BAB 2
LANDASAN TEORI

2.1 Permainan (*Game*)

Permainan berasal dari arti kata bahasa inggris yaitu *game*. Permainan merupakan kegiatan atau sebuah aktivitas yang di dalamnya terdapat peraturan, cara bermain (*gameplay*), budaya. Permainan dalam hal ini, merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Permainan bertujuan untuk menghibur, biasanya permainan banyak disukai oleh anak-anak hingga orang dewasa. Berdasarkan perkembangannya pun permainan dibedakan menjadi dua yaitu, permainan tradisional dan permainan modern. Dari perbedaan dua permainan tersebut dapat dilihat sebagai berikut[10] :

1. Permainan modern
   - Tidak memerlukan biaya yang mahal.
   - Meningkatkan kekompakan anak.
   - Kerjasama antar anak yang kuat.
   - Tanpa memilih teman yang kaya ataupun miskin.
2. Permainan tradisional
   - Memerlukan biaya yang lebih banyak.
   - Bermain secara individu (karena permainan tergantung skill masing-masing anak semakin sering memainkan semakin hebat).
   - Permainan kerjasamanya kurang, lebih ber-ego dan gengsi.
   - Permainan tertentu yang bisa dinikmati semua anak.

Permainan tradisional juga mempunyai macam-macam permainan seperti gambar berikut ini:



Gambar 2.1 permainan lompat tali[10].



Gambar 2.2 permainan petak umpet[10].



Gambar 2.3 permainan enggrang[10].



Gambar 2.4 permainan kelereng[10].

Dan permainan modern pun mempunyai macam-macam permainan seperti gambar berikut ini:



Gambar 2.5 *Playstation*[10].



Gambar 2.6 *mobile game*[10].



Gambar 2.7 PC *game online*[10].

2.1.1 Pengertian *Game*

Pengertian permainan menurut beberapa ahli adalah [11]:

1. Menurut Chris Crawford, seorang designer *game* komputer mengemukakan bahwa *game*, pada intinya adalah sebuah interaktif, aktivitas yang berpusat pada sebuah pencapaian, mempunyai pelaku aktif, mempunyai lawan.
2. Menurut David parlett, *game* adalah sesuatu yang memiliki "akhir dan cara mencapainya" artinya mempunyai tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.
3. Menurut Roger Caillos seorang sosiolog Perancis, dalam bukunya yang berjudul *Les jeux et les hommes* menyatakan *game* adalah aktivitas yang mencakup karakteristik berikut: *fun* (bebas bermain adalah pilihan bukan kewajiban), *separate* (terpisah), *uncertain* (tidak menentu), *non-productive* (tidak produktif), *governed by rules* (ada aturan), *fictitious* (pura-pura).
4. Menurut Greg Costikyan, *game* adalah sebentuk karya seni dimana peserta yang disebut pemain, membuat keputusan untuk mengelola sumber daya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan.
5. Menurut Clark C. Abt, *game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan "dibatasi oleh konteks tertentu" (misalkan, dibatasi oleh peraturan).

2.1.2 Teori Permainan

Teori permainan yang pertama kali ditemukan oleh sekelompok ahli matematika pada tahun 1944. Teori itu dikemukakan oleh Jhon von Neumann dan 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 dan kekalahan dalam berbagai situasi"[5]. Dengan adanya teori permainan tersebut memberikan manfaat ke dalam beberapa hal :

1. Mengembangkan suatu analisa pengambilan keputusan dalam situasi persaingan.
2. Menguraikan metode kuantitatif yang semantik bagi pemain yang terlibat dalam persaingan untuk memilih strategi dalam pencapaian tujuan.
3. Member gambaran dan penjelasan kejadian dari situasi konflik persaingan.

Model-model teori permainan pun diklasifikasikan dengan sejumlah cara, seperti jumlah pemain, jumlah keuntungan, jumlah kerugian, dan jumlah strategi yang digunakan dalam permainan. Bila jumlah pemain ada dua, permainan disebut sebagai permainan dua pemain. Bila keuntungan atau kerugian sama dengan nol, disebut permainan jumlah nol. Representasi model matematis ini terdapat dalam *zero-sum games*.

Dalam representasi model matematisnya mempunyai dua jenis solusi optimal yaitu, pertama jenis permainan startegi murni (pure strategy game) dimana setiap pemain hanya menjalankan strategi tunggal, dan kedua permainan strategi campuran (mixed strategy game) dimana kedua pemain menjalankan strategi yang berbeda-beda.

2.1.3 Klasifikasi Permainan

2.1.3.1 Berdasarkan Jenis *Platform* yang Digunakan

Berdasarkan jenis *platform* yang digunakan game dibagi menjadi beberapa jenis, yaitu[13]:

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, contohnya Nitendo DS dan Sony PSP.
5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA.

2.1.3.2 Berdasarkan *Genre* Permainan *Game*

Berdasarkan *genre* permainan *game* dibagi menjadi beberapa jenis, yaitu[13]:

1. Aksi *Shooting*, *video game* jenis ini sangat memerlukan kecepatan *reflex*, koordinasi mata-tangan, juga *timing*, inti dari *game* jenis ini adalah tembak, tembak dan tembak. Permainan yang termasuk di dalam *genre* ini sebagai berikut :
   A. *First Person Shooting* (FPS), *video game* yang ciri utamanya adalah penggunaan sudut pandang orang pertama dengan tampilan layar yang mensimulasikan apa yang dilihat melalui mata karakter

yang dimainkan seperti permainan *game Counter Strike* dan *Call of Duty*.

B. *Drive n' Shoot*, *video game* yang ciri utamanya adalah menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan seperti permainan *game Spy Hunter*, *Rock and Roll Racing*, *Road Rash*.

C. *Shoot 'em up*, *video game* yang ciri utamanya adalah menggunakan unsur pesawat dengan tujuan utama menembak dan menghancurkan lawan seperti permainan *game Raiden*, 1924, dan *Gradius*.

D. *Beat 'em up*, *video game* yang ciri utamanya adalah perkelahian (hajar menghajar) seperti permainan *game* Shinobi dan *Legend of Kage*.

E. *Light gun shooting*, *video game* yang ciri utamanya adalah menggunakan senjata seperti permainan *game Virtual Cop* dan *Time Crisis*.

2. Pertarungan *Fighting*, 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. Petualangan, Permainan *game* yang termasuk di dalam *genre* ini sebagai berikut :

A. Aksi - Petualangan, *video game* yang ciri utamanya adalah 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. Permainan *game* 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, seperti permainan *game* Tomb Rider, Grand Theft Auto dan Prince of Persia termasuk didalamnya.

B. Petualangan, *video game* jenis petualangan berbeda dengan *video game* aksi – petualangan karena mempunyai ciri utama adalah kelihaian pemain dalam bergerak dan reflex, berlari, melompat hingga memecut atau menembak tidak diperlukan disini. *Video game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berfikir 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. *Video game* yang jenis permainannya termasuk ke dalamnya sebagai berikut :

   1. Petualangan dengan teks atau dengan sistem tunjuk dan klik, permainan *game* yang termasuk adalah *King Quest, Space Quest, Heroes Quest, Monkey Island, Sam and Max*.
   2. Novel atau film *interaktif*, seperti *game dating* yang banyak beredar di jepang seperti permainan *game Dragon lair* dan *Night Trap*.

4. Simulasi, Kontruksi 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.

5. *Role Playing Game* (RPG), *Video game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh atau 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 permainan *game* Final Fantasy, Dragon Quest dan Xenogears. Sedangkan pengembang *game* **RPG** Eropa, cenderung membuat karakter kita bebas memilih jalan cerita sendiri secara non-linear, seperti permainan *game* Ultima, Never Winter Nights, baldurs gate, Elder Scroll, dan Fallout.
6. 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. Permainan *game* jenis ini terbagi sebagai berikut :

A. *Real Time Strategy*, permainan *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 permainan *game* : Starcraft, Warcraft , dan Command and Conquer.

B. *Turn Based Strategy*, permainan game yang berjalan secara bergiliran, saat kita mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya, layaknya permainan *game* catur[12]. Contoh permainan *game* : Front Mission, Super robot wars, Final Fantasy tactics, Heroes of might and magic, Master of Orion. Sebenarnya ada yang memilah lagi menjadi jenis permainan *game* tactical dan strategi, namun cenderung untuk menggabungkannya karena perbedaannya hanya ada di masalah skala dan kompleksnya dalam manajemen sumber daya saja.

7. *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 seperti permainan *game* Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman.

8. 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 permainan *game* sebagai berikut :
   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 permaina *game* : Apache 64, Comanche, Abrams, YF-23, F-16 fighting eagle.
   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 permainan *game* : 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 permainan *game* : Wing Commander, Freelancer , Star Wars X-Wing, Star Wars Tie Fighter.

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* dan *Activision*, *subgenre* Simulasi Mecha ini memungkinkan pemainnya untuk mengendalikan robot dan menggunakannya untuk menghancurkan gedung, helikopter dan tentu saja robot lainnya. Contoh permainan *game* : Mechwarrior, Gundam Last war Chronicles, dan Armored Core.

E. 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. Contoh permainan *game* : Seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater.

2.1.3.3 Berdasarkan Kategori *Mode* Permainan *Game*

Berdasarkan kategori *mode* permainan *game* dibagi menjadi beberapa jenis, yaitu[13]:

1. *Multiplayer Online*, permainan *game* yang lagi trend di Indonesia bahkan dunia,menjadi salah satu titik balik mengapa dunia *game* dan *internet* di Indonesia dapat berkembang. Dan karena dimainkan online dan dengan sistem pembayaran menggunakan voucher, pembajakan sudah tidak menjadi masalah lagi. Permainan *game* yang dapat dimainkan secara bersamaan oleh lebih dari 2 orang (bahkan dapat mencapai puluhan ribu orang dalam satu waktu) membuat pemain dapat bermain bersama dalam satu dunia *virtual* dari sekedar *chatting* hingga membunuh naga bersama teman yang entah bermain di mana. Umumnya permainan tipe ini dimainkan di PC dan bertema RPG, walau ada juga yang bertema *music* atau *action*. Contoh permainan *game* : Ragnarok online, O2jam, World of Warcraft, Ayo Dance, Lineage, Rose online.

2. *Casual Games*, sesuai namanya permainan *game* yang *casual* itu tidak kompleks, memainkannya rileks dan sangat mudah untuk dipelajari (bahkan cenderung langsung bisa dimainkan). Jenis ini biasanya memerlukan spesifikasi komputer yang standar pada jamannya dan ukurannya tidak lebih dari 100 MB karena biasanya dapat di *download versi demo*-nya di *website* resminya. *Genre* permainannya biasanya *puzzle* atau *action* sederhana dan umumnya dapat dimainkan hanya menggunakan mouse (biasanya *game* lain menggunakan banyak tombol tergantung game-nya). Contoh permainan *game* : Diner Dash, Sally Salon, Bejeweled, Zuma, Feeding Frenzy, Insaniquarium.
3. *Edugames*, *video game* jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, entah untuk belajr mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. Developer yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan *design visual* ataupun animasinya. Contoh permainan *game* : Bobi Bola, Dora the explorer, Petualangan Billy dan Tracy.
4. *Advergames*, sering mengunjungi *website* merk-merk kesayangan anda. Permen coklat M&M, Coca-cola, Nike, A-Mild, atau Rexona. Anda pasti menemukan *game-game* yang dapat dimainkan lalu dapat anda beritahukan atau mengundang langsung ke teman-teman anda. jenis permainan *game* yang biasanya mudah dimainkan ini mengusung dan menampilkan produk atau brand mereka baik secara gamblang maupun tersembunyi. Di era tumbuhnya media-media baru berteknologi tinggi sekarang ini, dunia periklanan memang sudah tidak lagi terbatas pada TV, koran, majalah, *billboard* dan radio, *video game* sekarang telah menjadi sarana beriklan atau membangun *brand-awareness* yang efektif. Baik melalui internet maupun di mainkan di

event-event mereka, *edugames* terasa semakin dibutuhkan untuk menjaring calon konsumen bagi produk yang menggunakan *advergames* ini. Contoh produk di indonesia yang membuat *advergames*: A-Mild, Rexona teens, Axe.

