OpenGL

O que é OpenGL

OpenGL é uma interface de software para dispositivos de hardware. Esta interface consiste em cerca de 150 comandos distintos usados para especificar os objetos e operações necessárias para produzir aplicativos tridimensionais interativos. OpenGL foi desenvolvido com uma aerodinâmica, independente de interface de hardware para ser implementado em múltiplas plataformas de hardware.

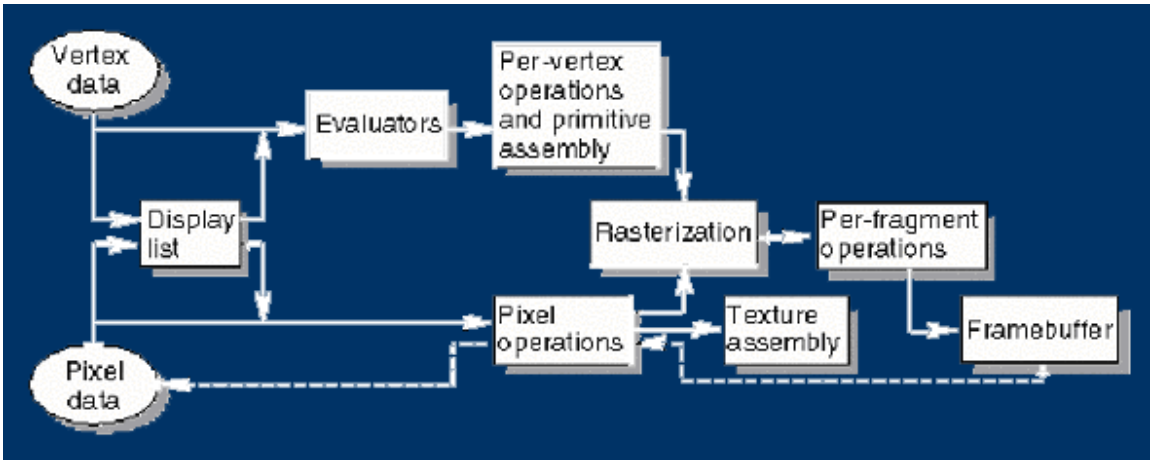
Diante das funcionalidades providas pelo OpenGL, tal biblioteca tem se tornado um padrão amplamente adotado na indústria de desenvolvimento de aplicações. Este fato tem sido encorajado também pela facilidade de aprendizado, pela estabilidade das rotinas, pela boa documentação disponível e pelos resultados visuais consistentes para qualquer sistema de exibição concordante com este padrão.

As especificações do OpenGL não descrevem as interações entre OpenGL e o sistema de janelas utilizado (Windows, X Window etc). Assim, tarefas comuns em uma aplicação, tais como criar janelas gráficas, gerenciar eventos provenientes de mouse e teclado, e apresentação de menus ficam a cargo de bibliotecas próprias de cada sistema operacional. Uma biblioteca bastante utilizada é a (OpenGL ToolKit) para gerenciamento de janelas.

Desde sua introdução em 1992, OpenGL transformou-se num padrão extensamente utilizado pelas indústrias. OpenGL promove a inovação e acelera o desenvolvimento de aplicações incorporando um grande conjunto de funções de render, de texturas, de efeitos especiais, e de outras poderosas funções de visualização.

O “Pipeline” do OpenGL

A maior parte das implementações do OpenGL tem uma ordem de operações a serem executadas. Uma serie de estágios de processos chamam o “pipeline” de renderização do OpenGL.
O diagrama seguinte mostra como o OpenGL, obtém e processa os dados.



Listas de Exposição

Todos os dados, se descrevem uma geometria ou pixels, podem ser conservados em uma lista da exposição para uso corrente ou serem usados mais tarde. (A alternativa além de manter dados em uma lista de exposição é processar os dados imediatamente – também conhecido como modo

imediato.) Quando uma lista da exposição é executada, os dados retidos são enviados da lista apenas como se fossem enviados pela aplicação no modo imediato.

