah termasuk game tetapi kompetisi. Perbedaan utama antara game dan kompetisi

adalah dalam hal interaksi diantara peserta. Pada kompetisi tidak terjadi interaksi diantara pemain, kompetisi yang membolehkan adanya interaksi diantara pemainnya termasuk game.

4. *Children Games* (Permainan Anak) Aktifitas seperti berlari, sembunyi, melempar dan menangkap adalah menjadi ciri utama game anak-anak. Umumnya game ini menekankan pada aktifitas kelompok sebagai latihan untuk berkehidupan sosial. Walaupun dalam game ini terdapat juga upaya untuk saling mengalahkan secara mental atau fisik namun tujuan utamanya bukanlah untuk meraih kemenangan satu diatas yang lain tetapi sebagai ilustrasi kerjasama dalam kehidupan manusia. Penggunaan sejumlah alat untuk membantu ilustrasi dapat meningkatkan improvisasi game dan meningkatkan keterlibatan yang lebih baik dari pemain.
5. *Computer Games* (Permainan Komputer) Game ini dimainkan lewat bantuan alat komputer. Terdapat 5 alat yang dapat dikategorikan sebagai komputer, yaitu :
    a. *Expensive dedicated machine*, mesin yang dioperasikan dengan koin untuk memainkankannya.
    b. *Inexpensive dedicated machine*, disebut juga dengan *hand held machine*. Alat game watch termasuk dalam katagori ini.
    c. *Multiprogram home*, mesin seperti Atari, Nintendo termasuk dalam kelompok komputer ini
    d. Personal computer
    e. Mainframe computer

Komputer game berbeda dengan jenis game yang lain karena tidak ada pergerakan secara fisik atau interaksi langsung dengan object kecuali lewat perantaraan komputer. *Software* yang dibuat harus dapat menangkap reaksi yang cepat dari interaksi yang dihasilkan dengan pemain. Karena itu *software* untuk computer games harus bersifat real time. Kompleksitas game adalah bergantung dari kemampuan merepresentasikan aturan dan lingkungan game dalam program yang dibuat.

Diantara sekian banyak definisi game, maka definisi yang umum untuk computer game adalah game Komputer adalah sebuah program software dimana satu atau lebih pemain berusaha untuk membuat keputusan lewat kontrol terhadap object dan resource guna memenuhi satu tujuan tertentu [6].

2.1.2 Sejarah Singkat Game

Sejarah teknologi game komputer secara langsung berhubungan dengan perkembangan komputer itu sendiri. Komputer dengan kecepatan processor tinggi, grafis yang lebih mendekati realita, dan media penyimpanan yang lebih besar sebenarnya dimaksudkan untuk memenuhi kebutuhan dalam bermain games.

Sebelum dimulai, akan kita pahami dulu apa arti dari game (game komputer). Game adalah sebuah permainan *interactive* yang membutuhkan komputer untuk bermain. Program komputer menerima input dari si pemain melalui pengendali dan menampilkan lingkungan buatan melalui TV atau layar monitor [7].

a. Game generasi pertama

Tahun 1952, di Universitas Cambridge, A.S Douglas menulis sebuah tesis untuk gelar sebuah PhD-nya mengenai interaksi antara komputer dan manusia. Dalam tesisnya itu ia menciptakan game komputer dari sebuah permainan tradisional bernama Tic-Tac-Toe. Game ini diprogram dengan memakai komputer EDVAC vacuum tube yang memiliki layer berupa cathode ray tube (CRT).

Kemudian di tahun 1958. William Haginbotham menciptakan video game pertamanya. Berbeda dengan Douglas, video game pertamanya yang berjudul Tennis for Two diciptakan dan dimainkan di osciloscope.

Tapi game komputer pertama yang benar-benar diciptakan menggunakan komputer betulan adalah Spacewar. Saat itu di tahun 60-an, komputer adalah barang sangat mewah. Dan biasanya komputer dipakai untuk kepentingan riset dalam dunia militer. Tapi seseorang bernama Steve Russel memiliki ketertarikan akan hal lain. Dia dan teman-temannya sangat nge-fan dengan kisah fiksi ilmiah berjudul Skylark karangan Edward E Smith. Dari situ mereka membuat sebuah game bernama Spacewar. Kebetulan juga, Steve Russel bekerja menggunakan sebuah komputer mainframe bernama MIT PDP-1 yang biasa dipakai untuk perhitungan statistik. Dengan komputer itulah dia membuat Spacewar di tahun 1961.

b. Game generasi kedua

Di tahun 1971, Nolan Bushnell bersama dengan Ted Dabney menciptakan game ber-genre arcade yang pertama. Dinamai Komputer Space, game itu didasari oleh Spacewar. Tahun 1972, Nolan dan Ted memulai Atari komputer. Kemudian

dia mengembangkan game berjudul Pong yaitu game pertama yang tersedia untuk publik. Karena game-game sebelumnya hanya ada di dalam komputer mainframe untuk kesenangan sendiri saja. Asal usul Pong dimulai saat Nolan ingin membuat game sederhana dan mudah dimengerti. Dengan memory dan micro processor kelas rendah, kemampuan proses yang terbatas dan grafis yang sederhana, akhirnya dia membuat versi elektronik dari permainan ping pong yang kemudian menjadi Pong.

Sedikit selingan, Pong kemudian berevolusi menjadi sebuah game bernama Breakout. Game itu diciptakan oleh Steve Jobs untuk Atari. Dari situ, Steve Jobs dan temannya Steve Wozniak mulai berpikir untuk menciptakan sebuah PC. Kemudian mereka meminjam semua peralatan yang dipakai dalam proyek Breakout dan membuat sebuah prototype bernama Apple I. dan itu adalah cikal bakal dari komputer Apple Macintosh yang ada sekarang.

