BAB 2
TINJAUAN PUSTAKA

2.1 Game

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

2.1.1 Pengertian Game

Game merupakan sebuah bentuk seni dimana para penggunanya disebut dengan pemain (*player*), seorang pemain 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.

2.1.2. Klasifikasi *Game*

2.1.2.1. Berdasarkan Jenis Alat Yang Digunakan

Ada beberapa jenis alat di dunia *game* yang selalu dipilih oleh pengguna *game*, yaitu [5]:

1. *Arcade games*, yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah atau tempat khusus dan memiliki *box* atau mesin yang memang khusus di design untuk jenis *game* 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 Computer*.
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 *game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.
5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA.

2.1.2.2. Berdasarkan Sudut Pandang Permainan

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

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

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

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*).

2.1.2.3. Berdasarkan Genre Permainannya

1. *Action Game*

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.

2. *Adventure Game*

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.

3. *Fighting Game*

Fighting Game adalah tipe *game* yang intinya harus menjatuhkan lawan tandingnya, entah itu dengan pukulan, tendangan, combo, maupun dengan jurus special.

4. *Shooting Game*

Shooting (tembak-tembakan, atau hajar-hajaran bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya). Video game jenis ini sangat memerlukan

kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari game jenis ini adalah tembak, tembak dan tembak.

5. RPG (*Role Playing Game*)

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.

6. *Puzzle Game*

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 kecerdasan buatan.

7. *Racing Game*

Racing Game adalah tipe *game* yang tujuannya adalah mencapai garis *finish* dari suatu *race*, dalam *game* ini biasanya pemain dapat memilih dan membeli kendaraan, memodifikasi, melakukan *upgrade* mesin.

8. *Simulation Game*

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.

9. RTS (*Real Time Strategy*)

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.

10. *Sport Game*

Sport Game adalah tipe *game* yang berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau tim. Contoh *game* tipe ini antara lain sepakbola, bola basket, tenis, dan bilyard.

11. *Casual Game*

Casual Game adalah suatu game yang permainannya tidak kompleks, mainnya rileks dan sangat mudah untuk dipelajari (bahkan cenderung langsung bisa dimainkan).

2.1.2.4. Berdasarkan Batasan Usia

Berikut ini adalah klasifikasi permainan berdasarkan pembatasan usia dari pemainnya, menurut ESRB (*Entertainment Software Rating Board*) adalah sebagai berikut [6]:

1. *Early Childhood* (3+) memiliki isi yang mungkin cocok untuk usia 3 keatas. Tidak mengandung materi yang tidak pantas seperti adegan kekerasan.
2. *Everyone* (6+) memiliki konten yang mungkin cocok untuk usia 6 keatas. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan tidak jarang menggunakan bahasa yang ringan.
3. *Everyone* (10+) memiliki konten yang mungkin cocok untuk usia 10 keatas. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi atau kekerasan ringan, bahasa ringan dan juga minimal tema sugestif.
4. *Teen* (13+) memiliki konten yang mungkin cocok untuk usia 13 keatas. Judul-judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, sedikit darah, simulasi perjudian, dan tidak jarang menggunakan bahasa yang kuat.
5. *Mature* (17+) memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi intens kekerasan, darah, konten seksual dan bahasa yang kuat.

6. *Adults Only* (18+) memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun keatas. Judul-judul dalam kategori ini mungkin termasuk adegan yang intens, kekerasan dan juga mengandung konten seksual.

2.2 Kecerdasan Buatan *(Artificial Intelligence)*

2.2.1 Definisi AI

Sebagian kalangan menerjemahkan *Artificial Intelligence* sebagai kecerdasan buatan, kecerdasan artifisial, intelijensia artifisial, atau intelejensia buatan. Pada bukunya[9], Suyanto sengaja tidak menerjemahkan istilah *Artificial Intelligence* ke bahasa Indonesia karena istilah tersebut sudah sangat akrab bagi orang Indonesia. Begitu juga dengan singkatan istilah tersebut, yaitu AI, sudah sangat melekat di berbagai media ilmiah maupun non ilmiah.

