BAB 2
LANDASAN TEORI

2.1 Multimedia

2.1.1 Pengertian Multimedia

Multimedia adalah kombinasi dari komputer dan *video* atau multimedia secara umum merupakan kombinasi tiga lapisan, yaitu: suara, gambar dan teks, atau multimedia adalah kombinasi dari paling sedikit dua menjadi *input* atau *output* dari data, media ini dapat *audio* (suara, musik), animasi, *video*, teks, grafis dan gambar [3].

Multimedia merupakan alat yang dapat menciptakan presentasi yang dinamis dan interaktif yang mengkombinasikan teks, grafik, animasi, *audio*, dan gambar *video*. Definisi yang lain dari multimedia, yaitu dengan menempatkan dalam konteks, seperti yang dilakukan oleh Hofstetter, multimedia adalah pemanfaatan komputer untuk membuat dan menggabungkan teks, grafis, *audio*, gambar bergerak (*video* dan animasi) dengan menggabungkan *link* dan *tool* yang memungkinkan pemakai melakukan navigasi, berinteraksi, berkreasi dan berkomunikasi. Dalam definisi ini terkandung empat komponen penting multimedia. Pertama harus ada komputer yang mengkoordinasikan apa yang dilihat dan didengar. Yang berinteraksi dengan kedua, harus ada *link* yang menghubungkan kita dengan informasi. Ketiga, harus ada alat navigasi yang memandu kita, menjelajah jaringan informasi yang saling terhubung. Kempat, multimedia menyediakan tempat kepada kita untuk mengumpulkan, memproses,

dan mengkomunikasikan informasi dan ide kita sendiri, jika salah satu komponen tidak ada. Maka bukan multimedia dalam arti yang luas namanya. Misalnya, jika tidak ada komputer untuk berinteraksi, maka itu namanya media campuran, bukan multimedia. Jika tidak ada link yang menghadirkan sebuah struktur dan dimensi, maka namanya rak buku, bukan multimedia. Kalau tidak ada alat navigasi yang memungkinkan kita memilih jalannya suatu tindakan maka itu namanya film, bukan multimedia. Demikian juga jika kita tidak mempunyai ruang untuk berkreasi dan menyumbangkan ide sendiri, maka namanya televisi, bukan multimedia. Dari beberapa definisi diatas, maka multimedia ada yang *online* (internet) dan multimedia ada yang *offline* (tradisional).

Kelebihan multimedia adalah menarik indera dan minat, karena merupakan gabungan antara pandangan, suara dan gerakan. Lembaga Riset dan Penerbitan Komputer yaitu *Computer Technology Research* (CTR), menyatakan bahwa orang hanya mampu mengingat 20% dari yang dilihat dan 30% dari yang didengar. Tetapi orang dapat mengingat 50% dari yang dilihat dan didengar dan 30% dari yang dilihat, didengar dan dilakukan sekaligus. Maka multimedia sangatlah efektif. Multimedia menjadi *tool* yang ampuh untuk pengajaran dan pendidikan serta untuk meraih keunggulan bersaing perusahaan [3].

2.1.2 Pengertian Multimedia Pembelajaran Interaktif

Multimedia adalah media yang menggabungkan dua unsur atau lebih media yang terdiri dari teks, grafis, gambar, foto, *audio, video* dan animasi secara

terintegrasi. Multimedia dapat diartikan sebagai perpaduan dari berbagai media yang terdiri dari teks, grafis, gambar diam, animasi, suara dan *video* untuk menyampaikan pesan kepada publik. Berdasarkan pendapat-pendapat tersebut dapat disimpulkan, multimedia merupakan perpaduan antara berbagai media (*format file*) yang berupa teks, gambar (*vektor* atau *bitmap*), grafis, *sound* animasi terintegrasi dan telah dikemas menjadi *file digital* (komputerasi), digunakan untuk menyampaikan pesan kepada publik.

Karakteristik multimedia ada empat hal, yaitu:

a. mengoptimalkan seluruh media yang ada

b. interaktif

c. kontrol pada pengguna (*user*)

d. terprogram

Jenis multimedia dapat dibedakan menjadi dua, yaitu**:** multimedia linier dan multimedia interaktif. Multimedia linier adalah suatu multimedia yang tidak dilengkapi dengan alat pengontrol apapun yang dapat dioperasikan oleh user Multimedia ini berjalan sekuensial (berurutan),contohnya: TV dan film. Sedangkan multimedia interaktif adalah suatu multimedia yang dilengkapi dengan alat pengontrol yang dapat dioperasikan oleh pengguna, sehingga pengguna dapat memilih apa yang dikehendaki untuk proses selanjutnya. Contoh multimedia interaktif adalah: multimedia pembelajaran interaktif, aplikasi *game*, dan lain-lain.

2.1.3 Konsep Multimedia

1. Teks

Tampilan dalam bentuk teks atau yang lebih dikenal dengan istilah tipografi merupakan elemen yang cukup penting dalam pembuatan multimedia. Sebagian besar multimedia menggunakan teks karena sangat efektif untuk menyampaikan ide dan panduan kepada pengguna. Teks merupakan bentuk data multimedia yang paling mudah disimpan dan dikenali, serta *file* teks mempunyai struktur yang sederhana. Teks biasanya mengacu pada kata, kalimat, segala sesuatu yang tertulis atau ditayangkan.

Beberapa hal yang perlu diperhatikan mengenai penggunaan teks dalam suatu aplikasi multimedia adalah :

a. Gunakanlah huruf (*font*) yang sesuai dengan tema aplikasi multimedia yang akan dibuat.

b. Pastikan huruf (*font*) yang dipakai tersedia di sistem komputer lain.

c. Pemilihan bentuk dan warna yang sesuai dengan tema aplikasi multimedia.

d. Pastikan teks tersebut terbaca.

e. Usahakan ringkas tetapi padat.

2. Grafik

Grafik atau gambar merupakan sarana pembentukan informasi yang lebih mudah untuk dipahami. Gambar juga merupakan salah satu komponen penting dalam multimedia karena dapat meringkas dan menyajikan data kompleks serta mampu menyampaikan banyak kata. Gambar dalam publikasi multimedia lebih menarik perhatian dan dapat mengurangi kebosanan dibandingkan dengan teks, sebab manusia selalu berorientasi terhadap *visual*.

3. Audio

Teknologi *audio* juga berperan penting dalam penyampaian informasi, tanpa adanya audio dalam sebuah multimedia maka hasilnya tidak lengkap. Suara atau *audio* di dalam multimedia biasanya berupa suara musik, suara dari *voice record* dan efek–efek suara lain.