Pada tahun 1980, Atari mengeluarkan game berjudul Asteroid dan Lunar Lander. Kedua game tersebut adalah game pertama yang didaftarkan pada kantor hak cipta untuk mendapatkan paten. Asteroid merupakan game yang penuh inovasi baru dalam grafisnya. Daripada menggunakan metode raster, game ini merupakan grafis vector line seperti yang ada pada osciloscope.

Pengenalan Atari Video komputer System (Atari 2600) dengan CPU biasa dan slot untuk kasetnya, menjadi suatu era baru dalam dunia game. Di tahun 1980 itu juga menunjukkan penjualan yang meningkat dari PC yang biasa dipakai untuk game.

Dan kini komputer bersaing dengan mesin konsol seperti PS atau Xbox. Dan komputer tidak hanya sebagai mesin untuk kepentingan bisnis semata, tapi juga untuk hiburan seperti game [8].

2.1.3 Permainan Video (*Video Game*)

Permainan video (*video game*) adalah permainan yang menggunakan interaksi antarmuka pengguna melalui gambar yang dihasilkan oleh piranti video. Permainan video umumnya menyediakan sistem penghargaan, misalnya *score* yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan video tersebut. Sistem elektronik yang digunakan untuk menjalankan permainan video disebut *platform*, contohnya adalah komputer pribadi (*personal computer*) ataupun konsol permainan (*game console*).

1. Permainan Komputer (*Computer Game*)

   Permainan komputer adalah permainan video yang dimainkan pada komputer pribadi, dan bukan pada konsol permainan maupun mesin ding-dong. Permainan komputer telah berevolusi dari sistem grafis sederhana sampai menjadi sangat kompleks dan mutakhir.

   Permainan *online* (*online game*) adalah jenis permainan video atau permainan komputer dengan menggunakan jaringan komputer dan juga internet sebagai media interaksinya.

   Permainan online terdiri dari dua unsur utama, yaitu *server* dan *client*. *Server* adalah penyedia layanan gaming yang merupakan basis agar *client*-

client yang terhubung dapat memainkan permainan dapat melakukan komunikasi dengan baik. Suatu *server* pada prinsipnya hanya melakukan administrasi permainan dan menghubungkan *client-client*. Sedangkan *client* adalah pengguna permainan dan memakai kemampuan *server*. Contoh permainan *online* adalah *Ragnarok Online, Risk Your Life, Xian* dll.

2. Konsol Permainan (*Game Console*)

   Konsol permainan adalah sebuah mesin elektronik yang dirancang khusus untuk memainkan permainan video. Perangkat keluarannya biasa berupa monitor komputer atau televisi dan alat masukan utamanya berupa sebuah pengendali. Konsol permainan yang pertama kali dibuat adalah *Atari*, kemudian dilanjutkan dengan *Nintendo* yang sukses pada tahun 1985-1989. Konsol permainan modern sekarang ini antara lain adalah *Playstation* buatan perusahaan *Sony*. Untuk *Console* yang berbentuk kecil dan dapat dibawah kemana-mana biasa disebut *Portable Console* [8].

2.1.4 Klasifikasi Permainan (*Game Classification*)

Berikut ini adalah klasifikasi permainan berdasarkan pembatasan usia dari pemainnya, menurut ESRB (*Entertainment Software Rating Board*) adalah sebagai berikut :

1. ***Early Childhood (3+)*** memiliki isi yang mungkin cocok untuk usia 3 keatas. Tidak mengandung materi yang tidak pantas seperti adegan kekerasan.

2. ***Everyone (6+)*** memiliki konten yang mungkin cocok untuk usia 6 keatas. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan tidak jarang menggunakan bahasa yang ringan.
3. ***Everyone (10+)*** memiliki konten yang mungkin cocok untuk usia 10 keatas. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi atau kekerasan ringan, bahasa ringan dan juga minimal tema sugestif.
4. ***Teen (13+)*** memiliki konten yang mungkin cocok untuk usia 13 keatas. Judul-judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, sedikit darah, simulasi perjudian, dan tidak jarang menggunakan bahasa yang kuat.
5. ***Mature (17+)*** memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi *intens* kekerasan, darah, konten seksual dan bahasa yang kuat.
6. ***Adults Only (18+)*** memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun keatas. Judul-judul dalam kategori ini mungkin termasuk adegan yang *intens*, kekerasan dan juga mengandung konten seksual.
7. ***RP (Rating Pending)*** telah disampaikan kepada ESRB dan sedang menunggu penilaian akhir. (Simbol ini hanya muncul dalam iklan sebelum permainan dibebaskan) [10].

2.1.5 Kategori Game Berdasarkan Platform yang Digunakan

Kategori game berdasarkan platform yang digunakan adalah sebagai berikut :

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.

2.1.6 Kategori Game Berdasarkan Jenisnya

Kategori game berdasarkan jenisnya diantaranya sebagai berikut :[6]

1. Aksi – Shooting, (tembak-tembakan, atau hajar-hajaran bisa juga tusuk-tusukan, 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. Termasuk didalamnya :

a. *First person shooting (FPS)* seperti Counter Strike dan Call of Duty
b. *Drive n' shoot,* menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan, contoh : Spy Hunter, Rock and Roll Racing, Road Rash.
c. *Shoot em' up,* seperti Raiden, 1942, dan gradius.
d. *Beat 'em up* (tonjok hajar) seperti Double Dragon dan Final Fight, lalu *hack and slash* (tusuk tebas) seperti Shinobi dan Legend of Kage.
e. *Light gun shooting,* yang menggunakan alat yang umumnya berbentuk seperti senjata, seperti Virtua Cop dan Time Crisis.

