BAB II

LANDASAN TEORI

2.1. Konsep Dasar *Game*

2.1.1. Pengertian *Game*

Game adalah sebuah permainan komputer interaktif yang di kendalikan oleh mikroprosesor. Komputer dapat menciptakan bahan-bahan maya untuk digunakan dalam sebuah permainan seperti kartu dan dadu.Sebuah permainan komputer atau *video game* menggunakan satu atau lebih alat input, biasanya sebuah tombol atau kombinasi dari *joystick*, sebuah *keyboard* dan *mouse* dan *trackball* atau sebuah *controller* ataupun sebuah alat yang mempunyai sensor gerak.

Sebuah *video game* adalah permainan yang biasanya melibatkan *player* berinteraksi dengan alat pengendali atau *controller* untuk menghasilkan umpan balik secara visual dalam sebuah layar video. *Video game* biasanya juga mempunyai sistem pemberian hadiah yang diberikan kepada pemain apabila menyelesaikan tugas-tugas tertentu yang berada didalam aturan atau *rule set game* tersebut. Tipe-tipe peralatan elektronik dimana video *game* dapat dimainkan disebut dengan *platform* dan contoh dari *platform* adalah PC *(personal komputer)* dan mesin *video game* (video *game* console). *Video game* juga hadir dalam semua tingkatan teknologi mulai dari komputer sampai dengan peralatan genggam seperti *handphone* dan *PDA*.

Selain elemen dasar dari umpan balik berbasis video, *video game* juga menggunakan banyak system lain untuk menyediakan interaksi dan informasi kepada pemain. Contoh umumnya adalah penggunaan system *audio (speaker)* dan peralatan interaktif seperti umpan balik dengan getaran.

2.1.2. Pengertian *PC Game*

Sebuah *personal komputer* adalah sebuah *microkomputer* yang harganya, ukurannya, dan kemampuannya sesuai untuk individual. Istilah PC di populerkan oleh *Apple* dengan *Apple II* nya pada awal tahun 1980an, dan setelah itu oleh IBM dengan IBM *Personal Komputer*. *Personal Komputer* juga dikenal sebagai komputer rumah. *PC game* merupakan *game* yang dimainkan pada sebuah *personal komputer* (PC). Sebuah *PC game* membutuhkan sebuah sistem operasi seperti *Windows, Linux* ataupun *Mac* untuk menjalankannya. Sistem operasi yang paling populer untuk memainkan *game adalah Microsoft Windows*.

2.1.3. Jenis-Jenis *Game (genre)*

Video game secara umum dikategori-kategorikan kedalam *genre*. Karena kurangnya penyetujuan syarat atas *genre* atau kriteria untuk definisi sebuah *genre*, klasifikasi dari *game* menjadi tidak konsisten atau sistematis.

Dibawah ini merupakan daftar *genre-genre* yang sering dimainkan orang. Banyak dari kategori *game* dibawah ini bertabrakan satu sama lainnya, contohnya pada *game Legend of Zelda* yang mempunyai elemen *action, adventure,* dan *role-playing.*

a. *Action*

Game action mungkin adalah *genre game* yang paling basic, dan pastinya salah satu yang terbesar. *Game action* di karakteristik kan oleh *gameplay* nya yang fokus kepada aksi yang mengharuskan pemain melakukannya secara reflek, dalam *realtime*. *Game fighting* dan *first person shooters* termasuk dalam kategori ini.

b. *Action-adventure*

Game action-adventure fokus dalam penjelajahan, dan biasanya melibatkan pengumpulan benda, penyelesaian teka-teki yang sederhana, dan pertempuran. *Game action-adventure* yang pertama adalah *game Atari 2600* yang berjudul *adventure (1978).*

c. *First-person shooter*

Gambar. 2. 1. *Half-Life,* sebuah *game first-person shooter*

Game first-person shooter, biasanya disebut dengan *FPS*, menggambarkan menembak dan bertempur dalam perspektif orang pertama dari karakter yang mereka kontrol. Perspektif ini berarti memberikan pemain perasaan “ada disana”, dan mengizinkan pemain fokus pada membidik. Kebanyakan *FPS* sangatlah cepat dan memerlukan reflek yang sangat cepat dalam tingkat kesulitan yang tinggi.

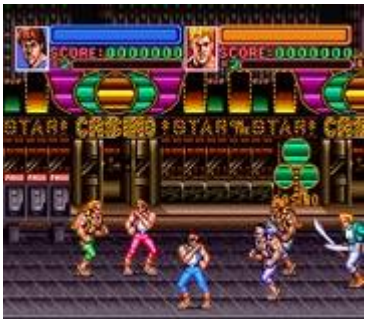
d. *Fighting*

Gambar. 2. 2. *Street Fighter II*, sebuah game perkelahian satu lawan satu

Game perkelahian adalah pertarungan satu lawan satu antara dua karakter yang salah satunya dikendalikan oleh komputer. *Game* ini biasanya fokus dalam bela diri dan bentuk lain dari pertempuran tangan kosong. Banyak dari gerakan yang dilakukan oleh petarungnya di dramatisir dan tidak mungkin secara fisik. Beberapa karakter yang bertarung juga menggunakan senjata seperti pedang dan atau serangan jarak jauh seperti serangan tenaga dalam.

Game perkelahian terkenal antara lain adalah *Mortal Kombat*, *Street Fighter*, *Super Smash Bros.*, *King of Fighters*, *Soul Edge* dan *Soul Calibur*, *Tekken*, *Dead or Alive*, dan *Virtua Fighter*

e. *Beat 'em up*



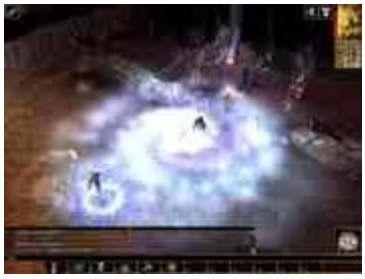
Gambar. 2. 3. Cuplikan gambar dari *Super Double Dragon* di *SNES*

Game ini adalah *game* dimana ada pertarungan jarak dekat satu lawan banyak, bertempur melawan sejumlah musuh yang dikontrol oleh komputer.

Gaya bertarungnya biasanya lebih sederhana daripada *game* fighting satu lawan satu. *Genre* ini popular pada tahun 1987 dengan dikeluarkannya Double Dragon, diikuti sejumlah banyak *game-game* serupa. *Gameplay* dalam *game* ini melibatkan pemain bertarung melewati sebuah seri *level* yang semakin sulit. Kritik mengatakan bahwa *game* seperti ini terkesan terus diulang-ulang dan membosankan. Di waktu sekarang, *genre* ini sudah bergabung dengan *genre* action-adventure.