4. *Video*

Video adalah gambar – gambar yang saling berurutan sehingga menimbulkan efek gerak. Pembuatan *video* dalam tampilan multimedia bertujuan untuk membuat tampilan yang dihasilkan lebih menarik.

5. Animasi

Animasi merupakan kumpulan gambar yang ditampilkan secara bergantian dan berurutan sehingga terlihat bergerak dan hidup. Pergerakan animasi akan lebih mudah dimengerti daripada objek atau gambar diam. Selain itu, animasi lebih menarik dan mudah dimengerti daripada hanya sekedar

gambar karena lebih komunikatif dalam menyampaikan suatu tujuan. Animasi dibagi menjadi beberapa macam, yaitu :

a. Animasi Sel

Sel animasi biasanya merupakan lembaran - lembaran yang membentuk sebuah *frame* animasi tunggal. Sel animasi merupakan sel yang terpisah dari lembaran latar belakang dan sebuah sel untuk masing-masing objek yang bergerak secara mandiri diatas latar belakang.

b. Animasi *Frame*

Animasi *frame* adalah bentuk animasi yang paling sederhana.Di umpamakan sebuah buku yang mempunyai gambar berseri di tepi halaman berurutan. Jika membuka buku dengan cepat, maka gambar kelihatan bergerak. Pada komputer multimedia, animasi buku tersebut menampilkan sebuah gambar yang berurutan secara cepat. Antara gambar satu (*frame* satu) dengan gambar lain (*frame* lain) berbeda.

c. Animasi *Sprite*

Animasi *sprite* serupa dengan teknik animasi tradisional, yaitu objek yang diletakkan dan dianimasikan pada bagian puncak grafik dengan latar belakang diam. *Sprite* adalah setiap bagian dari animasi yang bergerak secara mandiri, misalnya burung terbang, planet berotasi atau logo berputar

d. Animasi Lintas (*Animation Path*)

Animasi lintasan adalah animasi dari objek yang bergerak sepanjang garis kurva yang ditentukan sebagai lintasan.

e. Animasi *Spline*

Spline adalah representasi matematis dari kurva. Bila objek bergerak, biasanya tidak mengikuti garis lurus, misalnya berbentuk kurva. Program animasi komputer memungkinkan untuk membuat animasi *spline* dengan lintasan gerakan berbentuk kurva.

f. Animasi *Vektor*

Animasi *vektor* menjadikan objek bergerak dengan memvariasikan ketiga parameter ujung pangkal, arah dan panjang pada segmen-segmen garis yang menentukan objek.

g. Animasi Karakter

Animasi karakter merupakan sebuah cabang khusus animasi. Animasi karakter semacam yang dilihat dalam film kartun. Animasi ini berbeda dengan animasi lainnya, misalnya grafik bergerak animasi logo yang melibatkan bentuk organik yang komplek dengan penggandaan yang banyak, gerakan hirarkis.

h. *Computational Animation*

Dengan *computational animation*, untuk menggerakkan objek dilayar cukup memvariasikan koordiant x dan y nya. Koordinat x merupakan posisi *horizontal* objek, yaitu berapa jauh kiri-kanan layar. Koordinat y merupakan posisi *vertikal*, yakni berapa jauh atas-bawah layar.

i. *Morphing*

Morphing artinya mengubah satu bentuk menjadi bentuk lain dengan menampilkan serangkaian *frame* yang menciptakan gerakan halus begitu bentuk dirinya menjadi bentuk.

2.1.4 Multimedia Berbasis Komputer

Saat ini teknologi komputer tidak hanya digunakan sebagai sarana komputasi pengolahan kata *word processor*, tetapi juga sebagai sarana belajar multimedia yang memungkinkan mahasiswa membuat desain dan rekayasa suatu konsep dan ilmu pengetahuan. Sajian multimedia berbasis komputer dapat diartikan sebagai teknologi yang mengoptimalkan peran komputer sebagai sarana untuk menampilkan dan merekayasa teks, grafik, dan dalam sebuah tampilan yang terintegrasi. Dengan tampilan yang dapat mengkombinasikan berbagai unsur penyampaian informasi dan pesan, komputer dapat dirancang dan digunakan sebagai media teknologi yang efektif untuk mempelajari dan menyampaikan informasi yang dibutuhkan oleh manusia [4].

2.1.5 Perangkat Lunak Multimedia

Perangkat lunak multimedia adalah komponen-komponen dalam data processing system, berupa program-program untuk mengontrol bekerjanya sistem komputer multimedia. Fungsi perangkat lunak multimedia antara lain mengidentifikasikan program multimedia dan menyiapkan aplikasi program multimedia sehingga tata kerja seluruh peralatan komputer multimedia jadi terkontrol serta mengatur dan membuat pekerjaan agar yang berkaitan dengan multimedia lebih efisien.

Bahasa tingkat rendah lebih dekat hubungannya dengan mesin, maka bahasa ini disebut juga Machine Oriented, misalnya *Assembly*. Bahasa ini merupakan bahasa cepat yang digunakan dalam banyak sistem multimedia, khususnya "*game*", contohnya *Nintendo Entertainment System (NES), Super Nintendo Entertainment System (SNES), Game Boy, SEGA's Saturn* dan *Genesis*. *Assembly* mempunyai kemampuan besar untuk menyelesaikan masalah spesifik yang kritis terhadap waktu. Bahasa ini lebih cepat dalam mengerjakan sesuatu, karena bahasa lain berlaku sebagai "*buffer layer*" terhadap instruksi *assembly*. *Assembly* berkomunikasi langsung dengan instruksi prosesor. Bahasa *assembly* tetap digunakan dalam konsul generasi terbaru dan peralatan yang melekat karena kecepatannya. Ada beberapa variasi berbeda dari bahasa *assembly*:

80x86 atau i86 : *assembly* berbasis-Intel

68000 : *assembly* berbasis komputer *Apple (Macintosh)*

Z80 : Digunakan untuk program *game Nintendo Game Boy*

SH-2 : Digunakan dalam *SEGA Saturn*

ASM : Digunakan untuk *Game* Boy Tingkat Lanjut

MIPS : Digunakan untuk *game Pocket PC*

2.2 *Game*

Video *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.

2.2.1 Pengertian *Game*

Dalam bahasa Indonesia *game* diartikan sebagai permainan. Tetapi akhir-akhir ini terdapat banyak *game* yang digunakan untuk simulasi dan training. Hal ini berkembang karena fungsi *game* yang semakin banyak. Oleh karena itu dibutuhkan definisi *game* yang cukup beragam, sehingga perlu adanya definisi dari *game* yang lebih jelas jelas. Pada awalnya *game* adalah aplikasi atau console komputer, yang fungsinya adalah untuk menyajikan hiburan pada pemainnya.

