UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

COMPARAÇÃO ENTRE AS BIBLIOTECAS GRÁFICAS DIRECT3D E OPENGL

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ALEXANDRE OTTO STRUBE

BLUMENAU, JUNHO/2001

2001/1-01

COMPARAÇÃO ENTRE AS BIBLIOTECAS GRÁFICAS DIRECT3D E OPENGL

ALEXANDRE OTTO STRUBE

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Paulo Cesar Rodacki Gomes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Paulo César Rodacki Gomes

_____

Dalton Solano dos Reis

_____

Roberto Heinzle

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

AGRADECIMENTOS

Gostaria de agradecer à todas as pessoas que foram importantes nesta importante fase de minha vida. Em primeiro lugar a meus pais, Ralf e Nilce, que tiveram grande paciência com um filho que trocou o dia pela noite para terminar este trabalho, à minha namorada Katia, que acompanhou e tem suportado um programador mal-humorado por todos estes anos, ao Juliano Carvalho Sansão, que me deu não só um maravilhoso livro de OpenGL como me deu boas dicas na implementação do protótipo. Gostaria de agradecer também aos meus amigos, cuja contribuição para este trabalho é envolta demais em mistério para ser elucidada: Chalinho, Mendes, Oitenta, Leandro, Gica, Kathrine, Nélvio, Cris e todos os outros que eu não citei, mas também não esqueci.

Gostaria de agradecer também ao Brian Cabral e ao Mark Segal, da Silicon Graphics, ao Patrick Brown da IBM, ao Mark Kilgard do GLUT, e Sharda Iyer da Vertex Computer Systems, por todas suas perguntas aparentemente insanas, porém cruciais com a parte de OpenGL, e por terem suportado minha avalanche de e-mails, sempre com uma resposta inesperada no meio da madrugada. Também à Beverly Boyd, da California Berkeley University, por dicas para me aprofundar no estranho mundo do C++.

Agradeço também o meu orientador para este trabalho, o professor Paulo César Rodacki Gomes, que sempre era interrompido em seu trabalho por um formando sempre com pressa e que nunca ficava por perto tempo suficiente para ele entender o que se passava nesta mente insana.

RESUMO

Este trabalho faz uma comparação entre as bibliotecas gráficas OpenGL e Direct3D utilizando-se para tal a norma NBR13596 (ABNT, 1996), relativa à qualidade de software. Submetendo ambas as bibliotecas ao mesmo processo de avaliação de qualidade, é possível compará-las. O ambiente de desenvolvimento é o Microsoft Visual C++, tendo como plataforma de desenvolvimento e execução o Windows98. Para visualização do protótipo é utilizado a placa aceleradora 3d da marca 3dfx, modelo *Voodoo 4*. Como exemplo de aplicação, foi desenvolvido um protótipo de uma cena tridimensional implementada em cada uma das bibliotecas. O material apresentado poderá dar um ponto de partida para a escolha de uma das bibliotecas avaliadas para desenvolvimento.

ABSTRACT

This is a comparison between the graphical libraries OpenGL and Direct3D using for that NBR13596's guides for software quality. Doing the same evaluation process with both libraries it is possible to compare them. The development environment is the Microsoft's Visual C++, using Windows 98 as host and target platform. For prototype visualization it's used the Voodoo 4 3d accelerator from 3dfx. As application example a tridimensional scene was built and rendered using each library. This paper can act as a start point for the process of choosing one of the evaluated libraries for development.

1 INTRODUÇÃO

O uso de bibliotecas gráficas para desenvolvimento de aplicativos tem-se intensificado à medida que a complexidade dos mesmos aumenta. Desde jogos cada vez mais detalhados até novas aplicações que fazem uso intensivo de gráficos a necessidade de conhecimento de pelo menos uma coleção de funções gráficas é essencial ao desenvolvedor avançado. Existem atualmente disponíveis ao desenvolvedor geral duas APIs (*Application Program Interface*, daqui em diante citado como bibliotecas), possuindo destinos finais semelhantes, porém com tecnologias, origens e procedimentos completamente diferentes: o DirectX, da Microsoft, e o OpenGL, uma iniciativa independente de empresas. Segundo Microsoft (2001), o DirectX é "um conjunto de baixo nível de bibliotecas para criação de jogos e outras aplicações multimídia de alta performance. Inclui suporte para gráficos bidimensionais e tridimensionais, efeitos sonoros e música, dispositivos de entrada e suporte para aplicações em rede". Já, segundo OpenGL (2001), o OpenGL é mais específico: "É um ambiente para desenvolver aplicações 2D e 3D portáveis e interativas".

