BAB II
LANDASAN TEORI

II.1 *Game*

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya.Dalam *game*, ada target-target yang ingin dicapai pemainnya.Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, *play* dan budaya.Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan.

Game bertujuan untuk menghibur, *game* banyak disukai oleh anak-anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi *game* juga bisa merugikan karena apabila kita sudah kecanduan *game* kita akan lupa waktu dan akan mengganggu kegiatan atau aktivitas yang sedang kita lakukan [**4**].

II.1.1 Jenis – Jenis Game

Berdasarkan jenisnya *game* dibagi menjadi beberapa kategori diantaranya:

Berdasarkan *platfrom* atau alat yang digunakan [**4**] :

1. Arcade *game*s, yaitu video *game* dengan mesin yang memang khusus di desain untuk jenis video *game*s tertentu dan memiliki fitur seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil dan lain-lain.
2. PC *Game*s , yaitu video *game* yang dimainkan menggunakan Personal Computers.

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

Berdasarkan *genre* permainanya [**4**] :

1. *Shooting*

   Shooting merupakan salah satu *genre* dalam *game* biasanya mensimulasikan 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

2. *Fighting*

   Fighting (pertarungan) jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari game ini adalah penguasaan *skill*karakter yang di mainkan, pengenalan karakter dan timing sangatlah penting, dan berbeda seperti game Aksi pada umumnya yang umumnya hanya melawan *Artificial Intellegence* atau melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya.

3. 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 petualangan dengan teks atau sistem tunjuk dan klik.

4. 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.

5. *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 dan lain-lain.

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.

7. Olahraga

   Biasanya *game genre* ini permainannya diusahakan serealistik mungkin di buat se mirip mungkin dengan aslinya walau kadang ada yang menambah unsur fiksi dalam *game* olahraga tersebut.

8. Edukasi

   Video Game jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, baik itu untuk belajr mengenal warna, mengenal huruf dan angka, matematika, belajar bahasa asing dan lain-lain. *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 desain visual ataupun animasinya.

II.1.1.1. *Turn-based Strategy Game*

Turn-based strategy (TBS) game adalah turunan dari *game* dengan *genre* strategi dimana pemainnya bergantian dalam bermain. Hal ini sangat bertolak belakang dengan game-game lain dimana para pemainnya bermain bersama secara simultan.*Game* TBS berawal dari permainan catur kuno dimana player saling bergiliran menjalankan bidak caturnya.*Game* TBS mempunyai *gameplay* yang menawarkan pemainnya memilih satu dariberbagai macam keputusan yang disediakan pada *game*.Saat ini game TBS banyak dijumpai dalam *video game*.Berbagai jenis *gameplay* turunan *game* TBS ini pun bermunculan seperti menghabisi musuh (*annihilate*), menguasai wilayah (*dominate*), bahkan dengan misi-misi lebih spesifik lagi [**5**].

Pemain dalam game TBS tidak bergantung pada kecepatan, refleks, dan akurasi ketika bermain game lainnya karena sifat *gameplay*-nya yang saling bergiliran dalam bertindak membuat pemainnya cenderung lebih mengandalkan logika dan intuisi dalam mengambil tindakan. Game ini pertama kali muncul dalam *video game* pada tahun 1977 dan sampai sekarang masih terus dikembangkan.

II.2. *Artificial Intelligence* (AI)

*Artificial Intelligence*atau kecerdasan buatan merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan. Definisi Kecerdasan Buatan, Berbagai definisi diungkapkan oleh para ahli untuk dapat memberi gambaran mengenai kecerdasan buatan beberapa diantaranya :

1. Kecerdasan Buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas [**6**]**.**
2. Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia **[7**]**.**

Dari definisi diatas maka dapat ditarik kesimpulan bahwa kecerdasan buatan (*Artificial Intellegence*) adalah suatu ilmu yang mempelajari cara membuat komputer yang didalamnya terdapat pengetahuan – pengetahuan yang diperlukan untuk mengaplikasikannya, sehingga komputer ini dapat melakukan pekerjaan – pekerjaan yang dilakukan oleh manusia.

