BAB II

LANDASAN TEORI

2.1 *Computer Game*

Computer game atau juga *game* komputer adalah *video game* yang dimainkan pada sebuah *Personal Computer* (PC). Menurut Mark J.P Wolf dalam buku nya "*The Medium of the Video Game*" , permainan dalam *video* dan *computer game* dapat diklasifikan menjadi 42 tipe [LIO08],[1] yaitu :

1. *Abstrac*

   Jenis *game* yang di presentasikan secara sederhana. Contoh : Tetris, Arkanoid.

2. *Adaptation*

   Jenis *game* yang diadaptasi dari aktifitas sehari-hari. Contoh: Solitaire, Black Jack (adaptasi permainan kartu), Rubik Cube.

3. *Adventure*

   Jenis *game* yang bersetting "dunia", biasanya dibuat untuk bermain secara *multiplayer*. Contoh: Tomb Raider.

4. *Artificial Life*

   Jenis game yang mengajak kita untuk dapat merawat kehidupan (pertumbuhan, kematian) suatu karakter digital. Contoh: Tamagochi, The Sims.

[1] Artikel yang diambil dari website ini hanya mengutip (bab 6 : *Genre and The Video Game*) dari buku Mark J.P Wolf *"The Medium of The Video Games"*.

5. *Board Games*

   Jenis *game* yang diadaptasi dari permainan papan (*board game)* atau *game* yang mirip seperti *board game.* Contoh: Monopoli, Othello.

6. *Capturing*

   Jenis *game* yang bertujuan menangkap suatu karakter yang secara aktif berusaha melarikan diri dan menghilang dari karakter pemain. Contoh: Hole Hunter, Gopher.

7. *Card Games*

   Jenis *game* yang berdasarkan dari berbagai permainan kartu yang ada. Contoh: Black Jack, Poker.

8. *Catching*

   Sama seperti *capturing* hanya karakter yang dikejar tidak secara aktif berusaha menghilang dari kejaran karakter pemain. Contoh: Big Bird's Egg Catch.

9. *Chase* (Lihat *Catching, Capturing, Driving, Escape, Flying,* dan *Racing*).

10. *Collecting*

    Jenis *game* yang bertujuan mengumpulkan benda-benda yang tidak bergerak. Contoh: PacMan, Snake.

11. *Combat*

    Jenis *game* yang terdiri dua pemain atau lebih yang saling bertempur untuk saling mengalahkan. Contoh: Warlords.

12. *Demo*

    Jenis *game* yang khusus dikeluarkan untuk demo atau promosi.

13. *Diagnostic*

    Jenis *game* yang khusus dikeluarkan untuk mencoba dari sistem yang akan dipakai.

14. *Dodging*

    Jenis *game* yang tujuan utamanya menghindari benda-benda tertentu yang bergerak. Contoh: Frogging.

15. *Driving*

    Jenis *game* yang berdasar pada kemampuan dasar berkendara (menyetir, manuver, kontrol kecepatan dan lain-lain). Contoh: Night Driver, Pole Position.

16. *Educational*

    Jenis *game* yang dibuat untuk memberikan pengajaran. Contoh: Spelling Game, Word Games.

17. *Escape*

    Jenis *game* yang tujuannya melarikan diri dari pengejar. Contoh: Pac-Man, Ms.Pac-Man.

18. *Fighting*

    Jenis *game* yang melibatkan karakter-karakter yang saling berhadapan dan berkelahi. Contoh: Mortal Kombat, Street Fighter.

19. *Flying*

    Jenis *game* yang berdasar pada kemampuan untuk mengendarai pesawat terbang (lepas landas, mendarat, manuver, kontrol kecepatan dan lain-lain. Contoh: Flight Unlimited.

20. *Gambling*

    Jenis *game* yang mengajak pemain untuk mempertaruhkan "harta" miliknya (judi). Contoh: Black Jack, Poker.

21. *Interactive Movie*

    Jenis *game* yang jalan ceritanya dapat bercabang dan ditentukan oleh pemain. Contoh: Dragon's Lair, Johny Mnemonic.

22. *Management Simulation*

    Jenis *game* dimana pemain diharuskan untuk mengatur sumber daya yang akan digunakan untuk membangun semacam komunitas, institusi atau kerajaan. Contoh: Sim City.

