BAB 2

LANDASAN TEORI

2.1 Permainan (*Game*)

Permainan (game) adalah sebuah aktivitas rekreasi dengan tujuan bersenang-senang, mengisi waktu luang, atau berolahraga ringan. Permainan (*game*) biasanya dapat dilakukan sendiri atau bersama-sama [4]. Permainan (*game*) terdiri dari permainan tradisional dan modern.

2.2 Permainan Tradisional

Permainan tradisional menurut James Danandjaja adalah salah satu bentuk yang berupa permainan anak-anak, yang beredar secara lisan diantara anggota kolektif tertentu, berbentuk tradisional dan diwarisi turun temurun serta banyak mempunyai variasi. Sifat atau ciri dari permainan tradisional anak sudah tua usianya, tidak diketahui asal-usulnya, siapa penciptanya dan darimana asalanya. Biasanya disebarkan dari mulut ke mulu dan kadang-kadang mengalami perubahan nama atau bentuk meskipun dasarnya sama. Jika dilihat dari akar katanya, permainan tradisional tidak lain adalah kegiatan yang diatur oleh suatu peraturan permainan yang merupakan pewarisan dari generasi terdahulu yang dilakukan manusia (anak-anak) dengan tujuan mendapatkan kegembiraan.

Menurut Atik Soepandi, Sekar dan kawan-kawan, yang disebut permainan adalalah perbuatan untuk menghibur hati baik yang mempergunakan alat atau pun tidak mempergunakan alat. Sedangkan yang dimaksud tradisional adalah segala apa yang dituturkan atau diwariskan secara turun temurun dari orang tua atau nenek moyang. Jadi permainan tradisional adalah segala perbuatan baik mempergunakan alat atau tidak yang diwariskan turun temurun dari nenek moyang, sebagai sarana hiburan atau untuk menyenangkan hati.

Permainan tradisional ini bisa dikategorikan dalam tiga golongan :

1. Permainan untuk bermain (rekreatif) yaitu permainan yang pada umumnya dilakukan untuk mengisi waktu senggang.

2. Pemainan untuk bertanding (kompetitif), memiliki ciri-ciri : terorganisir, bersifat kompetitif, dimainkan oleh paling sedikit 2 orang, mempunyai kriteria yang menentukan siapa yang menang dan yang kalah, serta mempunyai peraturan yang diterima bersama oleh pesertanya.
3. Permainan yang bersifat edukatif, terdapat unsur-unsur pendidikan didalamnya.

2.2.1 Macam-macam Permainan Tradisional

Dari sekian banyak permainan tradisional yang ada di Indonesia, berikut adalah beberapa di antaranya :

1. Egrang

   Permainan egrang muncul sebelum Indonesia merdeka, tepatnya dibawa oleh orang Belanda. Egrang merupakan permainan yang menggunakan batang kayu atau bambu yang diberi pijakan untuk berjalan. Egrang menggunakan dua bilah bambu apus atau wulung yang lurus dan sudah tua sepanjang 2 meter yang dilubangi dengan jarak sekitar 30 – 50 cm atau bisa lebih tinggi lagi. Lubang tersebut dilesakkan potongan bambu lain sepanjang 20 – 30 cm untuk dijadikan pijakan. Egrang diklasifikasikan menjadi dua permainan yaitu adu balap dan saling menjatuhkan dengan cara memukul kaki bambu. Dalam permainan adu balap, masing-masing pemain berdiri di atas garis yang sama dan saling pacu menuju ke garis *finish*. Pemain yang lebih dulu mencapai *finish* adalah pemenangnya.
2. Benteng

   Permainan ini dimainkan oleh dua kelompok, masing-masing kelompok terdiri dari 4 sampai 8 orang. Kedua kelompok kemudian akan memilih suatu tempat sebagai markas, biasanya sebuah tiang, batu atau pilar yang disebut sebagai “benteng”. Tujuan utama permainan ini adalah untuk menyerang dan mengambil alih “benteng” lawan dengan menyentuh tiang atau pilar yang telah dipilih oleh lawan dan meneriakkan kata benteng. Kemenangan juga bisa diaih dengan “menawan” seluruh anggota lawan dengan menyentuh tubuh mereka. Untuk menentukan siapa yang berhak

menjadi “penawan”, ditentukan dari siapa yang paling akhir menyentuh “benteng” mereka.

3. Kelereng

   Kelereng adalah mainan kecil berbentuk bulat yang terbuat dari kaca atau tanah liat. Ukuran kelereng sangat bermacam-macam, umumnya 1,25 cm. Permainan kelereng ini biasanya dimainkan oleh anak sekolah dasar umur 7 tahun.

4. Layang-layang

   Permainan layang-layang, juga dikenali dengan nama wau merupakan satu aktivitas menerbangkan layang-layang tersebut di udara. Pada musim kemarau di Indonesia anak-anak selalau bermain layang-layang karena anginnya besar.

5. Kasti

   Kasti atau Gebokan merupakan sejenis olahraga bola seperti halnya olahraga *softball* atau *baseball*. Permainan yang dilakukan 2 kelompok ini menggunakan bola tenis sebagai alat untuk menembak lawan dan tumpukan batu untuk disusun. Siapapun yang berhasil menumpuk batu tersebut dengan cepat tanpa terkena pukulan bola adalah kelompok yang memenangkan permainan.

2.3 *Video Game*

Video game merupakan salah satu media hiburan salah satu hiburan yang paling populer untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat sekarang, teknologi game telah mengalami kemajuan yang terbilang sangat pesat. Hali ditandai dengan berkembangnya jenis, produk, alat dan jenis interaksi game dengan penggunaan yang semakin beragam bentuknya.

2.3.1 Pengertian Game

Game merupakan sebuah bentuk seni dimana penggunanya, disebut dengan pemain (*player*), diharuskan membuat keputusan-keputusan dengan tujuan untuk mengelola sumber daya yang diperoleh dari kesempatan-kesempatan

bermain (token) miliknya, untuk mencapai suatu tujuan tertentu. *Video game* adalah bentuk game yang interaksi utamanya melibatkan media *video* (dan biasanya melibatkan audio). Berdasarkan representasi visualnya, *game* dapat dibedakan menjadi 2 jenis, yaitu game 2 dimensi (2D) dan 3 dimensi (3D). Game 2D adalah game, yang secara matematis, hanya melibatkan 2 elemen koordinat kartesius, yaitu x dan y, sehingga konsep kamera pada game 2D hanya menentukan "gambar" mana pada game yang dapat dilihat oleh pemain. Sedangkan game 3D adalah game yang selain melibatkan elemen x dan y, juga melibatkan elemen z pada perhitungannya, sehingga konsep kamera pada game 3D benar-benar menyerupai konsep kamera pada kehidupan nyata, yaitu selain digeser (seperti pada game 2D), juga dapat diputar dengan sumbu tertentu.

2.3.2 Sudut Pandang Permainan (*Game View Point)*

Sebuah *game* biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan *genre game* yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan :

1. *Side Scrolling*

   Adalah sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinkan *background* pada *game* seolah-olah bergeser mengikuti pergerakan karakter utama.