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, o iya, *combo*-pun menjadi esensial untuk mengalahkan lawan secepat mungkin. 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. *Seri Street Fighter, Tekken, Mortal Kombat, Soul Calibur* dan *King of Fighter* adalah contohnya.
3. Aksi – 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. Menurut penulis, game jenis ini sudah berkembang jauh hingga menjadi genre campuran *action beat-em up* juga, dan sekarang, di tahun 2000 an, jenis ini cenderung untuk memiliki visual 3D dan sudut pandang orang ke-tiga. *Tomb Rider, Grand Theft Auto dan Prince of Persia* termasuk didalamnya.

4. Petualangan. Bedanya dengan jenis video game aksi-petualangan, refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. Video 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. Termasuk didalamnya:

   a. *Petualangan dengan teks* atau sistem tunjuk dan klik, contoh: *Kings Quest, Space Quest, Heroes Quest, Monkey Island, Sam and Max,*

   b. *Novel atau film interaktif,* seperti game "*dating*" yang banyak beredar di jepang, *Dragons Lair* dan *Night Trap*.

5. Simulasi, Konstruksi dan manajemen. *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. Contoh: *Sim City, The Sims, Tamagotchi.*

6. Role Playing. Video 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, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman maupun mahluk peliharaan.Secara kebudayaan, pengembang game Jepang biasanya membuat *Role Playing Game* (RPG) ke arah cerita linear yang diarahkan seolah karakter kita adalah tokoh dalam cerita itu, seperti *Final Fantasy*, *Dragon Ques*t dan *Xenogears.* Sedangkan pengembang game RPG Eropa, cenderung membuat karakter kita bebas memilih jalan cerita sendiri secara non-linear, seperti Ultima, Never Winter Nights, baldurs gate, Elder Scroll, dan Fallout.

7. Strategi. Kebalikan dari video game jenis action yang berjalan cepat dan perlu refleks secepat kilat, video game jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. 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. Pemain game strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan game action. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumberdaya (uang, besi, kayu,minyak,dll), hingga ke pembelian dan peng-upgrade-an pasukan atau teknologi. Game jenis ini terbagi atas:

a. *Real time Strategy,* game berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: *Starcraft, Warcraft*, dan *Command and Conquer*.
b. *Turn based Strategy* , game yang berjalan secara bergiliran, saat kita mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya, layaknya catur. contoh: Front Mission, Super robot wars, Final Fantasy tactics, Heroes of might and magic, Master of Orion. Sebenarnya ada yang memilah lagi menjadi jenis tactical dan strategi, namun penulis cenderung untuk menggabungkannya karena perbedaannya hanya ada di masalah skala dan ke-kompleks-an dalam manajemen sumber daya-nya saja.

8. *Puzzle*. 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 permainan jenis ini adalah juga unsur permainan dalam video game petualangan maupun game edukasi. Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman.
9. Simulasi kendaraan. Video Game jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut. Terbagi atas beberapa jenis:
    a. Perang. Video game simulasi kendaraan yang sempat tenar di tahun 90-an ini mengajak pemain untuk menaiki kendaraan dan berperang melawan kendaraan lainnya. Dan kebanyakan diantaranya memiliki judul sama dengan nama kendaraannya. Contoh : Apache 64, Comanche, Abrams, YF-23, F-16 fighting eagle. Tetapi game kehidupan bajak laut seperti ‘Pirates!’ pun dapat dikategorikan disini.
    b. *Balapan.* Dari namanya sudah jelas, siapa sampai duluan di garis finish dialah pemenangnya! Terkadang malah pemain dapat memilih kendaraan, mendandani, upgrade mesin bahkan mengecatnya. Contoh: Top Gear, Test Drive, Sega Rally Championship, Daytona, Grand Turismo, Need For Speed, Mario Cart, ManXTT

c. *Luar Angkasa.* Walau masih dapat dikategorikan simulasi kendaraan perang, tetapi segala unsur fiksi ilmiah dan banyaknya judul yang beredar membuat subgenre ini pantas dikategorikan diluar simulasi kendaraan perang. Jenis ini memungkinkan pemain untuk menjelajah luar angkasa, berperang dengan mahluk alien, mendarat di planet antah berantah atau sekedar ingin merasakan bagaimana menjadi kapten di film fiksi ilmiah kesayangan kamu. Contoh: Wing Commander, Freelancer , Star Wars X-Wing, Star Wars Tie Fighter, dll

d. *Mecha.* Pendapat bahwa hampir tidak ada orang yang terekspos oleh film robot jepang saat kecilnya tidak memimpikan ingin mengendalikan robot, memang sulit dibantah. Dipopulerkan oleh serial Mechwarrior oleh Activision, subgenre Simulasi Mecha ini memungkinkan pemainnya untuk mengendalikan robot dan menggunakannya untuk menghancurkan gedung, helikopter dan tentu saja robot lainnya. Contoh: Mechwarrior, Gundam Last war Chronicles, dan Armored Core.

10. Olahraga. Singkat padat jelas, bermain sport di PC atau konsol anda. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, Seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater, dll [10].

2.1.7 Sejarah Game Pertama di Indonesia

Divinekids adalah *game* yang diakui dan mendapat award resmi dari MURI (Museum Rekor Indonesia) sebagai Game Pertama Indonesia pada tahun 2004 dan banyak majalah-majalah Game dan Komputer (Gamestation, PCGAME, Bobo, INDOPOS, Game21 dan lain sebagainya) . Sebenarnya sudah banyak game-game buatan orang-orang indonesia yang lain tetapi karena kekurangan syarat-syarat minimal tertentu seperti : Karakter menggunakan karakter asing, penyebaran yang kurang luas, bahasa yang digunakan bukan bahasa Indonesia, tidak terdaftar resmi Copyright ataupun Trademarknya dan lain sebagainya, karena itu tidak dipilih sebagai Game Pertama Indonesia Resmi.