Avaliadores

Todas as primitivas geométricas são eventualmente descritas por vértices. As curvas e as superfícies paramétricas podem inicialmente ser descritas pelos pontos de controle e pelas funções polinomiais chamadas funções base. Os avaliadores fornecem um método para derivar os vértices usados representar a superfície dos pontos de controle. O método é o mapeamento polinomial, que pode produzir a normal de superfície, as coordenadas da textura, as cores, e valores de coordenadas espaciais dos pontos de controle.

Operações por vértices

Para dados dos vértices, está seguida do "o estágio das operações por vértices", que converte os vértices em primitivas. Alguns dados do vértice (por exemplo, coordenadas espaciais) são transformados em matrizes 4 x 4 de pontos flutuantes. As coordenadas espaciais são projetadas de uma posição no mundo 3D a uma posição na tela. Se as características avançadas estiverem habilitadas permitidas, este estágio é mesmo mais ocupado. Se texturização for usada, as coordenadas da textura podem ser geradas e transformado aqui. Se o luzes forem habilitadas, os cálculos da luz serão executados usando os vértices transformados, a normal da superfície, a posição da fonte de luz, as propriedades de materiais, e a outras informações de luzes para produzir um valor da cor.

Montagem de Primitivas

Clipping, uma parte principal da montagem de primitivas, é o eliminação de partes da geometria que caem fora de uma parte do espaço, definido por um plano. O clipping do ponto simplesmente passa ou rejeita vértices; o clipping da linha ou do polígono pode adicionar vértices adicionais dependendo de como a linha ou o polígono são interligados. Em alguns casos, isto é seguido pela divisão da perspectiva, o qual faz com que os objetos geométricos distantes pareçam mais perto. Então as operações do viewport e da profundidade (coordenada de z) são aplicadas.

Operações de Pixels

Enquanto os dados geométricos pegam um caminho através do pipeline de renderização do OpenGL., dados de pixels tomam uma rota diferente. Os dados de Pixels em uma matriz na memória do sistema são empacotados e em seguida escalados, inclinados e processados por um mapa de pixels. Os resultados são escritos na memória da textura ou emitidos à uma etapa de rasterização. Se os dados do pixel forem lidos do framebuffer, operações de transferência de pixels (escala, polarização, mapeamento, “clamping”) são executadas. Então estes resultados são empacotados em um formato apropriado e retornados a uma matriz de memória do sistema

Montagem de Texturas

Uma aplicação OpenGL pode aplicar imagens de texturas em objetos geométricos, para tornar estes mais realísticos. Se diversas imagens de textura forem usadas, as mesmas deveram ser colocadas em objetos de modo que se possa facilmente comutar entre elas. Algumas exceções do OpenGL podem ter recursos especiais para acelerar o desempenho das texturas. Se existir memória especial disponível , os objetos de textura podem ter prioridade de controle do recurso limitado de memória.

Rasterização

Rasterização é a conversão de dados geométricos e do pixel em fragmentos. Cada quadrado do fragmento corresponde a um pixel no framebuffer. Os “stipples” da linha e do polígono, largura da linha, tamanho do ponto, modelo de sombra, e os cálculos da cobertura para suportar o antialiasing são feitos considerando a conexão dos vértices em linhas ou os pixels interior. Os valores da cor e da profundidade são atribuídos para cada quadrado do fragmento.

Operações fragmentadas

Antes que os valores estejam armazenados realmente no framebuffer, uma série de operações, que podem se alterar ou mesmo jogar para fora dos fragmentos, são executadas. Todas estas operações podem ser habilitadas ou desabilitadas. A primeira operação que pode ser encontrada é o “texturing”, onde um texel (elemento da textura) é gerado da memória da textura para cada fragmento e aplicado ao fragmento. Então os cálculos do “fog” podem ser aplicados, seguido pelo teste “scissor”, pelo teste do alfa, pelo teste do estêncil, e pelo teste do buffer de profundidade (o buffer de profundidade é para a remoção de faces ocultas da superfície). Então, “blending”, operação lógica de “dithering”, e mascaramento por um bitmask podem ser executadas.