Tujuan utama pembuatan *game* yang paling banyak untuk menyajikan hiburan. Tapi sekarang banyak *game* yang digunakan sebagai training dan simulasi maka definisi *game*pun berkembang. *Game* merupakan suatu struktur atau semi struktur dari sebuah kegiatan, yang biasanya digunakan untuk kesenangan dan digunakan juga sebagai media pendidikan. (Definisi "*game*" bisa juga digunakan untuk mendeskripsikan berbagai macam kegiatan simulasi seperti, training, analisis, prediksi, dan lain-lain).

Dari beberapa definisi diatas dapat disimpulkan bahwa *game* merupakan sebuah kegiatan yang melibatkan fisik dan mental. *Game* bisa juga diartikan juga sebagai simulasi, training, analisis dan prediksi, sehingga bisa dikembangkan sebagai media pembelajaran [5].

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 AI (*Artificial Intelligence*)

Kecerdasan buatan atau *Artificial Intelligence* (AI) istilah yang mungkin akan mengingatkan kita akan kehebatan optimus prime dalam film The Transformers. Kecerdasan buatan memang kerap diidentikkan dengan kemampuan robot yang dapat berperilaku seperti manusia. Berbagai definisi diungkapkan oleh para ahli untuk dapat memberi gambaran mengenai kecerdasan buatan beberapa diantaranya :

- Kecerdasan Buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.
- Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia
- Kecerdasan Buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan [6].

2.3.1 Jenis-Jenis Kecerdasan Buatan

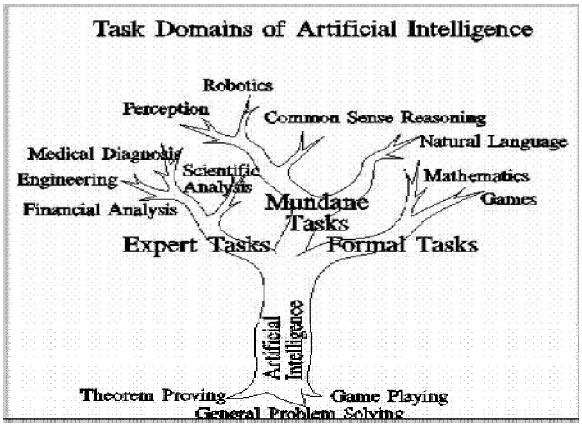
Dalam perkembangannya kecerdasan buatan dapat dikelompokkan sebagai berikut :

a. Sistem Pakar (*Expert System*), komputer sebagai sarana untuk menyimpan pengetahuan para pakar sehingga komputer memiliki keahlian menyelesaikan permasalahan dengan meniru keahlian yang dimiliki pakar.
b. Pengolahan Bahasa Alami (*Natural Language Processing*), *user* dapat berkomunikasi dengan komputer menggunakan bahasa sehari-hari, misal bahasa inggris, bahasa indonesia, dan sebagainya.
c. Pengenalan Ucapan (*Speech Recognition*), manusia dapat berkomunikasi dengan komputer menggunakan suara.
d. Robotika & Sistem Sensor.
e. *Computer Vision*, menginterpretasikan gambar atau objek-objek tampak melalui komputer.
f. *Intelligent Computer-Aided Instructi*on, komputer dapat digunakan sebagai tutor yang dapat melatih & mengajar.
g. *Game Playing*.
h. Soft Computing.
i. Area Aplikasi AI.

Ada dua hal yang sangat mendasar mengenai penelitian-penelitian AI, yaitu *knowledge representation* (representasi pengetahuan) dan *search* (pelacakan). Para peneliti AI terus mengembangkan berbagai jenis teknik baru

dalam menangani sejumlah permasalahan yang tergolong ke dalam AI seperti *vision* dan percakapan, pemrosesan bahasa alami dan permasalahan khusus seperti diagnosa medis.

AI seperti bidang ilmu lainnya juga memiliki sejumlah sub disiplin ilmu yang sering digunakan untuk pendekatan yang esensial bagi penyelesaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Berikut ini merupakan sejumlah bidang terapan utama dari kontribusinya terhadap perkembangan AI yang digambarkan bentuk pohon berikut ini :



Gambar 2.1 Beberapa bidang terapan AI [6].

Area aplikasi AI diantaranya yaitu:

- *Game Playing*.
- Penalaran Otomatis dan Pembuktian.
- Sistem Pakar.
- Pemahaman Bahasa Alami.

- Perencanaan dan Robotik.
- Jaringan Saraf Tiruan (JST).
- Algoritma Genetika.

2.3.2 Perbandingan Kecerdasan Buatan dengan Kecerdasan Alamiah.

Ada beberapa keuntungan kecerdasan buatan dibanding kecerdasan alamiah, yaitu:

- Lebih permanen.
- Memberikan kemudahan dalam duplikasi dan penyebaran.
- Relatif lebih murah dari kecerdasan alamiah.
- Konsisten dan teliti.
- Dapat didokumentasi.
- Dapat mengerjakan beberapa task dengan lebih cepat dan lebih baik dibanding manusia.

2.3.3 Perbedaan Komputasi AI dengan Proses Komputasi Konvensional.

Tabel 2.1 Pemrosesan data komputer konvensional [6].

Proses	Tugas
Kalkulasi	Mengerjakan operasi-operasi matematis seperti : +, =, x, :, atau mencari akar persamaan, menyelesaikan rumus/ persamaan

Logika	Mengerjakan operasi logika seperti and, or, invert
Penyimpanan	Menyimpan data dan gambar pada file
Retrieve	Mengakses data yang disimpan pada file
Translate	Mengonversi data dari satu bentuk ke bentuk yang lain
Sort	Memeriksa data dan menampilkan dalam urutan yang diinginkan
Edit	Melakukan perubahan, penambahan dan penghapusan data
Monitor	Mengamati event eksternal dan internal serta melakukan tindakan jika kondisi tertentu tercapai
Kontrol	Memberikan perintah atau mengendalikan peralatan luar

2.3.4 Perbandingan antara AI dengan pemrograman konvensional adalah seperti tabel berikut ini :

Tabel 2.2 Perbandingan AI dan pemrograman konvensional [6].

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	Terpisah antara kontrol dan pengetahuan	Kontrol terintegrasi dengan data
Output	Tidak harus lengkap	Harus lengkap
Maintenance dan update	Mudah karena menggunakan modul-modul	Umumnya susah dilakukan
Hardware	Workstation dan PC	Semua tipe

2.4 AI pada *Game* Development Pathfinding

AI bisa dikatakan cukup berperan pada sebuah *game*. musuh pada *game* biasa berinteraksi dengan *player* dalam berbagai hal, mulai dari bertarung, hingga berjalan. khusus proses berjalan, algoritma path finding adalah algoritma yang dapat dimanfaatkan membantu musuh menemukan alur jalannya.