Beberapa contoh *game genre* ini *Final Fight*, *Double Dragon*, *Battle Toads*, *Streets of Rage*, *Golden Axe*, dan yang modern *Dynasty Warriors*, *Viewtiful Joe*, *The Warriors* dan *Samurai Warriors*.

f. *Role-Playing*



Gambar. 2. 4. *Neverwinter Nights* sebuah role-playing game komputer yang terkenal

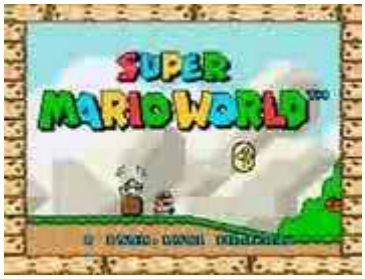
Role-Playing Game (*RPG*) seringkali memposisikan pemain dalam sebuah fantasia atau *science fiction* dan membawa *gameplay*nya melalui suatu jalan cerita yang rumit. Kebanyakan dari *game* ini membuat pemain beracting menjadi tipe “petualang”

spesifik yang mempunyai spesialisasi dalam sebuah kumpulan kemampuan-kemampuan (seperti pertarungan, mengeluarkan ilmu sihir). Tipe petualang yang bermacam-macam ini disebut "kelas-kelas" dan pemain biasanya mengendalikan satu atau lebih dari karakter-karakter ini.

g. ***Massively Multiplayer Online Games***

Massively Multiplayer Online Games merupakan sebuah dunia virtual dimana bisa terdapat ribuan pemain berinteraksi bersama melalui internet. Kebanyakan dari *game* ini berbasiskan pendaftaran, tetapi belakangan beberapa MMO mengizinkan pemain untuk membeli *game* nya sekali saja dan memainkannya secara online dengan gratis.

h. *Platform game*



Gambar. 2. 5 Cuplikan gambar dari *Super Mario World,* sebuah *game platform*

Platform game, juga biasa disebut *platformer*, adalah *game* yang dalam *gameplay* nya melibatkan perjalanan antar *platform* dengan cara meloncat (biasanya juga berayun dan memantul). *Genre* ini biasanya dihubungkan dengan tokoh-okoh

kartun seperti sonic the hedgehog, Mario, dan rayman, walaupun mungki mempunyai tema yang lainnya.

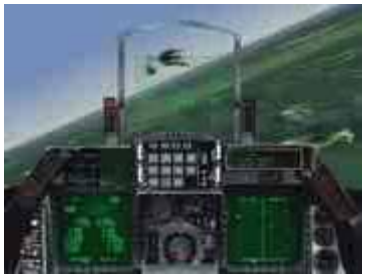
Secara tradisional, *game platform* berbentuk 2D, menampilkan lingkungan permainan dalam 1 perspektif saja, biasanya dari samping. Ini bisa dilakukan dengan menggunakan sprites, dan sangat mudah ditangani oleh komputer masa-masa awal. Komputer grafis 3D memungkinkan *game* jenis ini untuk bergerak ke segalah arah. Walaupun begitu, perspekti menyulitkan pemain untuk memperkirakan jarak, yang merupakan bagian penting bagi *game platform*.

Elemen-elemen tradisional dari *game* ini termasuk berlari, lompat, dan memanjat tangga atau pijakan. *Genre* ini seringkali meminjam elemen dari *genre* lain seperti fighting atau perkelahian dan shooting atau tembak-menembak.

Platform game yang cukup terkenal meliputi *Castlevania*, *Super Mario Bros.*, *Metroid*, *Sonic the Hedgehog*, *Mega Man*, *Donkey Kong*, *Contra*, *Lode Runner*, *Spyro the Dragon*, *Rayman* dan *Crash Bandicoot* dan seri *Prince of Persia* .

i. ***Simulation Games***

Game simulasi bertujuan untuk men simulasikan pengalaman seperti terbang didalam pesawat, serealistis dan sepraktis mungkin, dan juga terdapat hokum fisik seperti di dunia nyata. Beberapa *game* membutuhkan untuk banyak membaca sebelum *game* dimainkan, sementara *game* yang lain sudah termasuk petunjuk singkat.



Gambar. 2. 6. *Falcon 4.0* adalah contoh utama dari simulasi pertempuran udara.

Game simulasi seringkali dikaitkan dengan simulator sungguhan, dan sebagian dari *game* tersebut memang berguna dalam pelatihan militer sungguhan atau untuk kepentingan umum.

j. *Flight*

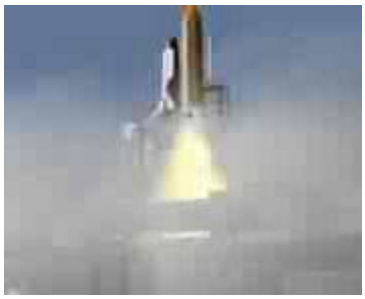
Combat flight simulator adalah sub*genre* simulasi paling terkenal. Falcon 4.0 dan IL-2 Sturmovik adalah contoh-contoh dari *genre* tersebut. Ada juga simulasi pesawat sipil, Microsoft Flight Simulator dan X-Plane adalah contoh yang paling terkenalnya.

k. *Military*

Ada *game* simulasi yang dibangun dengan tema peperangan yang berbeda-beda, ataupun dengan tipe-tipe yang berbeda. Contohnya, simulasi tank yang cukup terkenal adalah Abrams. Simulasi kapal dan kapal selam yang terkenal adalah Silent Hunter. Operation flashpoint: cold *war* crisis adalah sebuah simulator peperangan secara umum. Versi yang telah dikembangkan dari *game* ini digunakan untuk pelatihan militer sungguhan.

Game FPS militer yang bertujuan untuk menjadi lebih realistis disebut sebagai *tactical shooter* karena *game* ini biasanya membutuhkan strategi dan kerja sama untuk menang, dibandingkan dengan penggunaan *skill* dan reflek pada *game* realistis lainnya.

l. Space

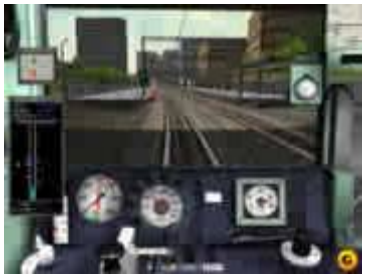


Gambar. 2 .7. *Space Shuttle Atlantis* meluncur dalam *Orbiter spaceflight simulator*

Space combat sims merupakan sebuah *subgenre* antara *game action* dan strategi. Contohnya adalah seri dari the elite yang mengawali adanya *genre* ini, dan *game* seperti *star wars:X-Wing*, *Descent: freespace, freelancer dan Homeworld.* Simulasi ruang angkasa berbeda dari sub*genre* lainnya, dan tidak secara umum disebut sebagai simulasi, karena *game* ini seringkali mensimulasikan objek yang tidak sungguhan ada. Walaupun begitu simulasi dari pesawat angkasa sungguhan juga ada, orbiter adalah contohnya.

