BAB 2
TINJAUAN PUSTAKA

2.1 Game

Game berasal dari kata bahasa inggris yang memiliki arti dasar Permainan. Permainan dalam hal ini merujuk pada pengertian "kelincahan intelektual" *(intellectual playability).* Game juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target-target yang ingin dicapai pemainnya. Sedangkan Agustinus Nilwan menjelaskan bahwa *game* di artikan sebagai suatu aktivitas tersetruktur atau juga digunakan sebagai alat pembelajaran.

Secara umum teori permainan dapat didefinisikan sebagai sebuah pendekatan terhadap kemungkinan strategi yang akan dipakai, yang disusun secara matematis agar bisa diterima secara logis dan rasional. Serta digunakan untuk mencari strategi terbaik dalam suatu aktivitas, dimana setiap pemain didalamnya sama - sama mencapai utilitas tertinggi[6].

2.2 Jenis Game

Terdapat banyak jenis *game* yang ada saat ini, diantaranya jenis *game* berdasarkan platform yang digunakan dan jenis *game* berdasarkan genre permainannya. Berikut uraian dari masing – masing jenis *game* tersebut.

2.2.1 Berdasarkan Jenis Platform yang Digunakan

Jenis *game* berdasarkan platform yang digunakan adalah sebagai berikut:

1. *Arcades Game*

   Arcade games atau kalau di Indonesia sering disebut dengan ding-dong, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang khusus didesain untuk jenis video games tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa "masuk" dan "menikmati", seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya).

2. *PC Games*

PC games adalah video games yang dimainkan menggunakan *Personal Computers*.

3. *Console games*

Console games adalah video games yang dimainkan menggunakan *console* tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.

4. *Handheld games*

Handheld games adalah video games yang dimainkan di *console* khusus video game yang dapat dibawa kemana-mana atau disebut juga dengan portable console, contohnya seperti Nintendo DS dan Sony PSP.

5. *Mobile games*

Mobile games adalah video game yang dimainkan dalam *mobile phone*, *smartphone*, PDA atau *handheld computer*, tidak termasuk yang dimainkan dengan *handheld video game* seperti Playstation Portable atau Nintendo DS. *Mobile game* tidak hanya dapat dimainkan melalui telepon selular, namun dapat dikembangkan dalam berbagai macam *mobile handset* seperti PDA, *Symbian OS* dan *Microsoft's Smartphone*. *Mobile game* dimainkan dengan menggunakan teknologi yang ada di dalam alat itu sendiri, contohnya *software game* yang telah ada dalam telepon selular itu sendiri. Untuk *game* yang dimainkan secara online, terdapat beberapa teknologi yang umum digunakan, misalnya dengan pesan teks (SMS), pesan multimedia (MMS) atau GPRS untuk mengidentifikasi lokasi video *games* yang dapat dimainkan atau khusus dimainkan untuk mobile phone atau PDA.

2.2.2 Berdasarkan Genre Permainannya

Jenis *game* berdasarkan genre permainannya adalah sebagai berikut:

1. *RTS (Real Time Strategy)*

Real Time Strategy adalah *genre* suatu permainan komputer yang memiliki ciri khas berupa permainan perang yang tiap pemainnya memiliki suatu negara, negara tersebut di kelola dalam hal pengumpulan sumberdaya (alam, manusia,

ekonomi), pengaturan strategi pasukan-pasukan tempur, diplomasi dengan negara tetanga, peningkatan ekonomi, pengembangan keyakinan, pengembangan pendidikan dari negara primitif menuju peradaban moderen. *RTS* dibedakan dari *turn-based strategi* dimana dalam *RTS* permainan tidak mengenal giliran. Setiap pemain dapat mengatur/memerintah pasukannya dalam waktu apapun. Dalam *RTS*, tema permainan dapat berupa sejarah (misalnya seri *Age of Empires*), fantasi (misalnya *Warcraft*) dan fiksi ilmiah (misalnya *Star Wars*).

2. FPS (First Person Shooter)

Jenis permainan tembak-menembak dengan tampilan pada layar pemain adalah sudut pandang tokoh karakter yang dimainkan, tiap tokoh karakter mempunyai kemampuan yang berbeda-beda dalam tingkat akurasi menembak, reflek menembak,dll. Permainan ini dapat melibatkan banyak orang. Permainan ini bisa berupa misi melumpuhkan penjahat maupun alien, kadang juga sejumlah pemain dibagi beberapa tim yang bertugas melumpuhkan tim lainnya, sebelum dilumpuhkan. Ciri utama lain adalah penggunaan senjata genggam jarak jauh.

Contoh : Duke Nukem 3D, Quake, Blood, Unreal, Unreal Tournament, seri Half-Life, Counter-Strike, seri Halo, Perfect Dark, TimeSplitters, Call of Duty, System Shock, dan GoldenEye 007.

3. RPG (Role Playing Game)

RPG *(Role Playing Game)* adalah sebuah permainan yang para pemainnya memainkan peran tokoh-tokoh khayalan dan berkolaborasi untuk merajut sebuah cerita bersama. Para pemain memilih aksi tokok-tokoh mereka berdasarkan karakteristik tokoh tersebut, dan keberhasilan aksi mereka tergantung dari sistem peraturan permainan yang telah ditentukan. Asal tetap mengikuti peraturan yang ditetapkan, para pemain bisa berimprovisasi membentuk arah dan hasil akhir permainan ini.