2. *Top Down*

   Adalah sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya *game* yang menggunakan sudut pandang permainan jenis ini adalah *shooter game*.

3. *Isometric*

   Adalah sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan *genre* RTS (*Real Time Strategy*) [5].

2.3.3 Jenis-jenis *Game*

Berikut ini akan dijabarkan beberapa jenis game berdasarkan cara pembuatannya, cara pemasarannya, mesin yang menjalankannya. Jenis game diantaranya adalah :

1. *Game* PC

   Game yang dimainkan pada PC (*Personal Computer*) yang memiliki kelebihan yaitu memiliki tampilan antarmuka yang baik untuk input maupun output, output visual kualitas tinggi karena layar computer biasanya memiliki resolusi yang jauh lebih tinggi dibandingkan dengan layar televisi biasa. Kekurangannya adalah spesifikasi komputer yang sangat bervariasai antara satu komputer dengan komputer lainnya menyebabkan beberapa game dapat ditampilkan dengan baik pada satu komputer, tetapi tidak berjalan dengan baik pada komputer yang lainnya.

2. *Game Console*

   Game yang dijalankan pada suatu mesin spesifik yang biasanya tersedia di rumah pribadi, seperti Microsoft Xbox 360, Nintendo Wii dan sebagainya.

3. *Game Arcade*

   Game yang dijalankan pada mesin dengan input dan output audio visual yang telah terintegrasi dan tersedia ditempat-tempat umum, seperti mal, bandara dan sebagainya.

4. *Game Online*

   Game yang hanya dapat dimainkan secara *online* melalui LAN atau internet [6].

2.3.4 *Genre Game*

Dari sekian banyak *Game* yang bermunculan di dunia, *game-game* tersebut dikelompokkan menjadi beberapa *genre*. Pada dasarnya, *genre-genre* game yang beredar di pasaran dapat dikelompokkan sebagai berikut.

1. *Action Game*

   Adalah tipe *game* dengan fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari

musuh atau menghindari rintangan. Pengembang *game* tipe ini perlu memastikan *game* yang dibuat dioptimasi sehingga pemain memiliki pengalaman bermain yang baik, yang tidak terganggu oleh *delay* proses yang lama, contohnya : *Metal Gear Solid*.

2. *Adventure Game*

Adalah tipe *game* yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, mencari pesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat *game* ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain, contohnya : *Tomb Raider*.

3. *Sport Game*

Adalah tipe *game* yang berupa kompetisi antara dua pemain atau lebih, dimana pemain dapat berupa *individual* atau tim. Contoh *game* tipe ini antara lain sepakbola, bola basket, tenis, dan bilyard. Tergantung seberapa cepat permainan yang terjadi, aplikasi *game* perlu dioptimalkan, contohnya : *Pro Evolution Soccer*.

4. *RPG* (*Role Playing Game*)

Adalah tipe *game* yang seringkali berupa *multi* karakter *game* di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang spesifik. Para pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafis yang khas untuk setiap karakter pemain ditambah dengan *storyline* yang mendebarkan akan sangat menarik dan memberikan pengalaman yang berbeda di dalam bermain, contohnya : *Final Fantasy*.

5. *Platform Game*

Adalah tipe *game* yang mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Tipe *game* ini sedikit serupa dengan *action game*, tetapi aksinya tidak secepat *action game*. Teknik *collision*

detection sangat sering dimanfaatkan pada tipe *game* ini, contohnya : *Sonic The* Hedgehog, dan *Mario Bros.*

6. *Puzzle Game*

Adalah tipe *game* yang umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. *Game* ini lebih bersifat *statis* dibanding *action game*. Pembuatan *game* tipe ini seringkali ditunjang dengan AI (*Artificial Intelligence*), contohnya : *7 Wonders of The Ancient World.*

7. *Sandbox Game*

Adalah tipe *game* yang umumnya ditujukan untuk menjelajahi suatu kota atau tempat dan bebas berinteraksi dengan objek disekitarnya. Mungkin *game* ini bisa di bilang mirip RPG tapi *Sandbox* sangat berbeda dengan RPG *game*, perbedaannya adalah tidak adanya peningkatan *level* karakternya, contohnya : *GTA*, dan *Bully*.

8. *FPS* (*First Person Shooter*)

Adalah tipe *game* yang mengunakan sudut pandang orang pertama untuk membidik atau membunuh musuh, sehingga kita hanya melihat tangannya saja dan tidak melihat tubuh karakter yang dimainkan, contohnya : *Call of Duty*.

9. *TPS* (*Third Person Shooter*)

Adalah tipe *game* yang mirip dengan FPS, tapi bedanya tipe *game* ini mengunakan sudut pandang orang ke-3, sehingga kita bisa melihat seluruh tubuh karakter yang dimainkan, contohnya : *Gears of Wars.*

10. *Music Game*

Adalah tipe *game* yang menuntut pemainnya untuk menekan tombol sesuai dengan tombol yang ada di layar dengan diiringi dengan musik, contohnya : *Guitar Hero*, dan *Dance Dance Revolution.*

11. *Fighting Game*

Adalah tipe *game* yang intinya harus menjatuhkan lawan tandingnya, entah itu dengan pukulan, tendangan, *combo*, maupun dengan jurus *special*, contohnya : *Tekken & Mortal Kombat.*

12. *RTS* (*Real Time Strategy*)

Adalah tipe *game* yang mengharuskan pemainnya menggunakan taktik dan strategi untuk jeli dalam melihat setiap peluang, kelemahan musuh dan bijaksana dalam menggunakan sumber daya yang ada, contohnya : *Command & Conquer*.

13. *Simulation Game*

Adalah tipe *game* yang memberikan pengalaman atau interaksi sedekat mungkin dengan keadaan yang aslinya, meskipun terkadang keadaan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detail dan pengalaman *realistik* menggunakan keadaan tersebut, contohnya : *Gran Turismo*.

14. *Racing Game*

Adalah tipe *game* yang tujuannya adalah mencapai garis finish dari suatu *race*, dalam *game* ini biasanya pemain dapat memilih dan membeli kendaraan, mendandani, melakukan *upgrade* mesin , contohnya : *Need For Speed*.

2.4 Kecerdasan Buatan / AI (*Artificial Intelligence*)

Berikut ini adalah beberapa definisi AI yang disampaikan oleh beberapa ahli. Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing.

1. Schalkoff

   Artificial Intelligence adalah bidang studi yang berusaha menerangkan dan meniru perilaku cerdas dalam bentuk proses komputasi.

2. Rich dan Knight

   Artificial Intelligence adalah studi tentang cara membuat komputer melakukan sesuatu yang sampai saat ini orang dapat melakukannya lebih baik.

3. Luger dan Stubblefield

   Artificial Intelligence adalah cabang ilmu komputer yang berhubungan dengan otomasi perilaku yang cerdas.

4. Haag dan Keen

   Kecerdasan Buatan adalah bidang studi yang berhubungan dengan penangkapan, pemodelan dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi informasi sehingga sistem tersebut dapat memfasilitasi proses pengambilan keputusan yang biasanya dilakukan oleh manusia.