2.1.3.4 Berdasarkan Umur Pengguna Permainan *Game*

Berdasarkan umur pengguna permainan *game* dibagi menjadi beberapa jenis, yaitu[12]:

1. EC (*Early Childhood*), permainan *game* ini memiliki konten yang mungkin cocok untuk anak usia 3 ke atas. Dan tidak memiliki kontent yang dianggap tidak pantas dilihat oleh anak dengan usia tersebut.



Gambar 2.8 EC (*Early Childhood*)[12].

2. E (*Everyone*), permainan *game* ini memiliki konten yang mungkin cocok untuk anak dengan usia 6 atau lebih tua. Judul dalam kategori ini mungkin berisi kartun, fantasi atau kekerasan ringan atau menggunakan bahasa ringan (tidak kasar).



Gambar 2.9 E (*Everyone*)[12].

3. E10+ (*Everyone 10 and older*), permainan *game* ini memiliki konten yang mungkin cocok untuk usia 10 atau lebih tua. Judul dalam kategori ini mungkin berisi kartun, fantasi atau kekerasan ringan, bahasa ringan atau dengan tema sugestif minimal.



Gambar 2.10 E10+ (*Everyone 10 and older*)[12].

4. T (*Teen*), permainan *game* ini memiliki konten yang mungkin cocok untuk usia 13 atau lebih tua. Judul dalam kategori *game* ini mungkin berisi kekerasan, tema sugestif, humor kasar, minimal darah, simulasi perjudian, atau menggunakan bahasa kuat yang minim.



Gambar 2.11 T (*Teen*)[12].

5. M (*Mature*), permainan *game* ini memiliki konten yang mungkin cocok untuk orang usia 17 atau lebih tua. Judul dalam kategori ini mungkin berisi kekerasan *intens*, darah dan gores, konten seksual dan bahasa kuat.



Gambar 2.12 M (*Mature*)[12].

6. AO (*Adults Only*), permainan *game* ini memiliki konten yang hanya dan harus dimainkan oleh orang 18 tahun keatas. Judul dalam kategori ini mungkin termasuk adegan kekerasan berkepanjangan intens dan konten seksual grafis dengan kontent yang berisi gambar atau video.



Gambar 2.13 AO (*Adults Only*)[13].

7. RP (*Rating Pending*), Judul terdaftar sebagai RP (Rating Pending) telah disampaikan kepada ESRB dan sedang menunggu penilaian akhir. (Simbol ini hanya akan muncul dalam iklan sebelum merilis *game*.)



Gambar 2.14 RP (*Rating Pending*)[13].

2.1.4 *Game Design*

Game Design merupakan proses menciptakan konten dan aturan permainan. *Design game* yang baik adalah proses menciptakan tujuan dimana seorang pemain merasa termotivasi untuk mencapainya dan seorang pemain harus mengikuti aturan saat dia membuat keputusan yang berarti dalam mengejar tujuan-tujuan tersebut. Menurut Brenda Brathwaite dan Ian Schreiber dalam bukunya *Challenges For Game Designers*, merancang *game* mencakup sebagai berikut[14]:

a. *World Design*, membuat desain keseluruhan latarbelakang cerita, pengaturan, dan tema dari permainan.
b. *System Design*, membuat desain aturan dan pola matematis yang mendasari dalam sebuah permainan. Ini adalah tugas desain yang umum untuk semua *game*, karena semua permainan memiliki aturan.
c. *Content Design*, membuat desain karakter, item, teka-teki, dan misi.
d. *Game Writing*, membuat penulisan dialog, text, dan cerita dalam dunia permainan.

e. *Level Design*, menciptakan level dalam permainan berdasarkan tampilan permainan dan penempatan objek hingga tantangan dalam tampilan permainan.
f. *User Interface*, membuat desain yang terdiri dari dua hal yaitu, bagaimana pemain dapat berinteraksi dengan permainan, dan bagaimana pemain menerima informasi dan mengambil hasil dari permainan.

Para pengembang *game* komersial biasanya mempersiapkan dokumen-dokumen yang berisi *design game* yang sangat panjang sebelum memulai pembuatan *game*. Setelah itu dari *design game* yang telah dibuat kemudian dapat diketahui semua elemen-elemen berbeda yang dibutuhkan dalam pembuatan *game*, misalnya karakter user, karakter musuh, animasi serangan dan sebagainya. Membuat *game* akan membutuhkan gambar dari tiap elemen-elemen yang ada, *background image*, dan lagu. Semua hal tersebut dapat dikatakan sebagai *resource game*. Hal-hal yang penting dalam *game* adalah sebagai berikut [15]:

1. *Sprites*

   Sprites adalah kumpulan gambar-gambar yang digunakan untuk icon, karakter ataupun benda bergerak dalam *game*. *Sprites* dapat dibuat dari gambar-gambar yang sudah dibuat dalam sebuah *art packages*, download dari internet maupun dibuat secara manual dengan menggunakan perangkat lunak seperti *Adobe photoshop*.
2. *Object*

   Sprites tidak dapat melakukan sesuatu dengan sendirinya. *Sprites* hanya menyimpan gambar-gambar dari elemen yang berbeda dalam sebuah fame. *Object* adalah bagian dari *game* yang mengontrol bagaimana elemen-elemen tersebut bergerak dan berinteraksi satu sama lain. Contohnya adalah karakter dalam *game*.
3. *Events* dan *Actions*

   Events adalah hal-hal penting yang terjadi didalam sebuah *game*, seperti ketika *objects* saling bertubrukan maupun ketika user menekan tombol pada *keyboard*. *Actions* adalah hal-hal yang terjadi sebagai respon dari

sebuah *event*, seperti perubahan arah pergerakan *objects*, perubahan *score*, atau memainkan sebuah lagu.

4. *Backgrounds*

   Backgrounds adalah gambar-gambar yang digunakan sebagai gambar latar belakang *game* dan merupakan jenis lain dari *resource*, seperti *sprites* dan *objects*. Pada umumnya ukuran *backgrounds* sama dengan resolusi *game*.

5. *Sounds*

   Sounds juga termasuk *resource game*, jenis *sounds* dalam *game* ada dua, yaitu *music* dan *sound effect*. *Sounds* dapat mempengaruhi atmosfer dalam *game*. *Masters* berupa lagu yang terus mengalun selama *game* dimainkan, namun lagu yang dimainkan pada umumnya bergantung pada tempat atau kondisi *game*. *Sound effect* adalah suara yang hanya terdengar ketika terjadi suatu hal tertentu, misalkan penekanan tombol, karakter terkena pukulan dan sebagainya. Kegunaan dari *sound effect* adalah untuk membantu user mengetahui *action* yang dilakukan maupun hal yang sedang terjadi.

6. Status

   Status adalah sekumpulan informasi yang berguna untuk orang yang memainkan sebuah *game* untuk mengetahui kondisi permainan saat itu. Bergantung dari jenis *game*-nya, informasi yang disampaikan melalui status dapat bermacam-macam. Berikut beberapa macam status pada *game* beserta penjelasannya:

   - *Score* adalah nilai yang diperoleh oleh orang yang memainkan sebuah *game*. Status ini sangat sering digunakan pada sebuah game untuk mengetahui kemampuan yang memainkannya.
   - *High Score* adalah nilai tertinggi yang pernah dicapai orang yang memainkan sebuah *game*. Status ini biasanya terdapat pada *game* yang dapat diselesaikan dalam waktu yang cukup singkat, sehingga membuat orang yang memainkannya tertarik untuk mengulangi

permainan dari awal, dengan harapan mendapat *high score* yang lebih tinggi.

- HP (Hit Point) adalah status yang menunjukkan jumlah maksimal *damage* yang dapat diterima oleh karakter dalam *game*. Apabila bernilai 0 maka karakter tersebut dinyatakan mati/kalah. Status ini biasanya terdapat pada *game* ber-*genre fighting*, *action* atau RPG (*Role Playing Game*).
- SP (*Skill Point*) adalah sejumlah poin yang dapat digunakan untuk menggunakan *skill*. Jumlah yang dibutuhkan bergantung dari *skill* yang digunakan. Status ini biasanya terdapat pada *game* ber-*genre* RPG (*Role Playing Game*).
- *Strength* adalah jumlah kekuatan karakter dalam *game*. Nilai status biasanya menentukan kekuatan serangan karakter. Status ini biasanya terdapat pada *game* ber-*genre fighting*, *action* atau RPG (*Role Playing Game*).
- *Vitality* adalah jumlah pertahanan karakter dalam *game*. Nilai status ini biasanya akan mempengaruhi jumlah HP dan pertahanan yang dimiliki karakter. Status ini biasanya terdapat pada *game* ber-*genre fighting*, *action* atau RPG (*Role Playing Game*).
- *Speed* adalah kecepatan karakter dalam *game*. Nilai status ini biasanya akan mempengaruhi seberapa cepat karakter akan bergerak atau mendapatkan giliran serta menentukan mudah atau tidaknya terkena serangan musuh. Status ini biasanya terdapat pada *game* ber-*genre fighting*, *action* atau RPG (*Role Playing Game*).
- *Intelligence* adalah jumlah kekuatan sihir dari karakter dalam *game*. Nilai status ini biasanya akan berpengaruh pada jumlah SP, serangan sihir dan ketahanan terhadap serangan sihir. Status ini biasanya terdapat pada *game* ber-*genre action* atau RPG (*Role Playing Game*).

- *Luck* adalah keberuntungan karakter dalam *game*. Nilai status ini biasanya akan mempengaruhi karakter dalam mendapatkan hal-hal yang sulit didapatkan dalam *game,* seperti terjadinya *critical hit*.
- Status *point* adalah sejumlah poin yang biasanya digunakan untuk menambah status karakter seperti *Strength*, *Vitality*, *Intelligence*, *Speed* dan *Luck*. Status ini biasanya terdapat pada *game* ber-*genre action* atau RPG (*Role Playing Game*).
- *Time* adalah status yang digunakan untuk berbagai macam fungsi dalam *game*, misal untuk menentukan waktu yang tersisa untuk mencapai tujuan tertentu, jumlah waktu yang telah dihabiskan seseorang untuk memainkan *game* tersebut, dan lain sebagainya. Status ini biasanya terdapat pada *game* ber-*genre fighting*, *action* atau RPG (*Role Playing Game*).
- *Experience* dalam dunia *game* adalah sejumlah poin yang biasanya digunakan untuk menentukan *level* karakter dalam *game*. Status ini biasanya terdapat pada *game* ber-*genre action* atau RPG (*Role Playing Game*).