Contoh : World of Warcraft, The Lord of the Rings Online: Shadows of Angmar, Final Fantasy, Ragnarok, DoTA.

4. *Life Simulation Games*

Permainan simulasi kehidupan ini meliputi kegiatan individu dalam sebuah tokoh karakter. Dalam memainkan tokoh karakter tersebut pemain bertanggung jawab atas inteligen serta kemampuan fisik dari tokohnya tersebut. Tokoh karakter tersebut memerlukan kebutuhan layaknya manusia seperti kegiatan belajar, bekerja, belanja, bersosialisasi, memelihara hewan, memelihara lingkungan dan lain-lain. Lawan mainnya bisa berupa pemain lain yang memainkan karakter sebagai tetangga maupun komputer dengan kecerdasan buatan tingkat tinggi.

Contoh: SimLife, Second Life.

5. *Construction and Management Simulation Games*

Permainan yang mensimulasikan proyek membangun sebuah kota, pemain di haruskan membangun sebuah kota lengkap dengan fasilitas publik maupun fasilitas pemerintah seperti gedung, alat transportasi publik, taman, sekolah, rumah sakit, tempat beribadah, pabrik, bandara, stasiun, bank dan bangunan lainnya lainnya. saat membangun kota tersebut pemain juga harus memperhatikan sumber daya ekonomi, kenyamanan para penduduknya dalam beraktifitas yang mungkin terganggu saat pembuatan jembatan penyebrangan dan lain-lain.

Contoh: SimCity, Caesar.

6. *Vehicle Simulation*

Jenis permainan ini mensimulasi pengoperasian beberapa kendaraan, kendaraan bisa berupa pesawat terbang, pesawat tempur, kereta, kendaraan perang, maupun kendaraan konstruksi.

Contoh: FlightGear, Tram, Orbiter.

7. *Action Game*

Permainan jenis ini sangat berkaitan dengan tantangan fisik, seperti ketangkasan, reflek dari pemain. Dalam permainan ini pemain mengendalikan seorang tokoh karakter, tokoh karakter yang dimainkan bisa dihadapkan

dengan tokoh karakter lain yang bertarung berhadapan bisa juga menjalankan sebuah misi yang banyak rintangan, mengumpulkan objek tertentu, mengalahkan musuh, maupun menyelamatkan karakter lainnya.

Contoh: Street Fighter.

8. Adventure Game

Game petualangan adalah sebuah genre permainan dalam video *game* dimana pemain berperan sebagai protagonis dalam sebuah cerita interaktif diiringi berbagai teka-teki untuk dipecahkan. Hampir seluruh *game* dengan genre ini adalah permainan *single player* karena lebih menekankan dari segi sudut cerita dan karakternya, membuat desain *multi player* untuk *game* petualangan ini sulit dibuat.

Di dunia barat, popularitas genre ini berada dipuncak sekitar tahun 1980 hingga pertengahan tahun 1990-an meski kini petualangan dianggap memiliki pasar yang kecil seiring dengan berkembangnya genre lain yang lebih diminati seperti Role Playing Games, First Person Shooter atau Simulasi. Akan tetapi di Jepang, genre ini tetap populer dalam bentuk novel visual dimana 70% dari keseluruhan judul untuk jenis ini diluncurkan di negeri sakura tersebut. Masalahnya adalah *game* buatan Jepang menggunakan bahasa Jepang yang menuntut pemahaman huruf kanji pemain membuat kebanyakan orang tidak mampu menikmatinya. Akan tetapi bukan berarti *game* petualangan berbahasa inggris itu buruk. Faktanya, beberapa dari *game* tersebut merupakan permainan legendaris yang justru banyak dikenang oleh para gamer. Menurut sebuah situs khusus permainan bergenre *adventure*, terdapat 100 game petualangan terbaik sepanjang masa dari berbagai platform dan kebanyakan adalah *cross genre* seperti *action adventure*, *horror adventure*, *puzzle adventure* atau *simulation adventure.*

Permainan yang menggunakan tokoh karakter fiksi yang bertugas mengekplorasi memecahkan sebuah misteri atau kasus, memburu harta karun, maupun menyelamatkan tokoh karakter buatan. Banyak dari game ini diangkat dari sebuah novel populer maupun film biskop.

Contoh : Monkey Island, The Last Express.

9. Action Adventure Game

Permainan petualangan yang dikombinasikan dengan aksi bertarung, menghadapi rintangan maupun memecahkan teka-teki.

Contoh: Tomb Raider, Indiana Jones.

10. Manager Simulation

Permainan yang mensimulasikan menjadi seorang manager dalam sebuah klub sepakbola.

Contoh: Championship Manager.

11. Racing

Adalah salah satu jenis permainan yang bernuansakan balap kendaraan. Disini *user* bisa mengontrol sendiri kendaraan yang dimiliki maupun *user* bisa menongtrolnya bersama-sama dengan partner *user* yang lainnya.