Definisi kecerdasan buatan (*Artificial Intelligence*) atau disingkat AI, adalah suatu pengetahuan yang membuat komputer dapat meniru kecerdasan manusia sehingga diharapkan komputer dapat melakukan hal-hal yang memerlukan kecerdasan seperti melakukan penalaran untuk mencapai suatu kesimpulan atau melalukan translasi dari satu bahasa manusia ke bahasa manusia yang lain.

Dalam katagori AI ada yang dikatakan sebagai AI tingkat rendah dan AI sebagai tingkat tinggi. Misalnya permainan game seperti tic-tac-toe (permainan tiga dara), permainan catur dan permainan komputer yang lainnya, permainan ini dapat dikatan sebagai AI yang tingkatannya masih rendah karena keputusan yang diambilnya masih dapat diprediksi. Adapun AI sebagai tingkat tinggi misalnya pernahkah anda mendengar suatu cerita atau menyaksikan film-film fiksi yang menunjukkan suatu komputer dengan kepintaran yang melebihi manusia, yang dapat bercakap-cakap dengan manusia dan memberikan keputusan serta pendapat yang cerdik. Hal inilah yang dikatakan sebagai AI tingkat tinggi yang masih diimpikan oleh para ahli untuk mewujudkannya.

Menurut O'Brien atribut perilaku cerdas adalah :

1. Berfikir dan Bernalar
2. Memakai penalaran untuk memecahkan persoalan
3. Menyerap dan menerapkan pengetahuan
4. Meperlihatkan kreativitas dan imajinasi
5. Bekerja dengan situasi yang kompleks dan membingungkan
6. Melakukan tanggapan dengan cepat dan berhasil terhadap situasi baru
7. Mengenali elemen-elemen yang relatif penting dalam suatu situasi
8. Menangani informasi yang rancu, tak lengkap atau salah

Ruang lingkup penelitian dan pengembangan AI sekarang ini dapat dikelompokkan kedalam beberapa bidang yaitu:

1. *Game Playing* (Permainan game)
2. *General problem solving* (Pemecahan masalah secara umum)
3. *Natural language recognition* (Pengenalan bahasa alamiah)
4. *Speech recognition* (Pengenalan percakapan)
5. *Visual recognition* (Pengenalan penglihatan)
6. *Robotics*
7. *Expert System* (Sistem ahli)

AI atau *Artificial Intelligence* mempunyai beberapa kelebihan dibandingkan dengan kecerdasan alami (Kecerdasan Manusia). Kelebihan tersebut dipaparkan sebagai berikut:

1. AI lebih bersifat permanen

   Berbeda dengan AI, kecerdasan Alami yang dipunyai oleh seseorang tidak dapat disimpan. Ketika orang tersebut pindah kerja, pengetahuan yang dimilikinya ikut terbawa. AI lebih bersifat permanen karena tetap ada sepanjang sistem komputer dan program masih terpelihara.

2. AI menawarkan kemudahan untuk digandakan dan disebarkan

   Pemindahan pengetahuan dari satu orang ke orang lain memerlukan waktu yang panjang dan bahkan mungkin pengetahuan itu tidak dapat diduplikasi secara lengkap. Adapun pengetahuan dalam sistem komputer mudah sekali untuk disalin dan dipidahkan ke sistem lain.

3. AI dapat lebih murah daripada kecerdasan Alami

   Telah banyak dibuktikan bahwa biaya membeli jasa dengan komputer lebih murah daripada biaya untuk membiayai manusia yang melaksanakan tugas yang sama.

4. AI bersifat konsisten dan teliti
5. AI dapat didokumentasi

2.4.1 AI yang Digunakan

Algoritma adalah cara yang dapat ditempuh oleh komputer dalam mencapai suatu tujuan, terdiri atas langkah-langkah yang terdefinisi dengan baik, menerima input, melakukan proses, dan menghasilkan output meskipun tidak selalu, biasanya sebuah algoritma memiliki sifat bisa dihitung (*computable*) atau bisa dihitung. Sebuah algoritma dikatakan benar (correct, jika algoritma tersebut berhasil mengeluarkan output yang benar untuk semua kemungkinan input. Jika sebuah algoritma dikatakan 99% benar, algoritma tersebut tetap salah (*incorrect*). Agar algoritma tersebut dikatakan benar, algoritma tersebut harus benar 100% [14].

Algoritma A* (*A Star*) merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai : f(n) = g(n) + h(n). Dengan perhitungan biaya seperti ini, algoritma A* (*A Star*) adalah complete dan optimal. Sama dengan algoritma dasar *Best First Search*, algoritma A* (A *Star*) ini juga menggunakan *OPEN* dan *CLOSED*. Terdapat tiga kondisi bagi setiap suksesor yang dibangkitkan, yaitu :

1. Sudah berada di *OPEN*.
2. Sudah berada di *CLOSED*.
3. Tidak berada di *OPEN* maupun *CLOSED*.

Pada ketiga kondisi tersebut diberikan penanganan yang berbeda-beda. Jika suksesor sudah berada di *OPEN*, maka dilakukan pengecekan apakah perlu pengubahan parent atau tidak tergantung pada nilai g nya melalui parent lama atau parent baru. Jika melalui parent baru memberikan nilai g yang lebih kecil, maka dilakukan pengubahan parent. Jika pengubahan parent dilakukan, maka dilakukan pula perbaruan (update) nilai g dan f pada suksesor tersebut. Dengan perbaruan ini, suksesor tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai simpul terbaik (*best node*).