Menurut sebagian pengamat game ini adalah kurangnya investigasi MURI dalam hal melihat adanya game-game tersebut sebelum memberikan gelar game pertama ke David Setiabudi. Terlebih mereka tidak meminta pendapat dari pihak yang memahami dunia game Indonesia itu sendiri.

Pengamat-pengamat yang berpendapat berbeda dengan MURI dan majalah-majalah PC dan Game, memposisikan sejarah game Indonesia sebagai berikut: Sejarah Game Indonesia menurut Gamedevid. Selain itu juga masih ada banyak game-game lain yang belum terdokumentasi dengan baik. Beberapa game buatan Elex Media Komputindo dengan copyright resmi dan penyebaran yang luas ada juga yang dibuat sebelum sebelum dirilisnya Divine Kids. Game ini dibuat dan dibagikan secara gratis melalui CD majalah2 game/ komputer dan melalui Internet.

Game-game Divinekids berjumlah 38 lebih dan terus bertambah (2006). Pembuatnya adalah David Setiabudi dan mahasiswa-mahasiswi bimbingannya. Pembuatan musik gamenya dibuat oleh Rama Aurora, tunanetra berbakat membuat musik digital.

David Setiabudi membuat game juga tahun 1992 tetapi tidak diakui sebagai game pertama Indonesia karena kurang memenuhi syarat. Ada juga game-game buatan Korea yang terkenal diterjemahkan ke bahasa Indonesia & game-game yang penyebaran kurang luas & game yang berkaraker asing atau berbahasa asing & tidak terdaftar Copyright secara resmi tidak terpilih menjadi Game Pertama Indonesia. Ada yang berpendapat asal tahun tertulis paling tua itu patut disebut Game Pertama Indonesia tidak perduli dengan buatan negara mana, atau tersebar di lokal saja. Sampai sekarang masih diperdebatkan kelompok tertentu [8].

2.1.8 Genre Game

Game dibagi-bagi lagi ke dalam beberapa genre (jenis), diantaranya yaitu:[10]

1. *Action*
2. *Fighting*
3. *Education*
4. *Adventure*
5. *Racing*
6. *Strategy*

7. *Simulation*
8. *Puzzle*
9. *Flight*
10. *RPG (Roler Playing Game)*

2.2 Kecerdasan Buatan / AI (*Artificial Intelligence*)

AI (*Artificial Intelligence*) atau kecerdasan buatan merupakan cabang ilmu dari ilmu komputer yang konsern dengan pengautomatisasi tingkah laku cerdas. Pernyataan tersebut juga dapat dijadikan definisi dari AI. Definisi ini menunjukkan bahwa AI adalah bagian dari komputer sehingga harus didasarkan pada *sound theoretical* (teori suara) dan prinsip-prinsip aplikasi dari bidangnya. Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan, algoritma yang diperlukan untuk mengaplikasikan pengetahuan tersebut, serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya.

Namun, definisi di atas belum memadai karena pada kenyataannya istilah kecerdasan itu sendiri tidak didefinisikan atau dipahami dengan sangat baik. Kerap kali kita meyakini telah mengetahui tingkah laku cerdas saat melihatnya sendiri. Akan tetapi, keraguan juga akan muncul ketika orang lain menyatakan kecerdasan dalam cara khusus menurut cara mereka yang cukup membantu dalam evaluasi kecerdasan program komputer. Dengan demikian, belum ada suatu definisi yang cocok untuk *Artificial Intelligence* itu sendiri.

Definisi AI ini menjadi masalah tersendiri yang harus dijawab : apakah kecerdasan tersebut merupakan suatu *"single faculty"*, atau apakah hanya sebuah nama untuk kumpulan perbedaan dan tidak ada hubungannya dengan kemampuan? Apa perlunya mempelajari AI? Apakah ada hubungannya dengan proses belajar, kreativitas dan intuisi? Dapatkah kecerdasan disimpulkan dari tingkah yang dapat diobservasi atau membutuhkan pembuktian dari suatu mekanisme internal yang khusus? Bagaimana pengetahuan direpresentasikan dalam jaringan saraf dari kehidupan? Semua pertanyaan tersebut merupakan pertanyaan yang mungkin belum terjawab dan tentunya akan membantu kita untuk terus berusaha menggeluti, mempelajari dan mencari metode yang dapat menjadi solusi bagi pertanyaan ini, dan nantinya akan membentuk suatu intisari AI modern. AI menawarkan media dan uji teori dari kecerdasan. Teori-teori tersebut dapat dinyatakan dalam bahasa program komputer, diuji dan dibuktikan melalui eksekusi pada komputer.

Definisi tersebut dibagi dalam dua kotak utama, kotak bagian atas menunjukkan definisi berkaitan dengan proses pemikiran dan penalaran dan kotak yang satu menunjukkan tingkah laku.