II.2.1. Sejarah *Artificial Intelligence*

Bibit *Artificial Intelligence* (AI) pertama kali disebar hanya dua tahun setelah General Electric menerapkan komputer yang pertama kali digunakan untuk penggunaan bisnis.Tahun itu adalah tahun 1956 dan istilah AI pertama kali dibuat oleh John McCarthy sebagai tema suatu konferensi yang dilaksanakan di Dartmouth College. Pada tahun yang sama, program komputer AI pertama yang disebut *Logic Theorist* diumumkan. Kemampuan *Logic Theorist* yang terbatas untuk berfikir (membuktikan teorema-teorema kalkulus) mendorong para ilmuwan merancang program lain yang disebut *General Problem Solver* (GPS), yang ditujukan untuk digunakan dalam memecahkan segala macam masalah. Proyek ini ternyata membuat para ilmuwan yang perama kali menyusun program ini kewalahan, dan riset AI dikalahkan oleh aplikasi-aplikasi komputer yang tidak

terlalu ambisius seperti SIM dan DSS.Namun seiring waktu, riset yang terus menerus akhirnya membuahkan hasil dan AI telah menjadi wilayah aplikasi komputer yang solid [**8**].

II.3. *Knowledge Based System*

Pengetahuan (knowledge): kemampuan untuk kegiatan yang efektif. Knowledge adalah suatu pengetahuan yang dapat digunakan untuk menyimpulkan, dan menyelesaikan suatu masalah berdasarkan data yang ada. Dalam mengorganisasikan knowledge kita harus mengerti apa yang akan kita hadapi, masalah apa yang akan timbul, bagaimana cara kita memperoleh data, bagaimana urutan-urutan yang harus ditempuh dan bagaimana cara menyelesaikan masalah tersebut, dan juga bagaimana kita menyampaikannya pada user.

Knowledge based system merupakan bagian dari *artificial intelligence* yang menyediakan fungsi pengambilan keputusan secara cerdas dengan berdasarkan berbagai macam pertimbangan. Pengetahuan yang ditanamkan pada sistem didapat dari aturan-aturan yang akan digunakan pada sistem. Tujuan utama dari penggunaan *knowledge based system* adalah sebagai pembantu dalam pembuatan keputusan**9**]. *Knowledge based system* menggunakan metoda dan teknik pada *artificial intelligence*. Komponen utama pada *knowledge based system* adalah :

1. *Knowledge base*

   Knowledge base adalah bagian yang berisi fakta-fakta yang menjelaskan problem domain dengan menggunakan aturan – aturan untuk mengekspresikan logika masalah. Teknik representasi yang paling popular adalah penggunaan RULES. Suatu Rule menyatakan apa yang harus dilakukan pada suatu situasi dan terdiri dari 2 bagian, yaitu : KONDISI (benar atau salah) serta AKSI yang dilakukan bila kondisi bernilai ***benar***.

2. *Acquisition Mechanism*

   Acquisition Mechanism adalah bagian yang menterjemahkan aturan-aturan kedalam *knowledge base*. Aturan-aturan tersebut merupakan aturan yang akan

diterapkan pada sistem dan dirubah menjadi representasi data yang dimengerti oleh sistem.

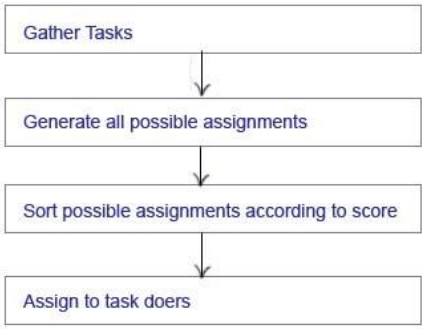
3. *Inference mechanism*

   Inference mechanism adalah bagian yang melaksanakan penalaran dengan cara menggunakan isi dari knowledge base melalui urutan tertentu. Saat konsultasi inference engine memeriksa rules pada knowledge base satu per satu, bila "rule's condition bernilai true" maka aksi tertentu akan dilaksanakan.