Pada materi ini dimulai dengan definisi AI, penulis lebih menyukai definisi AI yang dikemukakan oleh peter norvig pada bukunya *AI a Modern Approac*h. Membahas algoritma path finding sederhana, mulai dari *Basic Pathfinding, Random Movement Obstacle Avoidance, Tracing Around Obstacles, Path Following, Wall-tracing p*ath, hingga *Waypoint Navigation*. Setelah itu disebutkan materi tentang path finding yang ada pada AI yang terkelompok menjadi *Uninformed* dan *Informed search*.

Basic Pathfinding, pada *basic path finding* koordinat musuh akan berjalan mengikuti *increment* dari koordinat x, dan y dari objek. Jika koordinat x telah tercapai, namun y belum tercapai, maka y akan dilakukan *increment* hingga mencapai tujuan. Hal sebaliknya juga terjadi pada algoritma ini. Kelemahan algoritma ini adalah jika musuh menemui *obstacle* pada saat dia berjalan.

Random Movement Obstacle Avoidance, algoritma ini mengatasi kekurangan pada basic path finding. pada saat *basic path finding* menemui *obstacle*, maka musuh akan diberikan random arah, sehingga tidak menemukan *obstacle*.

Tracing Around Obstacles, berbeda dengan RMOA, algoritma ini melakukan penelusuran area ke arah tertentu baik ke kiri, atau ke kanan *obstacle*. algoritma TAO sederhana memiliki kekurangan pada pemberhentian *tracing around*, maka dari itu bisa digabungkan dengan algoritma yang lainnya.

Path Following adalah algoritma yang menjalankan musuh mengikuti arah atau pola jalan tertentu. algoritma ini cocok dipergunakan sebagai contoh musuh diharuskan berlari, berjalan mengikuti alur atau track atau jalan tertentu.

Wall-tracing path, algortima WTP akan mentrace melalui labirin sehingga memungkinkan musuh bergerak dari satu ruang ke ruang tertentu. algoritma ini mengikuti pola pergerakan tertentu, sehingga ada kemungkinan musuh tidak dapat menjelajahi semua ruang yang ada di labirin.

Waypoint Navigation menggunakan *node* yang telah dibuat, dan *predefined routing tabl*e dalam menentukan path. hampir sama seperti WTP, namun algortima ini lebih bagus dan cepat pemrosesannya, dan bisa dikatakan lebih feksibel [7].

2.4.1 Algoritma A* (*A Star*)

Algorima A*, dapat juga disebut sebagai Algoritma A Star, merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Jika kita mengetikkan Algoritma A* pada sebuah mesin pencari, seperti *google.com*, maka akan kita temukan lebih dari sepuluh ribu literatur mengenai algoritma A*

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point*, simpul (*nodes*), A, *open list*, *Closed list*, harga (*node*), halangan (*unwalkable*) [6].

Simpul tujuan yaitu simpul yang dituju. Rintangan adalah sebuah atribut yang menyatakan bahwa sebuah simpul tidak dapat dilalui oleh A. Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (*starting point*) menuju simpul tujuan dengan memperhatikan harga (F) terkecil. Diawali dengan menempatkan A pada *starting point*, kemudian memasukkan seluruh simpul yang bertetangga dan tidak memilik atribut rintangan dengan A ke dalam *open list*. Kemudian mencari nilai H terkecil dari simpul-simpul dalam *open list* tersebut. Kemudian memindahkan A ke simpul yang memiliki nilai H terkecil.

Simpul sebelum A disimpan sebagai parent dari A dan dimasukkan ke dalam *Closed list*. Jika terdapat simpul lain yang bertetangga dengan A (yang sudah berpindah) namun belum termasuk kedalam anggota *open list*, maka masukkan simpul-simpul tersebut ke dalam *open list*. Setelah itu, bandingkan nilai G yang ada dengan nilai G sebelumnya (pada langkah awal, tidak perlu dilakukan perbandingan nilai G). Jika nilai G sebelumnya lebih kecil maka A kembali ke posisi awal. Simpul yang pernah dicoba dimasukkan ke dalam *Closed list*. Hal terebut dilakukan berulang-ulang.

A* adalah algoritma pencarian pohon yang mencari jalur dari satu titik awal ke sebuah titik akhir yang telah ditentukan. Algoritma A* menggunakan pendekatan heuristik h(x) yang memberikan peringkat ke tiap-tiap titik x dengan cara memperkirakan rute terbaik yang dapat dilalui dari titik tersebut. Setelah itu

tiap-tiap titk x tersebut dicek satu-persatu berdasarkan urutan yang dibuat dengan pendekatan heuristik tersebut. Maka dari itulah algoritma A* adalah contoh dari best-first search. Algoritma ini pertama kali ditemukan pada tahun 1968 oleh Peter Hart, Nils Nilsson dan Bertram Raphael. Dalam tulisan mereka, algoritma ini dinamakan algoritma A. Penggunaan algoritma ini dengan fungsi heuristik yang tepat dapat memberikan hasil yang optimal, maka algoritma inipun disebut A*. Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point*, simpul (*node*s), A, *open list*, *Closed list*, harga (*node*), halangan (unwalkable).

Starting point adalah sebuah terminologi posisi awal sebuah benda. A adalah simpul yang sedang dijalankan algortima pencarian jalan terpendek. Simpul adalah petak-petak kecil sebagai representasi dari *area pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. *open list* adalah tempat menyimpan data simpul yang mungkin diakses dari *starting point* maupun simpul yang sedang dijalankan. *Closed list* adalah tempat menyimpan data simpul sebelum A yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan. Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke A, dan H, jumlah nilai perkiraan dari sebuah simpul ke simpul tujuan. Simpul tujuan yaitu simpul yang dituju. Rintangan adalah sebuah atribut yang menyatakan bahwa sebuah simpul tidak dapat dilalui oleh A. Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (*starting point*) menuju simpul tujuan dengan memperhatikan harga (F) terkecil.

A* memperhitungkan *node* dari *current state* ke tujuan denga fungsi heuristic, Algoritma ini juga mempertimbangkan *node* yang telah ditempuh selama ini dari initial state ke *current state*. Jadi jika ada jalan yang telah ditempuh sudah terlalu panjang dan ada jalan lain yang *node*-nya lebih kecil tetapi memberikan posisi yang sama dilihat dari goal, jalan yang lebih pendek yang akan dipilih.