Contoh: CTR (Crash Team Racing), Need For Speed.

12. Sport

Game bergenre ini bertemakan olahraga seperti pada umumnya. *User* dapat mengontrol manusia yang diibaratkan dirinya sendiri di dalam *game* tersebut, yang membedakan adalah *user* disini tidak benar-benar berolahraga, melainkan tokoh yang dimainkanlah yang melakukan kegiatan tersebut.

Contoh: PES 2010, Winning Eleven.

13. Art

Game yang bergenre ini sangat jarang sekali ditemukan, dikarenakan mungkin masih kurang peminatnya. Sehingga masih kurang perusahaan pembuat game yang mengeluarkannya.

2.3 Artificial Intelligence (AI)

Artificial intelligence (AI) atau biasa juga dikenal sebagai kecerdasan buatan adalah cabang ilmu komputer yang berfokus pada otomatisasi tingkah laku cerdas sebuah sistem. AI adalah bagian ilmu komputer yang harus didasarkan pada sound

theoretical dan prinsip-prinsip aplikasi dibidangnya. Prinsip ini adalah struktur data untuk merepresentasikan pengetahuan, algoritma yang diperlukan untuk mengaplikasikan pengetahuan, dan bahasa serta teknik pemrograman yang digunakan untuk implementasi[7].

Definisi kecerdasan buatan sendiri belum terlalu jelas. Berikut ini Tabel 2.1 menjelaskan beberapa definisi dari beberapa buku.

Tabel 2. 1 Beberapa Definisi AI

”*The existing new effort to make computers think...machine with mind, in full and literal sense*” (Haugeland, 1985)	”*The study of mental facultaties though the use of computational models”(Chaniak and mcDermott*, 1985)
”*The study of how to make computers do things at which, at the moment, people are better*”(Rich and Knight, 1991)	”*The branch of computer science that is concerned with tha automation of intelligent behavior*”(Luger and Stubblefield, 1993)

Dari beberapa definisi di atas dapat disimpulkan bahwa AI atau yang disebut dengan kecerdasan buatan adalah ilmu komputer yang di dalamnya terdapat tingkah laku cerdas dari sebuah sistem. Merepresentasikan pengetahuan, algoritma, bahasa, serta teknik pemrograman terhadap implementasi sistem.

2.3.1 Kategori AI

Dari definisi pada Tabel 2.1, definisi AI dapat dibagi menjadi empat kategori, yaitu :

A. *Thinking Humanly* : Pendekatan kognitif

Ada beberapa cara untuk menyatakan suatu program berpikir seperti manusia, yaitu melalui introspeksi dan eksperimen psikologi.

B. *Acting Humanly* : pendekatan uji Turing

Uji Turing dilakukan oleh Alan Turing tahun 1950. Pengujian dirancang untuk menyediakan suatu definisi operasional yang layak bagi kecerdasan. Objek yang diuji adalah komputer yang menjawab pertanyaan *interogator* melalui *teletype*. Jika *interogator* tidak dapat membedakan yang diinterogasi adalah manusia atau komputer maka komputer *berintelegensia* tersebut lolos dari uji Turing.

Komputer yang lolos uji Turing atau Turing Test harus memiliki kemampuan berikut :

1. *Natural Language Processing* untuk berkomunikasi dengan baik dalam bahasa Inggris atau yang lainnya.
2. *Knowledge Representation* untuk menyimpan informasi yang disediakan sebelum atau sampai dapat dijalankan.
3. *Automated Reasoning*, menggunakan informasi yang telah disimpan untuk menjawab pertanyaan dan menggambarkan kesimpulan terbaru.
4. *Machine Learning* untuk menyesuaikan keadaan dan mendeteksi serta memperhitungkan pola.

C. Thingking Rationally : The laws of Thought Approach

Dua hal dalam pendekatan ini adalah :

1. Tidak mudah membuat pengetahuan informal dan menyatakannya dalam term formal, diperlukan notasi logika yang kepastiannya <100%
2. Terdapat perbedaan yang mendasar antara kemampuan memecahkan masalah ”secara prinsip” dan ”secara praktik”.

D. Acting Rationally : The Rational Agent Approach

Pendekatan rational agent digunakan untuk membatasi bahasan karena aksi dan pikiran manusia dluar rasio (refleks dan intuisi) belum dapat ditiru komputer.

Dari beberapa perspektif, AI dapat dipandang sebagai :

1. Perspektif Kecerdasan (*Intelligence*), AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan oleh manusia.
2. Perspektif Pemrograman, AI juga meliputi studi tentang pemrograman simbolik, pemecahan masalah, proses pencarian (*search*).

3. Perspektif Bisnis, AI adalah alat bantu, metodologi bisnis yang menggunakan alat bantu tersebut menyelesaikan masalah-masalah bisnis.
4. Perpektif Penelitian, AI adalah akar studi area.