2.1.5 Metode Animasi

Animasi merupakan suatu teknik menampilkan gambar berurut sedemikian rupa sehingga penonton merasakan adanya ilusi gerakan (motion) pada gambar yang ditampilkan[16]. Animasi dapat dibuat dengan tiga teknik berbeda, yaitu *Image*, *Xoring*, dan *Make*, dan dalam pergerakannya dapat bertipe *object sprite* atau *object frame*, dan juga bisa dibedakan atas metode animasi yang digunakan antara animasi *frame*, *bibliting* dan *realtime*. Walaupun terbagi atas berbagai definisi berbeda, tapi dalam prakteknya teori-teori tersebut dapat digabungkan atau saling berhubungan sehingga tidak murni dipakai sendiri. Macam-macam animasi yang digunakan dalam membuat sebuah *game* akan dijelaskan sebagai berikut[29]:

a. Animasi dengan Teknik *Image*

Animasi dengan teknik ini adalah menyimpan *image* sebagai sebuah *sprite* dalam memori yang kemudian akan ditampilkan di *background*-nya. Dalam teknik ini animasi yang disimpan harus berlatar belakang sesuai *background*-nya. Animasi dengan teknik ini biasanya sulit dalam pembuatan gambarnya, sebab harus banyak dan melakukan penyamaan dan posisi. Akan tetapi teknik ini mudah dalam hal memainkan animasinya.

b. Animasi dengan Teknik *Xoring*

Teknik ini adalah teknik animasi yang mudah dan sederhana, sebab selain gambarnya satu *sprite*, cara menampilkannya juga jauh lebih mudah dibanding dengan teknik sebelumnya. Pembuatan gambarnya sangat mudah, sebab yang dibuat adalah *sprite*-nya saja dan tidak perlu menyamakan dengan *background*-nya. Kelemahan dari teknik ini adalah memiliki efek buruk yaitu tembus pandang dan mengganti warna *sprite*, maka hal ini tidak baik digunakan dalam animasi yang *background*-nya bergambar.

c. Animasi dengan Teknik *Make*

Animasi dengan teknik ini biasanya digunakan untuk animasi umum, tapi biasanya digunakan untuk proses pembuatan animasi. Animasi dengan teknik ini memiliki *sprite* yang terus menerus di generate oleh program, kemudian ditampilkan dengan perhitungan tertentu. Animasi ini biasanya dilakukan oleh 3D *modelling* dan *shading* oleh *software* seperti AutoCad, 3D Studio, Presidio 3D Workshop dan lain-lain. Operasi diatas haruslah di generate secara langsung dengan perhitungan sehingga saat pembuatan hampir bersamaan dengan saat menampilkannya.

d. Animasi dengan Tipe *Object Sprite*

Animasi ini menggunakan *sprite* sebagai pemeran utama sedangkan *object* lainnya hanya *background* diam. Prosesnya adalah mebuat gambar *sprite* dengan latar belakang warna hitam, lalu dibuat juga

sprite yang sama tetapi berwarna hitam dan latar belakangnya adalah warna tertinggi, kemudian ditempatkan dengan pertama-tama menyimpan *background* yang akan ditimpa oleh *sprite* dan ditempatkan *sprite* dengan Xor dan Ditimpa dengan warna tertinggi secara Xor.

e. Animasi dengan Tipe *Object Frame*

Animasi ini menitik beratkan animasi yang dimainkan hanya pada *sprite object*-nya saja, akan tetapi seluruh *background*-nya juga seolah-olah ikut digerakkan.

f. Metode Animasi *Frame*

Metode ini adalah metode animasi yang mendukung tipe *object frame*. Karena metode animasi dengan metode *fullscreen*, maka *frame* yang tampil haruslah disiapkan terlebih dahulu dalam bebera page sebelumnya. Karena hal tersebut maka pengambilan gambarnya haruslah sangat cepat, sehingga tidak menjadikan animasi lamban dan tersendat. Animasi *frame* ini haruslah menampilkan gambar *fullscreen* yang bergerak, agar efek tersendat dari pergantian *frame* tidak menyolok.

g. Metode Animasi *BitBlt*

Metode animasi ini biasanya disebut *sprite animation*, *array animation*, *blocked animation*, *partial screen animation*, *snapshot animation* atau *arcade animation*. Prinsip dari metode ini adalah menyimpan *image* dan memainkan animasinya dalam bentuk satu atau bebera *sprite* kecil. BitBlt ini digunakan untuk membuat program permainan dan program berbasis grafis terutama dalam membuat animasi sederhana.

h. Metode Animasi *Real-Time*

Dalam metode ini biasanya semua animasi yang sedang tampil atau yang akan dibuat dilakukan bersama sehingga tidak perlu disiapkan terlebih dahulu. Karena animasi ini lambat dan tersendat maka animasi dengan metode ini akan bagus jika pergerakan yang akan

dilakukan adalah tidak diketahui sebelumnya dan tiba-tiba muncul. Metode ini tidak disarankan untuk animasi biasa-biasa saja, akan tetapi sebaiknya digunakan untuk keperluan khusus seperti *rotating* dan *tweening*.

2.1.6 Transformasi 2 Dimensi

Transformasi 2 dimensi merupakan adalah pemodelan dari objek 2 dimensi yang melakukan berbagai operasi transformasi geometri, dan pada dasarnya transformasi ini berfungsi memindahkan objek tanpa merusak bentuk. Macam-macam transformasi geometri adalah sebagai berikut[17] :

1. Translasi

   Transformasi translasi merupakan suatu operasi yang menyebabkan perpindahan objek 2D dari satu tempat ke tempat yang lain. Perubahan ini berlaku dalam arah yang sejajar dengan sumbu X dan sumbu Y. Translasi dilakukan dengan penambahan translasi pada suatu titik koordinat dengan translation vector, yaitu (dx,dy), dimana dx adalah translasi menurut sumbu x dan dy adalah translasi menurut sumbu y. Koorinat baru titik yang ditranslasi dapat diperoleh dengan menggunakan rumus :

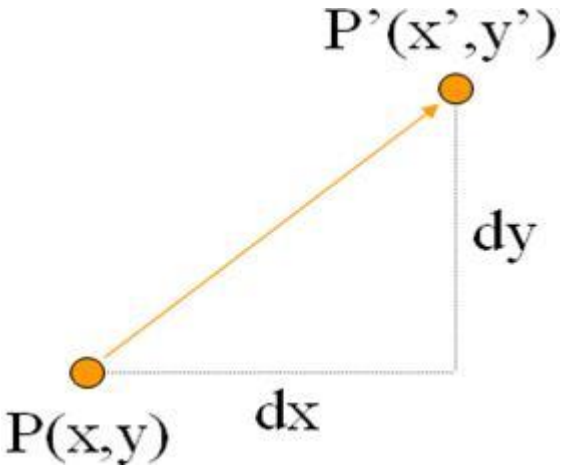
   (x,y) = titik asal sebelum translasi

   (x`,y`) = titik baru hasil translasi

   x`= x + dx

   y`= y + dy

   berikut ini adalah translasi atau perpindahan objek dari titik P ke titik P` secara linier :



Gambar 2.15 Translasi[17].

2. *Scalling*

Scalling atau disebut penskalaan adalah suatu operasi yang membuat suatu objek berubah ukurannya baik menjadi mengecil ataupun membesar secara seragam atau tidak seragam tergantung pada faktor penskalaan (scalling factor) yaitu (mx,my) yang diberikan. mx adalah faktor penskalaan menurut sumbu x dan my faktor penskalaan menurut sumbu y. Koordinat baru diperoleh dengan menggunakan rumus :

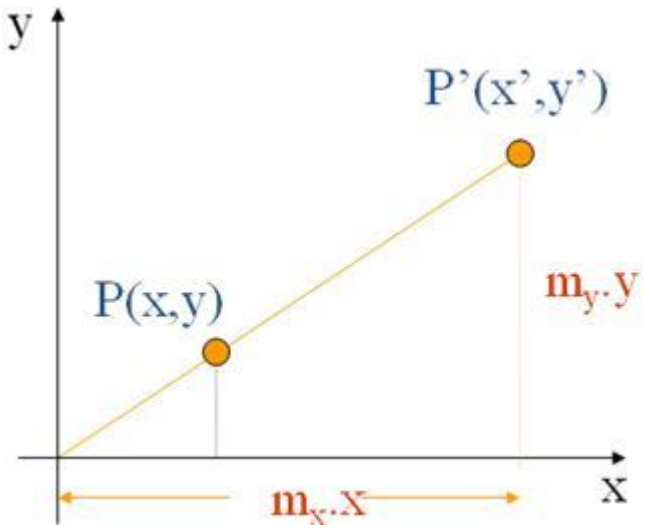
(x,y) = titik asal sebelum diskala

(x`,y`) = titik setelah diskala

x`= x * mx

y`= y * my

berikut ini adalah *scalling* atau perubahan ukuran objek dari titik P ke titik P` :



Gambar 2.16 *Scalling*[17].

3. Rotasi

Rotasi adalah suatu operasi yang menyebabkan objek bergerak berputar pada titik pusat atau pada sumbu putar yang dipilih berdasarkan sudut putaran tertentu. Untuk melakukan rotasi diperlukan sudut rotasi θ dan pivot point (xp,yp) dimana objek akan dirotasi. Putaran biasa dilakukan pada satu titik terhadap sesuatu sumbu tertentu misalnya sumbu x, sumbu y atau garis tertentu yang sejajar dengan sembarang sumbu tersebut. Titik acuan putaran dapat sembarang baik di titik pusat atau pada titik yang lain. Aturan dalam geometri, jika putaran dilakukan searah jarum jam, maka nilai sudutnya adalah negatif. Sebaliknya, jika dilakukan berlawanan arah dengan arah jarum jam nilai sudutnya adalah positif. Rumus transformasi untuk rotasi suatu titik(x,y) dengan sudut rotasi θ sebagai berikut :

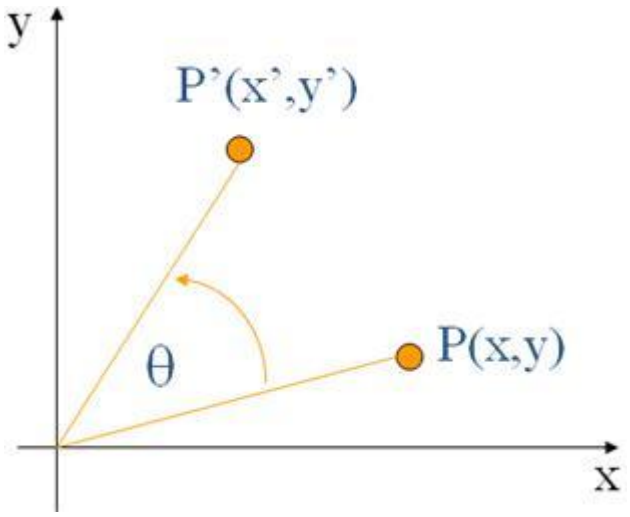
(x,y) = titik asal sebelum diskala

(x`,y`) = titik setelah diskala

x`= x cos (θ) – y sin(θ)

y`= x sin(θ) + y cos(θ)

Berikut rotasi atau perpindahan obyek dari titik P ke titik P', yang berupa pemindahan berputar sebesar sudut:



Gambar 2.17 Rotasi[17].

2.2 *Artificial Intelligence* (AI)

Kecerdasan buatan sering disebut juga dengan AI (*Artificial Intelligence*). AI merupakan salah satu bagian ilmu komputer yang mempelajari tentang bagaimana caranya agar komputer dapat melakukan pekerjaan seperti yang dapat dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kenidupan umat manusia. Sekarang komputer tidak hanya digunakan sebagai alat hitung, tetapi lebih dari itu, komputer diharapkan untuk dapat dimanfaatkan untuk mengerjakan segala sesuatu yang bias dikerjakan oleh manusia.

Agar komputer bisa bertindak seperti apa yang dapat dilakukan manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu pada *artificial intelligence* (AI), akan mencoba untuk memberikan beberapa metoda dengan membekali komputer oleh *artificial intelligence* (AI) agar komputer dapat menjadi pintar. Kecerdasan buatan dapat ditinjau dari berbagai sudut pandang, antara lain[18] :

1) Sudut Pandang Kecerdasan.