23. *Maze*

    Jenis *game* yang tujuan akhirnya membutuhkan penguasaan arah pada sebuah *maze* (lorong). Contoh: Pac-Man, Ms.Pac-Man, Doom.

24. *Obstacle Course*

    Jenis *game* yang tujuannya menelusuri jalur yang sulit.

25. *Pencil and Paper Games*

    Jenis *game* yang diadaptasi dari permainan yang dimainkan dengan pensil dan kertas. Contoh: Tic-Tac-Toe, Hangman.

26. *Pinball*

    Jenis *game* yang mensimulasikan permainan pinball. Contoh: Arcade Pinball, Astrocade Pinball, Electronic Pinball.

27. *Platform*

Jenis *game* yang tujuan akhirnya dicapai dengan bergerak melalui beberapa jalan bertingkat. Contoh: SpiderMan, Donkey Kong.

28. *Programming Games*

Jenis *game* dimana pemainnya menuliskan program singkat untuk mengontrol karakter dalam game tersebut. Contoh: AI Wars.

29. *Puzzle*

Jenis *game* yang tidak menonjolkan konflik antar pemain namun lebih kepada mencari tahu solusi atau metode penyelesaiannya. Contoh: Rubik Cube, Sokoban, Tetris.

30. *Quiz*

Jenis *game* yang tujuan akhirnya adalah menjawab suatu pertanyaan dengan benar. Contoh: Video Trivia,Wizz Quiz.

31. *Racing*

Jenis *game* yang tujuan utamanya adalah memenangkan balapan. Contoh: Gran Tourismo, Need For Speed.

32. *Role Playing*

Jenis *game* dimana tiap pemain menciptakan atau memiliki karakter, dimana karakter masing-masing memiliki kemampuan yang berbeda berdasar statistik. Contoh: Anvil of Dawn, Diablo.

33. *Rhythm and Dance*

Jenis *game* dimana pemain harus menselaraskan antara waktu dan ritme musik. Contoh: Guitar Hero, VOS.

34. *Shoot 'Em Up (Shooter)*

Jenis *game* yang bertujuan berperang (tembak menembak) melawan musuh yang biasanya cenderung lebih banyak dari pemain. Contoh: Doom, Freedom Fighter.

35. *Simulation* (Lihat *Management Simulation and Training Simulation*).

36. *Sports*

Jenis *game* yang diadaptasi dari permainan olahraga di dunia nyata. Contoh: Winning Eleven, PES, American Football.

37. *Strategy*

Jenis *game* yang menekankan pemakaian strategi. Contoh: Chess.

38. *Table Top Games*

Jenis *game* yang diadaptasi dari permainan *Table Top* (di atas meja). Contoh: Battle Ping Pong, Cool Pool.

39. *Target*

Jenis *game* yang bertujuan membidik dan menembak target. Contoh: Skeet Shoot, Wabbit.

40. *Text Adventure*

Jenis game yang menggunakan text sebagai media perantaranya. Contoh: Planetfall, Zork.

41. *Training Simulation*

Jenis *game* yang menggambarkan keadaan seperti di dunia nyata dengan tujuan melatih kemampuan (menyetir atau menerbangkan). Contoh: A-10 Attack, Comanche 3.

42. *Utility*

Program yang memiliki tujuan atau fungsi tertentu yang dibuat sedemikian rupa seperti *game.*Contoh: Mario Teaches Typing.

Dari pembagian klasifikasi menurut J.P Wolf diatas terlihat bahwa satu atau lebih *game* masuk dalam lebih dari satu tipe klasifikasi (lihat Tetris pada *puzzle* dan *abstract*, rubik cube pada *puzzle* dan *adaptation*). Hal ini dikarenakan tidak ada pembagian tipe *game* yang menjadi standar. Jadi dapat disimpulkan bahwa sesungguhnya pembagian tipe *game* bisa bervariasi.