2.3.2 Kecerdasan buatan dan Kecerdasan alami

Beberapa keuntungan kecerdasan buatan dibanding kecerdasan alami dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Keuntungan kecerdasan buatan dan kecerdasan alami

Keuntungan kecerdasan buatan	Keuntungan kecerdasan alami
1. Lebih permanen	Lebih kreatif
2. Memberikan kemudahan dalam duplikasi dan penyebaran	Dapat melakukan proses pembelajaran langsung. AI harus mendapat masukan berupa simbol dan representasi
Keuntungan kecerdasan buatan	**Keuntungan kecerdasan alami**
3. Relatif lebih murah	Fokus yang luas sebagai referensi untuk mengambil keputusan, AI fokusnya sempit.
4. Konsisten dan teliti	
5. Dapat didokumentasikan	
6. Dapat mengerjakan beberapa task lebih cepat dan lebih baik dibanding manusia.	

Meskipun banyak kelebihan, namun insting manusia dapat melakukan hal yang sulit diprogram oleh komputer, yaitu kemampuan mengenali hubungan, menilai kualitas dan menemukan pola suatu hubungan.

2.3.3 Ruang lingkup dan Elemen-elemen AI

Ruang lingkup dan elemen-elemen AI terdapat beberapa bagian diantaranya:

1. Sistem pakar

Komputer memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki oleh pakar.

2. *Natural language Processing*

Diharapkan user dapat berkomunikasi dengan komputer menggunakan bahasa sehari-hari.

3. *Speech recognition*

Melalui pengenalan ucapan, diharapkan manusia dapat berkomunikasi dengan komputer menggunakan suara

4. *Robotics & Sensory Systems*
5. *Computer Vision*

Menginterpretasikan gambar atau objek-objek tampak melalui komputer.

6. *Inteliigence Computer – aided instruction*

Komputer digunakan sebagai tutor yang dapat melatih dan mengajar.

7. *Game Playing*

Sesuatu yg membuat kita betah bermain *game* itu.

2.3.4 Komputasi alami dan Komputasi buatan

Perbandingan antara AI dengan pemrograman konvensional dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Perbandingan AI dan Pemrograman konvensional

Dimensi	**AI**	**Pemrograman Konvensional**
Processing	Simbolik	Algoritmik
Input	Tidak harus lengkap	Harus lengkap
Search	Heuristik	Algoritmik
Explanation	Tersedia	Tidak tersedia
Major Interest	Pengetahuan	Data dan informasi
Struktur	Terpisal antara kontrol dan pengetahuan	Kontrol terintegrasi dengan data
Output	Tidak harus lengkap	Harus lengkap
Maintenance dan update	Mudah karena menggunakan modul	Umumnya sulit dilakukan
Hardware	Workstation dan PC	Semua tipe
Kemampun pemikiran	Terbatas tapi dapat ditingkatkan	Tidak ada

2.3.5 Algoritma A*

Algoritma A* 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. Dengan perhitungan biaya seperti ini algoritma A* adalah *complete* dan *optimal* [8].

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point*, *current node*, *simpul*, *open list*, *closed list*, harga (*cost*), *walkability*, *target point*. Berikut adalah penjelasannya:

1. *Starting point* adalah sebuah terminologi untuk posisi awal sebuah benda.
2. *Current node* adalah *node* yang sedang dijalankan dalam algortima pencarian jalan terpendek.
3. *Node* adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.
4. *Open list* adalah adalah *list* yang menyimpan kemungkinan *path* yang akan diperiksa. *Open list* dibuat terurut berdasarkan nilai f. *Open list* digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada *path* tujuan. *Open list* berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik (*best node*).
5. *Closed list* adalah tempat untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik (*best node*) atau *list* yang menyimpan jalan yang sudah diperiksa dari open list. Artinya, *Closed list* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup). Kedua list (*Open list* dan *Closed list*) ini bertujuan juga untuk menghindari penelusuran jalan (rute) berkali-kali yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam *Open list*.
6. Nilai f adalah harga perkiraan suatu *path* yang teridentifikasi. Nilai f merupakan hasil dari f(n).
7. Nilai g hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke *path* sekarang.

8. Nilai h hasil dari fungsi h(n), adalah estimasi harga simpul yang dihitung dari simpul tujuan.
9. *Target point* yaitu *node* yang dituju.

Fungsi f sebagai estimasi fungsi evaluasi terhadap *node* n, dapat dituliskan:

$$f(n) = g(n) + h(n) \quad \ldots\ldots (2.1)$$

Keterangan :

f(n) = fungsi evaluasi (jumlah g(n) dengan h(n))