Algoritma A* dapat dijelaskan dengan pseudocode dibawah ini :

1. Masukan *node* awal ke openlist
2. *Loop* Langkah – langkah di bawah ini :

   a. Cari *node* (n) dengan nilai f(n) yang paling rendah dalam *open list*. *Node* ini sekarang menjadi current *node*.

   b. Keluarkan current *node* dari openlist dan masukan ke close list

   c. Untuk setiap tetangga dari current *node* lakukan berikut :

   1. Jika tidak dapat dilalui atau sudah ada dalam close list, abaikan.

   2. Jika belum ada di *open list* . Buat current *node* parent dari *node* tetangga ini. Simpan nilai f,g dan h dari *node* ini.

   3. Jika sudah ada di *open list*, cek bila *node* tetangga ini lebih baik, menggunakan nilai g sebagai ukuran. Jika lebih baik ganti parent dari *node* ini di openlist menjadi current *node*, lalu kalkulasi ulang nilai g dan f dari *node* ini.

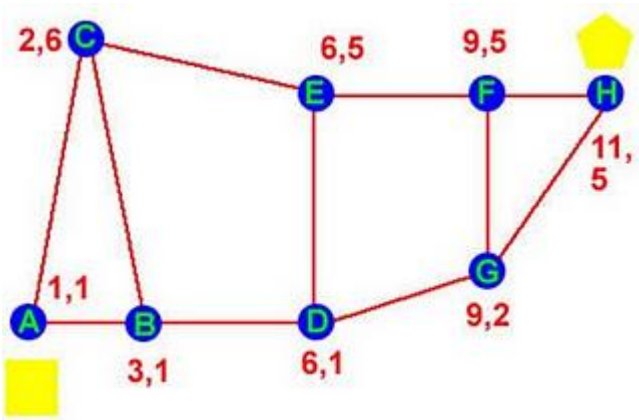
d. Hentikan *loop* jika :

1. *Node* tujuan telah ditambahkan ke *open list*, yang berate rute telah ditemukan.

2. Belum menemukan *node* goal sementara *open list* kosong atau berarti tidak ada rute.

3. Simpan rute. Secara ‘backward’, urut mulai dari*node* goal ke parent-nya terus sampai mencapai *node* awal sambil menyimpan *node* ke dalam sebuah array.

Pertama, kita misalkan ada persoalan menentukan rute seperti dibawah ini:



Gambar 2.2 Contoh Persoalan.

Segilima berwarna kuning adalah tempat tujuan dan kotak kuning adalah tempat asal, sedangkan titik biru adalah persimpangan setiap jalan.Setiap titik memiliki nilai X dan Y yang digunakan untuk perhitungan.

Langkah - langkah penentuan rute adalah :

1. Karena A adalah persimpangan terdekat dari tempat asal maka A masuk ke closelist.
2. Sedangkan H adalah persimpang terdekat dengan tempat tujuan, maka H masuk ke closelist. Kemudian dilakukan pengecekan percabangan dari A yaitu B dan C.B dan C masuk ke openlist.
3. Dilakukan perhintungan dengan algoritma A* diantar B dan C mana yang lebih dekat dengan H. Rumusnya :

$$d(x,y) = |\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} + .. + (x_1 - y_n)^2| = \sqrt{\sum_{i=0}^{n}}$$

4. Hasil pehitungan adalah

$$B - H = \sqrt{(11-5)^2} + (3-1)^2 + \sqrt{(11-3)^2} + (5-1)^2 = 15.26882723$$

$$B - H = \sqrt{(11-5)^2} + (2-6)^2 + \sqrt{(11-2)^2} + (5-6)^2 = 16.26648769$$

5. Maka persimpangan yang dipilih adalah B, maka B masuk ke closelist dan mengkosongkan openlist.
6. Lalu dilakukan pengecekan apakah H sudah masuk ke *close list*, jika belum maka dilakukan proses seperti no 2 - 6 hingga H masuk ke *close list*.
7. Sehingga jalur yang ditemukan adalah A - B - D - G - F.

2.4.2 *Collision detection*

Collision detection digunakan pada *video games*, simulasi fisika, dan geometrical komputasi. Pada *Collision detection* digunakan algoritma-algoritma untuk mengecek bertabrakan, contohnya adalah beririsan antara dua benda padat.

Pada awalnya video *game*s adalah *game* berbasis 2D dan memiliki karakter yang terbatas sehingga *Collision detection* secara manual dilakukan untuk mendeteksi adanya tabrakan. Namun, ketika dunia *game* mulai merambah dunia 3D dan dengan bertambahnya karakter pada *game* maka dibutuhkan algoritma pengecekan *Collision detection*. Jika tidak ada *Collision detection* maka karakter pada *game* bisa berjalan menembus tembok dan semua pembatas lainnya.

2.4.2.1 General Collisions Detections for *Game*s Using Ellipsoids

Teknik ini memungkinkan anda untuk memindahkan elips melalui *world* yang bukan hanya mendeteksi tubrukan tetapi juga beraksi seperti yang *gamers* inginkan dari *first person shooter* pada umumnya.

Teknik ini juga bisa digunakan untuk perpindahan melalui permukaan sebagaimana mudahnya dalam implementasi grafitasi, yang termasuk di dalamnya adalah perpindahan menurun ketika benda tersebut berdiri, secara otomatis memanjat tangga dan berpindah melalui tembok [9].

Beberapa kosakata yang berkaitan dengan ini :

a. *Source*

Titik Sumber di mana dimulainya objek yang tertubruk. Setiap objek yang tertubruk berjalan dari sumber menuju sebuah tubrukan dalam perjalanannya menuju suatu tujuan.

b. *Destination*

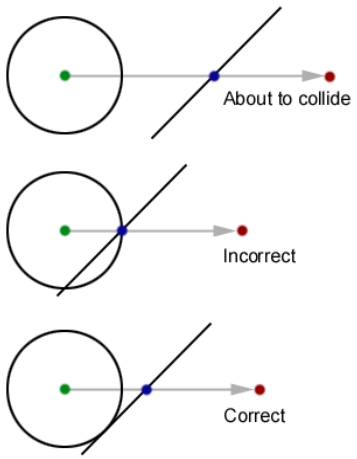
Lokasi yang dituju oleh objek yang akan mengalami tubrukan, tetapi tidak akan bisa sampai ke tujuan karena akan terlalu sulit untuk bertabrakan dengan benda lain.

c. *Velocity Vector*

Vektor yang mendefinisikan arah dan kecepatan yang akan menubruk objek yang sedang mengalami perjalanan (*travelling object*). Jika *travelling object* adalah 3 meter ke depan, maka arah *vector* ini adalah ke depan dan dengan panjang dari *vector* ini adalah 3 meter.