Funções gráficas do OPENGL

Buffer de acumulação : Trata-se de um buffer no qual múltiplos frames renderizados, podem ser compostos para produzir uma única imagem. Usado para efeitos tais como a profundidade de campo, “blur” de movimento, e de anti-aliasing da cena.

Alfa Blending : Provê mecanismos para criar objetos transparentes. Usando a informação alfa, um objeto pode ser definido como algo totalmente transparente até algo totalmente opaco.

Anti-aliasing : Um método de renderização utilizado para suavizar linhas e curvas. Esta técnica calcula a média da cor dos pixels junto à linha. Tem o efeito visual de suavizar a transição dos pixels na linha e daqueles junto à linha, assim fornecendo uma aparência mais suave.

Modo “Color-Index”: Buffer de Cores que armazena índices de cores das componentes vermelhas, verdes, azuis, e alfa das cores (RGBA).

Display Lists : Uma lista nomeada de comandos de OpenGL. Os índices de um Display list podem ser pré-processados e podem conseqüentemente executar mais eficientemente do que o mesmo conjunto de comandos do OpenGL executados no modo imediato.

Double buffering : Usado para fornecer uma animação suave dos objetos. Cada cena sucessiva de um objeto em movimento pode ser construída em “background” ou no buffer “invisível” e então apresentado. Isto permite que somente as imagens completas sejam sempre apresentadas na tela.

FeedBack : Um modo onde OpenGL retornará a informação geométrica processada (cores, posições do pixel, e assim por diante) à aplicação.

Gouraud Shading : Interpolação suave das cores através de um segmento de polígono ou de linha. As cores são atribuídas em vértices e linearmente interpoladas através da primitiva para produzir uma variação relativamente suave na cor.

Modo Imediato : A execução de comandos OpenGL quando eles são chamados, possui resultado melhor do que os “Display Lists”.

Iluminação e sombreamento de materiais : A habilidade de computar exatamente a cor de algum ponto dado as propriedades materiais para a superfície.

Operações de pixel : Armazena, transforma, traça e processa aumento e redução de imagens.

Executores polinomiais : Para suportar as NURBS (non-uniform rational B-splines).

Primitivas : Um ponto, uma linha, um polígono, um bitmap, ou uma imagem. Primitivas da rasterização : bitmaps e retângulos de pixels

Modo RGBA : Buffers de cores armazenam componentes vermelhos, verdes, azuis, e alfa da cor.

Seleção e colheita : Trata-se de um modo no qual o OpenGL determina se certa primitiva identificada do gráficos é renderizada em uma região no buffer de frame.

Planos do estêncil : Um buffer que é usado para mascarar pixels individuais no buffer de frame de cores.

Mapeamento de Texturas : O processo de aplicar uma imagem a uma primitiva gráfica. Esta técnica é usada gerar o realismo nas imagens.

Transformações : A habilidade de mudar a rotação, o tamanho, e a perspectiva de um objeto no espaço 3D coordenado.

Z-buffering : O Z-buffer é usado manter-se a par se uma porção de um objeto é mais próxima visor do que outra. É importante na remoção de superfície escondida

Instalação do OPENGL

As bibliotecas do OpenGL são distribuídas como parte dos sistemas operacionais da Microsoft, porém as mesmas podem ser baixadas no site oficial do OpenGL : http://www.opengl.org. Estas bibliotecas também estão disponíveis em outros sistemas operacionais por padrão, tais como, MacOS e Unix Solaris, além de estarem disponíveis no Linux.

Instalação do GLUT

O GLUT é um conjunto de ferramentas para escrita de programas OpenGL, independente do sistema de janelas. Ele implementa um sistema de janelas simples através de sua API, para os programas OpenGL. GLUT provê uma API portável, o que permite que programas trabalhem tanto em ambientes baseados em WIN32 quanto X11.