2.2 *Rubik Cube*

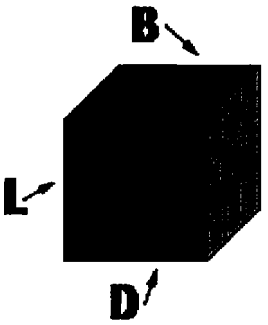
Rubik cube adalah permainan berjenis *puzzle* yang diciptakan pada tahun 1974 oleh seorang pemahat yang juga profesor arsitektur bernama Erno Rubik yang berasal dari Hungaria. Pada awal diciptakannya, permainan ini bernama *magic cube,* yang kemudian pada tahun 1980 nama tersebut diubah menjadi *rubik cube* oleh sebuah perusahaan mainan bernama Ideal Toys.

Gambar 2.1 Prototipe Awal *Rubik Cube*

Rubik cube tersusun atas dua puluh tujuh buah kubus yang saling berkaitan, namun pada kenyataannya sisi pusat kubus bukanlah kubus namun sebuah pengait yang berfungsi untuk menyatukan ke dua puluh enam kubus-kubus lainnya, yang juga berfungsi sebagai pusat dari perputaran kubus lainnya. Sebuah *rubik cube* memiliki enam warna berbeda pada tiap-tiap sisinya (merah, oranye, putih, kuning, biru dan hijau), dimana pada umumnya susunan warnanya adalah (merah berlawanan arah dengan oranye, putih berlawanan arah dengan kuning, biru berlawanan arah dengan hijau) seperti terlihat pada gambar berikut.

Gambar 2.2 Susunan Warna-Warna Pada *Rubik Cube*

Hingga kini telah dikenal standar notasi yang sering digunakan dalam permainan ini, yaitu standar yang dibuat oleh David Singmaster. Standar notasi ini digunakan untuk menandai sisi maupun arah perputaran sisi-sisi *rubik.*



Gambar 2.3 Sisi-sisi *Rubik*

R (Right) : sisi yang berlawanan dengan sisi *Left.*

L (Left) : sisi yang berada di sebelah kiri pemain (saat *rubik* lurus).

U (Up) : sisi yang berada pada bagian atas *rubik.*

D (Down) : sisi yang berlawanan dengan sisi *Up.*

F (Front) : sisi yang menghadap ke pemain.

B (Back) : sisi yang berlawanan dengan sisi *Front.*

Sementara itu terdapat tiga aturan perputaran *rubik* yang menjadi standar yaitu :

( ) : CW atau searah perputaran jarum jam (90°), tanpa simbol apapun. Contoh: F B L R.

(') : CCW berlawanan perputaran jarum jam (90°), disimbolkan dengan sebuah tanda petik . Contoh: F' B' L' R'

('') : Dua kali putaran *rubik* pada arah yang sama (180°), disimbolkan dengan dua tanda petik. Contoh: F'' B'' L'' R''.

(**2**) atau (^{2}) : Beda bentuk penulisan dari simbol ('') dua tanda kutip, yang sama saja maksudn yaitu dua kali perputaran *rubik* pada arah yang sama (180°). Contoh: F2 B2 L2 R2 atau F^2 B^2 L^2 R^2

Gambar 2.4 Arah Perputaran Rubik

Dapat dilihat pada Gambar 2.4 diatas (lihat dari kiri ke kanan) terdapat tiga macam perputaran *rubik* adalah:

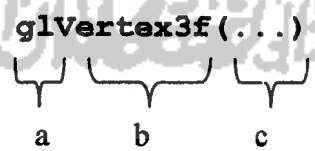
F – Sisi depan diputar 90° CW (searah jarum jam).

F' – Sisi depan diputar 90° CCW (berlawanan arah jarum jam).

F2 – Sisi depan diputar 180° CW (searah jarum jam).

2.3 OpenGL

''OpenGL is a three dimensional system. From application programmer's perspective, OpenGL primitives describe three dimensional objects that exist in a three dimensional world'', dari kutipan buku OpenGL A Primer [ANG05] tersebut dapat diberikan penjelasan bahwa OpenGL merupakan sebuah sistem tiga dimensi. Dimana dari sisi pemrogram, OpenGL dapat diartikan sebagai rutin-rutin dasar (primitif) yang menggambarkan obyek-obyek tiga dimensi pada dunia nyata (tiga dimensi), dimana hal tersebut diwujudkan dengan API (*Application Programming Interface*).