Knowledge based system digunakan sebagai *framework* pada sistem, dengan kata lain *knowledge based system* hanya sistem pendukung dalam pengambil keputusan berdasarkan berbagai macam pertimbangan. Dalam mencari nilai untuk dipertimbangkan dapat digunakan sistem cerdas lain.

II.3.1. Algoritma *Resource Assignment*

Algoritma *resource assignment* (RAA) adalah sebuah algoritma yang dapat menentukan sebuah nilai untuk sebuah objek yang memiliki *resource* berdasarkan nilai *resource* yang berasal dari objek lain maupun objek itu sendiri. Dengan kata lain, RAA adalah algoritma yang bergantung pada nilai *resource* yang dimiliki oleh objek-objek dalam menentukan sebuah nilai untuk sebuah objek [**9**]. Algoritma ini memiliki tiga parameter utama yaitu *objects*, *resource* dan *modifier*. Secara keseluruhan algoritma ini terbagi menjadi empat bagian yaitu *gather task, generate all possible assignments*, *sort possible assignment according to score* dan *assign to task doers* seperti yang ditunjukan pada Gambar II-1.



Gambar II-1 Bagian-Bagian Pada Algoritma *Resource Assignment* [1]

1. *Gather Task*

   Gather Task adalah proses pertama dari algoritma resource assignment yang mana pada tahap ini adalah menentukan banyaknya *objects* yang terlibat.

2. *Generate All Possible Assignments*

   Generate All Possible Assignments adalah proses penentuan nilai *modifier* yang mana nilai *modifier* ini akan digunakan untuk mengambil keputusan terhadap *objects*. Proses penentuan nilai *modifier* disini akan melibatkan *resource* yang terdapat pada *objects*. Nilai *modifier* terdapat pada setiap keputusan. Setiap keputusan tersebut akan diujikan pada setiap *objects*. Pengujian dilakukan berdasarkan aturan yang akan diterapkan pada sistem. Setiap terdapat aturan yang terpenuhi, maka nilai *modifier* untuk keputusan pada aturan tersebut akan ditambahkan.

3. *Sort Assignments According to Score*

   Sort Assignments According to Score adalah proses pengurutan setiap nilai *modifier* pada *objects*. Proses pengurutan ini bertujuan menentukan nilai *modifier* terbesar.

4. *Assign to Task Doers*

   Assign to Task Doers adalah proses untuk menjalankan keputusan dengan memperhatikan nilai *modifier* terbesar yang didapatkan dari tahap *Sort Assignments According to Score*.

Contoh sederhana dari algoritma resource assignment misalkan terdapat tiga objek yang harus menghasilkan satu keputusan dari dua keputusan yang ada yaitu A dan B. Setiap keputusan terdapat dua kondisi yang harus

dipenuhi. Nilai modifier akan ditambahkan ketika setiap kondisi pada setiap keputusan terpenuhi oleh objek tersebut. Dengan asumsi setiap objek telah diperiksa kondisinya berdasarkan aturan yang diterapkan maka gambaran untuk menentukan keputusan yang akan diambil pada contoh kasus ini ditunjukan pada Tabel II-1. Proses *Gather Task* terjadi ketika ditentukan berapa objek yang terlibat, berapa keputusan dan berapa kondisi pada setiap keputusan yang akan dijalankan serta bagaimana aturan kondisi pada setiap keputusan tersebut. Proses *Generate All Possible Assignments* terjadi ketika setiap objek diperiksa apakah sesuai dengan kondisi-kondisi yang telah ditentukan. Pada contoh kasus yang ditunjukan pada Tabel II-1, nilai Y berarti objek tersebut memenuhi sebuah kondisi sedangkan nilai N berarti objek tersebut tidak memenuhi sebuah kondisi. Jumlah kondisi yang terpenuhi adalah nilai *modifier*. Proses *Sort Assignments According to Score* terjadi ketika nilai modifier untuk keputusan ke-1 dan keputusan ke-2 diurutkan lalu menuju proses *Assign to Task Doers* dimana nilai modifier terbesar dipilih dan keputusan dengan nilai modifier terbesar dipilih untuk sebuah objek. Berikut adalah detail prosesnya :

1. Lakukan perhitungan pengambilan keputusan untuk objek ke-1
    a. Periksa kondisi ke-1 pada keputusan ke-1. Karena diasumsikan objek ke-1 memenuhi kondisi ke-1 pada keputusan ke-1 maka nilai modifier untuk keputusan ke-1 ditambah 1.