m. Train

Simulasi kereta mensimulasikan kendaraan, lingkungan dan seringkali aspek ekonomi yang berhubungan dengan transportasi kereta. Biasanya mengikuti sejarah dari evolusi kereta di berbagai negara, dan ledakan ekonomi yang seringkali terjadi dalam bisinis kereta api. Contoh dari *game* ini termasuk *Microsoft Train Simulator dan Trainz.*



Gambar. 2. 8. Tampilan dari *Microsoft Train Simulator*

n. *God games*

Sebuah *subgenre* popular dari *game* simulasi adalah *god game*, contoh dari *genre* ini termasuk populous,the sims yang sangat terkenal, "spore", dan *simAnt*, juga *SimEarth*. Tidak seperti *genre game* lainnya, *god game* seringkali tidak mempunyai tujuan yang memungkinkan pemain untuk memenangkan permainannya.

o. *Economic simulation games*

Game simulasi ekonomi, pada umumnya mencoba menghadirkan aspek keseluruhan dari sebuah ekonomi atau bisnis. Contoh populernya adalah Raiload Tycoon.

p. *City-building games*

Gambar. 2. 9. Contoh kota dari *Children of the Nile*

Dalam permainan pembangunan kota, yang mana sub *genre* spesialis dari permainan simulasi ekonomi, pemain bertindak sebagai perencana keseluruhan atau pemimpin yang harus memenuhikebutuhan dan keinginan karakter dalam *game* dengan cara membangun bangunan untuk makanan, tempat tinggal, kesehatan, keagamaan, pertumbuhan ekonomi dan lain-lain. Kesuksesan diraih saat anggaran kota membuat keuntungan terus-menerus dan *war*ga kota mengalami peningkatan kesejahteraan, kesehatan, dan lainnya. Disamping itu semua, perkembangan militer juga sering dimasukkan, fokusnya dalam kekuatan ekonomi.

Mungkin *game* yang paling terkenal di *genre* ini adalah *SimCity*, yang sekarang masih sangat popular dan mempunyai dampak yang besar kepada *game* sejenis lainnya. *SimCity* bagaimanapun juga termasuk dalam *genre God game* karena memberikan pemain kemampuan seperti tuhan untuk memanipulasi dunia.

q. *Government simulation games*

Sebuah permainan simulasi pemerintahan (*game* poliitk) melibatkan simulasi dari kebijakan-kebijakan, ppemerintahan atau politik dari sebuah negara, tapi biasanya tidak melibatkan peperangan, tipe *game* seperti ini disebut juga "*game* serius"

r. Sports

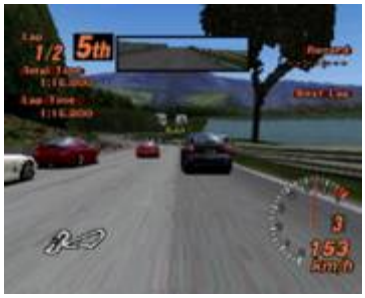
Game olahraga menampilkan permainan dari olahraga fisik tradisional seperti kriket, baseball, sepak bola, American football, tinju, golf, basket, skateboard, hoki es, tenis, bowling, rugby, dan lainnya. Beberapa fokus pada memainkan olahraganya dan beberapa fokus pada strategi dibalik olahraga tersebut (seperti *game* championship

manager). Lainnya menggunakan olahraga untuk efek komik (seperti *game Arch Rivals*). Salah satu yang paling terkenal dari *genre* ini adalah *Madden NFL.*

Genre ini ada pada masa awal sejarah video *game (pong)* dan masih popular sampai sekarang.

Gambar. 2. 10. *NBA Jam,* sebuah *game* olahraga.

s. **Racing**



Gambar. 2. 11. *Gran Turismo 2, game* balapan di *PlayStation*

Game balapan biasanya memposisikan pemain di kursi pengemudi dari kendaraan ber performa tinngi dan memerlukan pemain untuk berlomba melawan pengemudi lainnya ataupun melawan waktu. *Genre* ini merupakan *genre* yang selalu ada dalam permainan komputer dan banyak permainan yang diciptakan pada masa awal adalah bagian dari *genre* ini. Berawal dari tahun 1970, *genre* ini masih popular sampai sekarang dan terus mendorong perkembangan performa dan grafis. Ada dua sub *genre*

dari *genre* ini yaitu simulasi dan arcade. *Genre* arcade biasanya bersifat fantasi, kerusakan kendaraan yang tidak realistis. *Genre* simulasi lebih fokus terhadap handling kendaraan yang realistis dan performa yang sama dengan mobil sungguhan, dan biasanya mengizinkan pemain untuk mengikuti perlombaan sungguhan seperti Indianapolis dan dakkar. Balapan simulasi memperbolehkan pemain untuk merubah performa kendaraanya, mulai dari saluran pembuangan mesin sampai rasio gigi transmisi, tapi sekarang semua itu juga di aplikasikan pada *game arcade* seperti *need for speed: underground.*

Sebuah *sub genre* yang popular dari *genre* ini adalah *game* balapan kart, yang menyederhanakan handling kendaraan dan memperkenalkan rintangan yang bermacam-macam.

Game balapan *arcade* yang terkenal adalah *Out Run*, dan seri *Mario Kart*. *Game* balapan simulasi yang terkenal adalah *Gran Turismo*.

t. *Strategy*

Game strategi fokus pada perencanaan yang cermat dan manajemen sumber daya yang sangat baik dalam rangka mencapai kemenangan, dan selanjutnya *game* ini dikategorikan sebagai "permainan berpikir". *Game* ini bisa berbasiskan giliran atau *realtime*, tapi kadang campuran keduanya seperti X-com. *Genre* ini mempunyai penggemar yang konsisten sejak pertengahan 1980an. Walaupun sebagian besar dari *game* strategi adalah *game* peperangan, banyak juga yang tidak fokus pada peperangan seperti simulasi dan manajemen transaksi ekonomi, membangun sesuatu, mengatur konflik skala besar, dan lain-lain.

u. *Strategy wargame*s

Sebagian besar dari *game* strategi adalah *game* peperangan dan entah itu berbasiskan giliran maupun *realtime*. *Game* peperangan *realtime* secara umum disebut sebagai *real time Strategy* (*RTS*) atau *real time tactic*al (RTT). *Game* berbasis giliran aslinya adalah bentuk paling umum dari *game* strategi: komputer-komputer pada masa lampau terlalu lambat untuk interaksi secara *real time*.

v. Real-time *Strategy* dan *turn-based Strategy game*s

Kebanyakan dari *game* strategi bisa disebut sebagai "*game* perang strategi" karena mereka fokus pada pertempuran militer.Sub tipe dari *game* ini di klasifikasikan sebagai *real time* strategi yang biasanya fokus kepada pertempuran militer dan strategi berbasis giliran yang lebih fokus kepada strategi dan lebih memposisikan pemain menjadi jendral ataupun pemimpin suatu negara.

