BAB 2

LANDASAN TEORI

2.1 Teori Umum

2.1.1.Pemrograman Berorientasi Obyek

Menurut Goran Svenk, Pemrograman berorientasi obyek adalah pendekatan penyelesaian masalah dengan melakukan pengaplikasian logika yang digunakan untuk menyelesaikan masalah dalam kehidupan sehari-hari. Dimana konsep ini menggunakan obyek untuk berinteraksi satu sama lain untuk menyelesaikan suatu masalah. Interaksi dapat dilakukan dalam bentuk pengiriman pesan, dan obyek dapat merespon pesan tersebut dengan cara melakukan *action* dengan *method*. Pemrograman berorientasi obyek mengimplementasikan konsep-konsep penting sebagai berikut :

1. *Encapsulation*

   Encapsulation adalah konsep dimana data dan fungsi dikelompokkan secara bersama dalam suatu kapsul atau obyek. Konsep ini juga disebut sebagai *data abstraction*. Ini mengimplementasikan *data hiding*, dimana obyek dapat menyembunyikan data dari program dan mengatur akses data hanya untuk fungsi yang didefinisikan dapat mengakses data tersebut. Ini akan mengurangi kesalahan memodifikasi data dan kesalahan logika.

2. *Polymorphism*

   Polymorphism adalah konsep yang menggunakan fungsi yang sama di berbagai tipe obyek yang berbeda. *Polymorphism* merupakan kemampuan dua obyek yang berbeda untuk merespon pesan yang dikirimkan dengan cara mereka masing-masing. Dalam *OOP*, *Polymorphism* diimplementasikan dalam konsep *Overloading*, dimana *method* yang berbeda dalam satu obyek memiliki nama yang sama. Obyek akan memberitahu *method* mana yang akan dieksekusi sesuai dengan isi argument dari pesan tersebut. *OOP tools* ini membuat para *programmer* C++ dapat mengurangi waktu pengembangan program.

3. *Inheritance*

   Inheritance adalah konsep yang paling penting dalam pemrograman berorientasi obyek yang dapat mengimplementasikan *code reusability*. Ketika menggunakan *inheritance*, kode baru dapat mewarisi sifat dan data dari kode induk. Dengan *inheritance*, *programmer* juga dapat mengkombinasikan karakteristik dari berbagai kelas induk dan mewarisi karakteristik tersebut kepada kelas anak. Hal ini dapat mengurangi jumlah kode dan ukuran program serta membuat pembuatan program menjadi lebih mudah.

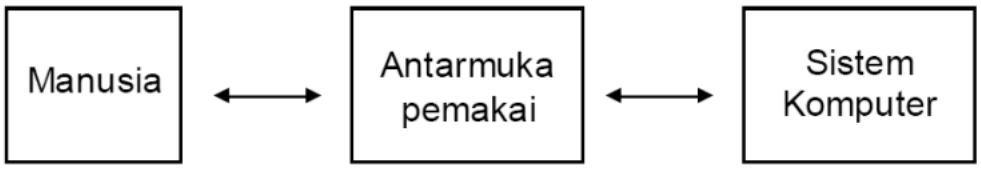
2.1.2.*Platform* (Windows 32-Bit)

Windows 32 bit menunjuk kepada sistem operasi windows atau aplikasi Windows yang diciptakan untuk berjalan pada *CPU* 32-bit. Istilah ini pada awalnya digunakan untuk membedakannya dengan aplikasi Windows 16 bit. Tapi dewasa ini, istilah ini lebih sering digunakan untuk membedakannya dengan aplikasi Windows 64 bit.

Windows adalah sebuah sistem operasi yang dikembangkan oleh Microsoft. Microsoft pertama kali memperkenalkan Windows pada 1985 sebagai *add-on* untuk MS-DOS dalam menyikapi cepatnya pertumbuhan ketertarikan pengguna terhadap aplikasi yang memiliki antarmuka pemakai. Sejak saat itu Microsoft terus mengembangkan Windows dan berhasil menjadikannya sebagai salah satu penguasa pasar personal komputer.

2.1.3.Interaksi Manusia dan Komputer

Menurut Shneiderman, interaksi manusia dan komputer atau yang biasa disebut *Human-Computer interaction* (HCI) adalah sebuah disiplin ilmu yang mempelajari bagaimana manusia berinteraksi dengan computer dan pengaruh dari interaksi antara manusia dan komputer tersebut.



Gambar 2.1 Fokus pada interaksi manusia dan komputer

Fokus utama pada Interaksi Manusia Komputer adalah perancangan evaluasi antarmuka pemakai. Antarmuka pemakai adalah bagian sistem komputer yang memungkinkan manusia berinteraksi dengan komputer. Ada 8 aturan emas yang harus diperhatikan dalam perancangan sistem antarmuka pemakai. Berikut adalah 8 aturan tersebut:

- Berusaha untuk konsisten
- Memungkinkan adanya penggunaan *shortcut*
- Memberikan umpan balik yang informatif
- Merancang dialog untuk menampilkan keadaan terakhir.
- Memberikan pencegahan kesalahan dan penanganan yang sederhana.
- Memungkinkan adanya pembalikan aksi yang mudah.
- Mendukung pusat kendali internal.
- Mengurangi beban ingatan jangka pendek.

Agar suatu sistem dikatakan *user friendly*, ada beberapa kriteria yang harus dipenuhi, yaitu:

- Proses pembelajaran yang tidak lama
- Tingkat kecepatan dalam penyampaian informasi
- Tingkat kesalahan
- Penghafalan sesudah melampaui jangka waktu tertentu
- Kepuasan pribadi
- Kondisi dan aksi alternative harus bisa dilihat oleh pemakai

- Harus ada perumusan konseptual yang baik dan sebuah gambaran sistem yang konsisten
- Pemetaan yang baik pada antarmuka yang dibuat
- Pemakai harus mendapat umpan balik yang terus menerus.