Modifier_1 = modifier_1 + 1	Modifier_1 = 0 + 1 = 1

    b. Periksa kondisi ke-2 pada keputusan ke-1. Karena diasumsikan objek ke-1 memenuhi kondisi ke-1 pada keputusan ke-1 maka nilai modifier untuk keputusan ke-2 ditambah 1.

Modifier_1 = modifier_1 + 1	Modifier_1 = 1 + 1 = 2

    c. Periksa kondisi ke-1 pada keputusan ke-2. Karena diasumsikan objek ke-1 tidak memenuhi kondisi ke-1 pada keputusan ke-2 maka nilai modifier untuk keputusan ke-2 tidak ditambah.

Modifier_2 = modifier_2	Modifier_2 = 0

d. Periksa kondisi ke-2 pada keputusan ke-2. Karena diasumsikan objek ke-1 memenuhi kondisi ke-1 pada keputusan ke-2 maka nilai modifier untuk keputusan ke-1 ditambah 1.

Modifier_2 = modifier_2 + 1	Modifier_2 = 0 + 1 = 1

e. Urutkan nilai modifier 1 dan modifier 2, lalu tentukan nilai modifier yang terbesar. Karena nilai modifier terbesar adalah nilai modifier untuk keputusan ke-1, maka keputusan yang dipilih oleh objek ke-1 adalah keputusan ke-1.

2. Lakukan perhitungan pengambilan keputusan untuk objek ke-2

a. Periksa kondisi ke-1 pada keputusan ke-1. Karena diasumsikan objek ke-2 memenuhi kondisi ke-1 pada keputusan ke-1 maka nilai modifier untuk keputusan ke-1 ditambah 1.

Modifier_1 = modifier_1 + 1	Modifier_1 = 0 + 1 = 1

b. Periksa kondisi ke-2 pada keputusan ke-1. Karena diasumsikan objek ke-2 tidak memenuhi kondisi ke-1 pada keputusan ke-1 maka nilai modifier untuk keputusan ke-2 tidak ditambah.

Modifier_1 = modifier_1	Modifier_1 = 1

c. Periksa kondisi ke-1 pada keputusan ke-2. Karena diasumsikan objek ke-2 memenuhi kondisi ke-1 pada keputusan ke-2 maka nilai modifier untuk keputusan ke-2 ditambah 1.

Modifier_2 = modifier_2 + 1	Modifier_2 = 0 + 1 = 1

d. Periksa kondisi ke-2 pada keputusan ke-2. Karena diasumsikan objek ke-2 memenuhi kondisi ke-1 pada keputusan ke-2 maka nilai modifier untuk keputusan ke-2 ditambah 1.

Modifier_2 = modifier_2 + 1	Modifier_2 = 1 + 1 = 1

e. Urutkan nilai modifier 1 dan modifier 2, lalu tentukan nilai modifier yang terbesar. Karena nilai modifier terbesar adalah nilai modifier untuk keputusan ke-2, maka keputusan yang dipilih oleh objek ke-2 adalah keputusan ke-2.

3. Lakukan perhitungan pengambilan keputusan untuk objek ke-3
   a. Periksa kondisi ke-1 pada keputusan ke-1. Karena diasumsikan objek ke-3 tidak memenuhi kondisi ke-1 pada keputusan ke-1 maka nilai modifier untuk keputusan ke-1 tidak ditambah.

Modifier_1 = modifier_1	Modifier_1 = 0

   b. Periksa kondisi ke-2 pada keputusan ke-1. Karena diasumsikan objek ke-3 tidak memenuhi kondisi ke-1 pada keputusan ke-1 maka nilai modifier untuk keputusan ke-2 tidak ditambah.