Diatas merupakan contoh salah satu fungsi dasar dari OpenGL, fungsi tersebut berguna untuk membuat lokasi sebuah titik, keterangan berikut ini:

a = kepustakaan GL (fungsi tersebut masuk dalam kepustakaan atau library opengl).

b = perintah dasar (fungsi tersebut merupakan fungsi penentu lokasi titik 3dimensi dengan argumen yang diisikan bertipe float).

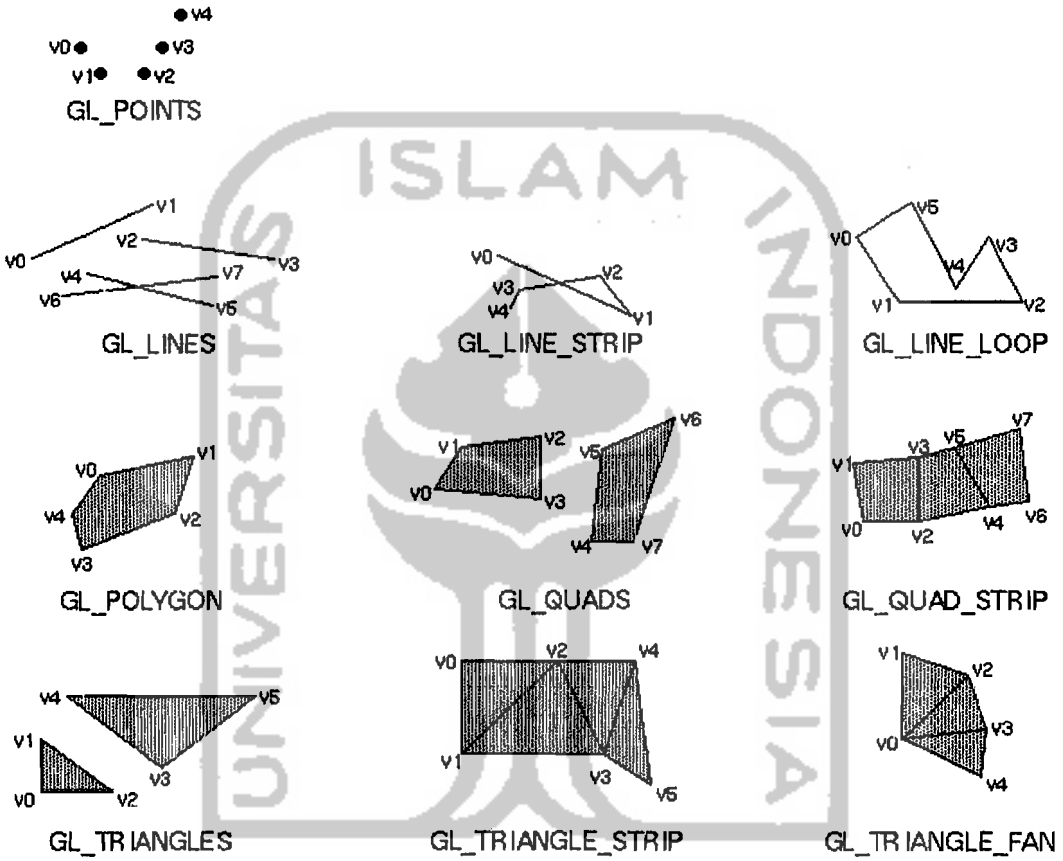
c = isian argumen atau variabel yang diisi (koordinat titik x,y dan z).

Tabel 2.1 Perintah-Perintah pada OpenGL

Perintah	**Arti**	**Keterangan**
glVertex2i(x,y)	Lokasi titik berada di (x,y)	Tipe argumen adalah integer dan 2Dimensi yaitu x dan y
glVertex2f(x,y)	Lokasi titik berada di (x,y)	Tipe argumen adalah float dan 2Dimensi yaitu x dan y
glVertex3i(x,y,z)	Lokasi titik berada di (x,y,z)	Tipe argumen adalah integer dan 3Dimensi yaitu x,y,z.
glVertex3f(x,y)	Lokasi titik berada di (x,y,z)	Tipe argumen adalah float dan 3Dimensi yaitu x dan y
glClearColor(R,G,B,α)	Warna latar belakang	Komponen warna Red, Green, Blue dan alpha
glColor3f(R,G,B)	Warna latar depan	Komponen warna Red, Green dan Blue
glColor4f(R,G,B,α)	Warna latar depan	Komponen warna Red, Green, Blue dan alpha
glPointSize(k)	Ukuran titik k piksel	k adalah bilangan integer
glBegin(GL_POINTS)	Titik	Titik individu