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. *Stuart Russel* dan *Peter Norvig* mengelompokkan definisi AI, yang diperoleh dari beberapa *textbook* berbeda, ke dalam empat kategori[9], yaitu :

a. ***Thinking humanly*** : *the cognitive modeling approach*

   Pendekatan ini dilakukan dengan dua cara sebagai berikut :

   1. Melalui introspeksi : mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir. Tetapi, seorang psikolog Barat mengatakan *"how do you know that you understand?"* Bagaimana anda tahu bahwa anda mengerti? Karena pada saat anda menyadari pemikiran anda. Sehingga definisi ini terkesan mengada-ngada dan tidak mungkin dilakukan.
   2. Melalui eksperimen-eksperimen psikologi.

b. ***Acting humanly*** : *the Turing test approach*

   Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berintelijensia untuk menguji apakah komputer tersebut mampu mengelabui seorang manusia yang menginterogasinya melalui *teletype* (komunikasi berbasis

teks jarak jauh). Jika *interrogator* tidak dapat membedakan yang diinterogasi adalah manusia atau komputer, maka komputer berintelijensia tersebut lolos dari *Turning test*. Komputer tersebut perlu memiliki kemampuan *Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics*. *Turing test* sengaja menghindari interaksi fisik antara *interrogator* dan komputer karena simulasi fisik manusia tidak memerlukan intelijensia.

c. ***Thinking rationally*** : *the laws of thought approach*

Terdapat dua masalah dalam pendekatan ini, yaitu :

1. Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam *formal term* yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.
2. Terdapat perbedaan besar antara dapat memecahkan masalah “dalam prinsip” dan memecahkannya “dalam dunia nyata”.

d. ***Acting rationally*** : *the rational agent approach*

Membuat inferensi yang logis merupakan bagian dari suatu *rational agent.* Hal ini disebabkan satu-satunya cara untuk melakukan aksi secara rasional adalah dengan menalar secara logis. Dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa aksi yang diberikan akan mencapai tujuan atau tidak. Jika mencapai tujuan, maka *agent* dapat melakukan aksi berdasarkan kesimpulan tersebut.

Thinking humanly dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni reflek dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika menggunakan definisi ini, maka banyak produk komputasi cerdas saat ini yang tidak layak disebut sebagai produk AI.

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally*. Oleh karena itu, definisi AI yang paling tepat untuk saat ini adalah *acting rationally* dengan pendekatan *rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut.

2.2.2 Teknik Pemecahan Masalah AI

Terdapat empat teknik dasar pemecahan masalah yang terdapat pada bidang *academic/traditional artificial intelligence* (AI)[9], diantaranya adalah sebagai berikut :

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. Langkah terakhir adalah memilih metode pencarian yang tepat sehingga dapat menemukan solusi terbaik.

Metode-metode pencarian pada teknik *searching* diantaranya[9] :

1. *Blind/Un-informed Search*
    a. *Breadth-First Search* (BFS)
    b. *Depth-First Search* (DFS)
    c. *Depth-Limited Search* (DLS)
    d. *Uniform Cost Search* (USC)
    e. *Iterative-Deepening Search* (IDS)
    f. *Bi-Directional Search* (BDS)
2. Metode Pencarian Heuristik
    a. *Generate-and-Test*
    b. *Hill Climbing*
    c. *Simulated Annealing*

d. *Best-First Search* (BFS)

e. *Greedy Best-First Search*

f. *A** (A *star*)

b. *Reasoning*

Teknik *reasoning* atau penalaran merupakan teknik penyelesaian masalah dengan cara merepresentasikan masalah ke dalam basis pengetahuan (*knowledge base)* 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, *user* harus mengetahui aturan yang berlaku untuk sistem yang akan dibangunnya. Tetapi, pada masalah tertentu terkadang suatu aturan tidak bisa didefinisikan secara benar ataupun lengkap. 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 pernah diketahui dapat ditemukan.