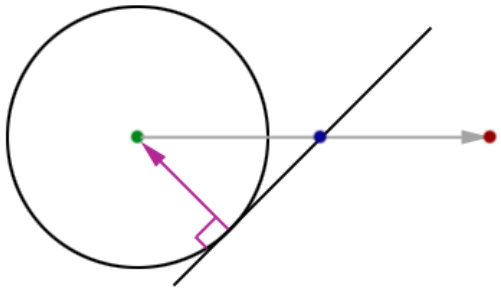
2.4.2.2 Permukaan dan Lingkaran

Tubrukan antara sebuah lingkaran dengan sebuah permukaan



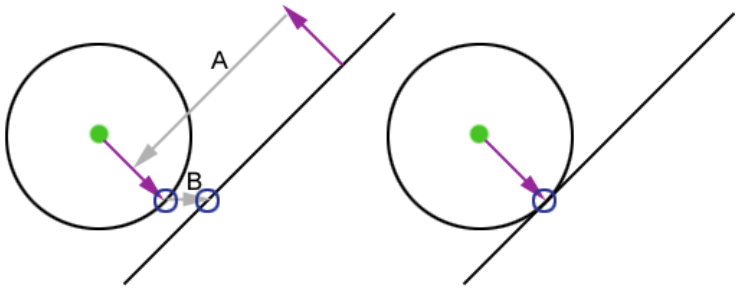
Gambar 2.3 Sebuah lingkaran menabrak sebuah permukaan (garis) [9].

Pada gambar 2.3, kita bisa melihat bahwa lingkaran bisa menabrak permukaan. Namun tubrukan lingkaran dengan permukaan tidak semudah yang kita bayangkan. Tubrukan tersebut bisa saja terjadi tidak benar yaitu ketika lingkaran sudah melewati garis.



Gambar 2.4 Tubrukan yang benar antara garis dan lingkaran [9].

Pada gambar 2.4, kita bisa melihat bahwa *vector* dibentuk dari titik perpotongan dan pusat dari lingkaran, adalah terjadi bersamaan dengan garis normal dari garis. Kita bisa membalik proses ini untuk memutuskan titik pada garis yang terdapat pada lingkaran yang akan berpotongan dengan garis sebelum kita melakukan tubrukan. Pada gambar 2.5 akan diperlihatkan lebih jelas lagi

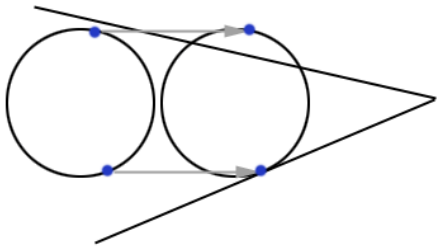


Gambar 2.5 Garis untuk menemukan titik potong lingkaran [9].

Gambar 2.5 menunjukkan bahwa garis normal dari garis bisa digunakan untu mendapatkan titik pada permukaan lingkaran yang akan beririsan dengan garis. Kita sebut dengan *titik potong lingkaran.* Untuk mendapatkan titik potong lingkaran kita harus menentukan garis normal dari garis ke jari-jari lingkaran lingkaran. Lalu tarik garis abu-abu dari garis normal menuju garis jari-jari yang disebut sebagai garis A dan kita akan melihat bahwa hasilnya adalah *vector* titik potong lingkaran (yang berupa garis yang berlawanan arah dari garis normal dan diambil dari titik tengah lingkaran). Dari titik yang didapatkan, kita pindahkan *velocity vector* (garis B berwarna abu-abu), hingga kita memotong garis. Sehingga lingkaran kita akan menuju garis dengan baik.

2.4.2.3 Bertubrukan dengan banyak garis

Jika kita bertubrukan dengan sebuah garis yang mana garis normalnya adalah jauh dari permukaan lingkaran kita maka kita akan membiarkan garis itu.



Gambar 2.6 Tubrukan yang terjadi dengan banyak garis [9].

Disamping dari ilusi *optic*, menunjukkan pada kita bahwa kita harus berhati-hati ketika ada lebih dari satu garis yang akan kita tabrak pada jalur kita. Solusinya mudah, kita harus menentukan semua tabrakan yang berpotensi kemudian memutuskan tabrakan yang mana yang akan terjadi pertama kali.

2.4.2.4 Memilih Tubrukan yang Potensial

Metode yang akan kita gunakan untuk melakukan ini bergantung pada data yang kita miliki, bagiamana kita mengorganisirnya, bagaimana kita berhubungan dengan tubrukan, dan sebagainya. Tapi ada beberapa solusi sederhana yang memenuhi kasus tersebut, yaitu:

1. Temukan kotak batasan dari lingkaran, terpusat pada posisi *source.*
2. Temukan kotak batasan dari lingkaran, terpusat pada posisi *destination.*
3. Temukan kotak batasan dari kedua kotak batasan tadi.
4. Temukan seluruh *polygon* yang beririsan dengan kotak batasan akhir.

2.5 Metode yang Digunakan

2.5.1 OOP (*Object Oriented Programming*)

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk

menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri [10].

Pemrograman orientasi-objek menekankan konsep berikut:

a. Kelas (*Class*) — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *'class of dog'* adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.
b. Objek (*Object*) - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
c. Abstraksi (*Abstract*) - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya,

dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

d. Enkapsulasi (*Encapsulation*) - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

e. Polimorfisme (*Polimorfism*) melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan

dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

f. Inheritas (*Inheritance*)- Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

2.5.2 UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun system *Unified Modeling Language* (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 adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi.

UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson.

Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan system [11].

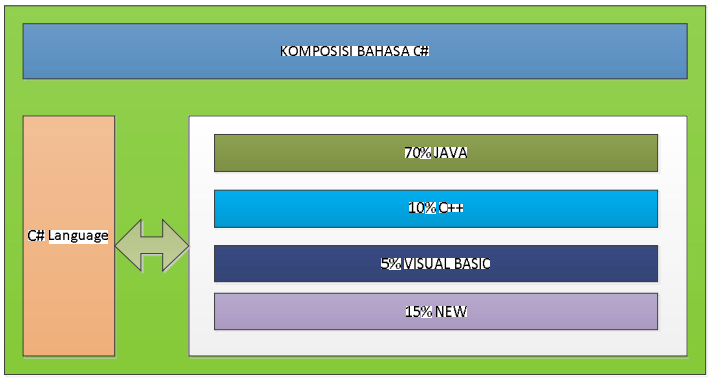
UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

a. *Use Case Diagram* untuk memodelkan proses bisnis.
b. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
c. *Sequence Diagram* untuk memodelkan pengiriman pesan (message) antar objects.
d. *Collaboration Diagram* untuk memodelkan interaksi antar objects.
e. *State Diagram* untuk memodelkan perilaku objects di dalam sistem.
f. *Activity Diagram* untuk memodelkan perilaku Use Cases dan objects di dalam *system*.
g. *Class Diagram* untuk memodelkan struktur kelas.
h. *Object Diagram* untuk memodelkan struktur object.
i. *Component Diagram* untuk memodelkan komponen object.
j. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

2.5.3 C#

Bahasa C# adalah sebuah bahasa pemrograman modern yang bersifat general-purpose, berorientasi objek, yang dapat digunakan untuk membuat program di atas arsitektur *Microsoft .NET Framework*. Bahasa C# ini memiliki kemiripan dengan bahasa Java, C dan C++ (selengkapnya dapat dilihat pada Sejarah Bahasa C#) [12].