Dari beberapa definisi AI dapat dibagi dalam empat kategori, yaitu :

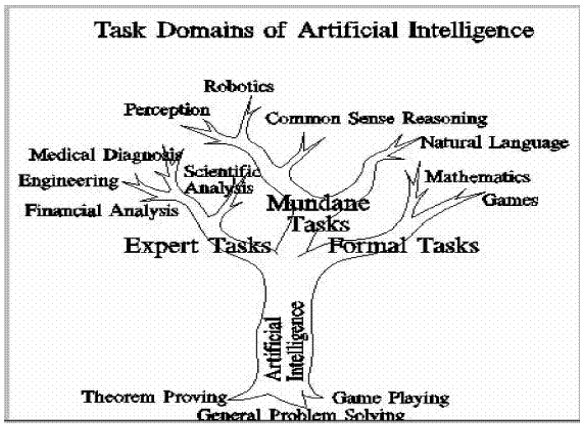
1. Sistem yang dapat berfikir seperti manusia *"Thinking humanly"*.
2. Sistem yang dapat bertingkah laku seperti manusia *"Acting Humanly"*.
3. Sistem yang dapat berfikir secara rasional *"Thinking rationally*".
4. Sistem yang dapat bertingkah laku secara rasional *"Acting rationally"*.

Secara historis, keempat pendekatan tersebut telah dilakukan. Pendekatan manusia haruslah merupakan suatu ilmu empiris, termasuk hipotesa dan konfirmasi percobaan dan pendekatan rasional meliputi kombinasi dari matematika dan rekayasa [2].

2.2.1 Area Aplikasi AI

Ada dua hal yang sangat mendasar mengenai penelitian-penelitian AI, yaitu *knowledge representation* (representasi pengetahuan) dan *search* (pelacakan). Para peneliti AI terus mengembangkan berbagai jenis teknik baru dalam menangani sejumlah permasalahan yang tergolong ke dalam AI seperti vision dan percakapan, pemrosesan bahasa alami dan permasalahan khusus seperti diagnosa medis.

AI seperti bidang ilmu lainnya juga memiliki sejumlah sub disiplin ilmu yang sering digunakan untuk pendekatan yang esensial bagi penyelesaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Berikut ini merupakan sejumlah bidang terapan utama dari kontribusinya terhadap perkembangan AI yang digambarkan bentuk pohon berikut ini :



Gambar 2.1 Beberapa bidang terapan AI

Area aplikasi AI diantaranya yaitu :

1. Game Playing.
2. Penalaran Otomatis dan Pembuktian.
3. Sistem Pakar.
4. Pemahaman Bahasa Alami.
5. Perencanaan dan Robotik.
6. Jaringan Saraf Tiruan (JST).
7. Algoritma Genetika [2].

2.2.2 Perbandingan Kecerdasan Buatan dengan Kecerdasan Alamiah

Ada beberapa keuntungan kecerdasan buatan dibanding kecerdasan alamiah, yaitu :

1. Lebih permanen.
2. Memberikan kemudahan dalam duplikasi dan penyebaran.
3. Relatif lebih murah dari kecerdasan alamiah.
4. Konsisten dan teliti.
5. Dapat didokumentasi.
6. Dapat mengerjakan beberapa task dengan lebih cepat dan lebih baik dibanding manusia.

2.2.3 Perbedaan Komputasi AI dengan Proses Komputasi Konvensional

Perbedaan komputeasi AI dengan proses komputasi konvensional dapat dilihat pada tabel dibawah ini :

Tabel 2.1 Pemrosesan data komputer konvensional

Proses	**Tugas**
Kalkulasi	Mengerjakan operasi-operasi matematis seperti : +, =, x, :, atau mencari akar persamaan, menyelesaikan rumus/ persamaan
Logika	Mengerjakan operasi logika seperti and, or, invert
Penyimpanan	Menyimpan data dan gambar pada file
Retrieve	Mengakses data yang disimpan pada file
Translate	Mengonversi data dari satu bentuk ke bentuk yang lain
Sort	Memeriksa data dan menampilkan dalam urutan yang diinginkan
Edit	Melakukan perubahan, penambahan dan penghapusan data
Monitor	Mengamati *event* eksternal dan internal serta melakukan tindakan jika kondisi tertentu tercapai
Kontrol	Memberikan perintah atau mengendalikan peralatan luar

Perbedaan antara AI dengan pemrograman konvensional adalah seperti tabel berikut ini :

Tabel 2.2 Perbedaan antara Pemrograman AI dan Konvensional

AI	Komputasi Konvensional
Representasi dan Manipulasi simbol	Algoritama
Memberitahu komputer tentang suatu masalah	Memerintah komputer untuk menyelesaikan masalah
Komputer diberi pengetahuan dan kemampuan inferensi	Memberi data kepada komputer dan program

2.3 AI yang digunakan

2.3.1 Algoritma DFS

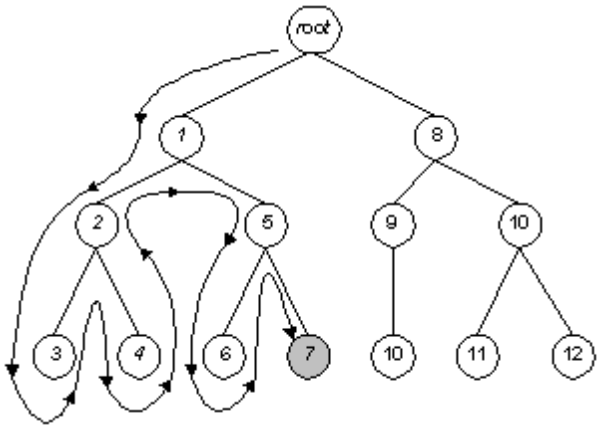
1. Pengertian Algoritma DFS (*Depth First-Search*)

Depth-first search (DFS) melakukan pencarian secara preorder. Mengunjungi anak suatu simpul sebelum simpul tetangganya. Berkaitan dengan mesin pencari, DFS ini cenderung mengindeks dokumen berdasarkan suatu link.

DFS adalah algoritma untuk melakukan traversal atau pencarian pada sebuah graf atau pohon. Dimulai dari simpul akar, pencarian akan dilakukan dan mengeksplorasi sejauh mungkin pada tiap cabang sebelum akhirnya melakukan *backtracking*.