Modifier_1 = modifier_1	Modifier_1 = 0

   c. Periksa kondisi ke-1 pada keputusan ke-2. Karena diasumsikan objek ke-3 memenuhi kondisi ke-1 pada keputusan ke-2 maka nilai modifier untuk keputusan ke-2 ditambah 1.

Modifier_2 = modifier_2	Modifier_2 = 0 + 1 = 1

   d. Periksa kondisi ke-2 pada keputusan ke-2. Karena diasumsikan objek ke-3 memenuhi kondisi ke-1 pada keputusan ke-2 maka nilai modifier untuk keputusan ke-2 ditambah 1.

Modifier_2 = modifier_2 + 1	Modifier_2 = 1 + 1 = 2

   e. Urutkan nilai modifier 1 dan modifier 2, lalu tentukan nilai modifier yang terbesar. Karena nilai modifier terbesar adalah nilai modifier untuk keputusan ke-2, maka keputusan yang dipilih oleh objek ke-3 adalah keputusan ke-2.

Tabel II-1 Contoh Pengambilan Keputusan Algoritma *Resource Assignment*

Objek	Keputusan ke-1			Keputusan ke-2			Keputusan yang dipilih
	C-1.1	C-1.2	Modifier 1	C-2.1	C-2.2	Modifier 2	
1	Y	Y	2	N	Y	1	keputusan ke-1
2	Y	N	1	Y	Y	2	keputusan ke-2
3	N	N	0	Y	Y	2	keputusan ke-2

Dimana :

C-1.1 adalah kondisi ke-1 pada keputusan ke-1

C-1.2 adalah kondisi ke-2 pada keputusan ke-1

C-2.1 adalah kondisi ke-1 pada keputusan ke-2

C-2.2 adalah kondisi ke-2 pada keputusan ke-2

II.4. Struktur Data

Struktur data adalah suatu kumpulan objek-objek data yang memiliki aturan (terorganisir) yang kemudian digolongkan berdasarkan operasi-operasi untuk memanipulasinya. Kompponen yang dimuat daam struktur data ini dapat berupa objek data elementer atau objek terstruktur lainnya. Contoh objek data terstruktur adalah *array*, *record*, *list*, *stack*/tumpukan, *queue*/antrian, *tree*/pohon, set/himpunan, *file*/berkas dan lain-lain [**10**].

Atribut-atribut penting untuk suatu tipe data terstruktur meliputi:

1. Jumlah komponen

   Berdasarkan perubahan umlah komponen selama eksekusi program, maka dapat dikelompokan menjadi struktur data statis yang jumlah komponennya tidak berubah dan struktur data dinamis yang jumlah komponennya dapat berubah.

2. Tipe untuk setiap komponen

   Apabila tipe data untuk seluruh komponennya harus sama, maka disebut struktur data homogen, dan bila dimungkinkan komponennya mempunyai tipe yang berbeda-beda, maka disebut struktur data heterogen.

3. Nama-nama untuk memilih komponen

   Hampir semua struktur data menyediakan operasi untuk mengakses komponen secara individu. Pada suatu array, hal ini dilakukan dengan sebuag subscript/indeks berupa angka.

4. Jumlah maksimum komponen

   Untuk sebuah struktur data dinamis mungkin perlu ditentukan dengan jelas.

5. Pengorganisasian semua komponennya.

   Susunan yang paling umum adalah berupa barisan linier seperti pada *array* berdimensi 1, *record*, *list, stack* dan *file*. Selanjutnya ada yang dapat dikembangkan menjadi struktur non linear seperti *array* multi dimensi dan juga pohon/*tree.*

Operasi-operasi pada struktur data meliputi:

1. Operasi seleksi komponen

   Operasi seleksi komponen adalah operasi untuk mengakses komponen struktur data dan membuatnya tersedia untuk pemrosesan dengan operasi lain. Operasi ini dapat dibedakan menjadi operasi akses secara random dimana struktur data yang diakses dapat berganti-ganti dan operasi akses secara berurutan/sekuensial dimana komponen-komponen diakses satu demi satu dalam urutan atau rangkaian pemrosesan.

2. Operasi struktur data secara keseluruhan

   Operasi ini memungkinkan struktur data secara keseluruhan sebagai argumenya dan menghasilkan struktur data yang baru.