Bahasa pemrograman ini dikembangkan oleh sebuah tim pengembang di Microsoft yang dipimpin oleh Anders Hejlsberg, seorang yang telah lama malang melintang di dunia pengembangan bahasa pemrograman karena memang ialah yang membuat *Borland Turbo Pascal, Borland Delphi*, dan juga Microsoft J++. Kini, C# telah distandarisasi oleh *European Computer Manufacturer Association* (ECMA) dan juga International *Organization for Standardization* (ISO) dan telah menginjak versi 5.0 yang mendukung beberapa fitur baru semacam *Language Integrated Query* (LINQ) dan lain-lainnya [13].



Gambar 2.7 Komposisi bahasa C# [13].

2.5.4 *Javascript*

Javascript adalah bahasa pemrograman berbasis prototipe yang berjalan disisi klien. Jika kita berbicara dalam konteks web, sederhananya, kita dapat memahami *Javascript* sebagai bahasa pemrograman yang berjalan khusus untuk di *browser* atau halaman web agar halaman web menjadi lebih hidup. Kalau dilihat dari suku katanya terdiri dari dua suku kata, yaitu Java dan Script. Java adalah Bahasa pemrograman berorientasi objek, sedangkan Script adalah serangkaian instruksi program. Secara fungsional, *Javascript* digunakan untuk menyediakan akses script pada objek yang dibenamkan (*embedded*). Contoh sederhana dari penggunaan *Javascript* adalah membuka halaman *pop up*, fungsi validasi pada form sebelum data dikirimkan ke *server*, merubah *imag*e kursor ketika melewati objek tertentu, dan lain lain.

2.5.4.1 Kelebihan *Javascript*

Javascript bekerja pada sisi *browser*. maksudnya begini : untuk menampilkan halaman web, *user* menuliskan alamat web di address bar url. setelah itu, *browser* "mengambil" file html (dengan file *Javascript* yang melekat padanya jika memang ada) ke *server* yang beralamat di URL yang diketikan oleh *user*. Selesai *file* diambil, file ditampilkan pada *browser*. Nah, setelah file *Javascript* berada pada *browser*, barulah script *Javascript* tersebut bekerja.

Efek dari *Javascript* yang bekerja pada sisi *browser* ini, *Javascript* dapat merespon perintah *user* dengan cepat, dan membuat halaman *web* menjadi lebih

responsif. *Javascript* melakukan apa yang tidak bisa dilakukan oleh HTML, PHP, dan CSS : menangani hal – hal yang membutuhkan respons cepat terhadap aksi dari *user*.

Contoh : fungsi validasi pada form. ketika anda mengisi sebuah form yang divalidasi menggunakan *Javascript*, anda mengetikkan data lalu mengetik submit, sebelum data dikirimkan ke *server*, data akan "dicek" terlebih dahulu pada *browser* menggunakan fungsi *Javascript* yang ada pada halaman web. sehingga, jika memang data yang anda isikan tidak valid, daripada membuang – buang waktu dengan mengirimkan data ke *server* baru di validasi di *server* dan lalu *server* mengirimkan respons balik mengenai ketidak validan input data anda, lebih baik cek validasi data form dilakukan secara lokal di *browser* menggunakan fungsi *Javascript* [14].

Javascript bergantung kepada *browser* (navigator) yang memanggil halaman web yang berisi skrip-skrip dari *Javascript* dan tentu saja terselip di dalam dokumen HTML. *Javascript* juga tidak memerlukan kompilator atau penterjemah khusus untuk menjalankannya (pada kenyataannya kompilator *Javascript* sendiri sudah termasuk di dalam *browser* tersebut). Lain halnya dengan bahasa "Java" (dengan mana *Javascript* selalu di banding bandingkan) yang memerlukan kompilator khusus untuk menterjemahkannya di sisi *user*/klien. Tabel daftar navigator dan versi dari *Javascript* :

Tabel 2.3 Navigator

Versi Javascript	Browser
Javascript 1.2	Netscape Navigator 4.0/4.05, Internet Explorer 4.0
Javascript 1.1	Netscape Navigator 3.0
Javascript 1.0	Netscape Navigator 2.0, Internet Explorer 3.0
Javascript 1.5	Netscape Navigator 6.0
Javascript 1.3	Netscape Navigator 4.06, Internet Explorer 5.0
Javascript 1.4	Netscape Navigator 6.0, Internet Explorer 5.5

[14].

Javascript merupakan suatu bahasa yang perkembangannya lambat di bandingkan dengan Java yang berkembang sangat cepat. Di *Javascript* kita tidak mungkin menyembunyikan kode skrip yang kita tulis, kode langsung di tulis di dalam dokumen HTML dan sangat mudah terlihat, sedangkan di Java, kode sudah berbentuk setengah terkompilasi (dalam bentuk applet) dan tidak mungkin terlihat dari dalam dokumen HTML, satu mesin virtual di sisi *user* yang bertanggung jawab untuk menterjemahkan program di dalam applet tersebut setiap kali halaman HTML yang memuat applet tersebut dipanggil oleh *browser*. Dibandingkan dengan applet java yang cukup lambat dibuka oleh *browser*, bisa kita katakan bahwa *Javascript* cukup cepat di panggil (di *load*) oleh navigator [15].

Javascript sendiri merupakan bahasa yang mudah dipahami, dalam artian diperlukan skill novice atau dasar untuk mengerti bahasa ini, jika anda sudah terbiasa dan mengenal konsep bahasa pemrograman visual, maupun Java ataupun C, akan sangat mudah untuk memahami konsep *Javascript*. Berikut ini satu tabel yang berisi beberapa perbandingan mendasar antara Java dan *Javascript* :

Tabel 2.4 Perbandingan *Javascript* dan JAVA [15].