O GLUT suporta :

- Janelas múltiplas para renderização OpenGl
- Resposta a eventos baseados em Callback de funções
- Uma rotina “idle” e “timers”
- Criação de menus pop-up simples
- Suporte pra bitmaps e fontes
- Uma miscelânea de funções para gerenciamento de janelas.

Instalando o GLUT no Borland C++ Builder 5 no ambiente Windows

Os arquivos de instalação do GLUT poderão ser obtidos em
http://www.opengl.org/developers/documentation/glut/index.html

1 – Baixe o arquivo do link acima e descompacte-os em uma pasta temporária
2 – Copie os arquivos glut.dll e glut32.dll a pasta c:\windows\system32 ou uma pasta que esteja no caminho do sistema operacional, no PATH.

3 – Copie os arquivos glut*.lib para <diretório de Borland C Builder>\lib
4 – Copie o arquivo glut.h para < diretório de Borland C Builder >\include\gl
5 – Como os arquivos originais são desenvolvidos para Microsoft Visual C++ você deverá executar os seguintes comandos dentro da pasta \lib, localizada dentro do diretório de instalação do Borland C++ :
- implib glut.lib c:\windows\system32
- implib glut32.lib c:\windows\system\glut32.dll

Obs.: Normalmente poderão ocorrer problemas no momento da compilação / linkedição de um programa OpenGL, com o Borland C++, porém isto se deve ao fato da instalação do mesmo ter sido feita no caminho \Arquivos de programas\Borland\CBuilder5, existe um pequeno bug. Para resolução deste problema instale o CBuilder e um diretório tipo C:\BC5 que funcionará corretamente.

Instalando o GLUT no MS Visual C++

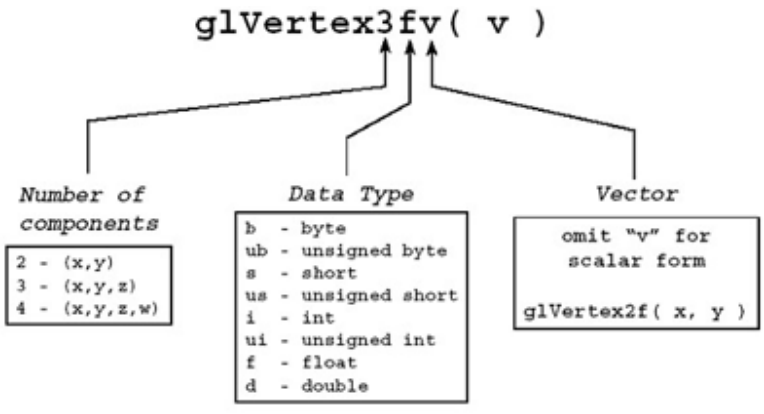
Como a esta API foi desenvolvida no próprio MS Visual C++, não é necessária nenhuma conversão da mesma para funcionamento. Porém é necessário a instalação da mesma. Assim ela deverá ser baixada do site http://www.opengl.org/developers/documentation/glut/index .

1 – Descompacte os arquivos em um diretório temporário.
2 – Copie os arquivos glut.dll e glut32.dll a pasta c:\windows\system32 ou uma pasta que esteja no caminho do sistema operacional, no PATH.
3 – Copie os arquivos glut*.lib para <diretório de instalação do visual c>\lib
4 – Copie o arquivo glut.h para <diretório de instalação do visual c>\include\gl
5 – Quando for compilar um programa deverão serem incluídas as seguintes bibliotecas na opção de linkedição do programa : opengl32.lib, glu32.lib, glut32.lib.

Sintaxe de Comandos do OpenGL

Os comandos da API OpenGL obedecem a um padrão bem definido para especificação dos nomes de funções e constantes. Todos os comandos utilizam-se do prefixo gl em letras minúsculas. Similarmente, OpenGL define constantes com as iniciais GL_, em letras maiúsculas, e usa um "underscore" para separar as palavras (Ex. GL_COLOR_BUFFER_BIT).