2.2.2.1 Algoritma A* (*A Star Algorithm*)

Algoritma A* (A Star) merupakan perbaikan dari metode BFS dengan memodifikasi fungsi heuristiknya. A* (A Star) akan meminimumkan total biaya lintasan. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal. Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS hanya saja dengan dua faktor tambahan yaitu :

1. Setiap sisi mempunyai *cost* yang berbeda-beda, seberapa *cost* untuk pergi dari satu simpul ke simpul lain.
2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian sehingga lebih kecil kemungkinan mencari ke arah yang salah.

Cost untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila ingin mencari jalan dengan waktu tercepat untuk dilalui. Algoritma A* bekerja dengan prinsip yaitu :

1. Simpul-simpul di *list* terbuka diurutkan oleh *cost* keseluruhan dari simpul awal ke simpul tujuan, dari *cost* terkecil sampai *cost* terbesar.*Cost* keseluruhan dihitung dari *cost* dari simpul awal ke simpul sekarang (*current node*) ditambah *cost* perkiraan menuju simpul tujuan.
2. Simpul di *list* tertutup bisa dimasukkan ke *list* terbuka bila jalan terpendek menuju simpul tersebut ditemukan.

Cost antara simpul adalah jaraknya dan perkiraan *cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan. Untuk lebih mudah dimengerti,berikut rumusnya.

$$f(n) = g(n) + h(n) \qquad (2.1)$$

keterangan :

f(n) = fungsi evaluasi

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

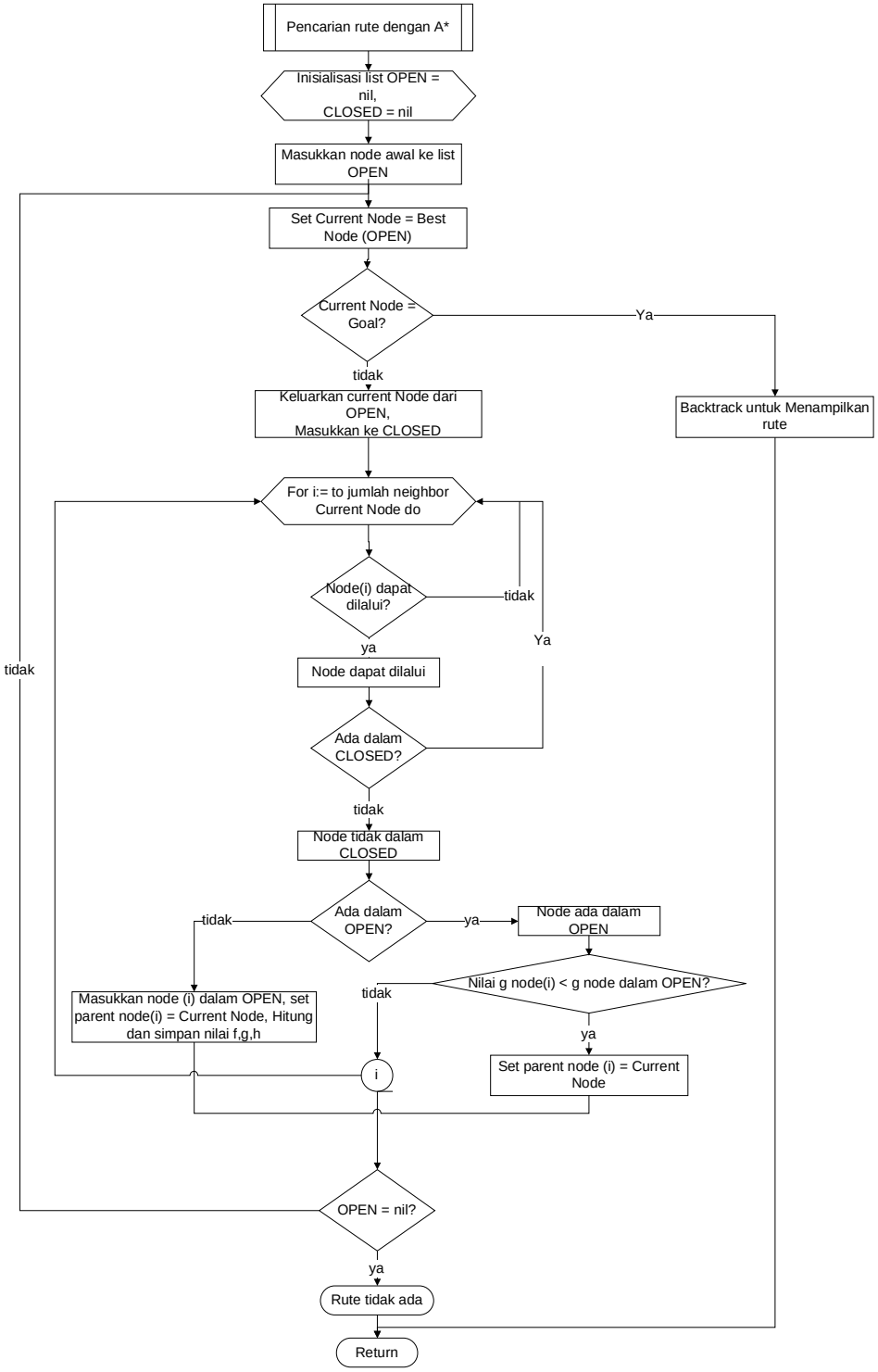
h(n) = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Node dengan nilai f(n) terendah merupakan solusi terbaik untuk diperiksa pertama. Dengan fungsi *heuristic* yang memenuhi kondisi tersebut maka pencarian dengan algoritma A* dapat optimal. Keoptimalan dari A* cukup langsung dinilai optimal jika h(n) adalah *admissible heuristic* yaitu nilai h(n) tidak akan memberikan penilaian lebih pada *cost* untuk mencapai tujuan. Salah satu contoh dari *admissible heuristic* adalah jarak dengan menarik garis lurus karena jarak terdekat dari dua titik adalah dengan menarik garis lurus.