2.1.4.Unified Modeling Language

Menurut Whitten, Bentley, dan Dittman, *Unified Modeling Language* atau biasa disebut *UML* adalah sebuah pendekatan untuk mempelajari obyek-obyek yang ada untuk melihat apakah obyek tersebut dapat digunakan kembali atau dimodifikasi untuk kegunaan baru, dan mendefinisikan obyek baru atau obyek yang telah dimodifikasi yang akan digabung dengan obyek yang sudah ada untuk membuat aplikasi bisnis.

Berdasarkan sudut pandangnya, *UML* terdiri dari sembilan diagram yang diklasifikasikan ke dalam lima kategori, yaitu:

Kategori 1: *Use-Case Model Diagram*

Use-case diagram menggambarkan interaksi antara sistem dengan luar sistem serta antara sustem dengan pengguna. Dengan kata lain, *use case* diagram menggambarkan siapa yang akan menggunakan sistem dan dalam cara apa pengguna berinteraksi dengan sistem.

Kategori 2 : *Static Structure Diagram*

UML menawarkan dua diagram untuk memodelkan struktur statis dari sistem informasi, yaitu:

- *Class diagram*

  Class diagram menggambarkan struktur sebuah sistem dalam bentuk obyek. Disini digambarkan obyek *class* yang membangun sistem serta hubungan antar *class*.

- *Object diagram*

  Object diagram mirip dengan *class diagram*, tapi selain menggambarkan *object class*, *object diagram* juga menggambarkan *object instance* yang menampilkan nilai atribut dari *instance*. Diagram ini biasa digunakan untuk membantu memahami struktur sistem dengan baik.

Kategori 3 : *Interaction diagram*

Interaction diagram memodelkan interaksi, terdiri dari sekumpulan obyek, hubungan antar obyek, serta pesan-pesan yang dikirimkan antar obyek tersebut. Diagram jenis ini memodelkan aspek dinamis dari sistem. UML memiliki dua diagram pada kategori ini, yaitu:

- *Sequence diagram*

  Sequence diagram menggambarkan bagaimana obyek-obyek yang ada saling berinteraksi melalui pesan dalam melakukan suatu operasi atau melakukan *use case*.

- *Collaboration diagram*

  Collaboration diagram mirip dengan *sequence diagram*, tetapi *collaboration diagram* lebih terfokus pada interaksi antar obyek dalam format jaringan.

Kategori 4 : *State diagram*

Seperti *interaction diagram*, *state diagram* juga memodelkan aspek dinamis dari sistem. *UML* memiliki diagram untuk memodelkan perilaku kompleks dari obyek dan diagram untuk memodelkan perilaku dari *use case* atau metode. Ada 2 diagram yang termasuk dalam jenis ini, yaitu:

- *Statechart diagram*

  Statechart diagram digunakan untuk memodelkan aspek dinamis dari suatu obyek. Pada *statechart diagram* diilustrasikan daur hidup obyek, berbagai keadaan obyek, dan peristiwa yang menyebabkan transisi dari keadaan yang satu ke keadaan yang lain.

- *Activity diagram*

  Activity diagram digunakan untuk menggambarkan urutan aktivitas secara berurutan dari proses bisnis atau *use case*.

Kategori 5 : *Implementation diagram*

Implementation diagram juga memodelkan struktur dari sistem informasi. Yang termasuk dalam *implementation diagram* adalah:

- *Component diagram*

  Component diagram digunakan untuk menggambarkan organisasi dari sistem dan ketergantungan dari komponen *software* dalam sistem. *Component diagram* dapat juga digunakan untuk menunjukkan bagaimana kode program dibagi-bagi menjadi modul-modul.

- *Deployment diagram*

  Deployment diagram mendeskripsikan arsitektur fisik dalam 'node' untuk *hardware* dan *software* dalam sistem. Pada bagian ini, digambarkan konfigurasi dari komponen *software, processor*, dan peralatan lain yang membangun arsitektur sistem secara *run-time*.

2.1.5.Bahasa Pemrogaman (C++)

Menurut Deitel, C++ adalah bahasa pemrograman yang dikembangkan oleh Bell Lab pada awal tahun 1970-an. Bahasa ini diturunkan dari bahasa sebelumnya, yaitu B yang diturunkan oleh bahasa sebelumnya, yaitu BCL. Awalnya bahasa tersebut dirancang sebagai bahasa pemrograman yang dijalankan pada sistem operasi UNIX. C merupakan bahasa pemrograman yang structural dimana penyelesaian atas suatu masalah dilakukan dengan membagi-bagi masalah tersebut menjadi sub-submasalah yang lebih kecil. Sedangkan C++ merupakan bahasa pemrograman yang memiliki sifat *Object Oriented Programming* (OOP). Karena mendukung konsep *Object Oriented Programming* maka C++ menjadi bahasa pemrograman yang dominan di industry dan pendidikan.

Obyek merupakan komponen penting software untuk mendukung *code reusability* yang merupakan model yang sesuai dengan kehidupan sebenarnya. Dengan pendekatan *object oriented design* maka pengembangan software akan lebih produktif dibandingkan dengan teknik pengembangan software sebelumnya.

2.1.6.Multimedia

Menurut Vaughan, multimedia berasal dari kata multi dan media. Jadi multimedia adalah rangkaian kombinasi dari manipulasi suatu teks, foto, seni grafis, suara, animasi, dan elemen *video*. Multimedia terdiri dari beberapa elemen yang sering dikombinasikan agar dapat memperoleh hasil yang terbaik. Berikut adalah elemen-elemen multimedia:

- Teks

  Teks merupakan jenis penyampaian informasi yang paling umum dan sederhana. Penggunaan teks pada multimedia memiliki beberapa keuntungan dan kelemahan. Keuntungan dengan menggunakan media teks adalah teks tidak membutuhkan ruang yang besar pada media penyimpanan. Tapi teks memliki beberapa kekurangan seperti penggunaan yang terlalu lama akan membuat pembaca menjadi bosan serta membuat mata lelah.