JavaScript	Java
Bahasa yang diintepretasikan langsung oleh browser	Bahasa yang setengah terkompilasi dan memerlukan Java Virtual Machine untuk menterjemahkannya
Kode terintegrasi dengan HTML	Kode(applet) terpisah dari dokumen HTML, dipanggil pada saat membuka dokumen HTML
Bahasa dengan karakteristik yang terbatas	Bahasa dengan karakteristik yang luas (pendeklarasian jenis variabel)
Hubungan dinamis, referensi dari obyek diverifikasi pada saat loading.	Hubungan statis, obyek harus ada pada saat program di loading (di kompilasi)
Kode program bisa di akses	Kode program tersembunyi

Javascript adalah bahasa yang "case sensitive" artinya membedakan penamaan variabel dan fungsi yang menggunakan huruf besar dan huruf kecil, contoh variabel atau fungsi dengan nama TEST berbeda dengan variabel dengan nama test. Dan yang terakhir seperti bahasa Java ataupun C, setiap instruksi diakhiri dengan karakter titik koma (;).

2.5.5 *Game Engine*

Game engine adalah perangkat lunak yang dirancang untuk membuat dan mengembangkan video *game*. Fungsi utama yang diberikan oleh *game engine* meliputi rendering untuk 2D atau 3D *graphic*, *Collision detection*, *sound*, scripting, animasi, *Artificial Intelligence, networking, memory management, threading* dan *scene graph*. *Game engine* memberikan perangkat untuk *visual developmen*t dengan tambahan komponen perangkat lunak yang dapat dipakai berulang kali. Perangkat ini pada umumnya memberikan *integrated development*

environment yang dapat mempermudah, serta mempercepat pengembangan *game* [16].

Game engine juga sering disebut juga "*Game Middleware"*. *Game Middleware* adalah software yang didalamnya sudah terdapat elemen-elemen yang dibutuhkan oleh seorang *game developer*, kebanyakan dari *game middleware* memberikan fasilitas guna menunjang pengembangan *game* dengan mudah seperti *graphics*, *sound* dan *Artificial Intelligence*. Beberapa elemen yang ada di dalam *game engine* adalah :

a. Tools/Data

Dalam pengembangan *game*, dibutuhkan data yang tidak semudah menuliskan text files. Dalam pengembangan *game*, paling tidak dibutuhkan beberapa tools seperti 3d *model editor, level editor dan graphics programs*. Bahkan jika diperlukan, seringkali kita mengembangkan *game engine* tersebut dengan menambahkan beberapa code dan fitur yang diperlukan.

b. System

System sendiri adalah bagian dari *game engine* yang berfungsi untuk melakukan komunikasi dengan *hardware* yang berada di dalam mesin. Jika *game engine* sudah dibuat dengan baik maka system ini adalah satu-satunya bagian yang membutuhkan perubahan yang cukup banyak apabila dilakukan implementasi pada platform yang berbeda. Di dalam system sendiri terdapat beberapa sub system yaitu *graphics, input, sound, timer, configuration*. System sendiri bertanggung jawab untuk melakukan inisialisasi, update dan mematikan sub *system* yang terdapat di dalamnya.

c. *Console*

Dengan menambahkan console, kita dapat merubah setting *game* dan setting *game engine* di dalam *game* tanpa perlu melakukan restart pada *game* tersebut. Console sendiri lebih sering digunakan dalam proses *debugging*. Apabila *game engine* tersebut mengalami error kita tinggal mengoutputkan *error message* tersebut ke dalam *console* tanpa harus melakukan *restart*. *Console* sendiri dapat dihidupkan dan dimatikan sesuai keinginan.

d. *Support*

Support adalah bagian yang paling sering digunakan oleh *system* di dalam *game engine* . *Support* sendiri berisi rumus-rumus matematika yang biasa digunakan, *vector, matrix, memory manager, file loader*. Merupakan dasar dari *game engine* dan hampir digunakan semua projek *game engine*.

e. *Renderer/Engine Core*

Pada *game engine* , *engine core / renderer* terdiri dari beberapa sub yaitu *visibility*, *Collision detection* dan *Response, Camera, Static Geometry, Dynamic Geometry, Particle Systems, Billboarding, Meshes, Skybox, Lighting, Fogging, Vertex Shading,* dan *Output.*

f. *Game Interface*

Game interface sendiri merupakan layer diantara *game engine* dan *game* itu sendiri. Berfungsi sebagai *control* yang bertujuan untuk memberikan *interface* apabila di dalam *game engine* tersebut terdapat fungsi fungsi yang bersifat dinamis sehingga memudahkan untuk mengembangkan *game* tersebut.

g. The Game

Merupakan inti dari penggunaan *game engine* sendiri, sehingga terserah kita bagaimana mengembangkan *game* tersebut.

2.5.6 Unity 3D

Unity adalah *Tools* yang terintegrasi untuk menciptakan video *game* 3D atau konten interaktif lainnya seperti visualisasi arsitektur atau real-time 3D animasi. Lingukangan pengembangan *game* yang dibuat oleh Unity dapat berjalan di *Windows, MAC OSX*. Dan permainan yang dihasilkan dapat dipublish dan dijalakan di *Windows, MAC OSX, iOS, iPhone, iPod, Android*, dan *Web*. Hal ini juga permainan *browser* semacam *Flash* dengan menggunakan *Unity Web player* yang didukung oleh sistem oprasi *Windows* dan *MAC OS*, tetapi tidak untuk *Linux*.

Unity 3D terdiri dari editor untuk mengembangkan atau merancang content dari *game engine* untuk mengeksekusi produk akhir dan mempublisnya. Unity 3D mirip dengan *game engine* lain seperti *Blender game engine , Shiva, Unreal engine , Cry engine* dan *game* studio lainnya yang juga menggunakan grafis terintegrasi sebagai metode utama pembangunan *game* [17].

2.5.7 *3D Studio Max*

3D Studio Max atau biasa dikenal dengan *3D Max* adalah suatu *software* (perangkat lunak) untuk membuat sebuah grafik vektor 3 dimensi dan animasi. ditulis oleh *Autodesk Media & Ente*rtainment, dulunya dikenal sebagai *Discreet* dan *Kinetix*. *3D Studio Max* dikembangkan dari pendahulunya yaitu *3D Studio for DOS*, tetapi untuk *platform Win32*.

Kinetix kemudian bergabung dengan akuisisi terakhir *Autodesk*, *Discreet Logic*. Yang sampai saat penulis membuat tugas akhir ini yang terbaru adalah *3D Studio Max* 2012. Para desain grafis banyak menggunakan *software* ini digunakan untuk membuat sebuah film animasi, arsitektur rumah, ataupun membuat logo suatu perusahaan [18].

Untuk menjalankan aplikasi ini membutuhkan spesifikasi komputer yang cukup tinggi agar mendapat kenyamanan saat menggunakannya. Karena banyak proses yang membutuhkan grafik yang tinggi untuk melakukan rendering dan sebagainya.