Algoritma pencarian mendalam (DFS) mencari solusi dengan mengunjungi simpul akar, lalu simpul-simpul yang bertetangga dengan simpul akar (setingkat di bawahnya), terus sampai simpul paling dalam pada bagian

tersebut. Setelah itu, dicari simpul yang telah dikunjungi pada tingkat terdekat dan terdalam, lalu simpul yang bertetangga dengan simpul ini dikunjungi, demikian seterusnya sampai seluruh simpul telah dikunjungi. Berikut gambar yang mengiilustrasikan urutan simpul yang dikunjungi pada algoritma BFS



Gambar 2.2 Ilustrasi urutan kunjungan simpul pada algoritma DFS

Dari gambar di atas, dapat dilihat bahwa dengan algoritma DFS, setiap anak simpul pertama yang bertetangga dengan simpul akar dikunjungi sampai tingkat terdalamnya lebih dahulu, lalu seluruh simpul pada subpohon tersebut, sebelum simpul lain yang juga bertetangga dengan simpul akar [3].

a. Cara Kerja Algoritma DFS

Dalam algoritma DFS, simpul yang telah dikunjungi disimpan dalam suatu tumpukan (*stack*). Antrian ini digunakan untuk mengacu simpul-simpul yang akan dikunjungi sesuai urutan tumpukan (masuk terakhir, keluar pertama) dan mempermudah proses runut-balik jika simpul sudah tidak mempunyai anak (simpul pada kedalaman maksimal). Untuk memperjelas cara kerja algoritma DFS beserta tumpukan yang digunakannya, berikut langkah-langkah algoritma DFS:

1. Masukkan simpul ujung (akar) ke dalam tumpukan
2. Ambil simpul dari tumpukan teratas, lalu cek apakah simpul merupakan solusi
3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam tumpukan
5. Jika tumpukan kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan
6. Ulangi pencarian dari langkah kedua

b. Properti Algoritma DFS

Berbeda dengan algoritma BFS yang hanya melakukan pencarian solusi pada graf sampai menemukannya, sehingga membutuhkan waktu yang lama, algoritma DFS menggunakan metode heuristik, sehingga solusi dapat ditemukan lebih cepat.

Kompleksitas ruang algoritma DFS juga jauh lebih rendah dari algoritma BFS, karena hanya menyimpan simpul-simpul pada suatu subpohon. Akan tetapi, kompleksitas waktu algoritma DFS sama dengan kompleksitas waktu algoritma BFS. Algoritma DFS juga dapat mencari solusi secara tuntas seperti algoritma BFS, berarti algoritma ini akan menemukan solusi terbaik dan terdekat bagaimanapun bentuk graf tersebut. Walaupun demikian, masih terdapat kelemahan algoritma DFS, yaitu jika pada graf yang sangat besar kedalaman graf terus bertambah dan tidak diketahui akhirnya, sehingga memori tidak cukup. Hal

ini dapat diatasi dengan pengembangan algoritma DFS yaitu pencarian mendalam berulang (*Iterative Deepening Search* - *IDS*).

c. Aplikasi Algoritma DFS dalam Teori Graf

Seperti halnya algoritma BFS, algoritma DFS juga dapat digunakan dengan mudah pada teori graf. Bahkan banyak persoalan yang tidak dapat diselesaikan algoritma BFS yang dapat diselesaikan oleh algoritma DFS. Berikut contoh-contoh penggunaan algoritma BFS dalam

persoalan teori graf:

d. Pencarian Komponen Terhubung dalam Graf

Sama seperti pada algoritma BFS, dalam algoritma DFS, seluruh simpul yang dikunjungi juga pasti merupakan simpul yang terhubung dengan simpul akar, karena simpul-simpul tersebut hanya dikunjungi jika bertetangga dengan simpul ayahnya, sehingga seluruh simpul yang terhubung dengan simpul akar juga pasti dikunjungi.

Dalam hal pencarian komponen terhubung dalam graf, algoritma BFS dan DFS menghasilkan pencarian yang memiliki kecepatan dan kebutuhan memori sama, karena setiap simpul harus dikunjungi. Hasil pencarian juga sama, walaupun proses yang dilakukan berbeda. Kedua algoritma dapat menghasilkan komponen terhubung terbesar pada graf yang mencakup simpul awal (akar). Oleh karena itu, kedua algoritma dapat digunakan untuk menyelesaikan persoalan ini dengan mangkus.

e. Pengurutan Topologi

Dalam teori graf, pengurutan topologi dari graf DAG (graf berarah tanpa siklus) adalah pengurutan linear simpul-simpul di dalam graf yang bersesuaian dengan urutan di mana x dikunjungi sebelum y jika terdapat lintasan dari x ke y. Pengurutan topologi dapat diaplikasikan pada penjadwalan sekuensial suatu pekerjaan. Dengan menggunakan algoritma DFS, pengurutan topologi dilakukan dengan algoritma berikut:

1. Lakukan algoritma DFS pada graf G
2. Inisialisasi *list* kosong
3. Jika suatu simpul telah selesai dipanggil seluruh anaknya, masukkan simpul ke dalam l*i*st
4. Kembalikan list yang telah berisi semua simpul.

Urutan setiap simpul didapat dari posisinya dalam *list*, mulai dari awal *list*.

f. Pencarian Komponen Terhubung Kuat

Jika algoritma BFS dapat mencari komponen terhubung pada graf tak berarah, algoritma DFS dapat melakukan sesuatu yang lebih rumit, yaitu mencari komponen terhubung kuat pada graf berarah. Graf berarah disebut terhubung kuat jika untuk setiap pasangan simpul A dan B terdapat lintasan dari A ke B serta dari B ke A. Komponen terhubung kuat pada graf berarah merupakan subgraf terhubung kuat maksimal pada suatu graf. Komponen ini membentuk partisi dari graf.