- *Image*

  Image merupakan representasi khusus dari obyek yang disusun oleh matriks nilai numerik yang merepresentasikan setiap *pixel* yang

diciptakan dengan program *image editing*. *Image* memiliki peran yang penting dalam multimedia karena dengan 1 gambar bisa mewakili ribuan kata dan tidak mengenal batasan bahasa-bahasa yang ada di dunia.

- Suara

  Suara merupakan elemen yang menggunakan telinga sebagai sarana untuk menangkap informasi. Dengan menggunakan suara, informasi yang diterima bisa lebih banyak karena selain melalui komunikasi visual, komunikasi verbal juga ikut membantu penyaluran informasi.

- Animasi

  Animasi adalah kumpulan *frame* gambar yang disusun sedemikian rupa sehingga menghasilkan kesan gerakan. Animasi memegang peranan yang cukup penting karena animasi bisa membuat tampilan menjadi lebih menarik. Tapi perlu diingat jangan sampai animasi yang dibuat menggangu konsentrasi pengguna.

- Video

  Video merupakan elemen yang paling kompleks dan paling berat karena merupakan gabungan dari animasi dan suara. Hal ini juga menjadikan video sebagai sarana informasi yang paling lengkap, akan tetapi video membutuhkan ruang yang cukup besar dan persyaratan *hardware* yang lebih tinggi.

2.2 Teori Khusus

2.2.1 Pengertian *Game*

Sampai saat ini, masih bisa ditemukan banyaknya diskusi mengenai pengertian dari *game* sehingga muncul berbagai macam definisi yang berbeda-beda. Sehingga akan lebih mudah jika mengatakan apa yang bukan merupakan sebuah *game*.

Sebuah film bukanlah sebuah *game*

Hal utama yang membedakan sebuah *game* dan sebuah film adalah pada film tidak ada partisipasi secara aktif dari penonton. Penonton tidak memiliki kendali atas film tersebut serta tidak dapat memberi keputusan yang mempengaruhi hasil akhir dari film tersebut. Selain itu, hasil akhir dari sebuah film sudah ditetapkan sebelumnya. Hal ini merupakan aspek penting dari sebuah film. Sedangkan dalam *game* adalah hal yang sebaliknya, orang-orang tidak menyukai jika hasil akhir dari suatu *game* telah ditentukan. Mereka ingin mempengaruhi hasil akhir, mereka ingin mempunyai kendali.

Sebuah mainan bukanlah sebuah *game*

Pada mainan tidak ada tujuan yang telah ditentukan sebelumnya, walaupun pada saat bermain orang cenderung menetapkan tujuan tersebut. Beberapa *game* komputer sebenarnya menyerupai mainan. Sebagai contoh, pada *game* SimCity atau The Sims tidak ada tujuan yang telah ditetapkan

sebelumnya. User dapat menciptakan sendiri tujuan yang ingin dicapai, tapi hal ini bukanlah suatu akhir yang natural.

Sebuah program menggambar bukanlah sebuah *game*

Program menggambar sangat menarik untuk dimainkan dan melatih kreativitas, tapi pada program menggambar tidak ada tujuan yang jelas. User menetapkan tujuan tersebut dan user juga yang menentukan apakah tujuan tersebut sudah dicapai atau belum.

Sebuah *puzzle* bukanlah sebuah *game*

Sebenarnya banyak *game* yang menyertakan elemen *puzzle* dalam permainannya. Tapi perlu dicatat bahwa sebuah *puzzle* bersifat statis, sedangkan *game* bersifat dinamis dan terus berubah sepanjang permainan. Sebuah *game* bisa dikatakan memuaskan jika bisa dimainkan berulang-ulang dan ada berbagai strategi yang bisa diterapkan untuk mencapai kesuksesan.

Dari penjelasan diatas, bisa ditarik kesimpulan bahwa sebuah (komputer) *game* adalah:

"sebuah program dimana di dalamnya satu atau lebih pemain membuat keputusan melalui kontrol atas obyek-obyek dan sumber daya yang terdapat pada *game*, dalam mencapai suatu tujuan."

Definisi di atas sama sekali tidak berbicara mengenai grafis, efek suara, atau *movies* yang terdapat di dalam *game*. Hal-hal tersebut memang memiliki peranan dalam menciptakan sebuah *game* yang menarik, tapi bukan merupakan hal yang esensial. Penjelasan lebih detail mengenai definisi tersebut adalah sebagai berikut:

Sebuah *computer game* adalah sebuah program

Kebanyakan *computer game* memiliki elemen *real-time*, permainan tetap berlanjut meskipun pemain tidak melakukan apapun. Hal ini bisa menambah kesenangan serta memunculkan perasaan hadir dalam dunia *game* yang lebih baik. Selain itu, *computer game* dapat disesuaikan sehingga mampu memuaskan jenis-jenis pemain yang berbeda, baik seorang pemula ataupun bukan. *Computer game* juga dapat menjadi lebih kompleks karena *game* tersebut bisa membantu pemain memahami berbagai macam aspek serta mengajari pemain cara bermain. Dengan menambahkan grafis dan musik yang indah, *Computer game* dapat menciptakan lingkungan yang lebih real.

Sebuah computer *game* melibatkan pemain

Game bukanlah sesuatu untuk ditonton, pemain harus terlibat di dalam permainan. *Game* tertentu ditujukan untuk pemain tertentu, misalnya *game* untuk anak-anak pasti berbeda dengan *game* untuk orang dewasa.

Bermain *game* adalah tentang membuat keputusan

Pemain membuat keputusan yang mempengaruhi jalannya permainan. Pada *game* yang memiliki aksi-aksi yang cepat, membuat keputusan biasanya ada pada kearah mana pemain harus bergerak, atau harus menggunakan senjata apa. Menyeimbangkan keputusan dan efek dari keputusan tersebut adalah hal yang penting dalam menciptakan permainan yang menarik.

Bermain *game* adalah tentang memegang kendali

Pemain harus dibuat seolah-olah memiliki kontrol didalam *game* tersebut. Lebih banyak kebebasan yang dirasakan oleh pemain akan lebih baik. Tapi dalam *game*, elemen kejutan dan efek-efek dramatis juga memegang peranan penting. Dan hal tersebut bisa dimunculkan secara lebih baik ketika pemain tidak sedang memegang kendali. Jika ingin membatasi pergerakan pemain, lakukankah dengan cara yang alami.