Contoh dari *game* strategi *real time* adalah *Warcraft*, *StarCraft*, *Command and Conquer*, *Age of Empires* dan *Total Annihilation*. Contoh dari *game* strategi berbasiskan giliran adalah *Sid Meier's Civilization*, the *Heroes of Might and Magic* , *Advance Wars*, *Fire Emblem* dan Shattered Union. Contoh dari *game* gabungan antara *real time* dan giliran adalah Rise of Nations: Rise of Legends, Age of Empires III, dan yang akan datang Sins of a Solar Empire.

w. Real-time *tactic*al and turn-based *tactic*al *game*s

Genre berbeda yang fokus kepada jumlah kumpulan unit pasukan, dan tidak memperhatikan pengumpulan sumber daya dan aspek produksi unit dari *game RTS*

disebut sebagai *real time tactic*al (RTT). Contoh dari sub *genre* ini adalah *War*hammer: Dark Omen *game*, salah satu RTT murni, Ground Control yang futuristic, mengambil aspek pertempuran dari command and conquer dan membentuknya menjadi suatu *game* RTT murni. *Game-game* seperti ini murni fokus terhadap aspek taktikal, sangat kontras dengan *game RTS* yang fokus terhadap ekonomi dan produksi unit.

2.1.4 . Komponen-komponen *game*

Game memiliki 5 komponen penting yaitu:

a. Fitur

Fitur merupakan hal yang bisa membedakan setiap *game* yang ada. Fitur juga bisa menggambarkan jalan cerita *game* kedalam bentuk-bentuk yang dapat dilihat maupun dirasakan.

b. *Gameplay*

Gameplay membantu pengembang *game* untuk mengetahui cara kerja suatu *game*, dimana fitur-fitur yang ada akan membentuk suatu *gameplay*.

c. *Interface*

Interface merupakan semua tampilan yang ada dalam suatu *game*. Sebuah *interface* yang baik adalah *interface* yang tidak membosankan dan memudahkan pemain *game*.

d. Aturan/*rules*

Merupakan kumpulan aturan-aturan dalam sebuah *game*.

e. Desain *Level*

Desain *level* mencakup style, background, dan jalan cerita dari sebuah *game*.

2.1.5 . *Game Balance*

Game balance merupakan salah satu yang terpenting dari sebuah *game*. Apabila sebuah *game* tidak seimbang maka kemungkinan besar ada fitur *game* yang sangat jarang digunakan sehingga menjadi sia-sia.

Ada 3 jenis *game balance*:

a. *Player – Player balance*

Yang termasuk dalam kategori ini kebanyakan adalah multi*player game* karena masing-masing *player* mempunyai keuntungan yang sama, kecuali keahlian masing-masing pemain.

b. *Player – Gameplay balance*

Didalam kategori ini, keberhasilan pemain dalam setiap *level*nya akan mendapatkan imbalan sesuai dengan keberhasilan yang telah dicapainya.

c. *Gameplay – Gameplay balance*

Dalam kategori ini, setiap fitur yang terdapat dalam sebuah *game* harus seimbang secara keseluruhan, contohnya apabila terdapat senjata yang lebih kuat daripada senjata lainnya, maka senjata tersebut harus berada di *level game* yang lebih tinggi dari senjata lainnya yang kurang kuat.

2.2. *Engine Game*

2.2.1. Pengertian *Engine Game*

Game engine merupakan sebuah komponen inti *software* dari sebuah *game* ataupun aplikasi lain yang menggunakan grafik secara *real time*. *Game engine* mempunyai *development tools* dengan tampilan visual dan langsung terintegrasi didalam *IDE (integrated development environment)* sehingga *tools – tools development* tersebut dapat digunakan kembali untuk mengembangkan *game* yang lain. Hal ini membuat *game engine* sering disebut *game middleware* karena *game engine* mempercepat pengembangan sebuah *game*, mengurangi biaya pemgembangan *game*, tingkat kompleksitas dari pengembangan *game,* yang merupakan faktor utama untuk bersaing didalam industri *game*. Beberapa *game engine* hanya memberikan fitur untuk melakukan 3D *rendering*, dan tidak memberikan fitur – fitur yang dibutuhkan untuk membuat *game* secara keseluruhan. *Game engine* seperti ini disebut sebagai 3D *engine*, dan sangat bergantung pada pengembang *game* untuk menambahkan fitur – fitur yang tidak terdapat didalamnya dengan menggunakan komponen dari *game engine* yang lain. Contoh dari *game engine* tersebut adalah ; *RealmForge, Ogre, Jmonkey*. *Game engine* yang memberikan fitur 3D *engine* dikembangkan dengan menggunakan grafik *API*

(application programming interface) seperti Directx dan openGl, yang memberikan akses ke *GPU (Graphical Processing Unit)* dari video card. *Game engine* secara lisensi terbagi menjadi dua yaitu; open source dan komersial. Salah satu contoh *game engine* open source yang terkenal adalah *Crystal Space* dan *Ogre,* sedangkan *game engine* komersial terkenal dan banyak digunakan saat ini adalah *Torque game engine*.

2.2.2. Jenis *Engine Game*

Pada awalnya jenis – jenis *game engine* dapat dikelompokkan menjadi; *game engine FPS (First Person Shooter)*, *game engine RTS (Real Time Strategy),* dan beberapa jenis *game engine* lainnya, tetapi pengelompokan jenis *game engine* semakin lama menjadi semakin tidak jelas dikarenakan faktor – faktor sebagai berikut :

- *Game engine* yang ada saat ini mampu mengembangkan berbagai macam genre *engine*, walaupun dikembangkan dari *game engine* yang hanya dikhususkan pada satu genre saja. *engine MMORPG Lineage II* merupakan salah satu contoh *engine* yang dikembangkan dari *game engine* yang berbeda genre dengan *Lineage II*, yaitu *game engine Unreal II* yang ber-genre *FPS (First Person Shooter)*.

- *Game engine* yang dianggap baru terkadang masih menggabungkan teknologi lama dengan teknologi baru yang ada di *game engine* tersebut, hal ini dikarenakan teknologi yang dianggap maju saat ini akan menjadi standar yang diharapkan pada tahun – tahun mendatang. Contoh implementasi penggabungan teknologi lama dan teknologi baru adalah, *Red Faction (2001)*

yang memberikan fitur dinding dan tanah yang dapat hancur, ini merupakan hal yang tidak biasa pada *game engine* saat itu hingga menjadi standar baru *game engine* di tahun 2004.

Faktor – faktor tersebut di atas, bukan menjadi kelemahan dari *game engine*, tetapi menunjukkan perkembangan *game engine* terus meningkat dan pengembang *engine* mampu memaksimalkan kemampuan dari *game engine*.