Em algumas funções algumas seqüências de caracteres extras aparecem,como sufixo, no nome destas funções (Ex. glColor3f e glVertex3f). É verdade que a parte "Color" do comando glColor3f(), não é suficiente para definir o comando como um conjunto de cores correntes. Particularmente, o a parte "3" do sufixo indica que três argumentos são necessários; outra versão de "Color" necessita de 4 argumentos. O "f" do sufixo indica que os argumentos são do tipo números de ponto flutuante. Através destas definições para diferentes tipos de formatos e permitido ao OpenGL, aceitar dados no seu próprio formato definido.



Alguns comandos OpenGL aceitam até 8 tipos diferentes de argumentos. As letras usadas como sufixo, como dito anteriormente, irão determinar o tipo de dados correspondente à padronização ISO para implementação em linguagem C.

Sufixos de comandos e tipos de dados para argumentos

Sufixo	Tipo de Dados	Tipo correspondente na linguagem C	Definição de tipo para OpenGL
b	Inteiro de 8 bits	Signed char	GLbyte
s	Inteiro de 16 bits	Short	GLshort
i	Inteiro de 32 bits	Int ou long	GLint,GLsizei
f	Ponto flutuante de 32 bits	Float	GLfloat,GLclampf
d	Ponto flutuante de 64 bits	double	GLdouble,GLclampd
ub	Inteiro não sinalizado de 8 bits	Unsigned char	GLubyte,GLboolean
us	Inteiro não sinalizado de 16 bits	Unsigned short	GLushort
ui	Inteiro não sinalizado de 32 bits	Unsigned int ou unsigned long	GLuint,GLenum,GLbitfield

Rotinas de Callback

As funções de callback são aquelas executadas quando qualquer evento ocorre no sistema, eventos tais como : redimensionamento de janela o desenho da mesma, entradas de usuários através de teclado, mouse, ou outro dispositivo de entrada, e ocorrência de animações. Assim o desenvolvedor pode associar uma ação especifica à ocorrência de determinado evento.
GLUT oferece suporte a muitos diferentes tipos de ações de callback incluído :
o glutDisplayFunc() – chamada quando um pixel na janela necessita ser atualizado.

o glutReshapeFunc() – chamado quando a janela é redimensionada

o glutKeyboardFunc() – chamada quando uma tecla do teclado é pressionada

o glutMouseFunc() – Chamada quando o usuário pressiona um botão do mouse

o glutMotionFunc() - chamada quando o usuário movimenta o mouse enquanto mantém um botão do mesmo pressionado

o glutPassiveMouseFunc() – chamado quando o mouse é movimentado, independente do estado dos botões.

o glutIdleFunc() – uma função de callback chamada quando nada está acontecendo. Muito útil para animações.

Estrutura Básica de Programas OpenGL

Um programa OpenGL deve conter, um mínimo de requisitos para sua perfeita execução.
Normalmente alguns passos devem ser seguidos para criação destes programas. Estes passos são :

• declaração dos arquivos de header para o OpenGL
• Configurar e abrir a janela.
• inicializar os estados no OpenGL
• Registrar as funções de “callback”
• renderização;
• redimensionamento
• Entradas : teclado, mouse, etc.
• Entrar no loop de processamento de eventos

O programa exemplo abaixo irá ilustrar estes passos

```
#include <GL/gl.h>
#include <GL/glut.h>
void main( int argc, char** argv )
{
     int mode = GLUT_DOUBLE | GLUT_RGB;
     glutInitDisplayMode( mode );
     glutInitWindowSize(400,350);
     glutInitWindowPosition(10,10);
     glutCreateWindow( argv[0] );
     init();
     glutDisplayFunc( display );
     glutReshapeFunc( resize );
     glutKeyboardFunc( key );
     glutIdleFunc( idle );
     glutMainLoop();

}
```

• Os arquivos de header, contém as rotinas e declarações necessárias para utilização do OpenGL/GLUT com a linguagem c/c++.