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

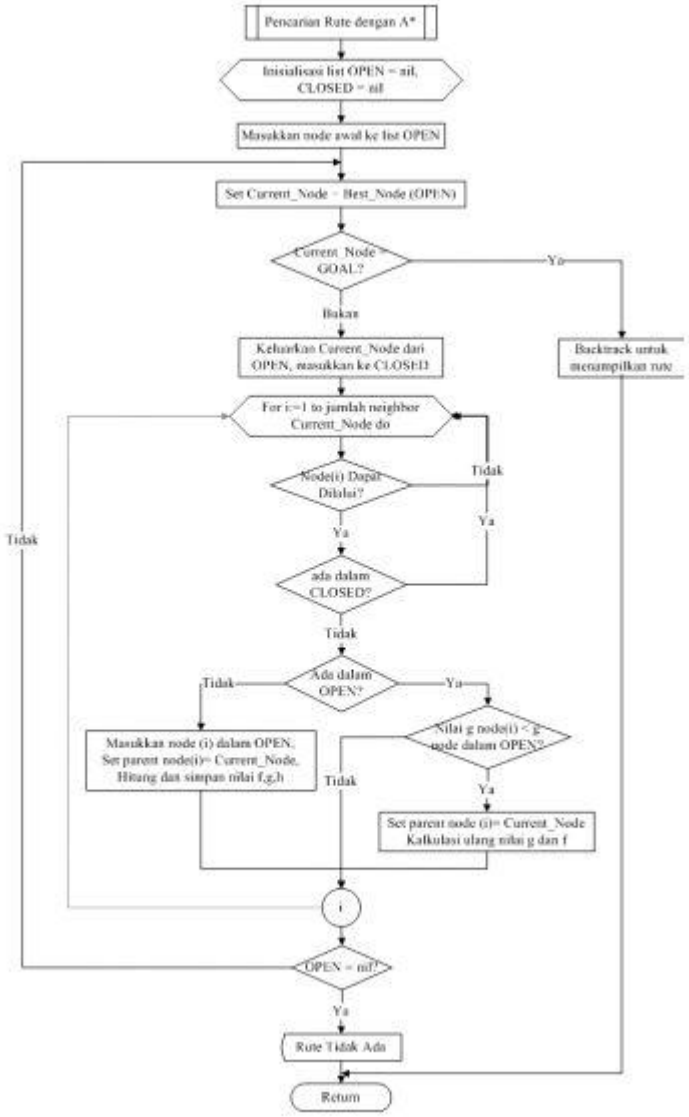
h(n) = estimasi biaya untuk sampai pada suatu tujuan mulai dari n.

Adapun langkah-langkah algortima A* nya adalah sebagai berikut.

1. Dimulai dari *starting point*.
2. Tambahkan *starting point* ke dalam *open list.*
3. Carilah biaya F terendah pada setiap simpul dalam *open list. Node* dengan biaya F terendah inilah yang disebut dengan *current node.*
4. Masukkan ke dalam *closed list.*
5. Untuk setiap 8 simpul (*neighbor node*) yang berdekatan dengan *current node:*
   a. Jika tidak walkable atau jika termasuk *closed list,* maka abaikan.
   b. Jika tidak ada pada *open list,* maka tambahkan ke *open list*.
   c. Jika sudah ada pada *open list,* 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 came from dari *current node,* dan menghitung ulang nilai G dan F dari simpul ini.
6. Berhenti ketika anda menambahkan *target point* ke dalam *closed list*, dalam hal ini jalan telah ditemukan atau gagal untuk menemukan *target point*, dan *open list* kosong. Dalam hal ini, tidak ada jalan.

7. Simpan jalan, bekerja mundur dari *target point*, pergi dari masing-masing simpul ke simpul came from sampai mencapai *starting point*.

Flowchart dari Algoritma A* itu sendiri dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Flowchart Algoritma A*[16]

2.4 Tools yang Digunakan

Berikut ini dijelaskan mengenai beberapa *tool* yang digunakan dalam pembangunan *game Say No To Drugs.*

2.4.1 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode ini didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Pemrograman berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek[9].

Keuntungan menggunakan pemrograman berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Meningkatkan produktivitas karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut *(reusable).*

2. Kecepatan pengembangan

   Kecepatan pengembangan karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.

3. Kemudahan pemeliharaan

   Kemudahan pemeliharaan karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Adanya konsistensi karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Meningkatkan kualitas perangkat lunak karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat

pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

a. Kelas (*class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut.

b. Objek (*object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak.

c. Metode (*method*)

Metode atau operasi atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi structural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.

d. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek.

e. Abstraksi (*abstractions*)

Prinsip untuk mempresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapsulasi (*encapsulations*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

h. Antarmuka (*interface*)

Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

i. Reusabilily

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi atau Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

k. Komunikasi Antar Objek

Komunikasi antar-objek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

l. Polimorfisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

m. Package

Package adalah sebuah kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.4.2 UML (Unified Modelling Language)

UML *(Unified Modelling Language)* adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model, deskripsi atau perangkat lunak) dari system perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan system yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan[10].

2.4.2.1 Bagian-Bagian *UML*

Bagian-bagian utama dari UML adalah *view*, diagram, *model element*, dan *general mechanism*.

a. *View*

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram.

Beberapa jenis *view* dalam UML antara lain: *use case view*, *logical view*, *component view*, *concurrency view*, dan *deployment view*.

b. *Use case view*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya.

View ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang *(designer),* pengembang *(developer),* dan penguji sistem *(tester).*

c. *Logical View*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object, danrelationship*) dan kolaborasi dinamis yang terjadi ketika objek mengirim pesan ke objek lain dalam suatu fungsi tertentu.

View ini digambarkan dalam *class diagrams* untuk *struktur statis* dan dalam *state, sequence, collaboration,* dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang *(designer)* dan pengembang *(developer).*

d. *Component view*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administratif lainnya.

View ini digambarkan dalam *component view* dan digunakan untuk pengembang *(developer).*

e. *Concurrency view*

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration*, dan *activity diagrams*) dan diagram implementasi (*component dan deployment diagrams*) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

f. *Deployment view*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya.

View ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

g. *Diagram*

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu.

Adapun jenis diagram antara lain :

1. *Use Case Diagram*

Use case diagram adalah abstraksi dari interaksi antara sistem dan aktor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata user. Sedangkan *use case diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.

2. *Class Diagram*

Class adalah dekripsi kelompok obyek-obyek dengan properti, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class-class yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

3. *Component Diagram*

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. *Component* merupakan implementasi software dari sebuah atau lebih *class*. *Component* dapat berupa *source code*, *component biner*, atau *executable component*. Sebuah *component* berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga *component diagram* merepresentasikan dunia riil yaitu *component* software yang mengandung *component*, *interface* dan *relationship*.

4. *Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan objek yang

dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

5. State Diagram

Menggambarkan semua state (kondisi) yang dimiliki oleh suatu objek dari suatu *class* dan keadaan yang menyebabkan state berubah. Kejadian dapat berupa objek lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah state yang terdefinisi dengan baik dan kondisi *class* berubah oleh state yang berbeda.

6. Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

7. Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration diagrams* menggambarkan objek dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence diagrams*, tapi jika penekanannya pada konteks gunakan *collaboration diagram*.

8. Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

2.4.2.2 Tujuan Penggunaan *UML*

Adapun tujuan dari penggunaan UML (*Unified Modelling Language*) adalah:

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahas pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

3. Memberikan model yang siap pakai, bahsa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.

UML bisa juga berfungsi sebagai sebuah (*blue print*) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bias diketahui informasi secara detail tentang *coding* program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram *(reserve enginering).*

2.4.2.3 Perangkat Lunak yang Mendukung Pembuatan Diagram UML

Berikut ini adalah beberapa perangkat lunak yang mendukung pembuatan diagram UML.

1. *StarUML*

StarUML adalah sebuah proyek *open source* untuk mengembangkan cepat, *fleksibel, extensible, featureful,* dan bebas-tersedia UML / *platform* MDA berjalan pada platform Win32. Tujuan dari proyek *StarUML* adalah untuk membangun sebuah alat pemodelan perangkat lunak dan juga *platform* yang menarik adalah pengganti alat UML komersial seperti *Rational Rose*, bersama dan sebagainya.

2. *Acceleo*

Acceleo adalah generator kode yang mengubah model menjadi kode. *Acceleo* mudah digunakan dan menyediakan "dari rak" generator (Jee,. Bersih, Php …) dan template editor untuk *Eclipse.*

3. *ArgoUML*

ArgoUML adalah *open source* UML *modeling tool* terkemuka dan termasuk dukungan untuk semua diagram UML standar 1,4. Ini berjalan pada setiap platform Java dan tersedia dalam bahasa sepuluh. *ArgoUML* ditulis seluruhnya di Java dan menggunakan *Java Kelas Foundation*. Hal ini memungkinkan *ArgoUML* untuk berjalan di hampir semua *platform.*

2.4.3 Pemrograman Java

Java adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Sun Microsystems sejak tahun 1991. Bahasa ini dikembangkan dengan

model yang mirip dengan bahasa C++ dan *Smalltalk*, namun dirancang agar lebih mudah dipakai dan *platform independent*, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur komputer. Bahasa ini juga dirancang untuk pemrograman di internet sehingga dirancang agar aman dan portable[11].

Beberapa keunggulan java yaitu java merupakan bahasa yang sederhana. Java dirancang agar mudah dipelajari dan digunakan secara efektif. Java tidak menyediakan fitur-fitur rumit bahasa pemrograman tingkat tinggi, serta banyak pekerjaan pemrograman yang mulanya harus dilakukan manual, sekarang digantikan dikerjakan Java secara otomatis seperti dealokasi memori. Bagi pemrogram yang sudah mengenal bahasa C++ akan cepat belajar susunan bahasa Java namun harus waspada karena mungkin Java mengambil arah (semantiks) yang berbeda dibanding C++.

Seluruh objek diprogram harus dideklarasikan lebih dulu sebelum digunakan. Ini merupakan keunggulan Java yaitu *Statically Typed*. Pemaksaan ini memungkinkan kompilator Java menentukan dan melaporkan terjadinya pertentangan (ketidakkompatibelan) tipe yang merupakan barikade awal untuk mencegah kesalahan yang tidak perlu (seperti mengurangkan variabel bertipe integer dengan variabel bertipe string).

Pencegahan sedini mungkin diharapkan menghasilkan program yang bersih. Kebaikan lain fitur ini adalah kode program lebih dapat dioptimasi untuk menghasilkan program berkinerja tinggi.

Java menggunakan model pengamanan tiga lapis *(three-layer security model)* untuk melindungi sistem dari *untrusted Java code*. Pertama, *bytecode verifier* membaca *bytecode* sebelum dijalankan dan menjamin *bytecode* memenuhi aturan-aturan dasar bahasa Java. Kedua, *class loader* menangani pemuatan kelas Java ke *runtime interpreter*. Ketiga, manajer keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem file, port jaringan, proses eksternal dan sistem window.