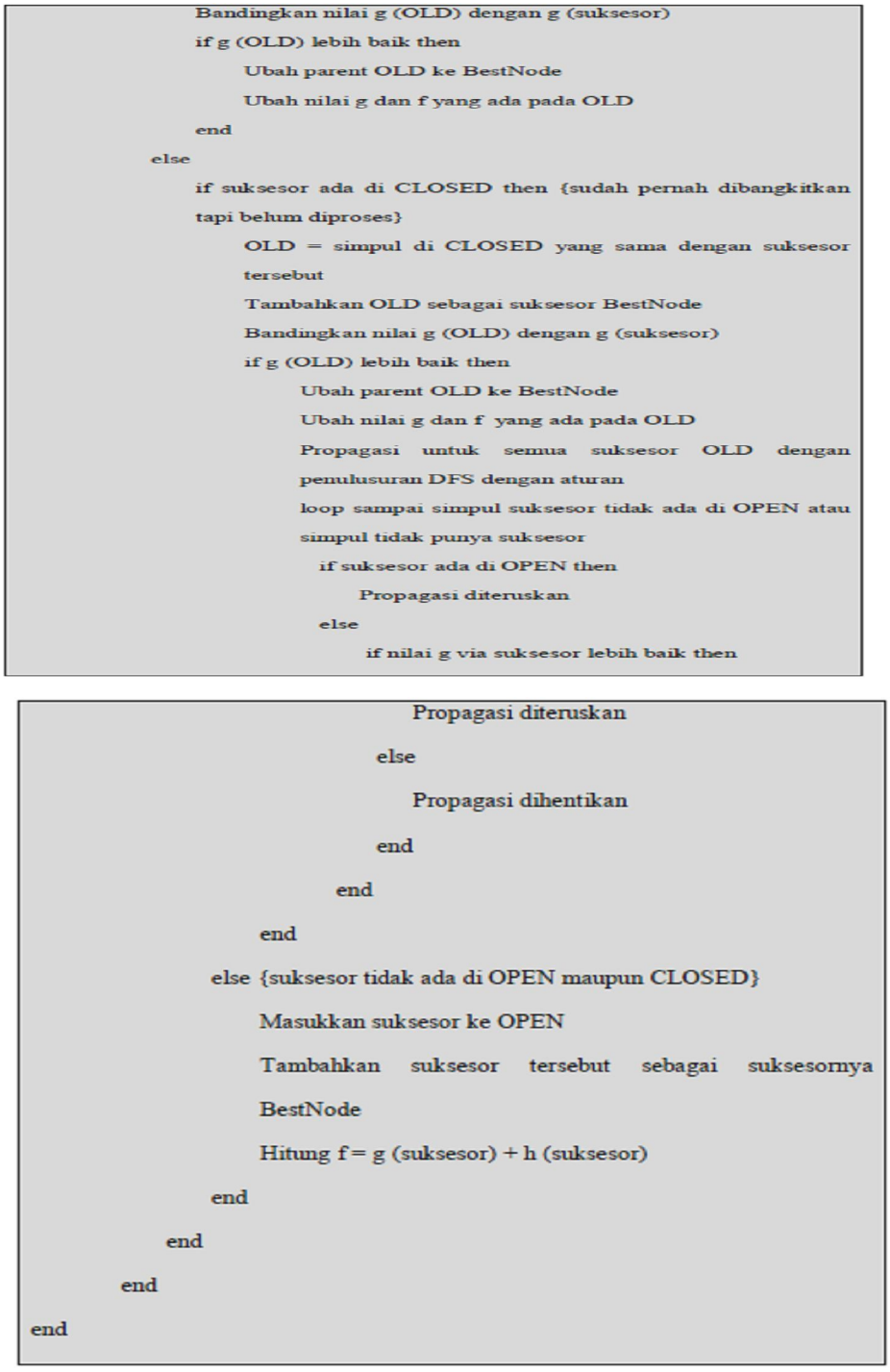
Jika suksesor sudah pernah berada di *CLOSED*, maka akan dilakukan pengecekan apakah perlu pengubahan parent atau tidak. Jika ya, maka dilakukan

perbaruan nilai g dan f pada suksesor tersebut serta pada semua "anak cucunya" yang sudah pernah berada di *OPEN*. Dengan perbaruan ini, maka anak cucunya tersebut memiliki kesempatan lebih besar untuk terpilih sebagai simpul terbaik (*best node*).

Jika suksesor tidak berada di *OPEN* maupun *CLOSED*, maka suksesor tersebut dimasukkan ke dalam *OPEN*. Tambahkan suksesor tersebut sebagai suksesornya best node. Hitung biaya suksesor tersebut dengan rumus f = g + h. *Pseudo-code* algoritma A* (*Star*) dapat di lihat pada gambar 2.1 :

```
function A*(masalah) return solusi
        OPEN ← S
        CLOSED ← array kosong

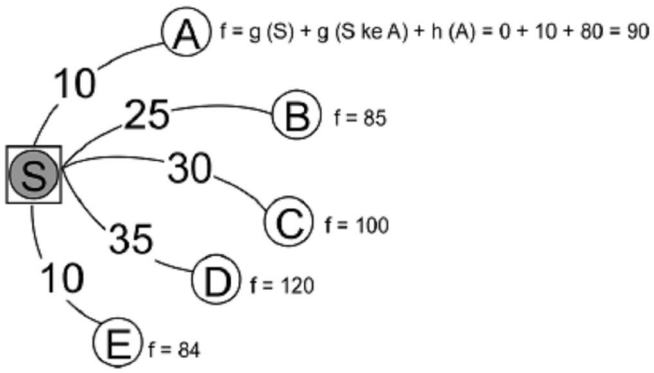
loop sampai goal ditemukan atau sampai tidak ada simpul didalam OPEN
        if OPEN = kosong then
            Gagal
        else
            BestNode = simpul yang ada di OPEN dengan f minimal
            Pindahkan simpul terbaik tersebut dari OPEN ke CLOSED
        if BestNode = goal then
            Sukses
        else
            Bangkitkan semua suksesor BestNode tapi jangan buat pointer
            Untuk setiap suksesor kerjakan :
            Hitung g (suksesor) = g (BestNode) + actual cost (dari BestNode ke
            suksesor)
            {Periksa suksesor}
            if suksesor ada di OPEN then {sudah pernah dibangkitkan tapi
            belum diproses}
                OLD = simpul di OPEN yang sama dengan suksesor tersebut
                Tambahkan OLD sebagai suksesor BestNode
                Buat pointer dari OLD ke BestNode
```



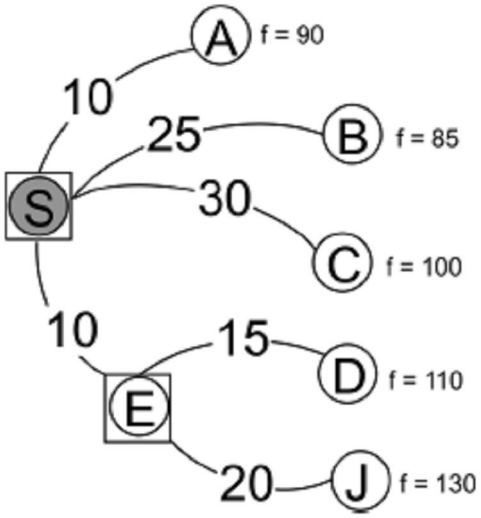
```
                Bandingkan nilai g (OLD) dengan g (suksesor)
                if g (OLD) lebih baik then
                    Ubah parent OLD ke BestNode
                    Ubah nilai g dan f yang ada pada OLD
                end
            else
                if suksesor ada di CLOSED then {sudah pernah dibangkitkan
                tapi belum diproses}
                    OLD = simpul di CLOSED yang sama dengan suksesor
                    tersebut
                    Tambahkan OLD sebagai suksesor BestNode
                    Bandingkan nilai g (OLD) dengan g (suksesor)
                    if g (OLD) lebih baik then
                        Ubah parent OLD ke BestNode
                        Ubah nilai g dan f yang ada pada OLD
                        Propagasi untuk semua suksesor OLD dengan
                        penulusuran DFS dengan aturan
                        loop sampai simpul suksesor tidak ada di OPEN atau
                        simpul tidak punya suksesor
                          if suksesor ada di OPEN then
                              Propagasi diteruskan
                          else
                              if nilai g via suksesor lebih baik then
                                  Propagasi diteruskan
                              else
                                  Propagasi dihentikan
                              end
                          end
                    end
                else {suksesor tidak ada di OPEN maupun CLOSED}
                    Masukkan suksesor ke OPEN
                    Tambahkan suksesor tersebut sebagai suksesornya
                    BestNode
                    Hitung f = g (suksesor) + h (suksesor)
                end
            end
        end
end
```