• As funções glutInitDisplayMode() e glutCreateWindow() compõem o passo de configuração da janela. O modo da janela que é argumento para a função glutInitDisplayMode(),indicam a criação de uma janela double-buffered (GLUT_DOUBLE) com o modo de cores RGBA (GLUT_RGB). O primeiro significa que os comandos de desenho são executados para criar uma cena fora da tela para depois rapidamente colocá-la na view (ou janela de visualização). Este método é geralmente utilizado para produzir efeitos de animação. O modo de cores RGBA significa que as cores são especificadas através do fornecimento de intensidades dos componentes red, green e blue separadas. A função glutCreateWindow() cria a janela com base no parâmetros definidos nas funções glutInitWindowSize (Tamanho da janela em pixels) e glutInitWindowPosition (coordenadas para criação da janela). Esta janela conterá o nome especificado em seu parâmetro de entrada.

• Em seguida é chamada a rotina init(), a qual contem a primeira inicialização do programa. Neste momento serão inicializados quaisquer estados OpenGL, que serão executados na execução do programa.

• O próximo passo será o registro das funções de callback, que estarão sendo utilizadas na execução do programa.

• Finalmente o programa irá entrar em um processo de loop, o qual interpreta os eventos e chamadas das rotinas especificadas como callback.

Exemplo de um programa OpenGL

Este exemplo simples, apenas desenha um quadrado na tela.

```
/* Exemplo1.c - Marcionílio Barbosa Sobrinho
 * Programa simples que apresenta o desenho de um quadrado
 * Objetivo : Demonstrar funções de gerenciamento de
 *            janelas e funções de callback
 * Referência do Código: OpenGL Programming Guide - RedBook
*/

#include <windows.h>
#include <GL/gl.h>
#include <GL/glut.h;

void display(void)
{
   /* Limpa o Buffer de Pixels */
   glClear (GL_COLOR_BUFFER_BIT);

   // Define a cor padrão como branco
   glColor3f (1.0, 1.0, 1.0);

   /* desenha um simples retângulo com as coordenadas
   * (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0)
  */

   glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
   glEnd();

   /* Inicia o processo de desenho através dos
      dados bufferizados
   */
   glFlush ();
}

void init (void)
{
    /* Seleciona a cor de fundo para limpeza da tela */
    glClearColor (0.0, 0.0, 0.0, 0.0);

    /* inicializa os valores de visualização */
    glMatrixMode(GL_PROJECTION);

    /* Faz com que a matriz corrente seja inicializada
            com a matriz identidade (nenhuma transformação é acumulada)
    */
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}


/*
  Cria a janela
*/
int main(int argc, char** argv)
{

   /*
     Estabelece o modo de exibição a ser utilizado pela
          janela a ser criada neste caso utiliza-se de um buffer
          simples, ou seja, a apresentação será imediata à execução
     Define o modo de cores como RGBA
   */
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
```

```
   /*
     Determina o tamanho em pixels da
     janela a ser criada
   */
   glutInitWindowSize (250, 250);

   /*
     Estabelece a posição inicial para criação da
     janela
   */
   glutInitWindowPosition (100, 100);

   /*
     Cria uma janela com base nos parâmetros especificados
     nas funções  glutInitWindowSize e glutInitWindowPosition
     com o nome de título especificado em seu argumento
   */
   glutCreateWindow ("Exemplo 1");

   /*
      Especifica os parâmetros inicias para as variáveis
      de estado do OpenGL
   */
   init ();

   // Associa a função display como uma função de callback
   glutDisplayFunc(display);

   /*
     Inicia a execução do programa OpenGL.
     O programa irá executar num loop infinito devendo
     o desenvolvedor especificar as condições de saída do mesmo
     através de interrupções no próprio programa ou através
     de comandos de mouse ou teclado como funções de callback
   */
   glutMainLoop();
   return 0;
}
```

Mais informações: http://www.ingleza.com.br/opengl/1.html