Obyek *game* dan sumber daya

Pemain biasanya mengendalikan sebuah obyek dalam suatu *game*, entah itu berupa karakter utama, mobil, atau benda-benda lainnya. Sedangkan dalam *game-game* yang lebih kompleks, pemain bisa mengendalikan lebih dari satu obyek. Selain obyek-obyek yang diatur oleh pemain ada juga obyek-obyek lain yang dikendalikan oleh komputer. Disamping mengendalikan obyek-obyek tertentu, pemain biasanya juga harus mengatur sumber daya. Hal ini paling jelas terlihat pada jenis *game* strategi dimana pemain harus mengatur sumber daya yang ada seperti makanan, kayu, emas, dan lain-lain.

Sebuah *game* harus memiliki tujuan

Tujuan adalah elemen krusial dari sebuah *game*. Pemain pasti ingin memenangkan *game* yang mereka mainkan, oleh karena itu diperlukan tujuan yang harus dicapai. Ketika tujuan dicapai, pemain biasanya akan mendapatkan hadiah.

2.2.2 Jenis-jenis *Game*

a. Arcade *game*s

Pada jenis *game* ini kecepatan reaksi memegang peranan penting. Contohnya adalah *game-game scrolling shooter*, *game* labirin seperti

pacman, beraneka macam platform *game*s, dan lain-lain. Biasanya jenis *game* ini menggunakan grafis 2 dimensi.

b. *Puzzle game*s

Kepintaran biasanya berperan penting pada jenis *game* ini. Contoh *puzzle game* yang cukup terkenal adalah Bejeweled dari Popcap Games. *Puzzle game* biasanya menggunakan grafis 2 dimensi.

c. *Role Playing Game*s (*RPG*)

Bagian paling penting dari *game RPG* adalah pengembangan karakter yang dikontrol oleh pemain. Karakter akan mempelajari jurus-jurus baru, menjadi lebih kuat, dan menemukan senjata yang lebih kuat. Dan pada saat yang bersamaan musuh-musuh yang ada juga menjadi lebih kuat. *Game RPG* biasanya disertai dengan alur cerita yang kuat dan pemain akan dituntut untuk mencaritahu apa yang sedang terjadi di dunia *game* tersebut. *Game RPG* biasanya menggunakan grafis *isometric* ataupun *full 3D*, tapi tidak menutup kemungkinan adanya *RPG* yang menggunakan grafis 2 dimensi. Contoh *game RPG* yang terkenal adalah seri Diablo dari Blizzard, seri Oblivion dari Bethesda, dan lain-lain.

d. *Strategy Game*s

Ada beberapa subgenre dari *game* strategi, misalkan *turn based strategy* dan *real time strategy*. Pada jenis *game* ini pemain biasanya tidak mengatur obyek secara langsung, melainkan mengatur strategi yang akan dijalankan oleh unit-unit di dalam *game*. Tampilan biasanya

menggunakan sudut isometris. Contoh *game real time strategy* misalnya Age of Empires, StarCraft. Contoh *game turn based strategy* misalnya Brigandine, Heroes Might and Magic series dan lain-lain.

e. *Management Game*s

Pada jenis *game* ini, pemain biasanya mengatur sebuah kota, taman bermain, perusahaan kereta api, dan lain-lain. Sudut pandang yang biasa digunakan adalah isometris. Mengatur sumber daya yang ada merupakan komponen penting dari *game* ini. *Game* ini termasuk *game* yang cukup rumit karena banyaknya sistem yang berjalan untuk mensimulasikan dunia *game* tersebut. Contohnya yang cukup popular adalah Tycoon *series* dan Sims City.

f. *Adventure Game*s

*Adventure game*s biasanya mementingkan sisi cerita. Jenis grafis yang digunakan bisa 2 dimensi ataupun 3 dimensi dan menggunakan antarmuka *point-and-click*. Contoh jenis *game* ini adalah Crash Bandicoot dan Ratchet and Clank.

g. First Person Shooter (*FPS*)

Game FPS bisa dianggap sebagai versi 3 dimensi dari *game-game arcade*. *Game FPS* membutuhkan dunia 3 dimensi agar lebih memunculkan perasaan hadir yang lebih nyata. Contoh *game FPS* adalah yang terkenal adalah Half Life, Doom, dan Quake.

h. Third Person Shooter

Game jenis ini biasanya menempatkan pemain sebagai seorang karakter di dunia yang liar. Perbedaan paling mencolok dari *game* jenis ini dengan *game RPG* adalah tidak adanya penekanan pada pengembangan karakter. Hal yang lebih ditekankan adalah aksi-aksi yang cepat dan penjelajahan dunia *game* tersebut. Banyak *game-game third person shooter* yang menggunakan fitur-fitur yang biasanya ada pada *game adventure*. Contohnya adalah Grand Theft Auto Series, Resident Evil, dan Tomb Raider.

i. *Sport game*s

Pada jenis *game* ini, olahraga seperti sepakbola, bola basket, dan *baseball* disimulasikan. Jenis *game* ini cukup ada banyak di pasaran dan cukup populer. Hal ini dikarenakan *game* seperti ini memiliki *replay-value* yang sangat tinggi sehingga bisa dimainkan berulang kali. Contoh *game* olahraga adalah Winning Eleven, Madden NFL, dan Wii Sport

j. *Racing Game*s

*Racing game*s merupakan *game* jenis *sport* yang cukup khusus karena memiliki berbagai macam kategori sendiri sehingga bisa dianggap menjadi 1 *genre* tersendiri. Beberapa *racing game*s, misalnya Gran Turismo berusaha menyajikan pengalaman berkendara yang seasli mungkin. Sedangkan pada seri Need For Speed lebih ditekankan aksi-aksi cepat.

k. *Simulator*

Game-game simulasi bertujuan untuk mensimulasikan sesuatu serealistis mungkin, misalkan menerbangkan pesawat. *Game* jenis ini populer karena banyak orang yang ingin mengerti bagaimana cara kerja suatu sistem dan bagaimana mengendalikan sistem tersebut. Contoh *game* simulasi adalah Microsoft Flight Simulator dan Football Manager.