Algortima A* secara ringkas langkah demi langkahnya adalah sebagai berikut:

1. Tambahkan *starting point* ke dalam *openset*.
2. Ulangi langkah berikut:
    a. Carilah biaya f terendah pada setiap simpul dalam *openset*. *Node* dengan biaya f terendah kemudian disebut *current node*.
    b. Masukkan ke dalam *closedset*.
    c. Untuk setiap 8 simpul (*neighbor node*) yang berdekatan dengan *currentnode*:
        1. Jika tidak *walkable* atau jika termasuk *closedset*, maka abaikan.
        2. Jika tidak ada pada *openset*, tambahkan ke *openset*.
        3. Jika sudah ada pada *openset*, periksa apakah ini jalan dari simpul ini ke *current node* yang lebih baik dengan menggunakan biaya g sebagai ukurannya. Simpul dengan biaya g yang lebih rendah berarti bahwa ini adalah jalan yang lebih baik. Jika demikian, buatlah simpul ini (*neighbor node*) sebagai *camefrom* dari *current node*, dan menghitung ulang nilai g dan f dari simpul ini.
    d. *Stop* Ketika :
        1. Menambahkan *targetpoin* ke dalam *closedset*, dalam hal ini jalan telah ditemukan, atau,
        2. Gagal untuk menemukan *targetpoin*. Dan *openset* kosong. Dalam kasus ini, tidak ada jalan.
        3. Simpan jalan. Bekerja mundur dari *targetpoin*, pergi dari masing-masing simpul ke simpul *camefrom* sampai mencapai *startingpoin*.