3. Penyisipan dan penghapusan komponen

   Operasi ini akan mengubah jumlah komponen dan berpengaruh pada representasi dan pengelolaan penyimpanan.

4. Pembuatan dan penghapusan struktur data.

II.5. Graph

Graph merupakan struktur data yang tersusun atas simpul-simpul atau titik-titik (Simpul) yang terhubung satu sama lain dengan garis-garis (*edge*). Secara matematis, graph dinyatakan sebagai berikut:

$$G = (V,E)$$

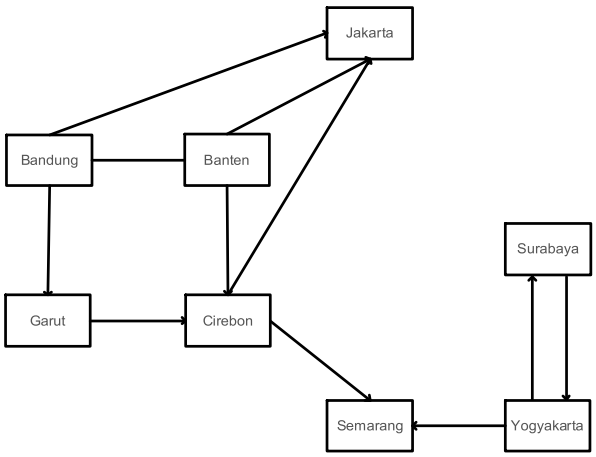
Dimana:

G = Graph

V= Simpul atau vertex, simpul

E = Sisi, busur atau edge

Terdapat banyak sekali jenis graph. Namun secara orientasi arah, graph dapat dibedakan menjadi dua, pertama graph berarah. Graph berarah merupakan graph yang mana sisinya memiliki arah. Sisi ini biasa disebut busur (*arc*). Pada graph berarah, urutan pasangan simpul sangat diperhatikan karena dapat menyatakan hal yang berbeda.

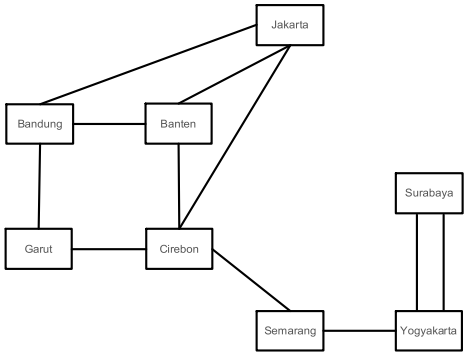


Gambar II-2 Graph Berarah

Pada Gambar II-2 dapat dilihat bahwa dua busur yang menghubungkan (Yogyakarta, Surabaya) dan (Surabaya, Yogyakarta) menyatakan hal yang berbeda. Untuk busur yang menghubungkan simpul Yogyakarta dan Surabaya, simpul Yogyakarta dinamakan sebagai simpul awal (*initial suksesor*) dan simpul Surabaya dinamakan sebagai simpul terminal. Sebaliknya, pada busur yang

menghubungkan simpul Surabaya dan Yogyakarta, simpul Surabaya merupakan simpul awal dan simpul Yogyakarta adalah simpul terminal.

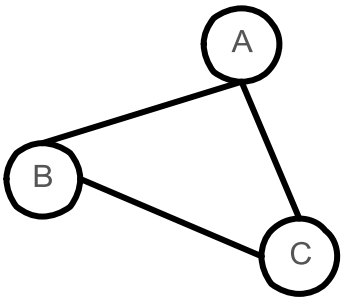
Yang kedua adalah graph tidak berarah. Graph tidak berarah merupakan graph yang sisinya tidak memiliki arah (kebalikan dari graph berarah). Pada graph tidak berarah, urutan pasangan simpul yang berhubungan tidak diperhatikan.