glBegin(GL_LINES)	Garis	Sepasang titik yang direpresentasikan sebagai sebuah garis
glBegin(GL_LINE_STRIP)	Poligaris	Beberapa garis yang terhubung
glBegin(GL_LINE_LOOP)	Poligaris tertutup (poligon)	Sama seperti diatas namun ditambahkan sebuah garis yang menghubungkan titik awal dan titik akhir
glBegin(GL_TRIANGLES)	Segitiga	Tiga buah titik yang diwujudkan dengan sebuah segitiga
glBegin(GL_TRIANGLE_STRIP)	Segitiga	Segitiga yang saling terkait
glBegin(GL_TRIANGLE_FAN)	Segitiga	Segitiga yang saling terkait dan membentuk kipas
glBegin(GL_QUADS)	Segiempat	Empat titik yang ditampilkan dalam bentuk sebuah poligon segi empat
glBegin(GL_QUAD_STRIP)	Segiempat	Empat titik yang ditampilkan dalam bentuk sebuah poligon segi empat yang saling terkait
glEnd()	Akhir perintah OpenGL	Menutup OpenGL

Dapat dilihat pada Tabel 2.1 diatas adalah perintah-perintah dasar yang digunakan pada pemrograman OpenGL, sedangkan pada Gambar 2.5 berikut adalah gambar objek-objek primitif yang digunakan dalam OpenGL.



Gambar 2.5 Objek-Objek Primitif OpenGL

2.4 Transformasi

Dalam buku OpenGL A Primer [ANG05] transformasi dinyatakan sebagai berikut: "*Transformation are the key to manipulating geometric objects, to animating scenes, and to obtaining the desired views*", yang kurang lebih dapat diartikan bahwa transformasi adalah kunci untuk memanipulasi objek-objek

geometris, membuat tampilan lebih hidup dan mendapatkan tampilan yang diinginkan. Pada dasarnya transformasi adalah memindahkan objek tanpa merusak bentuk dari objek tersebut.

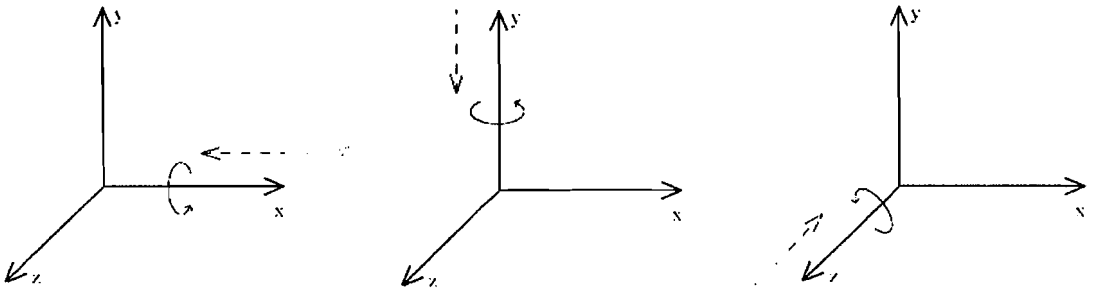
Sedangkan tujuan dari transformasi adalah [SUY03] :

1. Merubah atau menyesuaikan komposisi pandangan.
2. Memudahkan membuat objek yang simetris.
3. Melihat objek dari sudut pandang yang berbeda.
4. Memindahkan satu atau beberapa objek dari satu tempat ke tempat lain.

Transformasi meliputi translasi, penskalaan, rotasi (putaran) dan *shearing*. Dari beberapa proses transformasi tersebut disini penulis hanya akan membahas tentang proses rotasi (putaran), ini dikarenakan proses rotasi berkaitan langsung dengan proses perputaran sisi pada *rubik cube* (perangkat lunak yang akan dibuat).