2.2.3. Komponen-komponen *engine game*

Game engine memberikan fitur – fitur yang biasa terdapat didalam *engine*, seperti 3D atau 2D *rendering*, *LAN (local area network)*, efek suara, animasi, *artificial intelligence*, *networking, scripting*, dan lain sebagainya. Fitur – fitur ini adalah komponen utama dari *game engine*, dan masing - masing *game engine* bisa memiliki komponen yang berbeda. Secara garis besar komponen – komponen *game engine* terbagi menjadi 4 bagian :

1. **Grafik *Rendering***

   Rendering merupakan fitur utama dari *game engine* untuk menampilkan model secara 2D atau 3D. Rendering grafik membaca obyek atau model yang akan dimasukkan, menampilkan obyek atau model tersebut setiap pixelnya dilayar sehingga menghasilkan obyek atau model dengan detil.

2. ***Physic Game***

   Game engine memungkinkan untuk menentukan event yang terjadi pada obyek-obyek *engine*, dan mensimulasikan efek sebenarnya pada obyek-obyek tersebut sesuai karakteristik tiap obyek, seperti : memantul, pecah, dan lain sebagainya.

3. ***Platform Abstraction***

   Platform abstraction mempermudah untuk mengembangkan *engine* disetiap *platform* yang berbeda, dan beberapa *game engine* bisa mengembangkan *engine* pada *platform* konsole.

4. ***IDE (Integrated Development Environment)***

   Game engine menyederhanakan dan memudahkan proses pengembangan *engine*, seperti koding, penambahan visual atau efek suara, memasukkan *AI (Artificial Intelligence)* ke dalam *engine*, pengembangan *engine* dengan *networking* dan lain sebagainya.

Namun suatu *game* engine tidak wajib memiliki komponen – komponen seperti yang disebutkan diatas. Komponen – komponen tersebut merupakan garis besar dari beberapa *engine game* yang ada saat ini, sehingga bila terjadi kekurangan komponen pada *game engine* yang digunakan pengembang *engine* perlu melengkapi komponen tersebut dengan *tools – tools* yang lain.

2.2.4. *Engine Torque*

Torque adalah salah satu *game engine* yang banyak digunakan oleh pengembang *engine* saat ini. *Torque* merupakan versi modifikasi dari *game engine* yang digunakan

pada *engine FPS (First Person Shooter) Tribe 2* di tahun 2001 dan sampai saat ini lisensi dari *game engine Torque* dipegang oleh *GarageGames*. Beberapa contoh *engine* yang komersil yang dibangun dengan *game engine Torque* adalah; *Ultimate Hunting Duck, Wildlife Tycoon : Venture Africa.*

2.2.5. Komponen *Game Engine Torque*

Game engine Torque mempunyai garis besar komponen yang sama dengan *game engine* lainnya. *Game engine Torque* mempunyai script untuk networking, pembuatan *GUI (Graphical User Interface)* dan desain 3D atau 2D. *Game engine Torque* secara otomatis bisa membuat tampilan lapisan tanah yang detail, kemudian menggabungkannya dengan tekstur yang akan diletakkan diatas lapisan tanah tanpa perlu melakukan pengeditan lapisan tekstur dan lapisan tanah. Rendering yang ditawarkan *game engine Torque* adalah *environment mapping, gouraud shading, volumetric fog* dan beberapa efek lainnya, sedangkan untuk pengembangan *game network* seperti *game online* dan *lan game*, tetap menggunakan sistem arsitektur *client – server.*

2.2.6. Jenis – Jenis *Game Engine Torque*

Perkembangan *game* yang cepat dan munculnya berbagai macam *platform* dan genre, membuat *Torque* mengembangkan *game engine*-nya menjadi lebih spesifik. Sehingga muncul tiga jenis *game engine Torque* yang mempunyai fungsi – fungsi yang lebih spesifik, yaitu :

1. ***Torque Game Engine Advance***

   Torque engine game advance adalah *game engine Torque* yang dikembangkan untuk mampu mensupport perkembangan teknologi yang lebih maju seperti *shaders*, *pixel* per *lighting*, lapisan tanah yang beraneka ragam. Jenis *game engine Torque* ini lebih ditujukan kepada *platform* xbox dan xbox 360.

2. ***Torque Game Builder***

   Setelah *Torque game engine advance*, *Torque* membuat *game engine* yang ditujukan untuk mendesain *engine* 2D yang berbasiskan *game engine Torque*, dengan nama *Torque* 2D. *Torque* 2D kemudian berubah nama menjadi *Torque engine builder* karena *Torque* ingin membuat suatu *game-making suite.*

3. ***Torque Lighting Kit***

   Torque lighting kit merupakan modul tambahan dari *Torque engine game*. *Torque lighting kit* memberikan berbagai variasi efek *lighting* pada *Torque game engine*, dan pada versi terakhir *Torque lighting kit* memberikan tambahan efek seperti *dynamic lighting* dan *shadowing*.

2.2.7. *TorqueScript*

TorqueScript adalaah sebuah bahasa pemrograman yang kuat, fleksibel, dan mempunyai sintaks yang mirip sekali dengan bahasa *C++*. Dibawah ini akan dijelaskan seluruh fitur-fitur dari bahasa ini.

• ***Type-insensitive***

Dalam TorqueScript, tipe variable di konversikan seperlunya saja dan bisa dipakai secara bergantian. *TorqueScript* menyediakan beberapa beberapa tipe variable standar yang sudah umum dipakai.

• ***Case-insensitive***

TorqueScript tidak *case-sensitive* dalam menerjemahkan variable dan nama *function.*

• ***Statement termination***

Statement dalam *TorqueScript* dianggap selesai apabila ditandai dengan (;) sama seperti pada bahasa *programming* modern lainnya sepert *Java* dan *C++*.

• ***Full complement of operators***

TorqueScript menyediakan operator-operator standar yang sudah umum pada sebagian besar bahasa pemrograman, dan juga beberapa operator-operator yang lebih maju.

• ***Full complement of control structures***

Sama seperti bahasa pemrograman canggih lainnya*, TorqueScript* menyediakan susunan bahasa pemrograman yang sudah umum seperti*: if-then-else, for, while*, dan *switch.*

• ***Functions***

TorqueScript menyediakan kemampuan untuk membuat sebuah function dengan kemampuan *optional* untuk mengembalikan nilai. Parameter-parameter dikembalikan dengan metode *passed bt value* maupun *passed by reference.*

• ***Inheritance and Polymorphism***

TorqueScript memungkinkan *programmer* untuk membuat turunan dari sebuah objek didalam *engine* dan mengembangkan ataupun memodifikasi *method-method* dari objek tersebut.