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

2) Sudut Pandang Penelitian.

   Kecerdasan buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu dengan baik yang dilakukan oleh manusia. Topik yang sering dibahas oleh para peneliti meliputi *mudane task* (*vision and speech*, *translation*, *robot control*), *formal task* (*games*, matematika (*geometri*, kalkulus integral), *expert task* (analisis finansial, analisis medikal, analisis ilmu pengetahuan).

3) Sudut Pandang bisnis.

   Kecerdasan buatan adalah kumpulan peralatan *tools* yang sangat *powerfull* dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4) Sundut Pandang Pemrograman.

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

Secara umum, untuk membangun suatu sistem yang mampu menyelesaikan masalah, perlu dipertimbangkan 4 hal [18]:

1. Mendefinisikan masalah dengan tepat. Pendefinisian ini mencakup spesifikasi yang tepat mengenai keadaan awal dan solusi yang diharapkan.
2. Menganalisis masalah tersebut serta mencari beberapa teknik penyelesaian masalah yang sesuai.
3. Merepresentasikan pengetahuan yang perlu untuk menyelesaikan masalah tersebut.

Memilih teknik penyelesaian masalah yang terbaik.

2.2.1 Teknik Penyelesaian Masalah AI (*Artificial Intelligence*)

Terdapat empat teknik dasar penyelesaian masalah yang terdapat pada bidang *artificial intelegence* diantaranya adalah[19]:

A. *Searching*

Pada teknik searching atau pencarian ini terdiri dari beberapa langkah untuk merealisasikannya. Langkah pertama adalah mendefinisikan ruang masalah untuk suatu masalah yang dihadapi. Langkah kedua adalah mendefinisikan aturan produksi yang digunakan untuk mengubah suatu keadaan ke keadaan lainnya. Langkag terakhir adalah memilih metode pencarian yang tepat sehingga dapat menemukan solusi terbaik.

B. *Reasoning*

Teknik *reasoning* atau penalaran merupakan teknik penyelesaian masalah dengan cara merepresentasikan masalah ke dalam basis pengetahuan menggunakan *logic* atau bahasa formal (bahasa yang dipahami komputer). Teknik ini melakukan proses penalaran berdasarkan basis pengetahuannya untuk menemukan solusi.

C. *Planning*

Planning adalah suatu metode penyelesaian masalah dengan cara memecah masalah ke dalam sub-sub masalah yang lebih kecil, menyelesaikan sub-sub masalah satu demi satu, kemudian menggabungkan solusi-solusi dari sub-sub masalah tersebut menjadi sebuah solusi lengkap dengan tetap mengingat dan menangani interaksi yang terdapat pada sub-sub masalah tersebut.

D. *Learning*

Pada ketiga teknik sebelumnya, seseorang harus mengetahui aturan yang berlaku untuk system yang akan dibangunnya. Tetapi, pada masalah tertentu terkadang aturan tidak bisa didefinisikan secara benar ataupun lengkan. Hal tersebut mungkin dikarenakan data-data yang didapat tidak lengkap. Melalui teknik yang disebut *learning* ini, secara otomatis aturan yang diharapkan bisa berlaku umum untuk data-data yang belum pasti diketahui dapat ditemukan.

2.2.2 Algoritma Pencarian (*Searching*)

Metode pencarian disebut penting dalam menyelesaikan permasalahan karena setiap keadaan menggambarkan langkah-langkah untuk menyelesaikan permasalahan. Dalam sebuah permainan metode pencarian akan menentukan langkah apa yang harus dilakukan, dimana setiap langkah menggambarkan kemungkinan posisi dalam sebuah rangkaian deduktif.

Permasalahan pencarian dapat diselesaikan dengan beberapa metode pencarian sebagai berikut [20]:

1. Pencarian Buta (*Blind Search*)

   Blind search atau disebut juga pencarian buta merupakan metode pencarian sederhana yang berusaha mencari kemungkinan penyelesaian, dan pencarian ini tidak memiliki informasi awal. Ciri-ciri *blind search* adalah sebagai berikut[20]:

   - Membangkitkan simpul berdasarkan urutan.
   - Kalau ada solusi, solusi akan langsung ditemukan.
   - Hanya memiliki informasi tentang *node* yang telah dibuka (*node* selanjutnya tidak diketahui).

   Blind search pun di bagi menjadi 3 seperti, BFS (*Breadth-First Search*), DFS (*Depth-First Search*), UCS (*Uniform Cost Search*).

2. Pencarian Terbimbing (*Heuristic Search*)

   Heuristic search atau disebut juga pencarian terbimbing merupakan metode pencarian yang memperhatikan nilai *heuristic* (nilai perkiraan), dan pencarian ini akan dapat mencari jarak yang terpendek. *Heuristic* memperkirakan jarak ke *Goal* (tujuan) yang disebut dengan fungsi *heuristic*. Ciri-ciri *heuristic search* adalah sebagai berikut[20]:

   - Mengesampingkan usaha yang dapat memboroskan waktu.
   - Menggunakan fungsi *heuristic* untuk mengevaluasi keadaan dari masalah dan mendapatkan solusi yang diinginkan.
   - Kemungkinan dapat mengorbankan kelengkapan (*completeness*).

Heuristic search pun mempunyai contoh seperti, *Museum Procedure*, *Branch and Bound, Dynamic Programming, Best First Search,Greedy Search*, A* (A-*Star*) *Search*, dan *Hill Climbing Search*.

2.2.3 Algoritma A* (A-*Star*)

Algoritma A* (A-*Star*) adalah algoritma pencarian yang merupakan pengembangan dari algoritma *Best First Search* (BFS). Seperti halnya pada BFS, untuk menemukan solusi, A* juga di bimbing oleh fungsi heuristik, yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Heuristik merupakan penilai yang memberi harga pada tiap *verteks* yang memandu A* mendapatkan solusi yang diinginkan[18].

Dengan fungsi heuristik Algoritma ini membangkitkan *verteks* yang paling mendekati solusi. *Verteks* ini kemudian disimpan suksesornya ke dalam *list* sesuai dengan urutan yang paling mendekati solusi terbaik. Kemudian, *verteks* pertama pada *list* diambil, dibangkitkan suksesornya dan kemudian suksesor ini disimpan ke dalam *list* sesuai dengan urutan yang terbaik untuk solusi. *List verteks* ini disebut dengan *verteks* terbuka (*open node*)[21].

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.

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 :

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

Gambar 2.18 Fungsi *Heuristic*[22].

dengan:

f(n) = fungsi evaluasi

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

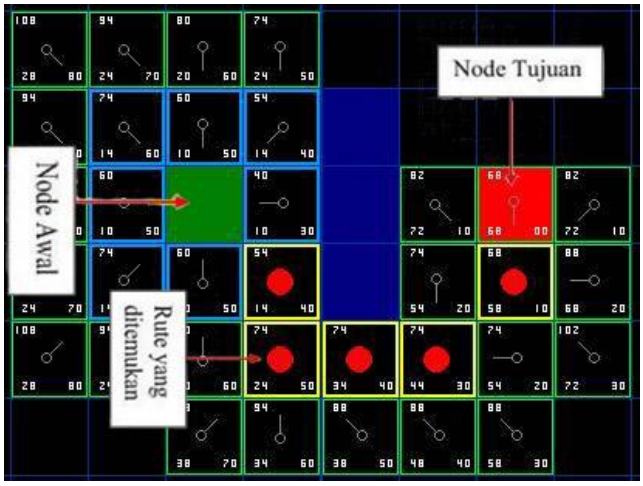
h(*n*) = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

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[22].

Metode A* dapat melakukan *backtracking* jika jalur yang ditempuh ternyata salah. Metode A* dapat melakukannya karena menyimpan jejak yang mungkin sebagai jalur yang optimal. Sebagai contohnya, jika kita sedang menuju suatu kota dan sampai pada persimpangan jalan, dan memutuskan untuk belok kiri daripada ke kanan, dan ternyata jika jalan yang dipilih ternyata salah, kita akan kembali ke persimpangan dan mengambil jalan satunya. Itulah yang dilakukan metode A* ini[22].

2.2.4 Fungsi *Heuristic*

A* sebagai 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* yang dipakai untuk pencarian rute mengisi nilai h pada algoritma A*. Ada beberapa fungsi *heuristic* umum yang bisa dipakai untuk algoritma BFS dan 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[21].



Gambar 2.19 *Rute* dengan langkah diagonal tidak diperbolehkan[21].

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

$$h(n) = (abs(n.x - goal.x) + abs(n.y - goal.y))$$

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

2.2.5 *Pathfinding*

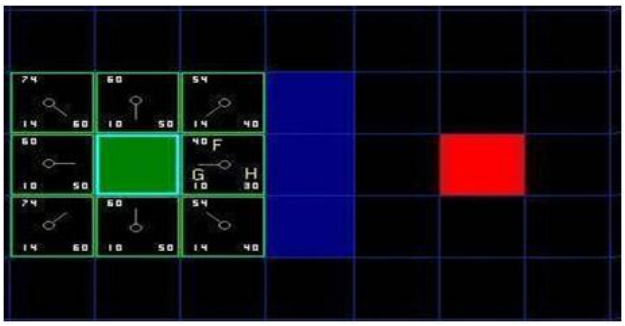
Pathfinding (pencarian jalan/*rute*) adalah salah satu bidang penerapan yang sering ditangani oleh kecerdasan buatan khususnya dengan menggunakan algoritma pencarian. Penerapan yang dapat dilakukan dengan *pathfinding* antara lain adalah pencarian *rute* dalam suatu game dan pencarian jalan/*rute* pada suatu peta. Algoritma pencarian yang dipakai harus dapat mengenali jalan dan elemen peta yang tidak dapat dilewati.

Bagaimana *Pathfinding* ini bekerja, pertama-tama peta permainan harus diolah atau diproses terlebih dahulu sebelum algoritma A * dapat bekerja. Hal ini akan melibatkan pemutusan peta ke titik yang berbeda atau lokasi, yang disebut *node*. *Node* ini digunakan untuk mencatat kemajuan pencarian. Selain memegang peta setiap *node* memiliki tiga atribut lainnya yaitu fungsi , tujuan dan heuristik yang umum dikenal sebagai f, g,dan h. Nilai yang berbeda dapat diberikan ke jalur

antara *node*, biasanya nilai-nilai ini akan mewakili jarak antara *node*. Atribut g, h, dan f didefinisikan sebagai berikut:

1. g adalah biaya yang didapatkan dari *node* awal ke *node* saat ini yaitu jumlah dari semua nilai di jalan antara awal dan *node* saat ini.
2. h singkatan heuristik yang merupakan perkiraan biaya dari *node* saat ini ke tujuan *node* (biasanya jarak garis lurus dari *node* ini ke tujuan).
3. f adalah jumlah dari g dan h dan merupakan estimasi terbaik dari biaya jalan akan melalui *node* saat ini. Intinya semakin rendah nilai f yang akan lebih efisien.