2.5 Rotasi

Pada pemrograman grafika komputer baik 2 dimensi maupun 3 dimensi rotasi adalah salah satu jenis transformasi yang sering digunakan, rotasi menyebabkan suatu objek bergerak berputar pada titik pusat atau pada sumbu putar tertentu. Putaran dapat dilakukan pada sumbu-*x* (*roll*), sumbu-*y* (*pitch*), sumbu-*z* (*yaw*) maupun sumbu putar lainnya. Menurut aturan geometri perputaran sudut positif adalah berlawanan dengan arah jarum jam, seperti dapat dilihat pada Gambar 2.6 berikut.



Gambar 2.6 Sumbu Putar Sudut Positif (Berlawanan Arah Jarum Jam)

Rotasi terhadap sumbu-*x* (*roll*) didefinisikan dengan persamaan sebagai berikut:

$$\begin{aligned} x' &= x \\ y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \end{aligned} \quad \textit{ekuivalen ...} \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotasi terhadap sumbu-*y* (*pitch*) didefinisikan dengan persamaan sebagai berikut:

$$\begin{aligned} x' &= z\sin\theta + x\cos\theta \\ y' &= y \\ z' &= z\cos\theta - x\sin\theta \end{aligned} \quad \textit{ekuivalen ...} \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotasi terhadap sumbu-*z* (*yaw*) didefinisikan dengan persamaan sebagai berikut:

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned} \quad \textit{ekuivalen ...} \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Dimana x', y' dan z' adalah koordinat titik yang baru setelah diputar sebesar sudut θ, dan x, y dan z adalah koordinat titik awal.

2.6 Microsoft Visual C++

Microsoft Visual C++ merupakan salah satu perangkat lunak yang digunakan dalam pengembangan (perangkat pengembangan) aplikasi yang menggunakan bahasa C++. Visual C++ merupakan sebuah IDE (*Integrated Development Environment*) atau lingkungan pengembangan terintegrasi yang menyediakan beragam fasilitas bagi pengembang aplikasi, seperti membuat kode program, mengedit aplikasi, mengkompilasi maupun menguji aplikasi secara lebih cepat dan lebih mudah. Beberapa komponen utama dalam Visual C++ adalah sebagai berikut [KAD04]:

- **Editor**

  Editor menyediakan sarana bagi pemrogram untuk menliskan program. Yang menarik, editor yang disediakan mampu mengenali kata-kata tercadang C++ dan akan memberi warna tersendiri terhadap kata-kata seperti itu. Keuntungannya, program menjadi lebih mudah dibaca dan sekiranya anda melakukan kesalahan dalam menuliskan kata-kata itu maka akan lebih cepat terdeteksi.

- **Kompiler**

  Kompiler adalah perangkat lunak yang berfungsi untuk menerjemahkan kode sumber (*source code*) kedalam bentuk bahasa mesin. Tentu saja piranti ini dapat memberikan pesan-pesan kesalahan jika terjadi kesalahan kaidah penulisan program yang terdeteksi pada tahap proses kompilasi. Hasil kompilasi berupa kode objek (*object code*) yang disimpan dalam berkas berekstensi (***.obj**).

- **Linker**

  Linker adalah perangkat lunak yang berfungsi menggabungkan berbagai modul yang dihasilkan oleh kompiler dan modul kode dari berbagai pustaka C++, serta membentuk menjadi kode yang dapat dieksekusi. Sebagaimana kompiler, linker juga dapat mendeteksi kesalahan. Kesalahan yang terjadi sewaktu proses *linking* bisa disebabkan karena ada bagian pustaka atau bagian program yang tidak ditemukan.

- **Pustaka**

  Visual C++ menyediakan berbagai pustaka (*library*) yang memudahkan pemrogram dalam melakukan berbagai operasi seperti melakukan berbagai operasi seperti menghitung akar kuadrat dan mengakses *database*. Pustaka-pustaka yang tersedia antara lain berupa:

  - *Standard C++ library* (berisi semua rutin yang tersedi pada kebanyakan kompiler C++)
  - *Microsoft Foundation Classes and Templates* (MFC&T), yang berkaitan dengan pemrograman Windows.

- **AppWizard**

  Perangkat ini bermanfaat untuk membangkitkan suatu kerangka dasar aplikasi Windows yang sangat memudahkan pemrogram untuk membuat aplikasi Windows.

- **ClassWizard**

  Perangkat ini bermanfaat untuk mengedit kelas-kelas yang dibangkitkan oleh AppWizard.