Dalam pencarian komponen terhubung kuat, digunakan dua kali algoritma DFS, yaitu pada graf serta graf transpose dari graf tersebut. Berikut algoritma DFS yang digunakan dalam pencarian komponen terhubung kuat:

1. Lakukan algoritma DFS pada graf G
2. Cari graf transpose dari G: GT
3. Lakukan algoritma DFS pada graf GT secara terbalik
4. Kembalikan output hutan DFS yang terbentuk pada langkah 3 sebagai suatu komponen terhubung kuat

2.4 Analisis Sistem

Analisis sistem yaitu proses penguraian suatu sistem yang akan dirancang oleh satu orang atau sekelompok orang, dimana informasi yang utuh akan diuraikan ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan.

Langkah-langkah yang harus dilakukan untuk menganalisis sebuah sistem yaitu :

1. Mengidentifikasi masalah
2. Memahami kerja sistem yang ada atau yang akan dibangun
3. Menganalisis sistem.
4. Membuat laporan hasil analisis.

2.5 *Tools* yang Digunakan

2.5.1 OOP (Object Oriented Programming)

OOP (*Object Oriented Programming*) 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.

Pemrograman orientasi-objek menekankan konsep berikut:

- **Kelas (*Class*)** — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class* of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah *class* sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
- **Objek (*Object*)** - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
- **Abstraksi (*Abstract*)** - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa

mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

- **Enkapsulasi (*Encapsulation*)** - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
- **Polimorfisme (*Polimorfism*)** melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam

pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai *polimorfisme* melalui penggunaan fungsi kelas-pertama.

- **Inheritas (*Inheritance*)**. Mengatur *polimorfisme* dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas) [12].

2.5.2 *UML* (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun system (Flowler, 2006). *Unified Modeling Language* (*UML*) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. *UML* adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. *UML* mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang *UML* sudah mulai banyak digunakan oleh para praktisi OOP. *UML* merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM [11].



Gambar 2.3 Logo UML

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. *UML* dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian *UML* dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan *UML* dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan system.

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

1. ***Use Case Diagram*** untuk memodelkan proses bisnis.
2. ***Conceptual Diagram*** untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. ***Sequence Diagram*** untuk memodelkan pengiriman pesan (*message*) antar *objects*.
4. ***Collaboration Diagram*** untuk memodelkan interaksi antar *objects*.

5. ***State Diagram*** untuk memodelkan perilaku *objects* di dalam sistem.
6. ***Activity Diagram*** untuk memodelkan perilaku *Use Cases* dan *objects* di dalam *system*.
7. ***Class Diagram*** untuk memodelkan struktur kelas.
8. ***Object Diagram*** untuk memodelkan struktur *object*.
9. ***Component Diagram*** untuk memodelkan komponen *object.*
10. ***Deployment Diagram*** untuk memodelkan distribusi aplikasi [11].

2.6 Game Engine

2.6.1 Definisi *Game Engine*

Game Engine adalah system perangkat lunak yang dirancang untuk menciptakan dan pengembangan video game. Ada banyak mesin permainan yang dirancang untuk bekerja pada konsol permainan video dan sistem operasi desktop seperti Microsoft Windows, Linux, dan Mac OS X. fungsionalitas inti biasanya disediakan oleh mesin permainan mencakup mesin render ("*renderer*") untuk 2D atau 3D grafis, mesin fisika atau tabrakan (dan tanggapan tabrakan), suara, script, animasi, kecerdasan buatan, jaringan, streaming, manajemen memori, threading, dukungan lokalisasi, dan adegan grafik. Proses pengembangan permainan sering dihemat oleh sebagian besar menggunakan kembali mesin permainan yang sama untuk menciptakan permainan yang berbeda.

2.6.2 Tipe – tipe *Game Engine*

Game engine biasanya datang dengan berbagai macam jenis dan ditujukan untuk berbagai kemampuan pemrogramman. Tapi dalam pembahasan sekarang

ini, ketimbang harus me-list semua jenis game engine yang ada di wikipedia, mendingan kita bahas garis besar nya saja.

Roll-your-own game engine. Banyak perusahaan game kecil seperti publisher indie biasanya menggunakan engine nya sendiri. Ini berarti mereka menggunakan API seperti XNA, DirectX atau OpenGL untuk membuat game engine mereka sendiri. Disisi lain, mereka kadang menggunakan libraries komersil atau malah open source. Terkadang mereka malah membuat semuanya dari nol.

Biasanya, game engine tipe ini lebih disukai karena selain kemungkinan besar tersedia gratis, juga memperbolehkan mereka, para developer, lebih fleksibel dalam mengintegrasikan komponen yang diinginkan untuk dibentuk sebagai game engine mereka sendiri. Kelemahannya, banyak engine yang dibuat dengan cara semacam ini malah menyerang balik developernya. Menara Games Studio membutuhkan satu tahun penuh untuk menyempurnakan game engine nya, hanya untuk di tulis ulang semuanya dalam beberapa hari penggunaan karena adanya bug kecil yang sangat mengganggu.

Mostly-ready game engines. Engine engine ini biasanya sudah menyediakan semuanya begitu diberikan pada developer / programmer. Semuanya termasuk contoh GUI, physiscs, libraries model dan texture, dan segalanya. Banyak dari mereka yang sudah benar benar matang, sehingga dapat langsung digunakan untuk scripting sejak hari pertama.