Untuk lebih jelasnya ilustrasi *pathfinding* yang digunakan musuh untuk mengejar pemain menggunakan Algoritma A* dapat dilihat pada gambar dibawah ini:



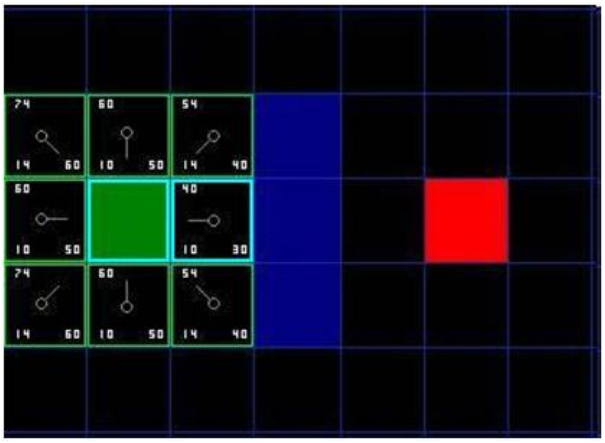
Gambar 2.20 *Pathfinding* A*[21].

Pada gambar diatas kotak berwarna hijau adalah node awal yang di ilustrasikan sebagai posisi awal musuh sedangkan kotak berwarna merah adalah tujuan yang merupakan posisi pemain. Tiga buah kotak biru merupakan penghalang yang tidak dapat dilewati oleh musuh. Angka-angka dalam kotak yang berada pada kiri atas adalah nilai f dan yang berada di kanan bawah adalah nilai h. gambar lingkaran ditengah kotak menunjukkan *parent* kotak tersebut. Untuk pergerakan kearah kanan vertikal *cost*nya adalah sepuluh, sedangkan pergerakan diagonal kanan memakan *cost* sebesar empat belas. Angka pada kanan bawah adalah h(n) yaitu harga estimasi dihitung dari posisi akhir. Simpul di sebelah kanan simpul awal memiliki h(n) = 30. Perhatikan pula bahwa jarak antara simpul tersebut dengan simpul tujuan adalah 3 kotak horizontal. Sehingga harganya

adalah 3 kali harga pergerakan horizontal, yaitu 3*10 = 30. Sedangkan h(n) di simpul sebelah kanan bawah adalah 40. Perhatikan bahwa jarak antara simpul ini dengan simpul tujuan adalah 3 kotak horizontal dan 1 kotak vertikal. Sehingga harga h(n)-nya adalah 3*10 + 1*10 = 40. Nilai yang tertera di bagian kiri-atas adalah f(n) = g(n) + h(n). Panah pada bagian tengah dari kotak menunjuk pada simpul yang menjadi parent dari simpul tersebut. Kedelapan simpul di atas kemudian dimasukkan ke list terbuka.

Selanjutnya kita akan memilih simpul dengan nilai f(n) terkecil dari list terbuka. Terhadap simpul yang dipilih kita lakukan :

1. Mengeluarkan simpul tersebut dari list terbuka dan memasukkan ke list tertutup.
2. Cek semua simpul yang terhubung langsung dengan simpul yang dipilih. Masukkan ke dalam list terbuka apabila simpul yang baru belum ada pada list terbuka serta men-set parent dari simpul baru tersebut ke simpul yang dipilih.
3. Apabila simpul yang dicek sudah terdapat pada list terbuka, cek nilai g(n) nya. Apabila nilai g(simpul dipilih) + harga untuk bergerak ke simpul dipilih < g(simpul dicek), maka ubah parent dari simpul yang dicek ke simpul yang dipilih. Jika tidak, jangan lakukan apa-apa.

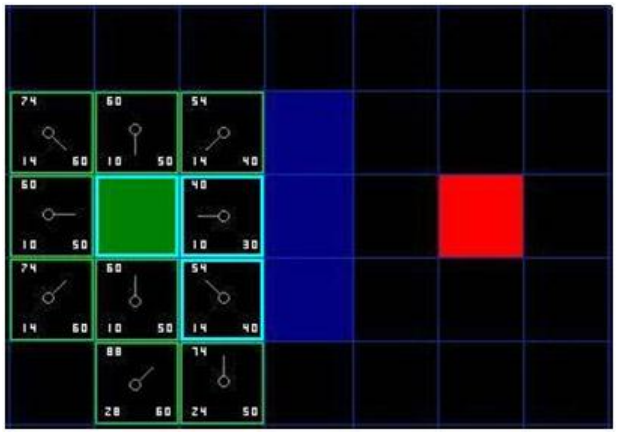


Gambar 2.21 *Pathfinding* A*[21].

Saat ini, simpul yang terpilih adalah simpul di sebelah kanan simpul awal. Kemudian seluruh simpul yang terhubung langsung dicek. Pada kasus

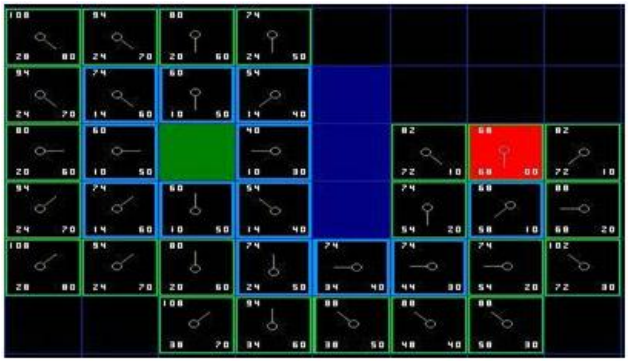
simpul di bawah simpul terpilih dicek, kita menemukan bahwa simpul tersebut sudah berada di dalam list terbuka. Karena itu, kita akan membandingkan g(n) simpul terpilih ditambah harga untuk pergerakan secara vertikal (10) dengan g(n) dari simpul yang dicek. Karena g(simpul terpilih) = 10, harga untuk bergerak vertikal = 10, dan g(simpul dicek) = 14, maka hubungan yang didapatkan adalah 10+10 > 14. Karena nilai g(simpul dicek) lebih kecil, maka kita tidak men-set parent simpul dicek menjadi simpul terpilih.

Dengan menggunakan cara yang sama, kita mendapatkan bahwa simpul terpilih yang baru adalah simpul di bawah dari simpul terpilih sebelumnya.



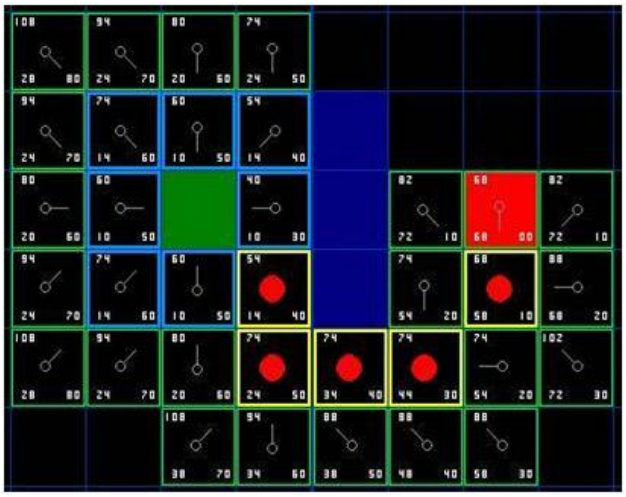
Gambar 2.22 *Pathfinding* A*[21].

Simpul yang terpilih kembali melakukan pengecekan terhadap semua simpul yang terhubung. Di sini kita definisikan bahwa simpul tidak dapat terhubung secara diagonal apabila berada di dekat penghalang. Apabila kita mengulangi cara ini terus menerus, kita akan mendapatkan hasil berikut :



Gambar 2.23 *Pathfinding* A*[21].

Selanjutnya kita hanya perlu menelusuri parent dari setiap simpul dimulai dari simpul tujuan, Seperti yang ditunjukkan gambar berikut :



Gambar 2.24 *Pathfinding* A*[21].

Pada gambar 2.24 *rute* yang ditemukan adalah bujur sangkar berbingkai kuning dengan lingkaran yang berwarna merah dengan demikian musuh dapat menemukan *rute* terpendek ke tujuan yaitu mengejar pemain.

2.3 *Interaksi Manusia dan Komputer*

Menurut Ben Shneiderman dalam bukunya yang berjudul *Designing the User Interface-Strategies for Effective Human-Computer Interaction*, bahwa Interaksi Manusia dan Komputer (IMK) atau *Human-Computer Interaction* (HCI) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya[23].

Selain itu dalam bukunya Ben Shneiderman mengemukakan, terdapat 8 aturan (*Eight Golden Rules*) yang digunakan dalam perancangan antarmuka adalah sebagai berikut [23]:

1. Berusaha untuk konsisten, urutan konsisten diperlukan pada situasi yang sama. Konsistensi harus digunakan pada prompt, menu, layar bantuan, warna tampilan, kapitalisasi, huruf, dan sebagainya.

2. Memungkinkan *frequent users* menggunakan *shortcuts*, ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.
3. Memberikan umpan balik yang informatif, untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik (feedback). Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya muncul suatu suara ketika salah menekan tombol pada waktu input data atau muncul pesan kesalahannya.
4. Merancang dialog yang memberikan penutupan (keadaan akhir), urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan memberikan indikasi bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.
5. Memberikan pencegahan kesalahan dan penanganan yang sederhana, Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sederhana dan mudah dipahami untuk penanganan kesalahan.
6. Memungkinkan pembalikan aksi yang mudah, hal ini dapat mengurangi kekuatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan, sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan
7. Mendukung pusat kendali internal (*internal locus of control*), pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

Mengurangi beban ingatan jangka pendek, keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, mnemonic, dan urutan tindakan.

2.4 *Unified Modeling Language* (UML)

UML adalah bahasa pemodelan standar yang terdiri dari serangkaian diagram, dikembangkan untuk membantu sistem dan pengembang perangkat lunak menyelesaikan tugas-tugas berikut: Spesifikasi, Visualisasi, Desain arsitektur, Konstruksi, Simulasi dan Pengujian serta Dokumentasi [27].

UML pada awalnya dikembangkan dengan ide mempromosikan komunikasi dan produktivitas antara pengembang sistem berorientasi objek, tetapi kekuatan tampak jelas dari UML telah menyebabkannya untuk membuat terobosan ke dalam setiap jenis sistem dan pengembangan perangkat lunak [27].

2.4.1 Konsep Dasar UML

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar dibawah.

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

Gambar 2.25 Konsep Dasar UML[26].

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

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

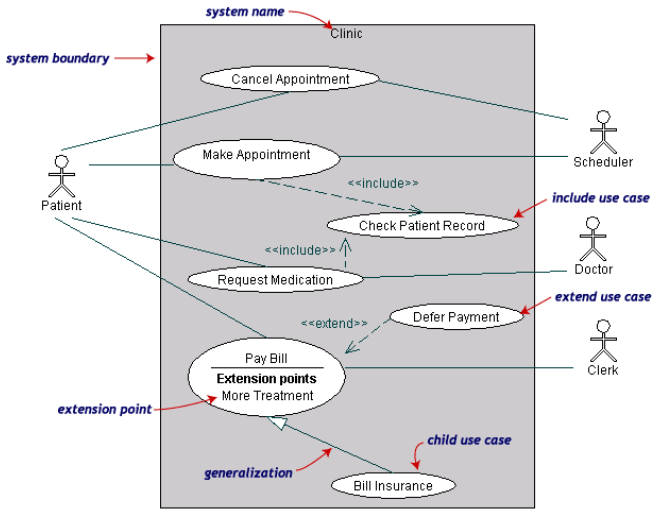
Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan

dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common.*

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar u*se case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Contoh *use case diagram:*



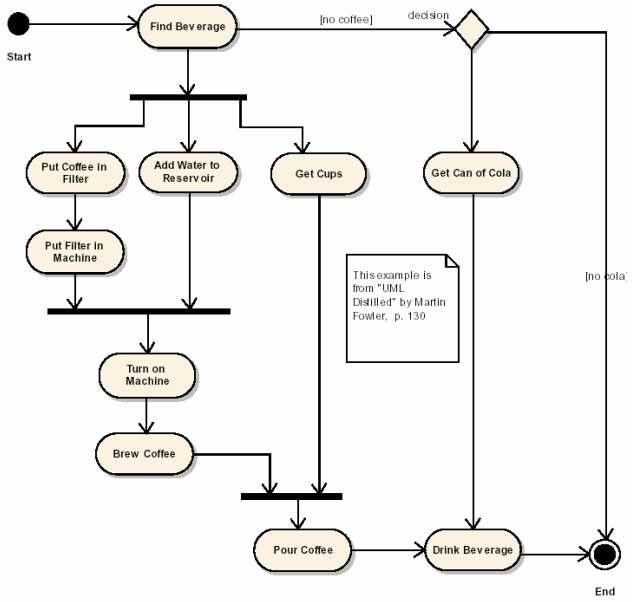
Gambar 2.26 Contoh *Use Case Diagram*[26].

2. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya *(internal processing).* Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel *(fork dan join)* digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Contoh *activity diagram* tanpa *swimlane*:



Gambar 2.27 Contoh *Activity Diagram* Tanpa *Swimlane*[26].

3. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

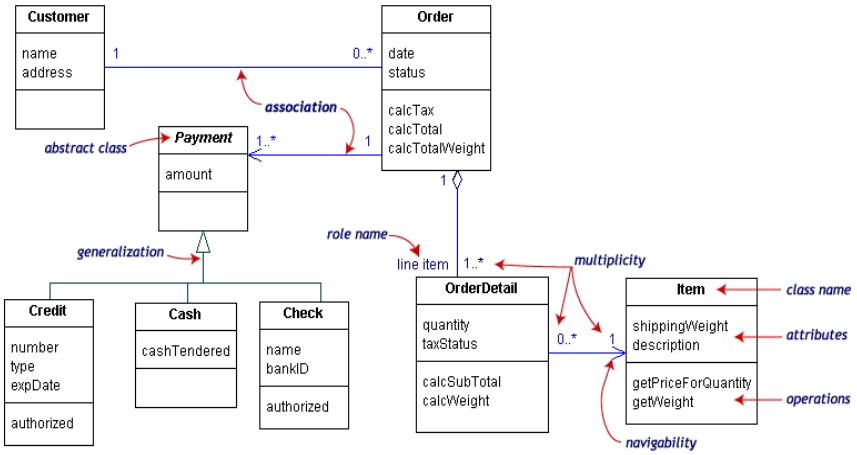
Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

4. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok, yaitu: Nama (dan stereotype), Atribut dan Metoda. Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
- *Public*, dapat dipanggil oleh siapa saja.

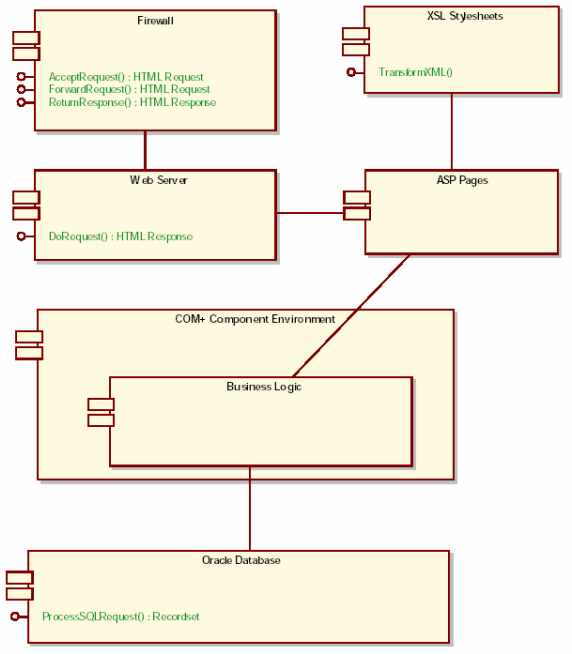


Gambar 2.28 Contoh *Class Diagram*[26].

5. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan *(dependency)* di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Contoh *component diagram:*



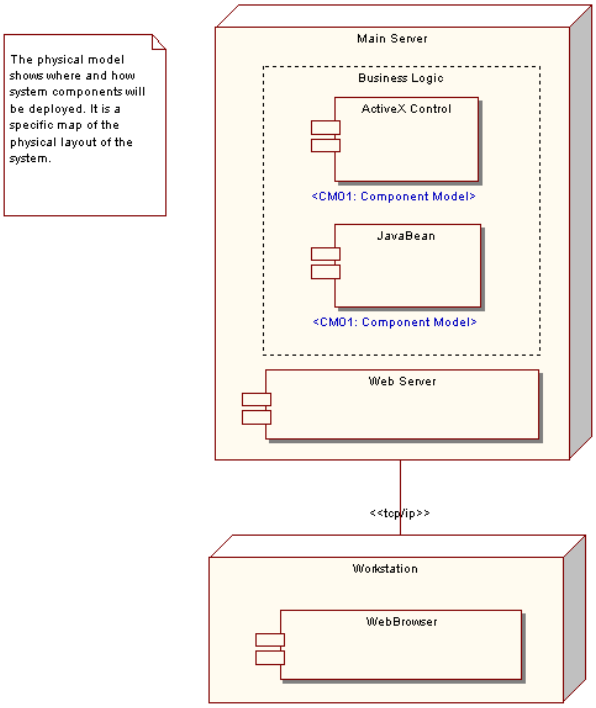
Gambar 2.29 Contoh *Component Diagram*[26].

6. *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

Contoh *deployment diagram*:



Gambar 2.30 Contoh *Deployment Diagram*[26].

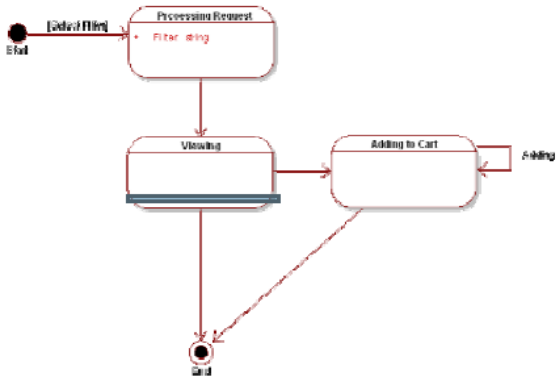
7. *State Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

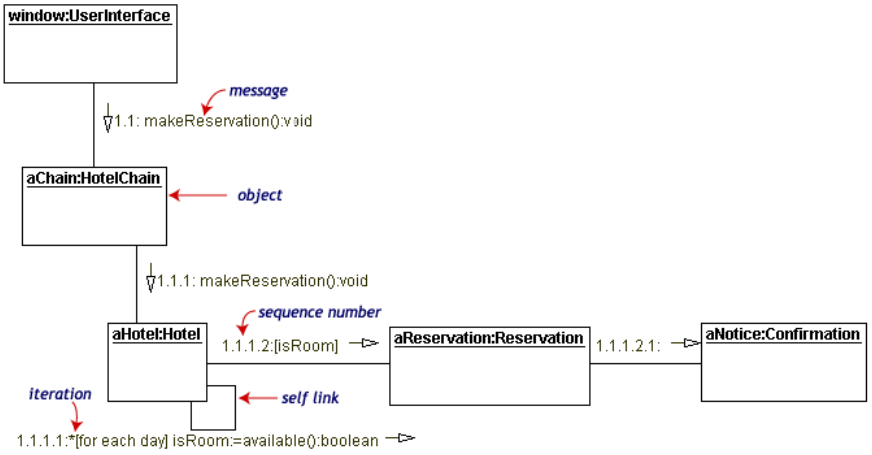
Contoh s*tatechart diagram:*



Gambar 2.31 Contoh *Statechart Diagram*[26].

8. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.



Gambar 2.32 Contoh *Collaboration Diagram*[26].

2.5 Java

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, desktop, web dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain. Java adalah bahasa pemrograman

yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source*[24].

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools* seperti, compiler, interpreter, penyusun dokumentasi, paket kelas dan sebagainya. Aplikasi dengan teknologi Java secara umum adalah aplikasi serba guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE)[24].

Terdapat dua komponen utama dari *Deployment Environment*. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas–kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada Web Browser. Hampir seluruh Web Browser komersial menyediakan *interpreter* dan *runtime environment* dari teknologi Java[24].

2.5.1 Karakteristik Java

Berdasarkan *white paper* resmi dari SUN, Java memiliki karakteristik berikut[24]:

1. Sederhana

   Bahasa pemrograman Java menggunakan sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan *pointer* yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection.*

2. Berorientasi objek *(Object Oriented)*

   Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3. Dapat didistribusi dengan mudah

   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

4. *Interpreter*

   Program Java dijalankan menggunakan interpreter yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada *platform* yang berbeda-beda.

5. *Robust*

   Java mempuyai reliabilitas yang tinggi. *Compiler* pada Java mempunyai kemampuan mendeteksi *error* secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai *runtime-Exception handling* untuk membantu mengatasi *error* pada pemrograman.

6. Aman

   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutral*

   Program Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada *platform* yang berbeda dengan *Java Virtual Machine.*

8. *Portabel*

   Source code maupun program Java dapat dengan mudah dibawa ke *platform* yang berbeda-beda tanpa harus dikompilasi ulang.

9. *Performance*

   Performance pada Java sering dikatakan kurang tinggi. Namun *performance* Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan *Inprise, Microsoft* ataupun *Symantec* yang menggunakan *Just In Time Compilers* (JIT).

10. *Multithreaded*

    Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. Dinamis

    Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu class dengan menambahkan *properties* ataupun *method* dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

2.6 Skala Pengukuran

Skala pengukuran merupakan kesepakatan yang digunakan sebagai acuan untuk menentukan panjang pendeknya interval yang ada dalam alat ukur, sehingga alat ukur tersebut bila digunakan dalam pengukuran akan menghasilkan data kuantitatif [25]. Dengan skala pengukuran ini, maka nilai variabel yang diukur 10 dengan instrumen tertentu dapat dinyatakan dalam bentuk angka, sehingga akan lebih akurat, efisien, dan komunikatif. Berbagai skala yang dapat digunakan untuk penelitian adalah [25]:

1. Skala *Likert*

Skala *Likert* digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Dengan skala *Likert*, maka variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item–item instrumen yang dapat berupa pernyataan atau pertanyaan. Jawaban setiap item instrumen yang menggunakan skala likert mempunyai gradasi dari sangat positif sampai sangat negatif, yang dapat berupa kata–kata antara lain: sangat setuju dengan skor 5, setuju dengan skor 4, ragu–ragu dengan skor 3, tidak setuju dengan skor 2, sangat tidak setuju dengan skor1.

Instrument peneletian yang menggunakan skla likert dapat dibuat dalam bentuk *checklist* ataupun pilihan ganda.

Contoh :

Berilah jawaban pertanyaan berikut sesuai dengan pernyataan anda, dengan cara memberi tanda (√) pada kolom yang tersedia.