2.2.3 *Game* Simulasi

Simulasi adalah proses membuat sesuatu terlihat nyata, padahal hal tersebut bukanlah hal yang nyata. Tindakan ini biasanya menyertakan karakterisitik utama dari suatu sistem yang disimulasikan. Dari pengertian ini bisa disimpulkan bahwa *game* simulasi adalah sebuah permainan yang mencoba menirukan berbagai aktivitas di dunia nyata yang digunakan untuk berbagai macam tujuan, misalnya pelatihan, analisa, ataupun prediksi. Biasanya jenis *game* ini tidak diharuskan memiliki tujuan yang jelas, karena hal utama yang perlu dicapai adalah pahamnya pemain terhadap karakter yang ia kendalikan. Contoh game simulasi yang cukup populer di kalangan *gamer* adalah seri The Sims dari developer Maxis.

2.2.4 Dasar-dasar *Game* Programming

Grafis

Grafis mungkin adalah komponen terpenting dalam sebuah *video game*. Dewasa ini pemrogaman grafis jauh lebih mudah dijangkau, bahkan

bagi *programmer* pemula sekalipun. 3D *graphic engine* berfungsi untuk menyembunyikan pemrograman yang kompleks sehingga pengguna bisa mengatur grafis dengan lebih mudah. OGRE(*Object-oriented Graphics Rendering Engine*) merupakan salah satu graphic engine yang telah banyak digunakan untuk membuat berbagai macam produk, dan salah satunya adalah *video game*.

3D *Models*

3D model adalah representasi komputer terhadap obyek yang bisa digambarkan pada layar melalui suatu proses yang disebut *rendering*. *Material* menentukan tampilan suatu obyek melalui pengaturan properti cahaya, warna, dan tekstur. Tekstur adalah sebuah gambar yang dibungkus di sekitar model. Kebanyakan dari obyek yang ditampilkan dalam grafis 3D adalah model 3D, mulai dari bangunan sampai karakter. Model-model ini dapat diciptakan dengan menggunakan *3D modeling tools*. Beberapa *3D modeling tools* yang cukup populer adalah Maya, Softimage XSI, dan Blender. Model-model yang dibuat dapat dieksport ke dalam *graphic engine* dengan menggunakan *3D model exporter tools*.

Material, Tekstur, dan Warna

Warna ditentukan oleh komposisi intensitas cahaya dari warna merah, biru, dan hijau. Terkadang warna juga memiliki nilai *alpha channel* yang merepresentasikan nilai transparansi dari suatu warna. Dalam grafis 3D, material digunakan untuk menentukan warna dari suatu model 3D. sebuah

material menentukan bagaimana model harus memantulkan tipe cahaya yang berbeda-beda dan mengaplikasikan tekstur kepada model. *Material* dapat diatur menggunakan *Level of Detail*(*LoD*) yang berbeda-beda, tergantung dari seberapa jauh jarak dari model dan *viewer*. Ketika berada pada jarak dekat, model sebaiknya dirender dengan detail sebanyak mungkin. Tetapi sebaliknya jika model berada pada jarak pandang cukup jauh, lebih baik tidak menampilkan model terlalu detail agar *performance* dapat lebih meningkat.

Color	Red value	Green value	Blue value
Red	1.0	0.0	0.0
Green	0.0	1.0	0.0
Blue	0.0	0.0	1.0
Orange	1.0	0.784	0.0
Pink	1.0	0.686	0.686
Cyan	0.0	1.0	1.0
Magenta	1.0	0.0	1.0
Yellow	1.0	1.0	0.0
Black	0.0	0.0	0.0
White	1.0	1.0	1.0
Gray	0.5	0.5	0.5
Light gray	0.75	0.75	0.75
Dark gray	0.25	0.25	0.25

Gambar 2.2 Intensitas warna merah, hijau, dan biru untuk warna umum

Pencahayaan

Ada 4 jenis cahaya dalam dunia 3D, *ambient, diffuse, emissive,* dan *specular*. Cahaya *ambient* adalah cahaya umum yang tidak datang dari suatu sumber cahaya tertentu. Cahaya *diffuse* muncul dari arah yang jelas dan dipantulkan secara merata oleh permukaan yang dikenainya. Cahaya *emissive* muncul dari suatu obyek tapi tidak mempengaruhi obyek-obyek disekitarnya. Cahaya *emissive* justru membuat obyek yang memancarkannya menjadi lebih cerah. Cahaya *specular* datang dari arah tertentu dan dipantulkan oleh suatu obyek berdasarkan tekstur permukaan serta sudut kedatangannya. Cahaya jenis ini biasa digunakan untuk membuat suatu obyek tampil bersinar.

Collision Detection dan Respons

Collision detection adalah proses menentukan apakah 2 obyek yang ada di dalam *game* bersentuhan atau tidak. Hal ini bisa dilakukan dengan perhitungan yang cukup kompleks. Obyek-obyek yang ada di dalam *game* perlu dibuat agar bereaksi secara wajar ketika bersentuhan satu sama lain. Hal ini bisa dilakukan dengan menambahkan c*ollision detection* dan *physics modeling library*. *Library* ini sangat membantu dalam menciptakan pengalaman bermain yang realistis.

Suara

Suara juga memiliki peran penting dalam menciptakan pengalaman bermain yang menyenangkan. *Audio library* bisa membantu para

pengembang *game* untuk memperkaya *game* dengan suara. Bahkan banyak dari antara *library* tersebut yang mendukung suara 3D. dalam sebuah *scene* 3D, obyek *game* yang mengeluarkan suara bisa berada pada jarak dan arah yang berbeda-beda dari pengguna. *Audio library* bertugas untuk menghitung faktor-faktor tersebut ketika memainkan sebuah suara. Suara yang berasal dari obyek yang berada dekat dengan pendengar akan mengeluarkan suara yang lebih keras dibandingkan dengan obyek yang berada jauh dari pendengar. Selain itu, suara dari satu sisi pendengar akan berbeda dengan suara yang berasal dari sisi lain.