Gambar 2.1 *Pseudo-code* algoritma A* [7]

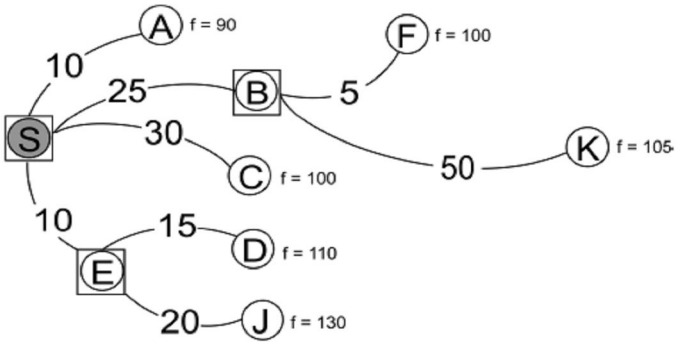
Untuk memperjelas pemahaman tentang algoritma A*, berikut adalah langkah-langkah dalam pencarian rute berdasarkan algoritma ini yang dapat dilihat pada gambar 2.2, gambar 2.3, gambar 2.4, gambar 2.5, gambar, 2.6 dan gambar 2.7 :



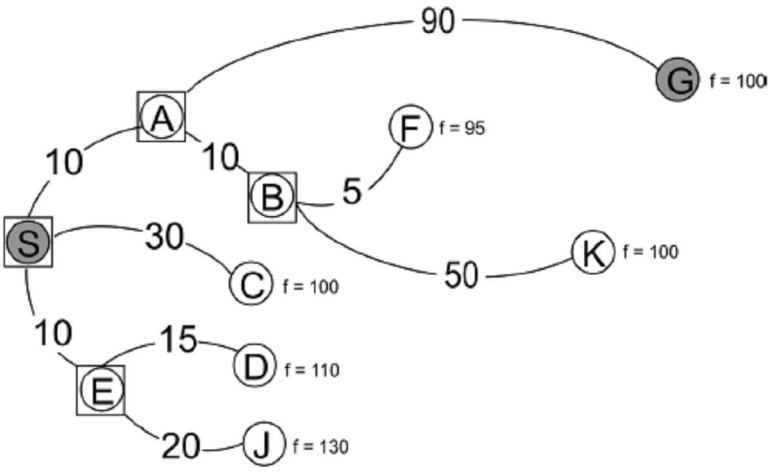
Gambar 2.2 Langkah 1 pencarian rute dengan algoritma A*



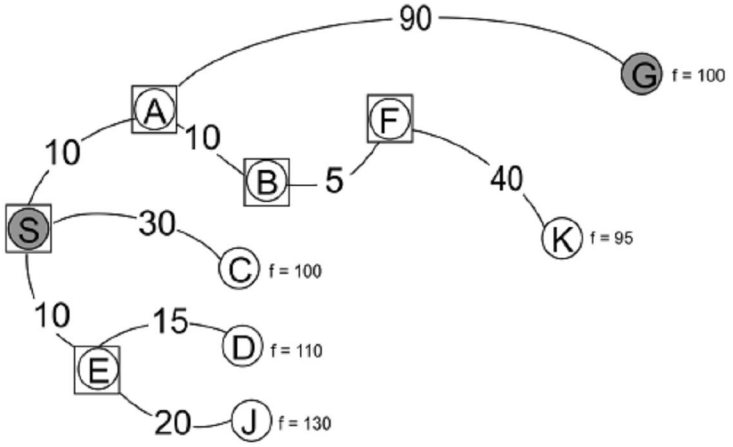
Gambar 2.3 Langkah 2 pencarian rute dengan algoritma A*



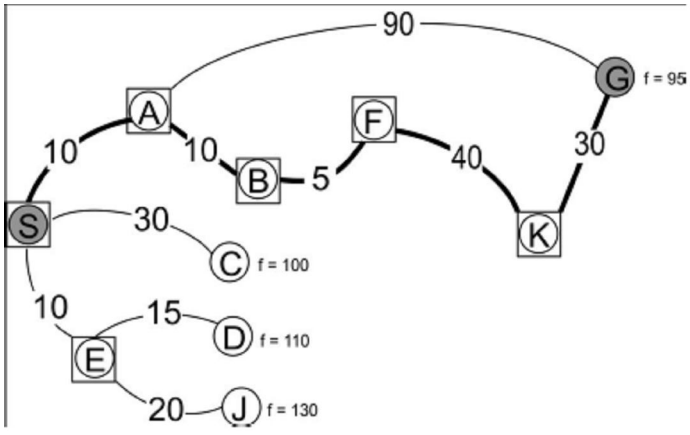
Gambar 2.4 Langkah 3 pencarian rute dengan algoritma A*



Gambar 2.5 Langkah 4 pencarian rute dengan algoritma A*



Gambar 2.6 Langkah 5 pencarian rute dengan algoritma A*



Gambar 2.7 Langkah 6 pencarian rute dengan algoritma A*

Simpul dengan tanda kotak menyatakan bahwa simpul tersebut berada di *CLOSED* (pernah terpilih sebagai *Best Node*) sedangkan simpul tanpa tanda kotak menyatakan simpul yang ada berada di OPEN.

Untuk langkah pertama, karena di *OPEN* hanya terdapat satu simpul yaitu S, maka S langsung terpilih sebagai BestNode dan dipindahkan ke *CLOSED*. Kemudian dibangkitkan semua suksesor S, yaitu : A, B, C, D dan E. Karena kelima suksesor tidak ada di *OPEN* dan *CLOSED*, maka kelimanya dimasukkan ke *OPEN*. Langkah pertama ini menghasilkan *OPEN* = [A, B, C, D, E] dan *CLOSED* = [S].

Langkah kedua, E dengan biaya terkecil (yaitu 84) terpilih sebagai *Best Node* dan dipindahkan ke *CLOSED*. Lalu semua suksesor E dibangkitkan, yaitu : D dan J. Karena belum pernah ada di *OPEN* maupun *CLOSED* sebelumnya maka J dimasukkan ke *OPEN*. Sedangkan simpul D sudah ada di *OPEN*, maka harus dicek apakah parent dari D perlu diganti atau tidak. Ternyata biaya dari S ke D melalui E (yaitu 10 + 15 = 25) lebih kecil daripada biaya S ke D (yaitu 35). Oleh karena itu, parent dari D harus dirubah, yang semula S menjadi E. Dengan perubahan parent ini maka nilai g dan f pada D juga diperbaharui (nilai g yang semula 35 menjadi 25, dan nilai f dari 120 menjadi 110). Langkah kedua ini menghasilkan *OPEN* = [A, B, C, D, J] dan *CLOSED* = [S, E].