Tabel 2.1 Tabel pertanyaan skala likert

No.	Pertanyaan	Jawaban				
		SS	ST	RG	TS	STS
1. 2.	Prosedur kerja yang baru itu akan segera diterapkan di perusahaan anda.		√			

SS = Sangat Setuju diberi skor 5
ST = Setuju diberi skor 4
RG = Ragu-ragu diberi skor 3
TS = Tidak Setuju diberi skor 2
STS = Sangat Tidak Setuju diberi skor 1

Kemudian dengan teknik pengumpulan data angket, maka instrument tersebut misalnya diberikan kepada 100 orang karyawan yang diambil secara random. Dari 100 orang pegawai setelah dilakukan analis misalnya:

25 Orang menjawab SS
40 Orang menjawab ST
5 Orang menjawab RG
20 Orang menjawab TS
10 Orang menjawab STS

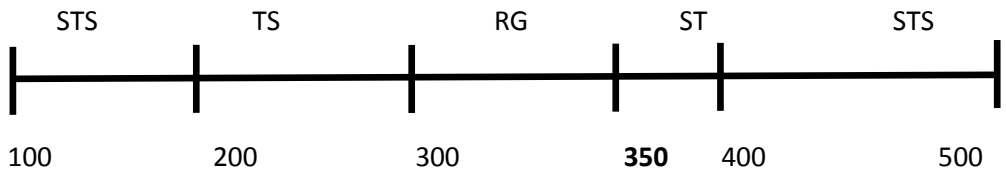
Berdasarkan data tersebut 65 orang (40+25) atau 65% karyawan menjawab setuju dan sangat setuju. Jadi kesimpulannya mayoritas karyawan setuju adanya metode kerja baru.

Data interval tersebut juga dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden. Berdasarkan skor yang telah ditetapkan dapat dihitung sebagai berikut :

Jumlah skor untuk 25 orang yang menjawab SS	= 25 x 5= 125
Jumlah skor untuk 40 orang yang menjawab ST	= 40 x 4= 160
Jumlah skor untuk 5 orang yang menjawab RG	= 5 x 3 = 15
Jumlah skor untuk 20 orang yang menjawab TS	= 20 x 2= 40
Jumlah skor untuk 10 orang yang menjawab STS	= 10 x 1= 10
Jumlah total	= 350

Jumlah skor ideal (kriterium) untuk seluruh item = 5 x 100 = 500 (seandainya semua menjawab SS). Jumlah skor yang diperoleh dari penelitian 350. Jadi berdasarkan data tu maka tingkat persetujuan terhadap metode kerja baru itu = (350 : 500) x 100% = 70% dari yang diharapkan (100%)

Secara kontinum dapat digambarkan seperti berikut:



Gambar 2.33 Interpretasi Skor Persetujuan Metode Kerja Baru[25].

Jadi berdasarkan data yang diperoleh dari 100 responden maka rata-rata 350 terletak pada daerah setuju.

2. Skala *Guttman*

Skala pengukuran dengan tipe ini, akan didapat jawaban yang tegas, yaitu ”ya–tidak”, ”benar–salah”, ”pernah–tidak pernah”,”positif–negatif” dan lain–lain. Data yang diperoleh dapat berupa data interval atau rasio 11 dikhotomi (dua alternatif). Jadi kalau pada skala *likert* terdapat 3, 4, 5, 6, 7 interval, dari kata “sangat setuju” sampai “sangat tidak setuju”, maka pada skala Guttman hanya ada dua interval yaitu ”setuju” atau ”tidak setuju”.

Penelitian menggunakan skala Guttman dilakukan bila ingin mendapatkan jawaban yang tegas terhadap suatu permasalahan yang ditanyakan.

Contoh :

1. Bagaimana pendapat anda, bila orang itu menjabat pimpinan di perusahaan ini ?
   a. Setuju
   b. Tidak setuju
2. Pernahkah pimpinan melakukan pemeriksaan di ruang kerja anda ?
   a. Tidak pernah
   b. Pernah

3. *Rating Scale*

Dari ketiga skala pengukuran seperti yang telah dikemukan, data yang diperoleh semuanya adalah data kualitatif yang kemudian dikuantitatifkan. Tetapi dengan *rating scale* data mentah yang diperoleh berupa angka kemudian ditafsirkan dalam pengertian kualitatif. Yang penting bagi penyusun instrumen dengan *rating scale* adalah harus dapat mengartikan setiap angka yang diberikan pada alternatif jawaban pada setiap item instrumen.

Contoh :

Seberapa baik data ruang kerja yang ada di Perusahaan A ?

Berilah jawaban dengan angka :

4. Bila tata ruang itu sangat baik
3. Bila tata ruang itu cukup baik
2. Bila tata ruang itu kurang baik
1. Bila tata ruang itu tidak baik

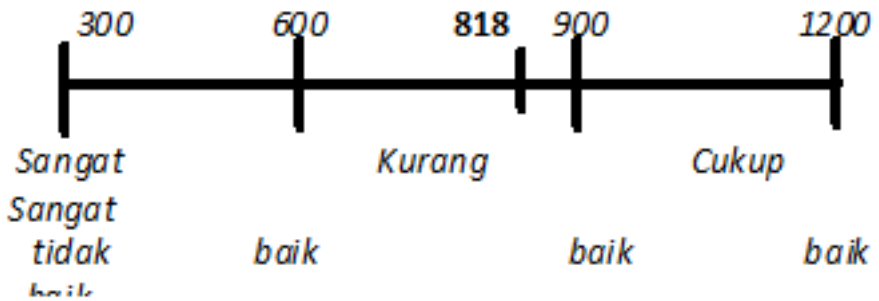
Jawaban dengan melingkari nomor jawaban yang tersedia sesuai dengan keadaan yang sebenarnya

Tabel 2.2 Tabel pertanyaan skala *rating scale*

No. Item	Pertanyaan tentang tata ruang kantor	Interval Jawaban			
1.	Penataan meja kerja sehingga arus kerja menjadi pendek	4	3	2	1
2.	Pencahayaan alam tiap ruangan	4	3	2	1
3.	Pencahayaan buatan/listrik tiap ruang sesuai dengan kebutuhan	4	3	2	1
4.	Warna lantai sehingga tidak menimbulkan pantulan cahaya yang dapat mengganggu pegawai	4	3	2	1
5.	Sirkulasi udara setiap ruangan	4	3	2	1
6.	Keserasian warna alat-alat kantor, perabot dengan ruangan	4	3	2	1
7.	Penempatan lemari arsip	4	3	2	1
8.	Penempatan ruangan pimpinan	4	3	2	1
9.	Meningkatkan keakraban sesama pegawai	4	3	2	1
10.	Kebersihan ruangan	4	3	2	1

Jumlah skor kriterium (bila setiap butir mendapat skor tertinggi) = 4 x 10 x 30 = 1200. Untuk ini skor tertinggi tiap butir = 4, jumlah butir = 10 dan jumlah responden = 30.

Jumlah skor hasil pengumpulan data = 818. Dengan demikian kualitas tata ruang kantor lembaga A menurut persepsi 30 responden itu 818 : 1200 = 68% dari kriteria yang ditetapkan. Hal ini secara kontinum dapat dibuat kategori sebagai berikut :



Gambar 2.34 Gambar Interpretasi Skor Persetujuan Tata Ruang Baru[25].

Nilai 818 termasuk dalam kategori interval “kurang baik dan cukup baik”. Tetapi lebih mendekati cukup baik.

4. *Semantic Deferential*

Skala pengukuran yang berbentuk *Semantic Deferential* dikembangkan oleh Osgood. Skala ini juga digunakan untuk mengukur sikap, hanya bentuknya tidak pilihan ganda maupun *checklist*, tetapi tersusun dalam satu garis kontinum yang jawabannya sangat positifnya terletak dibagian kanan garis, dan jawabannya sangat negatif terletak 12 di bagian kiri garis, atau sebaliknya[25].

Data yang diperoleh adalah data interval, dan biasanya skala ini digunakan untuk mengukur sikap atau karakteristik tertentu yang dipunyai oleh seseorang[25].

2.7 Pandawa

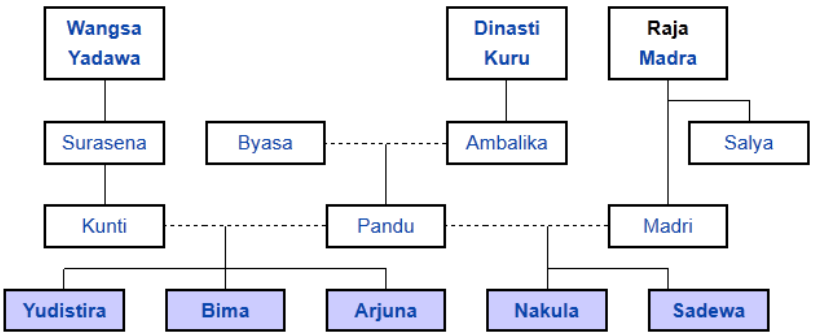
Pandawa adalah sebuah kata dari bahasa Sanskerta, yang secra harfiah berarti anak Pandu, yaitu salah satu Raja Hastinapura dalam wiracarita Mahabharata. Dengan demikian, maka Pandawa merupakan putra mahkota kerajaan tersebut. Dalam wiracarita *Mahabharata*, para Pandawa adalah protagonis sedangkan antagonis adalah para Korawa, yaitu putera Dretarastra, saudara ayah mereka (Pandu). Menurut susastra Hindu (*Mahabharata*), setiap anggota Pandawa merupakan penjelmaan (penitisan) dari Dewa tertentu, dan setiap anggota Pandawa memiliki nama lain tertentu. Misalkan nama "Werkodara" arti harfiahnya adalah "perut serigala". Kelima Pandawa menikah dengan Dropadi

yang diperebutkan dalam sebuah sayembara di Kerajaan Panchala, dan memiliki (masing-masing) seorang putera darinya[28].

Para Pandawa merupakan tokoh penting dalam bagian penting dalam wiracarita *Mahabharata*, yaitu pertempuran besar di daratan Kurukshetra antara para Pandawa dengan para Korawa serta sekutu-sekutu mereka. Kisah tersebut menjadi kisah penting dalam wiracarita Mahabharata, selain kisah Pandawa dan Korawa main dadu[28].

2.7.1 Silsilah

Para Pandawa terdiri dari lima orang pangeran, tiga di antaranya (Yudistira, Bima, dan Arjuna) merupakan putra kandung Kunti, sedangkan yang lainnya (Nakula dan Sadewa) merupakan putra kandung Madri, namun ayah mereka sama, yaitu Pandu.



Gambar 2.35 Silsilah[28].

2.7.2 Anggota

Yudistira merupakan saudara para Pandawa yang paling tua. Ia merupakan penjelmaan dari Dewa Yama dan lahir dari Kunti. Sifatnya sangat bijaksana, tidak memiliki musuh, dan hampir tak pernah berdusta seumur hidupnya. Memiliki moral yang sangat tinggi dan suka mema'afkan serta suka mengampuni musuh yang sudah menyerah. Memiliki julukan *Dhramasuta* (putera Dharma), *Ajathasatru* (yang tidak memiliki musuh), dan *Bhārata* (keturunan Maharaja Bharata). Ia menjadi seorang Maharaja dunia setelah perang akbar di Kurukshetra berakhir dan mengadakan upacara Aswamedha demi menyatukan kerajaan-kerajaan India Kuno agar berada di bawah pengaruhnya. Setelah pensiun, ia

melakukan perjalanan suci ke gunung Himalaya bersama dengan saudara-saudaranya yang lain sebagai tujuan akhir kehidupan mereka. Setelah menempuh perjalanan panjang, ia mendapatkan surge[28].