• ***Namespaces***

Sama seperti didalam *C++, TorqueScript* mendukung konsep dari *namespaces. Namespace* digunakan untuk me lokalkan nama dan identifikasi untuk mencegah tabrakan dalam pembacaan. Ini berarti, sebagai contoh, *programmer* bisa memiliki dua *function* yang berbeda bernama test() yang ada didalam dua *namespace* yang berbeda, tetapi keduanya digunakan dalam 1 *code.*

• ***Compiles dan executes code***

TorqueScript meng *compile script* bersamaan saat eksekusi, ini akan memberikan peningkatan kecepatan kerja dan juga menyediakan sebuah titik dimana tempat kemungkinan adanya kesalahan dalam *script*.

2.3. Dimensi

2.3.1. Pengertian Dimensi

Dimensi dapat diartikan sebagai banyak cara untuk menentukan posisi sebuah benda yang didasarkan terhadap acuan tertentu. Sebuah benda dikatakan berdimensi satu bila posisinya bisa ditentukan dengan sebuah angka. Benda berdimensi 2 memiliki posisi yang bisa ditentukan oleh 2 angka, misalnya suatu permukaan bola dapat diukur dengan angka derajat lintang dan angka derajat bujur.

2.3.2. Pengertian 2 Dimensi

Objek 2 dimensi direpresentasikan dalam sebuah bidang yang terdiri dari sumbu x dan y, sering juga disebut bidang cartesian. Objek yang bisa dibentuk dalam bidang ini dapat berupa titik, garis, maupun poligon. Sebuah objek titik terbentuk dari 2 koordinat (x,y) yang spesifik, dimana koordinat x menandakan suatu posisi yang terletak pada sumbu mendatar / horizontal dan koordinat y menandakan suatu posisi yang terletak pada sumbu tegak / vertikal.

Objek titik dalam bidang 2 dimensi dapat membentuk kumpulan objek garis yang saling terhubung, yang dapat membentuk suatu objek segi banyak tertutup (poligon) ataupun objek segi banyak terbuka. Masing-masing titik dari sebuah poligon disebut *vertex*

2.4. *UML (Unified Modelling Language)*

2.4.1. Pengertian *UML*

Menurut Bernd Brugge dan Allen H Dutoit (2000, p24), notasi membantu menyampaikan ide kompleks secara ringkas dan tepat. Dalam proyek yang melibatkan banyak anggota, seringkali latar belakang kebudayaan dan teknik yang berbeda, ketepatan dan kejelasan adalah aspek yang sangat penting mengingat begitu mudahnya terjadi salah pengertian. Agar suatu notasi dapat menjadi alat komunikasi yang akurat, notasi tersebut harus emiliki semantik yang telah didefinisikan dengan baik, dapat merepresentasikan aspek dari sistem secara tepat, dan dapat dimengerti dengan baik oleh para anggota tim.

Menurut Whitten, Bentley, dan Dittman (2004, p430), *Unified Modelling Langguage*™ (UML) adalah sebuah pendekatan untuk (1) mempelajari objek-objek yang ada untuk melihat apakah objek tersebut dapat digunakan kembali atau dimodifikasi untuk kegunaan baru, dan (2) mendefinisikan objek baru atau yang telah dimodifikasi yang akan digabungkan dengan objek yang ada untuk membuat aplikasi bisnis.

Menurut martin Fowler dan Kendall Scoot (2000, p13), UML adalah bahasa pemodelan, bukan suatu metode. UML tidak memiliki notasi process yang merupakan bagian penting dari metode.

Menurut Thimothy C. Lethbridge dan robert Laganiere (2002, p151), UML adalah standart bahasa grafis untuk memodelkan *software* berorientasi objek. UML dikembangkan pada pertengahan tahun 1990an oleh James Rumbaugh, Grady Booch, dan Ivan Jacobson dimana mereka telah mengembangkan notasi mereka masing-masing

pada awal 1990an. Pada November 1997, UML diresmikan sebagai standart untuk pemodelan objek oleh Object Management Group (OMG).

2.4.2. Jenis-jenis *UML*

Menurut Whitten, Bently, dan Dittman (2004, p441), UML terdiri atas sembilan *diagram* yang dikelompokan dalam lima kategori berdasarkan sudut pandangnya, yaitu :

Kategori 1 : Use – Case Model *Diagram*

Use-case diagram menggambarkan interaksi antara sistem dengan luar sistem sistem dengan *user.* Dengan kata lain, *use case diagram* secara grafis mengambarkan siapa yang akan menggunakan sistem dan dalam cara apa *user* ingin berinteraksi dengan sistem.

Kategori 2 : Static Stucture *Diagram*

UML mena*war*kan dua *diagram* untuk memodelkan struktur statis dari sistem informasi, yaitu :

a. Class Diagram

Class diagram melukiskan struktur sistem dalam bentuk objek. Disini digambarkan objek *class* yang membangun sistem beserta hubungan antar *class.*

b. Object Diagram

Object diagram serupa dengan *class diagram,* tetapi disamping mengambarkan objek *class,* digambarkan juga *instance* yang menampilkan nilai atribut dari *instance. Diagram* ini dapat digunakan untuk membantu tim pengembang memahami struktur sistem dengan baik.

Kategori 3 : *Interaction Diagram*

Iteraction diagram memodelkan interaksi, terdiri dari sekumpulan objek, hubungan dan pesan yang dikirimkan antar objek tersebut. *Diagram* ini memodelkan aspek dinamis dari sistem.

UML memiliki dua *diagram* untuk tujuan ini, yaitu :

a. Sequence Diagram

Sequence diagram secara grafis menggambarkan bagaiman objek saling berinteraksi melalui pesan dalam melakukan suatu operasi atau melakukan *use case*.

b. Collaboration Diagram

Collaboration diagram serupa dengan *sequence diagram*, tetapi yang difokuskan disini bukanlah urutan (*sequence*) melainkan interaksi antar objek dalam format jaringan.

Kategori 4 : State *Diagram*

State diagram juga memodelkan aspek dinamis dari sistem. UML memiliki *diagram* untuk memodelkan perilaku kompleks dari objek dan *diagram* untuk memodelkan perilaku dari *use case* atau model**.** *Diagram* tersebut adalah :

a. *Statechart Diagram*

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

b. *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 :

a. Component *Diagram*

Component diagram digunakan untuk menggambarkan organisasi dari sistem dan ketergantungan dari komponen *software* dalam sistem. *Component diaram* dapat juga digunakan untuk menunjukan bagaimana kode program dibagi menjadi modul-modul (atau komponen).

b. *Deployment Diagram*

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

2.4.2.1. *Use Case Diagram*

Menurut Martin Fowler dan Kendall Scott (2001, p39), skenario adalah urutan dari langkah langkah mendeskripsikan interaksi antar *user* dengan sistem. *Use case* adalah sekumpulan skenario yang digabungkan bersama untuk mencapai suatu tujuan tertentu.

Menurut Bernd Brugge dan Allen H Dutoit (2000, p25), *use case* digunakan selama masa pengumpulan kebutuhan dan analisis untuk mempresentasikan fungsionalitas dari sistem. *Use case* berpsat pada perilaku sistem pada sudut pandang luar.