Langkah ketiga, B dengan biaya terkecil (yaitu 85) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Lalu semua suksesor B dibangkitkan, yaitu : A, F dan K. Karena belum pernah ada di *OPEN* maupun *CLOSED*, maka F dan K dimasukkan ke *OPEN*. Sedangkan simpul A sudah ada di *OPEN*, maka harus dicek apakah parent dari A perlu diganti atau tidak. Ternyata, biaya dari S ke A melalui B (yaitu 25 + 10 = 35) lebih besar daripada biaya S ke A (yaitu 10). Oleh karena itu, parent dari A tidak perlu diubah (tetap S). Akhir dari langkah ke tiga ini menghasilkan *OPEN* = [A, C, D, F, J, K] dan *CLOSED* = [S, E, B].

Langkah keempat, A dengan biaya terkecil (yaitu 90) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Lalu semua suksesor A dibangkitkan, yaitu : B dan G. Karena belum pernah ada di *OPEN* maupun *CLOSED*, maka G dimasukkan ke *OPEN*. Sedangkan simpul B sudah ada di *CLOSED*, maka harus

dicek apakah *parent* dari B perlu diganti atau tidak. Ternyata biaya biaya dari S ke B melalui A (yaitu 10 + 10 = 20) lebih kecil daripada biaya dari S ke B (yaitu 25). Oleh karena itu parent dari B harus diubah, yang semula S menjadi A. Nilai g dan f pada B juga harus diperbarui (nilai g yang semula 25 menjadi 20, dan nilai f dari 85 menjadi 80). Nilai g dan f pada suksesor-suksesor B (sampai anak cucu) juga harus diperbarui menggunakan penulusuran DFS (*Depth First Search*). Dalam kasus ini, B hanya mempunyai dua anak (tidak punya cucu), yaitu F dan K. Nilai g (F) yang semula 30 diubah menjadi 25, dan nilai f (F) dari 100 menjadi 95. Nilai g (K) yang semula 75 diubah menjadi 70, dan nilai f (K) dari 105 menjadi 100). Akhirnya, *OPEN* = [C, D, F, G, J, K] dan *CLOSED* = [S, E, B, A].

Langkah kelima, F dengan biaya terkecil (yaitu 95) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Lalu semua suksesor F dibangkitkan, yaitu : K. Karena K sudah ada di *OPEN*, maka harus dicek apakah *parent* dari K perlu diganti atau tidak. Biaya dari S ke K melalui F ternyata lebih kecil daripada biaya dari S ke K melalui *parent* lama (B). Oleh karena itu *parent* dari K harus diubah, yang semula B menjadi F. Selanjutnya nilai g (K) yang semula 70 diubah menjadi 65, dan nilai f (K) dari 100 menjadi 95). Akhirnya, *OPEN* = [C, D, F, G, J, K] dan *CLOSED* = [S, E, B, A, F].

Langkah keenam, K dengan biaya terkecil (yaitu 95) terpilih sebagai *BestNode* dan dipindahkan ke *CLOSED*. Lalu semua suksesor K dibangkitkan, yaitu G. Karena G sudah ada di *OPEN*, maka harus dicek apakah *parent* dari G perlu diganti atau tidak. Biaya dari S ke G melalui K ternyata lebih kecil daripada biaya dari S ke G melalui *parent* lama (yaitu A). Oleh karena itu *parent* dari G harus diubah, yang semula A menjadi K. Selanjutnya nilai g (G) yang semula 100 diubah menjadi 95, dan nilai f (G) dari 100 menjadi 95). Pada akhir langkah keenam ini, *OPEN* = [C, D, G, J] dan *CLOSED* = [S, E, B, A, F, K].

Selanjutnya G dengan biaya terkecil (yaitu 95) terpilih sebagai *BestNode*. Karena *BestNode* sama dengan goal, berarti solusi sudah ditemukan. Rute dan total biaya bisa ditelusuri balik dari G menuju S karena setiap simpul hanya memiliki satu parent dan setiap simpul memiliki informasi biaya sebenarnya (g). Penelusuran balik menghasilkan rute S-A-B-F-K-G dengan total jarak sama

dengan 95 kilometer. Rute ini merupakan rute terpendek yang ada di graph tersebut. Jadi, algoritma A* (A *Star*) adalah optimal. Tanpa ada batasan waktu dan memory, A* (A *Star*) adalah *complete* (selalu menemukan solusi jika solusinya ada). Pada kasus tersebut A* (A *Star*) membangkitkan dan menyimpan 10 simpul (dari 13 simpul yang ada pada graph). Untuk masalah yang lebih kompleks, misalkan pencarian rute terpendek pada *graph* yang terdiri dari 100 juta simpul, A* (*A STAR*) akan menghadapi masalah waktu proses dan memori yang dibutuhkan. Untuk menyelesaikan kedua masalah tersebut, maka telah diusulkan berbagai variasi A* (*A Star*) dengan karakteristik yang sesuai untuk permasalahan tertentu [7].

2.5 Psikologi

Psikologi berasal dari kata dalam bahasa Yunani *Psychology* yang merupakan gabungan dan kata *psyche* dan *logos*. *Psyche* berarti jiwa dan *logos* berarti ilmu. Secara harafiah psikologi diartikan sebagal ilmu jiwa. Istilah *psyche* atau jiwa masih sulit didefinisikan karena jiwa itu merupakan objek yang bersifat abstrak, sulit dilihat wujudnya, meskipun tidak dapat dimungkiri keberadaannya. Dalam beberapa dasawarsa ini istilah jiwa sudah jarang dipakai dan diganti dengan istilah psikis [10].

Ada banyak ahli yang mengemukakan pendapat tentang psikologi, diantaranya:

1. Pengertian Psikologi menurut Ensiklopedi Nasional Indonesia Jilid 13, Psikologi adalah ilmu yang mempelajari perilaku manusia dan binatang baik yang dapat dilihat secara langsung maupun yang tidak dapat dilihat secara langsung.
2. Pengertian Psikologi menurut Dakir, psikologi membahas tingkah laku manusia dalam hubungannya dengan lingkungannya.
3. Pengertian Psikologi menurut Muhibbin Syah, psikologi adalah ilmu pengetahuan yang mempelajari tingkah laku terbuka dan tertutup pada manusia baik selaku individu maupun kelompok, dalam hubungannya dengan lingkungan. Tingkah laku terbuka adalah tingkah laku yang

bersifat psikomotor yang meliputi perbuatan berbicara, duduk , berjalan dan lain sebgainya, sedangkan tingkah laku tertutup meliputi berfikir, berkeyakinan, berperasaan dan lain sebagainya.

Dari beberapa definisi tersebut diatas dapat disimpulkan bahwa psikologi adalah ilmu pengetahuan yang mempelajari tingkah laku manusia, baik sebagai individu maupun dalam hubungannya dengan lingkungannya. Tingkah laku tersebut berupa tingkah laku yang tampak maupun tidak tampak, tingkah laku yang disadari maupun yang tidak disadari.

Dapat diketahui bahwa psikologi merupakan ilmu tentang tingkah laku. Pada hakekatnya tingkah laku manusia itu sangat luas, semua yang dialami dan dilakukan manusia merupakan tingkah laku. Semenjak bangun tidur sampai tidur kembali manusia dipenuhi oleh berbagai tingkah laku. Dengan demikian objek ilmu psikologi sangat luas.