Teks

Banyak *game* yang menggunakan teks sebagai sarana untuk berkomunikasi dengan user. Mulai dari memberikan instruksi atau hanya sekedar melaporkan berapa banyak poin yang telah didapatkan oleh user bisa dilakukan dengan cara menampilkan teks tersebut kepada user. Dalam banyak *game*, teks adalah bentuk komunikasi sesama pemain yang paling krusial.

Timer

Kecepatan berjalannya suatu *game* berbeda-beda pada setiap system dikarenakan perbedaan dari kecepatan *prosesor* masing-masing sistem. Untuk mengatasi masalah ini, programmer *game* menggunakan timer untuk mengontrol kecepatan animasi yang sedang berjalan. Jika suatu obyek berpindah sejauh jarak yang sama pada setiap *frame*(setiap kali layar

melakukan penggambaran ulang) maka obyek tersebut mungkin akan berpindah pada kecepatan yang berbeda di computer yang berbeda pula. *Timer* digunakan untuk membantu membuat animasi yang terlihat lebih natural dengan mengatur kecepatan yang menjaga mereka tetap bergerak dengan halus.

Pengalaman *User*

Game harus menyenangkan untuk dimainkan serta menarik perhatian pemain dalam berbagai hal. Dasar-dasar yang telah disebutkan sebelumnya berperan untuk menunjang pengalaman bermain secara keseluruhan. Perhatian pemain bisa didapat melalui grafis dan suara. Banyak aksi dalam *game* yang dikaitkan dengan grafis dan suara. Pemain perlu melakukan interaksi dengan *game* tersebut. Oleh karena itu dibutuhkan adanya alat untuk menerima inputan *user* seperti *keyboard*, *mouse*, dan *game controller*. Lalu komunikasi dengan *user* bisa dilakukan dengan menggunakan bantuan teks.

2.2.5 *Game Design*

Menurut Rollings dan Adams, *game design* adalah kegiatan mengimajinasikan *game*, mendefinisikan cara *game* tersebut bekerja, mendeskripsikan elemen-elemen yang membentuk *game*, lalu mentransmisikan informasi tersebut kepada tim yang akan mengembangkan *game*. *Game* design dibagi menjadi 3 kelompok utama yaitu:

- *Core Mechanics*

  Core mechanics adalah translasi dari visi *designer* menjadi sekumpulan aturan konsisten yang bisa diinterpretasikan oleh komputer. Menentukan *core mechanics* merupakan bagian "ilmiah" dalam mendesain sebuah *game*. *Core mechanics* merupakan jiwa dari sebuah *game*, oleh karena itu *core mechanics* harus mendapat perhatian khusus ketika mendesign sebuah *game*.

- *Storytelling* dan Narasi

  Setiap *game* menceritakan sebuah cerita. Kerumitan dan kedalaman cerita bergantung pada *game* tersebut. Tanpa cerita, atau tanpa adanya suatu cara bagi para pemain untuk menciptakan ceritanya sendiri, akan membuat *game* menjadi tidak menarik. Hal ini dikarenakan setiap cerita memiliki tekanan dramatis, yang bisa beruapa masalah yang tak terselesaikan, konflik, dan hal-hal lain yang membuat pendegarnya ingin terus menerus mengetahui kelanjutan dari cerita tersebut,

  Narasi adalah bagian dari cerita yang diceritakan oleh *game designer*. Narasi dalam *game* biasanya bersifat linear, tidak terpengaruh oleh aksi yang dilakukan oleh pemain dan selalu tetap. Banyak *designer* yang memandang ini sebagai suatu pembatasan kepada kebebasan pemain. Akhir-akhir ini, banyak sekali adanya diskusi mengenai ketidaklinearan pada *game* serta peningkatan apa saja yang akan dibawa.

- Interaktivitas

  Interaktivitas adalah cara pemain melihat, mendengar, dan melakukan tindakan dalam dunia *game*. Hal ini mencakup banyak hal seperti grafis, suara, antarmuka pemakai, dan hal-hal lain yang menghadirkan pengalaman bermain. Interaktivitas dimulai dari tampilan antarmuka pemakai. Hal ini dikarenakan antarmuka pemakailah yang menentukan kenyamanan dalam bermain *game*. Selain itu, tampilan grafis juga memegang peranan penting. Tetapi jangan sampai hal tersebut sampai mengurangi fokus pada *gameplay*.

2.2.6 Dimensi

Menurut Foley, Van Dam, Feiner, dan Hughes(1995), dimensi diartikan sebagai banyak cara untuk menentukan posisi suatu benda berdasarkan acuan tertentu. Sebuah benda dapat dikatakan berdimensi satu jika posisi benda tersebut dapat djtentukan dengan sebuah angka, misalnya sebuah kurva atau garis dimana posisinya dapat ditentukan oleh sebuah angka yang menyatakan jarak suatu titik terhadap titik awal. Sebuah benda dinyatakan berdimensi dua jika memiliki posisi yang dapat ditentukan oleh 2 angka. Misalnya, suatu permukaan bola dapat diukur dengan angka derajat lintang dan angka derajat bujur.

Dunia 3 dimensi (3D) tidak sesederhana dunia 2 dimensi. Hal ini dikarenakan dunia 3D memiliki 3 buah koordinat, sedangkan dunia 2D hanya memiliki 2 koordinat saja. 3 buah koordinat tersebut adalah *axis* x, y,

dan z, dimana x adalah *axis* mendatar, y, adalah *axis* tegak, dan z adalah *axis* yang menembus layar monitor. Dalam penglihatan matematisnya, obyek di dalam ruang 3 dimensi mempunyai kerumitan tersendiri, diantaranya:

- Setiap obyek bergerak mengikuti arah mata angin
- Digunakan teknik perhitungan collision antara koordinat x, y, dan z.
- Pergerakan obyek di dalam 3D harus dilakukan dalam perhitungan yang cepat, agar pergerakan yang ditampilkan terlihat halus dan bagus.