Secara umum dapat digambarkan dengan *flowchart* sebagai berikut:



Gambar 2.1 *Flowchart* Algoritma A*

2.3 *Unified Modelling Language* (UML)

Pada perkembangan teknik pemrograman berorientasi objek, muncul sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling* Language (UML). Adapun pengertian dari UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek[4].

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. Dalam hal ini UML merupakan suatu bahasa visual untuk melakukan pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

2.3.1 Diagram UML

UML menggunakan berbagai macam diagram dengan fungsi masing-masing untuk menggambarkan setiap proses dari sistem berorientasi objek. Berikut merupakan beberapa diagram UML diantaranya[4] :

a. ***Use Case Diagram***

Use Case atau diagram *use case* merupakan pemodelan yang digunakan untuk menggambarkan kelakuan (*behavior)* dari sistem yang akan dibuat[4]. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*[4].

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesar antarunit atau aktor.

b. ***Activity Diagram***

Diagram aktivitas atau *activity diagram* adalah sebuah diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis[4]. Dalam diagram aktivitas yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktivitas sistem, bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut[4] :

1. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* di mana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

C. ***Class Diagram***

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi[4].

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

d. ***Sequence Diagram***

Diagram sekuen adalah diagram yang menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antarobjek[4]. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambarkan adalah sebanyak pendefinisian *use case* yang memiliki proses itu sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

2.4 *Microsoft Visio*

Microsoft Visio (atau sering disebut Visio) adalah sebuah program aplikasi komputer yang sering digunakan untuk membuat diagram, diagram alir (*flowchart*), *brainstorm*, dan skema jaringan yang dirilis oleh *Microsoft* Corporation. Aplikasi ini menggunakan grafik vektor untuk membuat diagram-diagramnya.Visio aslinya bukanlah buatan *Microsoft Corporation*, melainkan buatan *Visio Corporation*, yang diakusisisi oleh Microsoft pada tahun 2000.

2.5 Construct 2

Construct2 adalah produk buatan Scirra, perusahaan yang berasal dari kota London, Inggris. Dengan menggunakan *game engine* ini, entah itu *developer* pemula maupun *expert* sangat dimanjakan dengan kemudahan pemakaian. Cukup membaca beberapa manual di situs resmi Scirra.

Game builder ini sebenarnya dirancang untuk *game* berbasis 2D. Dengan menggunakan Construct 2, pengembang permainan dapat mem-*publish*nya ke beberapa *platform* seperti:

1. HTML5 website
2. Google Chrome Webstore
3. Facebook
4. Phonegap (Android)
5. Windows Phone 8, bahkan Windows 8.

Untuk mempercantik *game*, disediakan pula 70 visual effect yang menggunakan *engine* WebGL. Construct 2 juga dilengkapi dengan 20 built-in *plugin* dan *behaviours* (perilaku objek) serta dapat menambah *sprite* animasi, objek teks, mengkoneksikan ke Facebook, menambah musik, manipulasi penyimpanan data *game*, dan penambahan efek-efek grafis layaknya Adobe Photoshop.

Pemanggilan fungsi-fungsi di Construct 2 hanya menggunakan pengaturan *Events* yang telah disediakan. *Events* merupakan pilihan-pilihan *Action* dan kondisi yang akan menjadi nyawa dalam *game* dan *game* dapat berjalan sesuai keinginan. Karena berbasis HTML5, *preview* saat ingin menjajal *game* pun hanya memanfaatkan *browser* (*localhost*).

2.6 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah jaminan kualitas dari elemen kritis perangkat lunak dan merepresentasikan review akhir dari spesifikasi, desain, dan pembuatan kode[14]. Pengujian perangkat lunak juga memberikan pandangan mengenai perangkat lunak secara obyektif dan independen, yang bermanfaat pada operasional bisnis untuk memahami tingkat risiko pada implementasinya. Teknik-teknik pengujian mencakup, tetapi tidak terbatas pada proses mengeksekusi suatu bagian program atau keseluruhan aplikasi dengan tujuan untuk menemukan kesalahan perangkat lunak. Pada umumnya terdapat 2 pengujian yaitu :

1. Pengujian Alpha

Pengujian alpha dilakukan pada sisi pengembang yang dianggap pemakai dan merekam semua kesalahan dan masalah pemakaian. Pengujian alpha dilakukan

pada sebuah lingkungan yang terkendali. Tahap pertama memulai pengujian produk. Pada fase ini produk masih dalam tahap pengujian dan belum sampai pada publik.