Gambar II-3 Gambar III-1 Graph Tidak Berarah

Dari Gambar II-3 dapat dilihat bahwa pasangan simpul (Bandung, Garut) dan (Garut, Bandung) menyatakan hal yang sama karena busur yang ada tidak memiliki orientasi arah. Graph dapat dinyatakan sebagai sebuah struktur data, atau secara spesifik dinamakan sebagai ADT (abstract data type) yang terdiri dari kumpulan simpul dan sisi yang membangun hubungan antar simpul. Secara umum terdapat dua macam representasi dari struktur data graph, yang pertama *adjacency list* dan yang kedua *adjacency matrix.*

Adjacency list merupakan bentuk representasi dari busur atau sisi dalam suatu graph sebagai satu senarai. Simpul yang terhubung oleh busur atau sisi dinyatakan sebagai simpul yang terkait. Dalam implementasinya, tabel hash digunakan untuk menghubungkan simpul dengan senarai yang berisi simpul-simpul yang saling terkait.



Gambar II-4 *Adjency List*

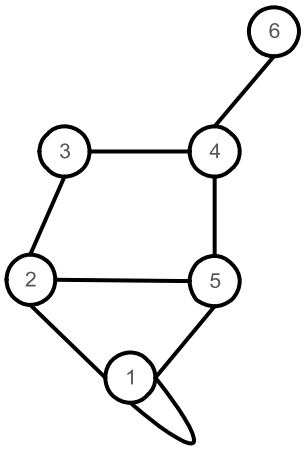
Graph pada Gambar II-4 *Adjency List* dapat digambarkan sebagai senarai {A,B}, {A,C} dan {B,C}. Representasi dari *adjacency list* dapat dilihat pada Tabel II-2.

Tabel II-2 Representasi Adjacency List

Simpul	Kedekatan	Senarai Dari Simpul yang Berhubungan
A	Dekat dengan	B,C
B	Dekat dengan	A,C
C	Dekat dengan	A,B

Salah satu kekurangan dari representasi *adjacency list* adalah tidak adanya tempat untuk menyimpan nilai yang melekat pada sisi yang mana contoh nilai ini adalah nilai jarak simpul.

Yang kedua adalah *adjacency matrix. Adjacency matrix* merupakan representasi matriks $n \quad x \quad n$ yang menyatakan hubungan antar simpul dalam graph. Kolom dan baris dari matriks merepresentasikan simpul-simpul, dan nilai entri dalam matriks ini menyatakan hubungan antar simpul. Berikut ini adalah representasi *adjacency matrix* dari graph tak berarah.



Gambar II-5 Graph Tak Berarah

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Gambar II-6 Representasi Adjacency Matrix Dari Gambar II-5

Kelebihan dari adjacency matrix ini adalah elemen matriksnya dapat diakses langsung melalui indeks. Sedangkan kekurangannya bila graph memiliki jumlah sisi atau busur yang relatif sedikit, kebutuhan ruang hash tabel untuk matriks menjadi boros dan tidak efisien karena komputer menyimpan elemen yang tidak perlu.

II.6. UML

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standard. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan yang lainnya harus mengikuti standar yang ada.UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya [**11**]. UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.

3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

II.4.1. Diagram-Diagram UML

Beberapa literatur menyebutkan bahwa UML menyediakan sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misalnya diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

a. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.
2. Diagram Paket (*Package Diagram*). Bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
3. Diagram *Use-Case.* Bersifat statis. Diagram ini memperlihatkan himpunan use-case dan actor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengordinasikan dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan *sequence* (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram Komunikasi (*Communication Diagram)*. Bersifat dinamis. Diagram sebagai pengganti diagramkolaborasi UML 1.4 yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (Statechart Diagram). Bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*state)*, transisi,

kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface)*, kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.

7. Diagram Aktivitas (*Activity Diagram)*. Bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram).* Bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalam satu atau lebih kelas-kelas, antaramuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram Deployment (*Deployment Diagram).* Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (run-time). Memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. Diagram deployment berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing).*
10. Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada UML dimungkinkan kita menggunakan diagram-diagram lain (misalnya data *flow diagram*, *Entity Relationship* diagram dan sebagainya) [**11**].

II.5. HTML5 Canvas

HTML5 adalah versi terkini dari HTML (*HyperText Markup Language*).HTML pertamakali dijadikan standarisasi sebagai bahasa yang digunakan untuk membuat halaman web pada 1993.HTML menggunakan *tag*<> dalam mendefinisikan fungsi dan konten dari halaman web.