Platform independence adalah kemampuan program bekerja di sistem operasi yang berbeda. Bahasa Java merupakan bahasa yang secara sempurna tidak

bergantung *platform*. Tipe variabel Java mempunyai ukuran sama di semua *platform* sehingga variabel bertipe integer berukuran sama tidak peduli dimana program java dikompilasi. Begitu telah tercipta file *class* dengan menggunakan kompilator Java di *platform* manapun, maka file *class* tersebut dapat dijalankan di *platform* manapun. Jadi "dimanapun dibuat, dimanapun dapat dijalankan". Slogan ini biasa diringkas sebagai *Write Once, Run Anywhere (WORA).*

Java termasuk bahasa *Multithreading*. *Thread* adalah untuk menyatakan program komputer melakukan lebih dari satu tugas di satu waktu yang sama. Java menyediakan kelas untuk menulis program *multithreaded*, program mempunyai lebih dari satu thread eksekusi pada saat yang sama sehingga memungkinkan program menangani beberapa tugas secara konkuren.

Program Java melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek-objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat di bahasa yang memungkinkan alokasi dinamis.

Java mempunyai mekanisme *exception-handling* yang ampuh. *Exception-handling* menyediakan cara untuk memisahkan antara bagian penanganan kesalahan dengan bagian kode normal sehingga menuntun ke struktur kode program yang lebih bersih dan menjadikan aplikasi lebih tegar. Ketika kesalahan yang serius ditemukan, program Java menciptakan *exception*. *Exception* dapat ditangkap dan dikelola program tanpa resiko membuat sistem menjadi turun. Program Java mendukung *native method* yaitu fungsi ditulis di bahasa lain, biasanya C/C++. Dukungan *native method* memungkinkan pemrogram menulis fungsi yang dapat dieksekusi lebih cepat dibanding fungsi ekivalen di java. *Native method* secara dinamis akan di-link ke program java, yaitu diasosiasikan dengan program saat berjalan[10].

2.4.4 Adobe Flash Professional

Adobe Flash Professional merupakan sebuah software yang didesain khusus oleh Adobe dan program aplikasi standar *authoring tool professional* yang

digunakan untuk membuat animasi dan bitmap yang sangat menarik untuk keperluan pembangunan situs web yang interaktif dan dinamis.

Flash didesain dengan kemampuan untuk membuat animasi 2 dimensi atau 3 dimensi yang handal dan ringan sehingga flash banyak digunakan untuk membangun dan memberikan efek animasi pada website, CD Interaktif dan yang lainnya, Selain itu software ini juga dapat digunakan untuk membuat animasi logo, movie, game, pembuatan navigasi pada situs website atau blog, tombol animasi, banner, menu interaktif, interaktif form isian, e-card, screen saver dan pembuatan aplikasi-aplikasi website lainnya.

Adapun fitur-fitur dari *Adobe Flash* sebagai berikut :

1. Support untuk HTML 5.

Terdapat ekstensi baru, dapat membuat konten interaktif HTML 5 dengan pembuatan animasi maupun gambar inti di Flash. Dapat di *exsport* sebagai *Java Script* sebagai sumber kerangka *Create JS open source*.

2. *Stage 3D targeting*

Turbocharge rendering dengan menggunakan *direct mode* dengan *open source Starling Framework* untuk terhubung dengan konten *hardware-accelerated* 2D.

3. *Advanced drawing tools*

Hasil *Design artwork* nya lebih tepat dan efisien, dengan *smart shapes* beserta *design tool* yang kuat.

4. *Industry-leading animation tools*

Create and edit motion menggunakan *Timeline and Motion Editor*, dan menggunakan *inverse kinematics* untuk membuat *motioan* yang natural/alami pada animasi karakternya.

5. *Professional video tools Easily incorporate*

Video terproyeksi, klip video dapat secara efisien terkonversi dengan adanya aplikasi *Adobe Media Encoder*.

6. *XML-based FLA source files*

Penyatuan proyek lebih mudah dengan implementasi XML berbasis format file FLA. Proyek terkompresi bertindak seperti folder, yang memungkinkan Anda untuk cepat mengelola dan memodifikasi aset.

7. *Creative Suite integration Perform roundtrip*

Memungkinkan suntingan gambar bitmap dengan *Adobe Photoshop ® CS6 software*, dan mengintegrasikan erat dengan *Adobe Flash Builder ® 4.6 software*.

8. *3D transformation Animate 2D objects*

3D space dengan *rotation tools* 3D yang menarik, yang memungkinkan untuk menganimasi seepanjang sumbu x, y, dan z menggunakan *apply local* atau *global transformation* ke objek.

9. *Spring attribute for the Bone tool Add*

Ekspresi seperti animasi real/hidup, dengan adanya attributes untuk *bone tool*. *Inverse kinematics engine* yang kuat dapat memberikan gerakan real.

10. *Adobe AIR mobile simulation Simulate*

Common mobile application berinteraksi seperti orientasi layar, adaya gerakan sentuh, dan *accelerometer* untuk membantu mempercepat pengujian[13].