Bima merupakan putra kedua Kunti dengan Pandu. Nama *bhimā* dalam bahasa Sanskerta memiliki arti "mengerikan". Ia merupakan penjelmaan dari Dewa Bayu sehingga memiliki nama julukan *Bayusutha*. Bima sangat kuat, lengannya panjang, tubuhnya tinggi, dan berwajah paling sangar di antara saudara-saudaranya. Meskipun demikian, ia memiliki hati yang baik. Pandai memainkan senjata gada. Senjata gadanya bernama Rujakpala dan pandai memasak. Bima juga gemar makan sehingga dijuluki *Werkodara*. Kemahirannya dalam berperang sangat dibutuhkan oleh para Pandawa agar mereka mampu memperoleh kemenangan dalam pertempuran akbar di Kurukshetra. Ia memiliki seorang putera dari ras rakshasa bernama Gatotkaca, turut serta membantu ayahnya berperang, namun gugur. Akhirnya Bima memenangkan peperangan dan menyerahkan tahta kepada kakaknya, Yudistira. Menjelang akhir hidupnya, ia melakukan perjalanan suci bersama para Pandawa ke gunung Himalaya. Di sana ia meninggal dan mendapatkan surga. Dalam pewayangan Jawa, dua putranya yang lain selain Gatotkaca ialah Antareja dan Antasena[28].

Arjuna merupakan putra bungsu Kunti dengan Pandu. Namanya (dalam bahasa Sanskerta) memiliki arti "yang bersinar", "yang bercahaya". Ia merupakan penjelmaan dari Dewa Indra, Sang Dewa perang. Arjuna memiliki kemahiran dalam ilmu memanah dan dianggap sebagai ksatria terbaik oleh Drona. Kemahirannnya dalam ilmu peperangan menjadikannya sebagai tumpuan para Pandawa agar mampu memperoleh kemenangan saat pertempuran akbar di Kurukshetra. Arjuna memiliki banyak nama panggilan, seperti misalnya *Dhananjaya* (perebut kekayaan – karena ia berhasil mengumpulkan upeti saat upacara Rajasuya yang diselenggarakan Yudistira); *Kirti* (yang bermahkota indah – karena ia diberi mahkota indah oleh Dewa Indra saat berada di surga); *Partha* (putera Kunti – karena ia merupakan putra Perta alias Kunti). Dalam pertempuran di Kurukshetra, ia berhasil memperoleh kemenangan dan Yudistira diangkat menjadi raja. Setelah Yudistira mangkat, ia melakukan perjalanan suci ke gunung

Himalaya bersama para Pandawa dan melepaskan segala kehidupan duniawai. Di sana ia meninggal dalam perjalanan dan mencapai surge[28].

Nakula merupakan salah satu putera kembar pasangan Madri dan Pandu. Ia merupakan penjelmaan Dewa kembar bernama Aswin, Sang Dewa pengobatan. Saudara kembarnya bernama Sadewa, yang lebih kecil darinya, dan merupakan penjelmaan Dewa Aswin juga. Setelah kedua orangtuanya meninggal, ia bersama adiknya diasuh oleh Kunti, istri Pandu yang lain. Nakula pandai memainkan senjata pedang. Dropadi berkata bahwa Nakula merupakan pria yang paling tampan di dunia dan merupakan seorang ksatria berpedang yang tangguh. Ia giat bekerja dan senang melayani kakak-kakaknya. Dalam masa pengasingan di hutan, Nakula dan tiga Pandawa yang lainnya sempat meninggal karena minum racun, namun ia hidup kembali atas permohonan Yudistira. Dalam penyamaran di Kerajaan Matsya yang dipimpin oleh Raja Wirata, ia berperan sebagai pengasuh kuda. Menjelang akhir hidupnya, ia mengikuti pejalanan suci ke gunung Himalaya bersama kakak-kakaknya. Di sana ia meninggal dalam perjalanan dan arwahnya mencapai surge[28].

Sadewa merupakan salah satu putera kembar pasangan Madri dan Pandu. Ia merupakan penjelmaan Dewa kembar bernama Aswin, Sang Dewa pengobatan. Saudara kembarnya bernama Nakula, yang lebih besar darinya, dan merupakan penjelmaan Dewa Aswin juga. Setelah kedua orangtuanya meninggal, ia bersama kakaknya diasuh oleh Kunti, istri Pandu yang lain. Sadewa adalah orang yang sangat rajin dan bijaksana. Sadewa juga merupakan seseorang yang ahli dalam ilmu astronomi. Yudistira pernah berkata bahwa Sadewa merupakan pria yang bijaksana, setara dengan Brihaspati, guru para Dewa. Ia giat bekerja dan senang melayani kakak-kakaknya. Dalam penyamaran di Kerajaan Matsya yang dipimpin oleh Raja Wirata, ia berperan sebagai pengembala sapi. Menjelang akhir hidupnya, ia mengikuti pejalanan suci ke gunung Himalaya bersama kakak-kakaknya. Di sana ia meninggal dalam perjalanan dan arwahnya mencapai surge[28].

2.7.3 Riwayat Singkat

2.7.3.1 Masa kanak-kanak

Pandawa lima yang terdiri atas Yudistira, Arjuna, Bima, Nakula dan Sadewa, memiliki saudara yang bernama Duryodana dan 99 adiknya yang merupakan anak dari Destarasta yang tak lain adalah paman mereka, sekaligus Raja Astina menggantikan saudaranya Prabu Pandudewanata yang tak lain adalah ayah dari Pandawa lima. Sewaktu kecil para kurawa sudah mendapatkan pikiran berek dari Pamannya Suman / Sengkuni. Suatu hari Duryodana berpikir ia bersama adiknya mustahil untuk dapat meneruskan tahta dinasti Kuru apabila sepupunya masih ada. Mereka semua (Pandawa lima dan sepupu-sepupunya atau yang dikenal juga sebagai Korawa) tinggal bersama dalam suatu kerajaan yang beribukota di Astina. Akhirnya berbagai niat jahat muncul dalam benaknya untuk menyingkirkan para Pandawa beserta ibunya[28].

2.7.3.2 Usaha pertama untuk menyingkirkan Pandawa

Dretarastra yang menggantikan tahta kerajaan yang sebelumnya dipimpin oleh Prabu Pandudewanata menyerahkan kembali tahta kerajaan Astina kepada putra sulung Prabu Pandu Arjuna sebagai putra mahkota tetapi ia langsung menyesali perbuatannya yang terlalu terburu-buru sehingga ia tidak memikirkan perasaan anaknya. Hal ini menyebabkan Duryodana iri hati dengan Arjuna, ia mencoba untuk membunuh para Pandawa beserta ibu mereka yang bernama Kunti. Rencana tersebut dipelopori oleh Pamannya Harya Suman / Sengkuni dengan mengajak tukang kayu kerajaan untuk membuat tempat pesta dari bahan yang mudah terbakar. Pada saat pesta, Kunthi dan para Pandawa Lima disuruh minum air yang sudah dimasuki obat tidur, dan dibakarlah lokasi pesta tersebut. Segala sesuatunya yang sudah direncanakan Duryodana dibocorkan oleh Widura yang merupakan paman dari Pandawa. Sebelum itu juga Bima juga telah diingatkan oleh seorang petapa yang datang ke dirinya bahwa akan ada bencana yang menimpannya oleh karena itu Bima pun sudah berwaspada terhadap segala kemungkinan. Untuk pertama kalinya Bima membawa ibunya Kunthi dan

keempat saudaranya lolos dalam perangkap Duryodana dan melarikan diri ke hutan rimba[28].

2.7.3.3 Para Pandawa mendapatkan Dropadi

Pandawa lima yang melarikan diri ke rimba mengetahui akan diadakan sayembara di Kerajaan Panchala dengan syarat, barang siapa yang dapat membidik sasaran dengan tepat boleh menikahkan putri Raja Panchala (Drupada) yang bernama Panchali atau Dropadi. Arjuna pun mengikuti sayembara itu dan berhasil memenangkannya, tetapi Bima dan Arjuna yang berkata kepada ibunya ketika ibunya tengah memasak, "Ibu, kami membawa sedekah yang terbaik!" Kunti, menjawab tanpa melihat, "Bagilah sama rata kepada saudaramu, Nak." Karena perkataan ibunya. Pancali pun bersuamikan lima orang[28].

2.7.3.4 Perselisihan antar keluarga

Pamannya (Dretarastra) yang mengetahui bahwa Pandawa lima ternyata belum mati pun mengundang mereka untuk kembali ke Hastinapura dan memberikan hadiah berupa tanah dari sebagian kerajaannya, yang akhirnya Pandawa lima membangun kota dari sebagian tanah yang diberikan pamannya itu hingga menjadi megah dan makmur yang diberi nama Indraprastha. Duryodana yang pernah datang ke Indraprastha iri melihat bangunan yang begitu indah, megah dan artistik itu. Setelah pulang ke Hastinapura ia langsung memanggil arsitek terkemuka untuk membangun pendapa yang tidak kalah indahnya dari pendapa di Indraprastha. Bersamaan dengan pembangunan pendapa di Hastinapura ia pun merencanakan sesuatu untuk merebut kerajaan milik Yudistira (Indraprastha) dan menjatuhkan Yudistira dan adik adiknya. Yang pada akhirnya Yudistra pun terjebak dalam rencananya Duryodana dan harus menjalani pengasingan selama 12 Tahun dan satu tahun untuk tidak dikenali, di dalam pengasingan itu Pandawa pun menyusun rencana untuk membalas dendam atas penghinaan yang telah dilakukan Duryodana dan adik adiknya, yang akhirnya memicu terjadinya perang besar antara Pandawa dan Korawa serta sekutu-sekutunya[28].

2.7.3.5 Pertempuran besar di Kurukshetra

Pertempuran besar di Kurukshetra (atau lebih dikenal dengan istilah Bharatayuddha di Indonesia) merupakan pertempuran sengit yang berlangsung selama delapan belas hari. Pihak Pandawa maupun pihak Korawa sama-sama memiliki ksatria-ksatria besar dan angkatan perang yang kuat. Pasukan kedua belah pihak hampir gugur semuanya, dan kemenangan berada di pihak Pandawa karena mereka berhasil bertahan hidup dari pertempuran sengit tersebut. Semua Korawa gugur di tangan mereka, kecuali Yuyutsu, satu-satunya Korawa yang memihak Pandawa sesaat sebelum pertempuran berlangsung[28].

2.7.3.6 Akhir Riwayat

Setelah Kresna wafat, Byasa menyarankan para Pandawa agar meninggalkan kehidupan duniawi dan hidup sebagai pertapa. Sebelum meninggalkan kerajaan, Yudistira menyerahkan tahta kepada Parikesit, cucu Arjuna. Para Pandawa beserta Dropadi melakukan perjalanan terakhir mereka di Gunung Himalaya. Sebelum sampai di puncak, satu persatu dari mereka meninggal dalam perjalanan. Hanya Yudistira yang masih bertahan hidup dan didampingi oleh seekor anjing yang setia. Sesampainya di puncak, Yudistira dijemput oleh Dewa Indra yang menaiki kereta kencana. Yudistira menolak untuk mencapai surga jika harus meninggalkan anjingnya. Karena sikap tulus yang ditunjukkan oleh Yudistira, anjing tersebut menampakkan wujud aslinya, yaitu Dewa Dharma. Dewa Dharma berkata bahwa Yudistira telah melewati ujian yang diberikan kepadanya dengan tenang dan ia berhak berada di surga[28].

Sesampainya di surga, Yudistira terkejut karena ia tidak melihat saudara-saudaranya, sebaliknya ia melihat Duryodana beserta sekutunya di surga. Dewa Indra berkata bahwa saudara-saudara Yudistira berada di neraka. Mendengar hal itu, Yudistira lebih memilih tinggal di neraka bersama saudara-saudaranya daripada tinggal di surga. Pada saat itu, pemandangan tiba-tiba berubah. Dewa Indra pun berkata bahwa hal tersebut merupakan salah satu ujian yang diberikan kepadanya, dan sebenarnya saudara Yudistira telah berada di surga. Yudistira pun mendapatkan surge[28].