Menurut Whitten, Bentley, dan Dittman (2004, p271-273), *use case diagram* terdiri dari *use case, actor (user)*, dan hubungan diantaranya. Actor adalah apa saja yang perluberinteraksi dengan sistem untuk bertukar informasi. *Actor* tidak harus manusia. *Actor* dapat juga merupakan organisasi, sistem informasi yang lain, alat alat di luar sistem seperti sensor, atau bahkan konsepakan waktu.

Menurut Bernd Brugge dan Allen H Dutoit (2000, p25), identifikasi dari *actor* dan *Associations* mrnghasilkan batasan sistem dan membedakan tugasyang dikerjakan sistem dengan tugas yang dikerjakan oleh lingkungan sistem.

Menurut Whitten, Bentley, dan Dittman (2004, p273-276), ada 5 tipe hubungan dengan *use case diagram*, yaitu:

1. *Associations*

Hubungan antara *actor* dan *use case* yang timbul dimana *use case* mendeskripsikan interaksi yang terjadi. *Associations* digambarkan dengan garis lurus menghubungkan *actor* dengan *use case. Associations* yang memiliki panah pada ujung yang menyentuh *use case* mengindikasikan bahwa *use case* dilakuakan *actor* pada ujung lainnya. *Associations* tanpa tanda panah mengindikasikan interaksi antara *use case* dan server luar dan *actor* penerima.

2. *Extends Use case*

Dapat mengandung fungsionalitas yang kompleks berisi beberapa langkah yang membuat use case menjadi sulit dimengerti. Untuk menyederhanakan *use case* dan

menjadikan lebih mudah dimengerti, jika dapat memecah langkah langkah kompleks tersebut menjadi *use case* terpisah. Hasilnya disebut *extension use case*. Hubungan antara *extension use case* dengan *use case* utamanya disebut *extends*. Hubungan *extends* digambarkan dengan garis dengan panah, bermula *extension use case* dan menunjuk pada use case utamanya. Setiap hubungan ini diberi label "<<extends>>."

3. *Uses* (atau *Includes*)

Seringkali ditemukan dua atau lebih *use case* melakukan langkah yang memiliki fungsionalitas yang sama. Cara terbaik adalah untuk memisahkannya menjadi *use case* terpisah yang disebut *abstract use case. Abstract use case* dapat digunakan oleh *use case* lain yang memerlukan fungsionalitasnya. Hubungan antara *abstract use case* dengan *use case* yang menggunakannya disebut dengan *uses*. Hubungan *uses* digambarkan dengan aris bertanda panah, bermula pada *use case* dan penunjuk pada *use case* yang menggunakannya. Setiap hubungan ini diberi label "<<uses>>."

4. *Depends On*

Hubungan antar *use case* dimana kondisi awal suatu *use case* bergantung pada kondisi akhir *use case* yan lain. Hubungan *depends on* digambarkan dengan garis bertanda panah, bermula pada *use case* yang menunjukan pada *use case* yang digantungkan. Setiap hubungan ini diberi label "<<depends on>>."

5. *Inheritance*

Saat dua atau lebih *actor* berbagi perilaku yang sama, cara yang terbaik adlah membuat *abstract actor* untuk elakukan perilaku tersebut agar mengurangi redudansi sistem, dengan demikian *actor* me*war*isi kemampuan untuk memulai sebuah *use case.*

2.4.2.2. *Class Diagram*

Menurut Whitten, Bentley, dan Dittman (2004, p431-435), *class* adalah kumpulan objek yang berbagi atribut dan perilaku yang sama. Objek adalah sesuatau yang dapat dilihat, disentuh, atau dapat dirasakan dan merupakan tempat *user* menyimpan data dan perilaku. Atribut adalah data yang merepresentasikan karakteristik dari suatu objek. Perilaku adalah kumpulan hal-hal yang dapat dilakukan oleh objek dan berhubungan dengan data yang dimilikinya.

Menurut Whitten, Bentley, dan Dittman (2004, p455), *class digram* adlah gambar dari struktur objek statis dalam sistem, menunjukan objek *class* yang menyusun sistem beserta hubungan antar objek *class* tersebut.

Pada waktu tingkatan-tingkatan *class* diidentifikasi, konsep ***inheritance*** diterapkan. *Inheritance* adalah konsep dimana metode dan/atau atribut yang telah didefinisikan dalam objek *class* dapat di*war*iskan atau digunakan kembali oleh objek *class* yang lain. Pendekatan yang mencari dan memanfaatkan persamaan antar objek dan *class* menunjuk pada ***generalization/specialization.*** *generalization/specialization* adalah teknik dimana atribut dan perilaku yang sama pada beberapa tipe objek *class* dikelompokan (atau diabstraksi) menjadi *clas* tersendiri, yang disebut *supertype.* Atribut dan metode objek *class supertype* di*war*iskan ke *class subtype.*

Menurut Whitten, Bentley, dan Dittman (2004, p435-438), macam-macam hubungan antar *class* adalah sebagai berikut :

1. *Association*

Menggambarkan hubungan antar *class.* Dalam *association* dapat ditampilkan kompleksitas atau derajat dari setiap *association.* Konsep ini disebut ***multiplicity.***

Multiplicity adalah nilai minimum dan maksimum dari keberadaan dari satu objek atau *class* untuk sebuah keberadaan objek atau *class* yang terhubung.

2. *Aggregation*

Hubungan antar *class* dimana suatu *class* 'utuh' mengandung satu atau lebih *class* 'bagian' yang lebih kecil. Dapat pula dilihat bahwa *class* 'bagian' merupakan bagian dari *class* 'utuh'. *Aggregation* digambarkan dengan wajik berlubang terhubung pada *class* 'utuh'.

3. *Composition*

Merupakan hubungan *aggregration* dimana *class* 'utuh' bertanggung jawab atas pembuatan dan penghancuran *class* 'bagian'. Jika *class* 'utuh' akan dihancurkan, *class* 'bagian' akan ikut hancur. *Composition* digambarkan dengan wajik penuh terhubung pada *class* 'utuh'.

Menurut Whitten, Bentley, dan Dittman (2004, p459), relasi *aggregation* adalah hubungan asimetris. Objek B merupakan bagian dari objek A tetapi objek A bukan bagian dari objek B. Relasi ini tidak menyertakan *inheritance,* sehingga objek B tidak me*war*isi atribut dan perilaku dari objek A.

2.4.2.3. *Sequence Diagram*

Menurut Whitten, Bentley, dan Dittman (2004, p702), *sequence diagram* adalah *diagram* UML yang memodelkan aspek logika dari *use case* dengan menggambarkan interaksi dari pesan-pesan antara objek dalam urutan waktu. *Sequence diagram* menunjukan secara detail bagaimana objek saling berinteraksi pada urutan waktu, dibaca dari atas ke bawah cera berurut mengikuti logika dari *use case.* Tulisan dari langkah-

langkah *use case* dapat ditulis dalam *pseudocode* pada sebelah kiri *diagram* untuk mempermudah pembacaan.