Permasalahan utama yang dihadapi adalah, layar monitor merupakan layar 2D. Untuk menampilkan obyek 3D pada layar 2D perlu dilakukan proyeksi 3D, yaitu memproyeksikan koordinat 3D dari suatu obyek sehingga menjadi koordinant 2D pada layar monitor.

2.2.7 *Game Engine*

Game engine adalah sebuah sistem perangkat lunak yang dirancang untuk menciptakan dan mengembangkan sebuah *game*. *Game engine* mempunyai *development tools* dengan tampilan visual dan langsung terintegrasi sehingga *tools-tools* tersebut dapat digunakan kembali untuk mengembangkan *game* lain. Dengan menggunakan *game engine*, maka pengembangan *game* akan menjadi lebih cepat, serta mengurangi biaya dan tingkat kerumitan. Contoh *game* engine yang cukup terkenal adalah Torque *game* engine.

Sebuah *game engine* memiliki beberapa komponen utama. Secara garis besar komponen-komponen *game* engine terbagi menjadi 4 bagian, yaitu:

- Grafik *rendering*

  Rendering merupakan fitur utama dari *game engine* sehingga mampu menampilkan model baik dalam 2 dimensi ataupun 3 dimensi. Grafik rendering membaca obyek yang diinput, lalu menampilkan setiap *pixel* dari obyek tersebut pada layar.

- *Physic game*

  Game engine memungkinkan untuk menentukan apa yang terjadi pada obyek dan mensimulasikan efek yang terjadi pada obyek sesuai dengan karakteristik obyek tersebut. Efek-efek tersebut bisa berupa: pecah, retak, memantul, dan sebagainya.

- *Platform abstraction*

  Platform abstraction mempermudah untuk mengembangkan *engine* di setiap *platform* yang berbeda. Hal ini ditujukan agar pengembang *game* tidak harus merombak ulang kode program jika ingin mengembangkan *game* yang sama ke *platform* yang berbeda.

- *Integrated Development Environment*

  Game engine menyederhanakan dan memudahkan proses pengembangan *game* seperti koding, penambahan efek visual dan suara, memasukkan *AI*, *collision detection*, fasilitas jaringan, dan sebagainya.

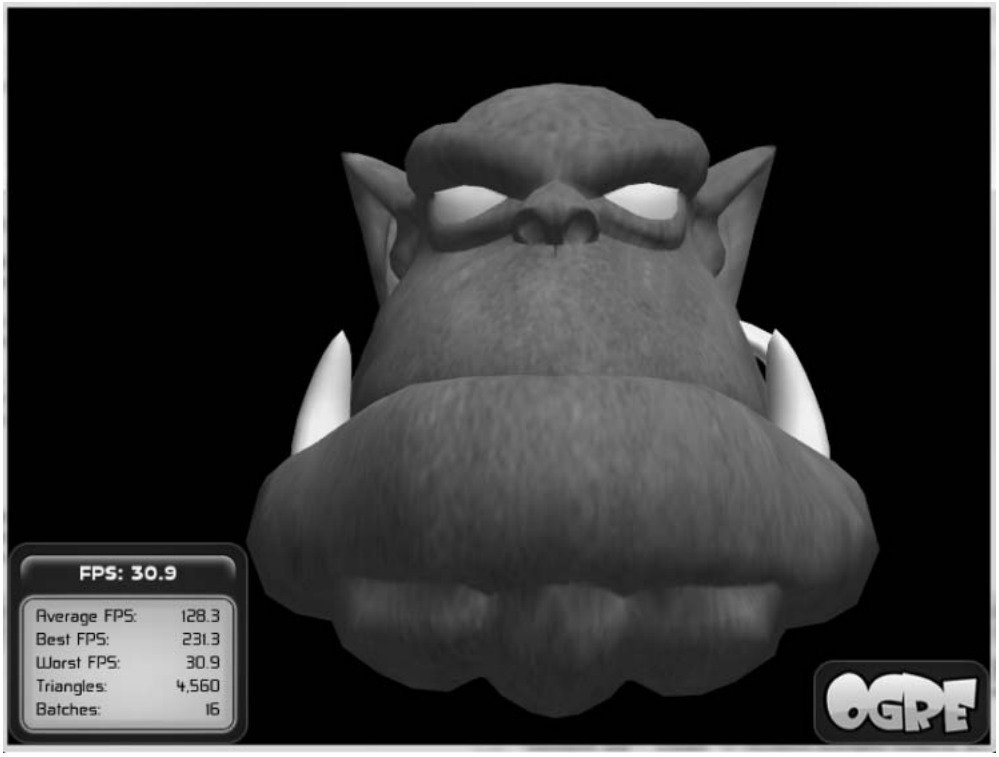
2.2.8 Graphics Engine

Ada beberapa *engine* yang hanya memberikan fitur untuk melakukan *3D rendering* saja dan tidak memberikan fitur-fitur yang biasanya dibutuhkan untuk membuat *game* secara keseluruhan. *Engine-engine* seperti ini dikenal dengan sebutan *graphics engine*. *Graphics engine* sangat bergantung pada fitur-fitur tambahan dari pihak ketiga, yang mampu membuat sebuah *graphics engine* berfungsi layaknya *game engine*. *Graphics engine* dikembangkan dengan menggunakan grafik *Application Programming Interface* seperti DirectX dan OpenGL yang memberikan akses ke *Graphical Processing Unit* dari kartu grafis. Contoh *graphics engine* yang cukup terkenal adalah OGRE.