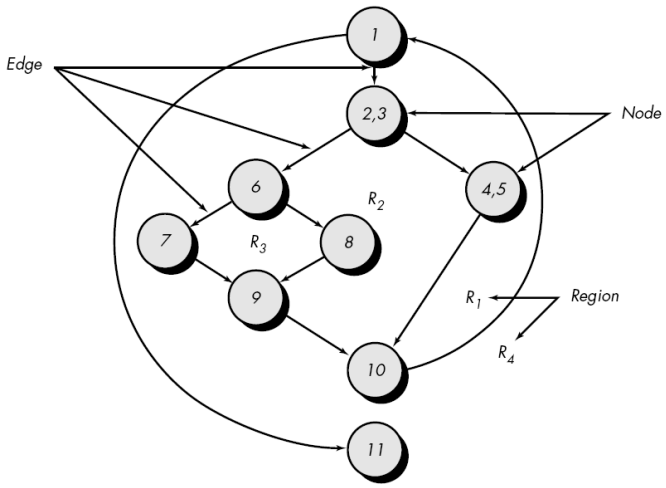
Terdapat 2 macam pendekatan pengujian, yaitu sebagai berikut:

a. White Box Testing

Pengujian white box, kadang disebut juga pengujian glass box, merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk memperoleh uji kasus. Dengan menggunakan metode pengujian white box, perekayasa sistem dapat melakukan test case untuk memberikan jaminan bahwa :

1. Semua jalur independen pada suatu modul ditelusuri minimal 1 kali.
2. Semua jalur keputusan logis True/False dilalui.
3. Semua loop dieksekusi pada batas yang tercantum dan batas operasionalnya.
4. Struktur data internal digunakan agar validitas terjamin.

Pengujian white box dapat dilakukan dengan pengujian *basis path*, metode ini merupakan salah satu teknik pengujian struktur kontrol untuk menjamin semua statemen dalam setiap jalur independen program dieksekusi minimal 1 kali. Perhitungan jalur independen dapat dilakukan melalui metrik *Cyclomatic Complexity*. Sebelum menghitung nilai *Cyclomatic Complexity*, terlebih dahulu diterjemahkan desain prosuderal ke bagan alir(*flow chart*), kemudian dibuat *flow graph*.



Gambar 2.2 Contoh *Flow Graph*[14]

Cyclomatic complexity digunakan untuk mencari jumlah path dalam satu flow graph. Dapat dipergunakan rumusan sebagai berikut :

1. Cyclomatix complexity V(G) untuk grafik alir dihitung dengan rumus:

$$V(G) = E - N + 2 \qquad (2.2)$$

Dimana:

E = jumlah edge pada grafik alir

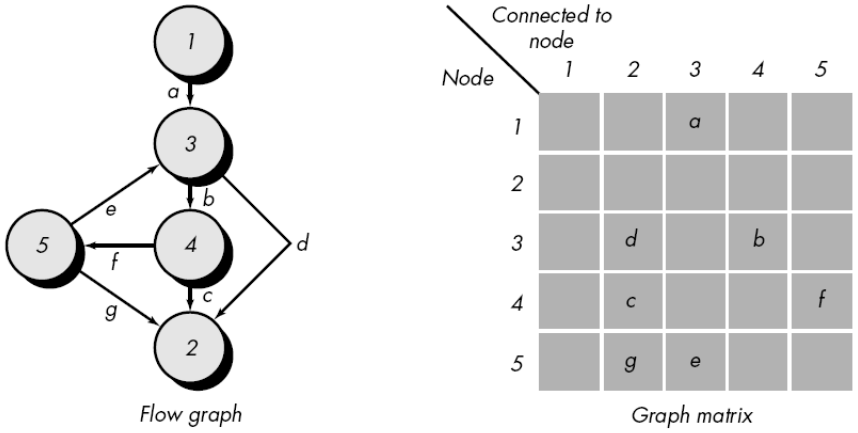
N = jumlah node pada grafik alir

2. Cyclomatix complexity V(G) juga dapat dihitung dengan rumus:

$$V(G) = P + 1 \qquad (2.3)$$

Dimana P = jumlah predicate node pada grafik alir

Untuk mengembangkan perangkat lunak yang membantu dalam pengujian basis path, *graph matrix* dapat sangat berguna. Graph matrix adalah matriks persegi yang terdiri dari jumlah baris dan kolom yang sama dengan jumlah node pada flow graph. Setiap baris dan kolom sesuai dengan identifikasi simpul, dan entri matriks sesuai dengan koneksi tepi antara node.



Gambar 2.3 Contoh *Graph Matrix*[14]

b. *Black Box Testing*

Pengujian black box, juga disebut pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak. Pengujian black-box memungkinkan

perekayasa perangkat lunak untuk mendapatkan set kondisi input yang akan sepenuhnya melaksanakan semua persyaratan fungsional untuk suatu program. Pengujian black-box bukan merupakan alternatif untuk teknik white-box. Sebaliknya, merupakan pendekatan komplementer yang kemungkinan akan mengungkap kelas yang berbeda dari kesalahan daripada metode white box. Pengujian pada Black Box berusaha menemukan kesalahan seperti:

a. Fungsi-fungsi yang tidak benar atau hilang.
b. Kesalahan *interface*.
c. Kesalahan dalam struktur data atau akses database eksternal.
d. Kesalahan kinerja.
e. Inisialisasi dan kesalahan terminasi.

2. Pengujian Beta

Pengujian beta digunakan untuk preview dalam sebuah produk perangkat lunak. pengujian beta hampir sama dengan pengujian alpha, namun pada tahap ini produk sudah didistribusikan ke publik. Pengujian beta juga bermanfaat Untuk mengetahui tanggapan dan penilaian pengguna atau publik terhadap produk perangkat lunak. Untuk mendapatkan tanggapan dari publik dapat menggunakan kuesioner, angket, poling. Setelah sampel terkumpul maka dilakukan pengukuran menggunakan skala pengukuran. Skala pengukuran yang digunakan adalah skala interval.

Skala interval merupakan skala pengukuran yang banyak digunakan untuk mengukur fenomena atau gejala sosial, dimana pihak responden diminta melakukan rangking terhadap preferensi tertentu sekaligus memberikan nilai terhadap preferensi tersebut. Jenis skala yang dapat digunakan untuk penelitian sosial, adalah skala *Likert*[13].

Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei. Nama skala ini diambil dari nama Rensis Likert, yang menerbitkan suatu laporan yang menjelaskan penggunaannya. Sewaktu menanggapi pertanyaan dalam skala Likert, responden menentukan tingkat persetujuan mereka terhadap suatu

pernyataan dengan memilih salah satu dari pilihan yang tersedia. Biasanya disediakan lima pilihan skala dengan format seperti:

1. Sangat Setuju
2. Setuju
3. Cukup Setuju
4. Kurang Setuju
5. Tidak Setuju

Perhitungan skala *likert* adalah sebagai berikut :

$$P = \frac{S}{skorideal} \times 100\% \quad (2.4)$$

Keterangan :

P = Nilai presentase yang dicari

S = Jumlah frekuensi dikalikan dengan nilai yang ditetapkan jawaban

skorideal = Nilai tertinggi dikalikan dengan jumlah sampel

100% = Bilangan tetap