Setiap objek digambarkan dalam simbol objek UML. Keterangan dari *use case* disimbolkan dengan garis vertikal yang disebut *lifeline.* Perilaku atau operasi yang perlu dilakukan oleh setiap objek digambarkan dengan empat persegi panjang pada *lifeline* objek. Empat persegi panjang ini mengambarkan kode program. Panah antar garis merepresentasikan interaksi atau pesan yang dikirimkan ke objek tertentu untuk memanggil operasi untuk memenuhi suatu permintaan.

2.4.2.4. *Statechart Diagram (State Diagram)*

Menurut Whitten, Bentley, dan Dittman (2004, p700-701), *statechart diagram* adalah *diagram* UML yang menggambarkan macam-macam keadaan yang dapat dimiliki objek, peristiwa yang memicu terjadinya transisi antar keadaan, dan aturan-aturan yang mengatur transisi tersebut. Suatu objek berganti keadaan saat nilai dan atributnya berubah. *Statechart diagram* tidak diperlukan untuk semua objek. *Statechart diagram* dibuat hanya untuk objek yang secrar jelas memiliki keadaan yang dapat diidentifikasi dan perilaku yang kompleks.

Statechart diagram diawali dengan keadaan awal (lingkaran hitam penuh) dan transisi melewati berbagai keadaan dalam daur hidup objek (segi empat tidak bersudut) sampai keadaan akhir (lingkaran hitam penuh dengan lingkaran di luarnya). Panah mengambarkan kejadian yang memicu perubahan keadaan ke keadaan yang lainnya.

2.4.2.5. *Activity Diagram*

Menurut Whitten, Bentley, dan Dittman (2004, p450-454), *activity diagram* adalah *diagram* yang dapat digunakan untuk mengambarkan secara grafis aliran dari sebuah proses bisnis, langkah-langkah dari sebuah *use case,* atau logika dari suatu perilaku objek (metode). *Activity diagram* serupa dengan *flowchart* dalam mengambarkan urutan alur aktivitas dari proses bisnis atau sebuah *use case,* tetapi berbeda dengan *flowchart* dalam penyediaan mekanisme untuk mengambarkan aktivitas yang muncul dalam waktu bersamaan.

Activity diagram dapat digunakan selama masa analisa dan desain. Sedikitnya sebuah *activity diagram* dapat dibuat dari sebuah *use case.* Jika *use case* terlalu panjang atau memiliki logika yang kompleks maka dapat dibuat lebih dari satu *activity diagram. System analyst* menggunakan *activity diagram* untuk memberi pemahaman lebih akan aliran dan urutan dari langkah-langkah dalam *use case.*

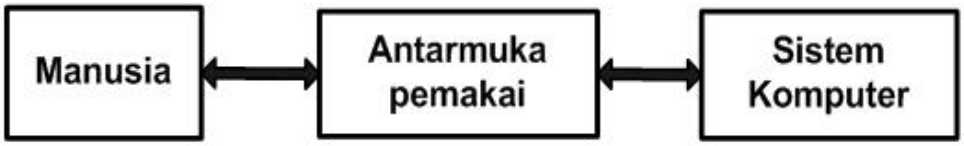
Lingkaran hitam penuh merepresentaikan awal dari proses. Segi empat tidak bersudut merepresentasikan sebuah aktivitas atau tuas yang perlu dilakukan. Panah menggambarkan pemicu yang menginisiasikan aktivitas yang berjalan bersamaan. Teks di dalam [ ] merepresentasikan pemicu yang merupakan hasil dari aktivitas pengambilam keputusan. Wajik merepresentasikan aktivistas pengambilam keputusan. Lingkaran hitam penuh dengan lingkaran diluarnyamenggambarkan akhir dari proses.

2.5. IMK (Interaksi Manusia dan Komputer)

2.5.1. Pengertian IMK (Interaksi Manusia dan Komputer)

Menurut Ben Shneiderman (1997), Interaksi Manusia dan Komputer (IMK) atau *Human-Komputer Interaction* (HCI) adlah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya.

Fokus pada IMK adalah perancangandan evaluasi antarmuka pemakai (*user interface*). Antarmuka pemakai adalah bagian sistem komputer yang memungkinkan manusia berinteraksi dengan komputer.



Gambar. 2. 12. Fokus Pada IMK (Interaksi Manusia dan Komputer)

2.5.2. Konsep Perancangan Antarmuka Pemakai

Dalam IMK, terdapat 8 (delapan) aturan emas (*golden rules*) yang digunakan dalam perancangan antarmuka pemakai yaitu :

a. Berusaha untuk konsisten

b. Memungkinkan *frequent users* menggunakan *shortcuts*

c. Memberikan umpan balik (*feedback*) yang informatif

d. Memberikan dialog yang memberikan penutupan (keadaan akhir)

e. Memberikan pencegahan kesalahan dan penanganan yang sederhana

f. Memungkinkan pembalikan aksi yang mudah

g. Mendukung pusat kendali internal (*internal locus of control*)

h. Mengurangi beban ingatan jangka pendek

2.6 Multimedia

2.6.1. Pengertian Multimedia

Menurut Wikipedia (http://id.wikipedia.org/wiki/multimedia). Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, grafik, video, animasi, dengan alat bantu (*tools*) dan koneksi (*link*) sehingga pengguna dapat bernavigasi, berinteraksi, berkarya, dan berkomunikasi. Multimedia sering digunakan dalam dunia hiburan.

2.6.2. Elemen Multimedia

Elemen-elemen yang terdapat dalam multimedia adalah :

a. Teks

Merupakan dasar penyampaian informasi, juga merupakan media paling sederhana yang direpresentasikan deangan *typeface* (jenis huruf) yang beragam agar harmonis dengan elemen media lainnya.

b. Citra diam (gambar)

Merupakan representasi spesial dari objek yang disusun sebagai matriks nilai numerik yang merepresentasikan setiap titik/*pixel* serta diciptakan dengan program *image editing.*

c. Suara atau audio

Merupakan fenomena fisik yang dihasilkan oleh pengantar materi.

Audio memiliki 3 kategori, yaitu :

- Ucapan (*speech*) : suara orang berbicara
- Musik (*music*) : hasil pendengaran alat musik
- Efek suara (*sound effect*) : suara lainnya, seperti : petir, gelas pecah, tembakan, dll

d. Animasi

Merupakan penayangan frame-frame gambar secara cepat untuk menghasilkan kesan gerakan.

e. Video

Video sama seperti animasi, tetapi disimpan dalam format khusus yang dapat menyimpan adegan dunia nyata atau rekaan dengan komputer. Video adalah elemen yang paling kompleks, dan paling memerulukan persyaratan *hardware* yang tinggi.