Biasanya game engine semacam ini memiliki batasan batasan, terutama jika dibandingkan dengan game engine sebelumnya yang benar benar terbuka

lebar. Hal ini ditujukan agar tidak terjadi terlalu banyak error yang mungkin terjadi setelah sebuah game yang menggunakan engine ini dirilis, dan masih memungkinkan game engine nya itu sendiri untuk mengoptimalkan kinerja game nya. Banyak dari game engine seperti ini, Unreal Engine, Source Engine, id Tech Engine dan sebagainya, yang sudah sangat optimal dibandingkan jika harus membuat dari awal. Hal ini dengan serta merta menyingkat sangat banyak waktu dan jelas, biaya dari para Developer game.

Point-and-click engines. Engine untuk point-and-click merupakan engine yang sangat amat dibatasi, tapi dibuat sangat user friendly. Kamu bahkan bisa mulai membuat gamemu sendiri menggunakan engine seperti Game Maker, Torque Game Builder dan Unity3D. Dengan sedikit memanfaatkan coding, kamu sudah bisa merilis game point-and-click yang kamu banget.

Kekurangannya terletak pada terbatasnya jenis interaksi yang bisa dilakukan, dan biasanya hal ini mencakup semuanya, mulai dari grafis, hingga tata suara. Tapi bukan berarti *game engine* jenis ini nggak berguna, bagi *developer* cerdas dan berdaya kreativitas tinggi, game engine bapuk seperti ini bisa dirubah menjadi sebuah game menyenangkan, seperti *Flow*. Game engine seperti ini memang ditujukan bagi developer yang ingin menyingkat waktu pemrogramman, dan secepatnya merilis game game mereka.

Beberapa contoh jenis game engine :

1. Freeware game engine/open source game engine

- Blender
- *Golden T Game Engine* (GTGE)

- DXFramework
- Ogre
- Aleph One
- Axiom Engine
- Allegro Library
- Box2D
- Build Engine
- Cube
- Cube 2
- DarkPlaces
- jMonkeyEngine (jME)
- Panda3D
- Sphere

2. Commercial engines/game engine berbayar (komersial)

- Alamo
- A.L.I.V.E
- BigWorld
- DXStudio
- Dunia Engine
- Euphoria
- GameStudio
- Jade Engine
- Jedi

- Medusa
- RPG Maker VX
- RPG Maker XP
- RPG Maker 2003
- RPG Maker 95
- Vision Engine

2.6.3 Java

Java 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 aplikasi-aplikasi 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.

2.6.3.1 Java Versi Lama (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" [12].

2.6.3.2 Java 2

Sun Myrosystems telah mendefinisikan tiga buah edisi dari Java 2 yaitu sebagai berikut :[12]

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

3. **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*.

2.6.3.3 Greenfoot

Greenfoot merupakan sebuah IDE berbasis java yang dikhususkan untuk pembuatan game sederhana. Greenfoot diprakarsai oleh Michael Kolliung pada tahun 2003. Prototype yang pertama dibangun oleh Poul Henriksson dan Michael Kolling pada tahun 2003/2004. semenjak tahun 2005 pembuatan greenfoot ini dilanjutkan dengan melibatkan anggota kelompok dari University of Kent dan Deakin University.

Pada awalnya, greenfoot bertujuan untuk menarik minat anak-anak untuk belajar pemrograman, greenfoot ini cocok untuk anak usia 13 tahun ke atas. Cara penggunaannya pun cukup mudah. Selain berbentuk teks seperti editor lainnya, pada greenfoot ini terdapat perangkat alat bantu seperti *class* browser, editor dan compiler. Greenfoot juga mendukung bahasa Java secara utuh. Dengan adanya bantuan perangkat-perangkat tadi, konsep OOP pada greenfoot lebih mudah dimengerti.

Didalam greenfoot terdapat project yang dinamakan scenario. Di dalam project tersebut, kita bisa membuat dua macam tipe *class* dari *library* greenfoot. Yaitu *class world* dan *actor*. *World* merupakan latar dari game yang akan dibuat. Untuk membuat *class world*, caranya adalah dengan meng-klik *new sub class* di bagian *world*. Maka akan mucul sebuah kotak dialog. Kemudian kita diminta

untuk memberi nama untuk kelas *world* tersebut. Setelah itu kita dapat memilih gambar sebagai latar belakang *class* world yang akan kita buat. Di dalam greenfoot sendiri telah tersedia beberapa gambar untuk latar belakang, seperti gambar tanah, pasir, batu, air, dll.

Class actor adalah sebuah kelas yang berfungsi sebagai karakter pada game yang akan kita buat. Untuk membuat actor caranya adalah dengan meng-klik icon aktor, kemudian pilih *new subclass*. caranya hampir mirip dengan pembuatan *class world*. Pada *class actor* juga telah disediakan beberapa gambar, sepertii gambar animal, people, symbol, dan lain-lain [13].

2.6.3.4 Corel Draw

Corel draw 12 adalah sebuah aplikasi grafis berbasis vector. Format vector adalah gambar yang membentuk sejumlah objek garis dan objek kurva berdasarkanrumusan matematis. Format vector lebih banyak digunakan untuk membentuk objek buatan, seperti menggambar objek dua dimensi, yang lebih ditekankan ke dalam pembuatan objek garis, lingkaran, polygon dan persegi panjang. Sedangkan untukobjek tiga dimensi, lebih ditekankan ke dalam pembuatan : bola, kubus dan tabung.

Objek vector, banyak digunakan dalam pembuatan pengolahan teks dan logo.Software yang banyak berhubungan dengan format vector yaitu Corel Draw, freehand, dan lain-lain. Ruang penyimpanan format vector relafitf lebih kecil dibandingkan dengan format bitmap [14].