2.2.9 OGRE

Pengertian OGRE

OGRE(*Object-oriented Graphics Rendering Engine*) adalah proyek *open source* yang diciptakan oleh Steve Streeting dan masih terus dikembangkan oleh Ogre Team sampai saat ini. OGRE adalah *graphic engine* yang bekerja secara *realtime*. Berdasarkan pengertian ini bisa diketahui bahwa OGRE hanya menangani grafis saja. Tetapi OGRE dapat digabungkan dengan *library-library* lain sehingga menjadi sebuah *game engine*. OGRE memungkinkan penggunanya untuk menampilkan grafis 3 dimensi pada aplikasi yang dibuat dengan cara yang berorientasi obyek.



Gambar 2.3 *Render Window* pada OGRE

Sejarah OGRE

- 1999

  Steve Streeting (Sinbad)menyadari bahwa proyek DIMClass miliknya tidak perlu didasari oleh Direct3D lagi. Sehingga ia mulai merencanakan membuat sebuah *library* yang bersifat API dan *platform independent.*

- 25 Februari 2000

  Terdaftar dalam proyek Sourceforge dengan nama OGRE. Belum dimulai adanya pengembangan karena masih ada hal-hal lain yang lebih diprioritaskan.

- Oktober 2000

  Mulai melakukan *refactoring* terhadap DIMClass.

- Desember 2000

  Kelas-kelas utama seperti *SceneManager*, *RenderSystem*, dan *SceneNode* mulai diperkenalkan.

- Februari 2001

  Platform manager pertama selesai (Win32). D3D7 adalah *rendersystem* pertama.

- Maret 2001

  Pertama kalinya dilakukan uji *render* terhadap sistem OGRE serta mulai digunakannya Doxygen untuk menghasilkan dokumentasi *API*.

- April 2001

  Membuat keputusan untuk melakukan perubahan desain untuk mengurangi depedensi dan meningkatkan fleksibilitas.

- Mei 2001

  Diciptakannya sistem *Entity* / *SubEntity* / *Mesh* / *SubMesh*

- Juni 2001

  Diperkenalkannya *FrameListener* dan *FrameEvent.*

- Juli 2001

  Dirilisnya OGRE 0.6.

- November 2001

  Diperkenalkannya *skyboxes*, *skydomes*, dan *skyplanes*.

- Desember 2001

  Dukungan BSP selesai.

- Januari 2002

  Diperkenalkannya kontroler.

- Februari 2002

  Cearny bergabung dengan tim inti.

- Maret 2002

  Dirilisnya OGRE 0.98b, diperkenalkannya *material sripts*.

- April 2002

  Temas bergabung dengan tim inti.

- Juni 2002

  Dirilisnya OGRE 0.99b, diperkenalkannya *particle system*.

- September 2002

  Dirilisnya OGRE 0.99d, adanya dukungan terhadap Linux, OpenGL *renderer*, *skeletal animation*, Milkshape *exporter*, dan *codec structures*.

- Oktober 2002

  Janders bergabung dengan tim inti. Dirilisnya OGRE 0.99e.

- Desember 2002

  Diperkenalkannya *Mesh LOD, Geomipmapping terrain*, dan *stencil operation*.

- Januari 2003

  Perilisan pertama Blender *exporter*.

- Februari 2003

  Dirilisnya OGRE 0.10.0 – 3DS Max *exporter*. D3D9 *renderer*.

- Pertengahan 2003

  Cearny dan Janders keluar dari tim inti.

- Juni 2003

  Dirilisnya OGRE 0.11.0, ditambahkannya *SceneQuery*, *ODE collision* didemonstrasikan dalam *BspCollision*, eksporter untuk Maya dan Wings, serta penambahan-penambahan lainnya.

- Mei 2003

  mental bergabung dengan tim inti.

- September 2003

  Dirilisnya OGRE 0.12.0, pertamakalinya *codebase* dibagi dalam kategori *maintenance* dan *development*.

- Januari 2004

  Dirilisnya OGRE 0.13.0, adanya perombakan besar terhadap material, dukungan vertex dan pixel shader terhadap *assembler* (D3D dan GL), dukungan terhadap Cg dan GLSL, diperkenalkannya kelas *Radian/Degree*.

- November 2004

  Softimage setuju untuk menjadi dari sponsor OGRE serta diizinkannya akses ke XSI SDK.

- Februari 2005

  Dirilisnya OGRE v1.0.0 *Final* Azathoth, adanya perombakan terhadap *resource system*, diperkenalkannya *hardware pixel buffer*, *HDR*, CEGui, dan eksporter XSI.

- Maret 2005

  OGRE menjadi '*Project of the Month*' di Sourceforge.

- Juli 2005

  Jeff 'nfz' Doyle bergabung dengan tim OGRE.

- September 2005

  Sinbad mengumumkan Kadath, *scene manager* berbayar dan eksporter yang bisa digunakan pada OGRE, XSI, serta Blender.

- Mei 2006

  Dirilisnya OGRE v1.2.0 *Final* Dagon.

- Maret 2007

  Dirilisnya OGRE v1.4.0 *Final* Eihort.

- April 2007

  Diumumkannya proyek Google Summer of Code 2007.

- Maret 2008

  Dibukanya aplikasi siswa untuk Google Summer of Code 2008.

- November 2008

  Dirilisnya OGRE 1.6.0 Final Shoggoth.

- Juli 2009

  David Rogers bergabung ke dalam tim, dengan membawa dukungan terhadap iPhone.

- Agustus 2009

  Diumumkannya proyek Google Summer of Code 2009.

- Oktober 2009

  Noam ‘Norman’ Gat bergabung dengan tim.

2.2.10 Autodesk Softimage

Autodesk Softimage atau biasa disebut dengan Softimage adalah aplikasi grafis 3D yang dimiliki Autodesk. Softimage bisa digunakan untuk membuat grafis komputer 3D, 3D modeling, dan animasi komputer. Software ini banya digunakan di industri film, *video game*, dan periklanan. Dirilis ada tahun 2000 sebagai penerus dari softimage 3D, XSI softimage dikembangkan oleh Softimage Corporation., yang merupakan anak perusahaan dari Avid Technology. Pada 23 Oktober 2008, Autodesk membeli Softimage dari Avid Technology seharga 35 juta US dollar. Lalu pada Februari 2009 namanya diganti menjadi Autodesk Softimage.