2.5.1 Variabel Psikologi

Variabel psikologi terdiri dari :

1. Sikap

   Sikap adalah hal penting dalam psikologi. Sikap adalah kecenderungan bertindak terhadap rangsangan yang datang dari lingkungan sosial seorang subyek. Kecenderungan itu dapat bersifat positif yang berupa perilaku menerima obyek maupun negatif yang berupa perilaku menolak obyek.

2. Motivasi

   Motivasi adalah energi aktif yang menyebabkan terjadinya suatu perubahan pada diri sesorang yang nampak pada gejala kejiwaan, perasaan, dan juga emosi, sehingga mendorong individu untuk bertindak atau melakukan sesuatu dikarenakan adanya tujuan, kebutuhan, atau keinginan yang harus terpuaskan.

3. Kesabaran

   Yaitu proses yang berhasil dengan sukses dalam menyesuaikan diri dengan kesulitan atau pengalaman hidup yang menantang terutama dalam kesiapan mental, emosi dan perilaku yang fleksibel. Penyesuaian diri

terhadap tuntutan eksternal dan internal dan salah satu faktor yang memberikan kontribusi bagaimana orang bisa beradaptasi dengan kesulitan

4. Stress

   adalah merupakan suatu kondisi ketegangan yang mempengaruhi emosi, proses berpikir dan kondisi diri seseorang. Stress yang terlalu besar dapat mengancam kemampuan seseorang untuk menghadapi lingkungannya.

5. Kognitif

   Kognitif berasal dari kata “cognitive” yang berarti hal yang berhubungan dengan pengamatan. Dalam ilmu Psikologi, Kognitif merupakan bagian dari gejala jiwa manusia. Kognitif merupakan gejala pengenalan yang terdiri dari penghayatan pengamatan tanggapan asosiasi, ingatan, fantasi, berpikir dan intelegensi.

2.6 Tools yang Digunakan

2.6.1 Android

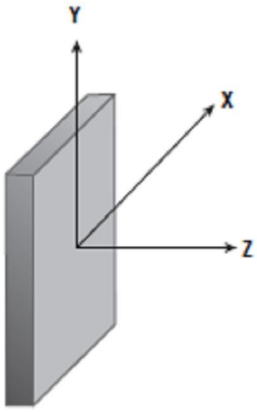
Android adalah sistem operasi yang digunakan di smartphone dan juga tablet PC. Fungsinya sama seperti sistem operasi Symbian di Nokia, iOS di Apple dan BlackBerry OS. Android tidak terikat ke satu merek *handphone* saja, beberapa vendor terkenal yang sudah memakai Android antara lain Samsung , Sony Ericsson, HTC, Nexus, Motorolla, dan lain-lain.

Android pertama kali dikembangkan oleh perusahaan bernama Android Inc., dan pada tahun 2005 di akuisisi oleh raksasa Internet Google. Android dibuat dengan basis kernel Linux yang telah dimodifikasi, dan untuk setiap release-nya diberi kode nama berdasarkan nama hidangan makanan. Keunggulan utama Android adalah gratis dan *open source*, yang membuat smartphone Android dijual lebih murah dibandingkan dengan Blackberry atau iPhone meski fitur (hardware) yang ditawarkan Android lebih baik. Beberapa fitur utama dari Android antara lain WiFi hotspot, *Multi-touch*, Multitasking, GPS, accelerometers, *support java*, mendukung banyak jaringan *(GSM/EDGE, IDEN, CDMA, EV-DO, UMTS,*

Bluetooth, Wi-Fi, LTE & WiMAX) serta juga kemampuan dasar handphone pada umumnya.

2.6.2 *Accelerometer*

Accelerometer, seperti namanya, digunakan untuk mengukur percepatan. Percepatan didefinisikan sebagai laju perubahan kecepatan, sehingga dapat mengukur seberapa cepat akselerasi pada perangkat berubah dalam arah tertentu. Dengan menggunakan *accelerometer*, anda dapat mendeteksi gerakan, laju perubahan dari kecepatan gerakan yang terjadi pada perangkat. Sangat penting untuk dicatat bahwa *accelerometer* tidak mengukur kecepatan, sehingga anda tidak bisa mengukur kecepatan secara langsung berdasarkan pembacaan *accelerometer* tunggal. Sebaliknya, anda perlu untuk mengukur perubahan dalam percepatan dari waktu ke waktu [10]. Percepatan dapat diukur oleh tiga sumbu arah : *forward-backward* (*longitudinal*), *left-right* (*literal*), *dan up-down* (*vertical*). Sensor manager melaporkan perubahan sensor di semua tiga sumbu arah. Seperti yang di ilustrasikan pada gambar di bawah ini :



Gambar 2.8 Tiga sumbu arah *accelerometer.*[10]

1. *Vertical* (*up-down*), dimana arah positif berarti ke atas gerakan perangkat yang sedang di tinggikan dan sebaliknya adalah nilai positif.

2. *Longitudinal* (*forward-backward*), dimana gerakan kedepan yang menandakan percepatan positif dan gerakan kebelakang yang menandakan percepatan negatif.
3. *Sideways lateral* (*left-right*), dimana gerakan kearah kanan dari perangkat yang mewakili nilai positif dan gerakan perangkat kearah kiri yang mewakili nilai negatif.

Sensor *manager* mempertimbangkan perangkat "diam" ketika menghadap ke atas pada permukaan yang datar dalam orientasi potret. Seperti dijelaskan sebelumnya, kita dapat memantau perubahan percepatan dengan menggunakan sensor *listener*. Untuk memanggil kelas *SensorListener* dari Sensor *Manager* yaitu dengan menggunakan *code SENSOR_ACCELEROMETER* yang nantinya dapat meminta pembaruan nilai-nilai *accelerometer* dan tingkat update sensor seperti potongan kode berikut :

```
SensorManager=
(SensorManager)getSistemService(Context.SENSOR_SERVICE);
sm.registerListener(mySensorListener,
SensorManager.SENSOR_ACCELEROMETER,
SensorManager.SENSOR_DELAY_UI);
```

Gambar 2.9 Potongan kode sensor *listener*

Sensor *Listener* yang kita buat harus menerapkan *onSensorChanged method* yang akan dipicu ketika perubahan terjadi dalam percepatan disepanjang salah satu dari tiga sumbu yang telah dijelaskan sebelumnya. *onSensorChanged method* menerima *array float* yang berisi nilai percepatan di ketiga sumbu dalam format baku. Sensor *Manager* mencakup konstanta indeks yang dapat kita gunakan untuk mengambil nilai percepatan yang dibutuhkan, seperti terlihat pada potongan kode di gambar 2.10 berikut :

```
SensorListener mySensorListener = new SensorListener() {
public void onSensorChanged(int sensor, float[] values) {
if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
float xAxis = values[SensorManager.DATA_X];
float yAxis = values[SensorManager.DATA_Y];
float zAxis = values[SensorManager.DATA_Z];
float raw_xAxis = values[SensorManager.RAW_DATA_X];
float raw_yAxis = values[SensorManager.RAW_DATA_Y];
float raw_zAxis = values[SensorManager.RAW_DATA_Z];
// TODO apply the acceleration changes to your application.
}
}
public void onAccuracyChanged(int sensor, int accuracy) { }

};
```

Gambar 2.10 Potongan kode Sensor *Manager*

2.6.3 OOP (*Object Oriented Programming*)

Java Merupakan salah satu bahasa pemrograman yang berorientasi objek. OOP Atau *Object Oriented Programing* Adalah merupakan model pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam model pemrograman ini dibungkus dalam objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Model pemrograman ini merupakan model pemrograman yang mudah untuk dikembangkan dan dikelola.