HTML5 Canvas adalah bagian pada HTML5 yang merupakan *bitmapped area* dan dapat dimanipulasi dengan menggunakan JavaScript. HTML5 Canvas akan me-*render* sepenuhnya dari area tersebut dari tiap-tiap *frame*menggunakan Canvas API yang dipanggil melalui JavaScript. HTML5 Canvas digunakan untuk mengolah dan menampilkan data dalam bentuk grafik pada HTML. [**12**]

II.6. Construct 2

Construct 2 adalah tools yang digunakan dalam membuat sebuah game tanpa memerlukan pengetahuan tentang pemrograman. caranya hanya melakukan drag and drop item yang tersedia, menambahkan perilaku mereka, dan membuat mereka menjadi bergerak dengan sebuah event.

Construct 2 memiliki interface yang sangat mirip dengan produk microsoft office. hal ini membuat mudah bagi orang-orang yang terbiasa dengan produk office. karena sifat HTML 5 masih eksperimental, ada beberapa fitur dalam pembuatan game yang mungkin tidak bekerja dengan baik di semua perangkat. Construct 2 merupakan *tools* yang dapat dipelajari dengan cepat, dalam pembuatan *game* dapat dilakukan dengan mengekspor permainan sendiri dan bekerja dalam *mobile device* [**13**].

II.7. PhoneGap

PhoneGap adalah sebuah *open source development tool* diciptakan oleh Nitobi yang berfungsi sebagai penghubung antara aplikasi berbasis HTML dan aplikasi berbasis mobile [**12**].PhoneGap bekerja dengan menggabungkan HTML5 Canvas dan JavaScript. Saat ini PhoneGap mendukung beberapa sistem operasi perangkat *mobile* terkemuka yaitu Android, iOS dan Blackberry. Pembuatan aplikasi menjadi lebih efisien dengan menggunakan PhoneGap dikarenakan cukup menggunakan HTML5 Canvas dan JavaScript untuk aplikasi pada Android, iOS dan Blackberry.

II.8. Android

Android adalah sebuah system operasi untuk perangkat mobile berbasis linux yang mencakup system operasi, *middleware* dan aplikasi.Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka.Android merupakan generasi baru *platform mobile*, *platform* yang memberikan pengembangan untuk melakukan pengembangan sesuai dengan yang diharapkan. Pengembangan aplikasi Android diperbolehkan untuk mendistribusikan aplikasi mereka di bawah skema lisensi apapun yang mereka inginkan.Pengembang memiliki beberapa pilihan ketika membuat aplikasi yang berbasis android. Sebagian besar pengembang menggunakan *eclipse* yang tersediasecara bebas untuk merancang dan mengembangkan aplikasi android, karena memiliki android *plug-in* yang tersedia untuk memfasilitasi pengembangan android. Selain itu, *eclipse* juga mendapat dukungan langsung dari google untuk menjadi IDE pengembangan aplikasi Android,ini terbukti dengan adanya penambahan *plug-in* untuk *eclipse* untuk membuat *project* android dimana *source software* langsung dari situs resminya google [**14**].

II.9. Adobe Photoshop

Photoshop adalah sebuah *image editor* atau program penyunting gambar yang berfungsi untuk membuat, menyunting , dan memodifikasi gambar-gambar digital yang terdapat didalam computer.

Photoshop merupakan program penyunting gambar yang paling hebat dan paling popular hingga saat ini. Kemampuan serta fasilitasnya yang lengkap membuatnya diminati oleh para seniman, profesional, maupun pemula yang membutuhkan sebuah program gambar yang lengkap namun mudah dalam penggunaannya.

Photoshop saat ini digunakan untuk membuat gambar-gambar untuk keperluan seperti iklan,brosur, poster, serta berbagai macam *output* cetak lainnya. Bagi para penggemar photografi, Photoshop juga dapat digunakan untuk me-*retouch* foto yang sudah rusak, hingga memodifikasi foto sehingga menjadi lebih baik [**15**].