2.4.5 Multimedia

Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, suara, gambar, animasi dan video dengan alat bantu (*tool*) dan koneksi (*link*) sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi. Multimedia sering digunakan dalam dunia hiburan. Selain dari dunia hiburan, multimedia juga diadopsi oleh dunia *game*.

Multimedia juga dapat diartikan sebagai penggunaan beberapa media yang berbeda dalam menyampaikan informasi berbentuk text, audio, grafik, animasi, dan video[14].

2.4.5.1 Kategori Multimedia

Multimedia dapat di definisikan menjadi 2 kategori, yaitu *Multimedia Content Production* dan *Multimedia Communication*. *Multimedia Content Production* adalah penggunaan beberapa media (*teks, audio, graphics, animation, video dan interactivity*) yang berbeda dalam menyampaikan suatu informasi atau

menghasilkan produk multimedia seperti video, audio, musik, film, game, *entertaintment*, dll.

Bisa juga dikatakan sebagai penggunaan beberapa teknologi yang berbeda yang memungkinkan untuk menggabungkan media *(teks, audio, graphics, animation, video, dan interactivity* dengan cara yang baru untuk tujuan komunikasi. Dalam kategori ini media yang digunakan adalah:

1. Media teks/tulisan
2. Media audio/suara
3. Media video
4. Media animasi
5. Media gambar
6. Media Interaktif
7. Media spesial effect

Multimedia Communication adalah penggunaan media (massa), seperti televisi, radio, media cetak dan internet untuk mempublikasikan / menyiarkan / mengkomunikasikan material periklanan, publikasi, entertaintment, berita, pendidikan, dll. Dalam kategori ini media yang digunakan adalah :

1. TV
2. Radio
3. Film
4. Media Cetak
5. Musik
6. Game
7. Entertainment
8. Tutorial
9. Internet

2.4.5.2 Elemen Multimedia

Menurut Hofstetter komponen multimedia terbagi atas lima jenis yaitu:

1. Teks

Teks merupakan elemen multimedia yang menjadi dasar untuk menyampaikan informasi, karena teks adalah jenis data yang paling sederhana dan membutuhkan tempat penyimpanan yang paling kecil. Teks merupakan cara yang paling efektif dalam mengemukakan ide-ide kepada pengguna, sehingga penyampaian informasi akan lebih mudah dimengerti oleh masyarakat. Jenis-jenis teks seperti *Printed Text*, yaitu teks yang dihasilkan oleh *word processor* atau *word editor* dengan cara diketik yang nantinya dapat dicetak. *Scanned Text* yaitu teks yang dihasilkan melalui proses scanning tanpa pengetikan, dan *Hypertext* yaitu jenis teks yang memberikan link ke suatu tempat / meloncat ke topik tertentu.

2. Grafik *(image)*

Sangat bermanfaat untuk mengilustrasi informasi yang akan disampaikan terutama informasi yang tidak dapat dijelaskan dengan kata-kata. Jenis-jenis grafik seperti bitmap yaitu gambar yang disimpan dalam bentuk kumpulan pixel, yang berkaitan dengan titik-titik pada layar monitor. *Digitized picture* adalah gambar hasil rekaman video atau kamera yang dipindahkan ke komputer dan diubah ke dalam bentuk bitmaps. *Hyperpictures*, sama seperti *hypertext* hanya saja dalam bentuk gambar.

3. Audio

Multimedia tidak akan lengkap jika tanpa audio (suara). Audio bisa berupa percakapan, musik atau efek suara.

Format dasar audio terdiri dari beberapa jenis :

a. WAVE

Merupakan format file digital audio yang disimpan dalam bentuk digital dengan eksistensi *WAV*.

b. *MIDI* (*Musical Instrument Digital Interface*)

MIDI memberikan cara yang lebih efisien dalam merekam music dibandingkan *wave*, kapasitas data yang dihasilkan juga jauh lebih kecil. *MIDI* disimpan dalam bentuk *MID*.

4. Video

Video menyediakan sumber yang kaya dan hidup untuk aplikasi multimedia. Dengan video dapat menerangkan hal-hal yang sulit digambarkan lewat kata-kata

atau gambar diam dan dapat menggambarkan emosi dan psikologi manusia secara lebih jelas.

5. Animasi

Animasi adalah simulasi gerakan yang dihasilkan dengan menayangkan rentetan *frame* ke layer. *Frame* adalah satu gambar tunggal pada rentetan gambar yang membentuk animasi.

Menurut Foley, Van Dam, Feiner dan Hughes (1997, p1057) *Animate* adalah untuk membuat sesuatu hidup, sebagian orang mengira bahwa animasi itu sama dengan motion (gerakan), tetapi animasi mencakup semua yang mengandung efek visual sehingga animasi mencakup perubahan posisi terhadap waktu, bentuk, warna, struktur, tekstur dari sebuah objek, posisi kamera, pencahayaan, orientasi dan fokus dan perubahan dalam teknik *rendering*.

2.4.6 Android

Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai macam piranti. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat piranti lunak untuk ponsel. Kemudian untuk mengembangkan android dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Telepon pertama yang memakai sistem operasi Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis telepon seluler yang menggunakan Android. Seiring perkembangannya, versi android semakin lama semakin ditingkatkan mengingat kemajuan teknologi yang semakin pesat[15].