Ada berbagai macam konsep yang dipakai dalam model pemrograman berbasis OOP diantaranya : Kelas, Objek, Abstrak, Polimorfisme, Enkapsulasi Dalam kesempatan ini kita akan pelajari Kelas, Objek serta beberapa macam akses modifier.

1. Kelas

   Kelas adalah sekumpulan data dan fungi dalam satu kesatuan untuk tujuan tertentu. Dalam contoh misalnya ada kelas “pensil” dimana kelas pensil ini merupakan turunan dari kelas alat_tulis, perhatikan kode berikut :

```
public class alat_tulis{protected public class pensil extends
alat_tulis{
String nama;
protected String harga;
String harga(String hrg){ merek; harga=hrg; String nama(String
nm){ return harga; nama=nm; }
return nama;
public static void main (String[]args) {}
pensil pl=new pensil();
String merek(String mrk){
pl.nama=”Pensil”;
merek=mrk; pl.merek=”Standart”;
return merek; pl.harga=”5000”;
}     System.out.println(“Nama “+pl.nama);
}           System.out.println(“Merek “+pl.merek);
             System.out.println(“Harga “+pl.harga);
             }
             }
```

Gambar 2.11 *Pseudo-code* kelas alat_tulis [10]

Dari kode tersebut bisa anda lihat kelas pensil bias memberikan nilai pada nama, merek padahal didalam kelas tersebut tidak mempunyai variabel itu, hal ini dikarenakan kelas pensil tersebut merupakan kelas turunan dari kelas alat tulis.

2. Objek

Objek bisa kita ibaratkan sebagai pembungkus suatu fungsi,data yang ada dalam pemrograman berorientasi objek. Contohnya adalah setiap komponen, variabel, fungsi di dalam java selalu diwakili oleh sebuah objek, perhatikan kode berikut :

Variabel : int[]angka=new int[5] ; angka merupakan objek variabel

Kelas : pensil pl =newm pensil(); pl merupakan objek kelas

3. Akses Modifier

Akses modifier merupakan hak akses yang diberikan kepada data/variabel, fungsi atau kelas dalam pemrograman berorientasi objek. Ada beberapa macam akses modifier yang harus anda ketahui diantaranya :

1. *Public* : Akses modifier yang mengijinkan semua kelas mengakses var,fungsi yang memliki tipe akses modifier ini.
2. *Private* : Akses modifier yang hanya bias diakses dalam satu kelas
3. *Protected* : Akses modifier yang hanya mengijinkan kelas turunan untuk mengakses var,fungsi yang memliki tipe akses modifier ini.
4. *Static* : Bukan akses modifier, Sebuah variabel atau fungsi yang mempunyai nilai kembalian yang nilaix akan tetap terjaga selama tidak ada perubahan nilai dalam var, dan fungsi tersebut selama aplikasi berjalan.

2.6.4 Java

Java adalah bahasa pemrograman berorientasi objek murni yang dibuat berdasarkan kemampuan-kemampuan terbaik bahasa pemrograman objek sebelumnya (C++, Ada, Simula). Java diciptakan oleh James Gosling, developer dari Sun Microsystems pada tahun 1991.

Karakteristik Java antara lain :

1. Sederhana (*Simple*)
2. Berorientasi Objek (*Object Oriented*)
3. Terdistribusi (*Distributed*)
4. *Interpreted*
5. *Robust*
6. Aman (*Secure*)
7. *Architecture Neutral*
8. *Portable*
9. *Performance*
10. *Multithreaded*
11. Dinamis

Macam-macam Java 2 *Software Developer Kit* (J2SDK) antara lain :

1. J2SE (Java 2 Standard Edition)
2. J2EE (Java 2 Enterprise Edition)
3. J2ME (Java 2 Micro Edition)

2.6.5 UML

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

UML menyediakan 10 macam diagram yang dapat digunakan untuk memodelkan aplikasi berorientasi objek yang dibuat, yaitu :

1. *Use Case Diagram* untuk memodelkan proses bisnis.
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence Diagram* untuk memodelkan pengiriman pesan (*message*) antar *objects*.
4. *Collaboration Diagram* untuk memodelkan interaksi antar *objects*.
5. *State Diagram* untuk memodelkan perilaku *objects* di dalam sistem.
6. *Activity Diagram* untuk memodelkan perilaku *Use Cases* dan *objects* di dalam *system*.
7. *Class Diagram* untuk memodelkan struktur kelas.
8. *Object Diagram* untuk memodelkan struktur *object*.
9. *Component Diagram* untuk memodelkan komponen *object*.
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

2.6.6 *Use Case Diagram*

Use case diagram menggambarkan tujuan penting fungsionalitas *sistem* yang akan ditampilkan pada user. Sebua use case merepresentasikan sebuah interaksi antara actor dengan sistem. *Use case* diagram dapat membantu dalam

penyusunan *requirement* sebuah *sistem*, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua fitur yang ada pada system [10].

2.6.7 *Class Diagram*

Class diagram digunakan untuk memperlihatkan *class* pada sistem dan hubungan antara *class*. *Class* diagram tidak menjelaskan fungsionalitas atau bagaimana *instances class* berinteraksi. *Class* menggambarkan keadaan (*atribut*/*property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (*method*/fungsi) [10].

2.6.8 *Activity Diagram*

Activity diagram digunakan untuk menggambarkan aliran aktivitas baik dalam proses bisnis maupun *use case*. Diagram ini juga dapat digunakan untuk memodelkan aksi yang akan dilakukan saat sebuah operasi dieksekusi dan memodelkan hasil dari aksi tersebut [10].

2.6.9 *Sequence Diagram*

Sequence diagram biasanya digunakan untuk menggambarkan skenario atau rangkaian lengkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan *output* tertentu. Diawali dari apa yang mend-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk *actor*, memiliki *lifeline vertical*. *Message* digambarkan sebagai garis berpanah dan satu objek ke objek lainnya. Pada fase desain berikutnya *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* [10].

2.6.10 *Photoshop*

Photoshop adalah software yang digunakan untuk memodifikasi gambar atau foto secara profesional baik meliputi modifikasi obyek yang sederhana maupun yang sulit sekalipun. Photoshop merupakan salah satu software yang berguna untuk mengolah gambar berbasis bitmap, yang mempunyai tool dan efek yang lengkap sehingga dapat menghasilkan gambar atau foto yang berkwalitas

tinggi. Kelengkapan fitur yang ada di dalam Photoshop inilah yang akhirnya membuat *software* ini banyak digunakan oleh desainer grafis profesional. Dan mungkin juga sampai saat ini masih belum ada software desain grafis lain yang bisa menyamai kelengkapan fitur dalam Photoshop.