O desenvolvedor em geral, à medida que necessita especificar uma aplicação que faça uso intensivo de tecnologia gráfica, acaba optando por uma ou outra tecnologia pelas mais aleatórias razões, sem dispor de um método científico que lhe possibilite a escolha mais adequada às suas necessidades. É exatamente a isto que se propõe este trabalho: comparar, dentro de determinados quesitos, qual biblioteca adequa-se mais a qual situação.

A norma brasileira NBR13596 (ABNT, 1996) analisa aspectos de qualidade de um software, podendo desta forma promover um comparativo entre as duas bibliotecas de funções, através da análise simultânea em ambas as bibliotecas, segundo os aspectos determinados pela norma, os quais são:

a) **funcionalidade**, que é a capacidade do *software* satisfazer quaisquer funções adequando estados e necessidades implicadas quando usados sob condições específicas;
b) **confiabilidade**, ou seja, a capacidade do *software* manter seu desempenho quando usado sob condições específicas;
c) **usabilidade**, que vem a ser a capacidade do *software* em ser fácil de usar e satisfazer o usuário, quando usado sob condições específicas;

d) **eficiência**, os recursos usados por um *software* contido no sistema para alcançar o desempenho requerido sob condições específicas;
e) **manutenabilidade**, os recursos necessários para fazer modificações específicas no *software*;
f) **portabilidade**, capacidade de transferir o *software* para outros ambientes.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal do trabalho é o estudo em profundidade das duas bibliotecas gráficas escolhidas, exercendo uma comparação entre elas baseada na norma ABNT (1996).

O objetivo específico do trabalho é a implementação de um protótipo onde elementos a serem comparados necessitem de uma demonstração visual dos resultados.

1.2 ESTRUTURA DO TRABALHO

A seguir serão descritos brevemente os capítulos do trabalho.

Este capítulo de introdução apresenta uma visão geral deste trabalho e seus objetivos.

O segundo capítulo trata sobre a fundamentação teórica para o desenvolvimento do trabalho. Conceitos, técnicas e ferramentas relevantes para o trabalho são apresentadas. Uma breve contextualização da computação gráfica tridimensional necessária para a compreensão das bibliotecas gráficas será apresentada. São também apresentadas brevemente as *APIs* OpenGL e Direct3D, seu relacionamento com os sistemas operacionais e com o *hardware.*

O desenvolvimento do trabalho é relatado no terceiro capítulo, onde requisitos, características, especificação, implementação, técnicas, ferramentas, resultados e uma discussão sobre o trabalho é feita. Especificamente, a biblioteca Direct3D terá um tópico específico, bem como a biblioteca OpenGL. Cada um destes tópicos contém uma descrição das capacidades, limitações e funcionalidade de cada uma destas bibliotecas. Será dado à norma NBR13596 um tópico, onde cada item de avaliação de qualidade de software será aplicado a cada uma das bibliotecas, procedendo assim com a comparação delas. Será abordado o projeto de uma cena tridimensional implementada em ambas as bibliotecas, a implementação do software construído a partir deste projeto.

O quarto e último capítulo apresenta os resultados obtidos e as dificuldades encontradas durante a execução do trabalho. São também apresentadas sugestões para próximos trabalhos e considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 A TERCEIRA DIMENSÃO

O termo tridimensional, ou 3D, significa que o objeto a ser descrito ou mostrado tem três dimensões de medida: largura, altura e profundidade. Um exemplo de um objeto bidimensional é uma folha de papel com um desenho ou algo escrito nela. Um objeto tridimensional é o lápis que está em cima da folha. Dependendo de onde se olha, o lápis é redondo (tem largura e altura) e é comprido (profundidade). Qual lado é altura ou largura é um fato relativo à sua perspectiva, mas o fato de existirem as três dimensões não se altera.

Por muito tempo os artistas souberam como fazer uma pintura parecer ter profundidade real. Uma pintura é inerentemente um objeto bidimensional, ela não é nada mais que um plano com tintas sobre si. De forma similar, gráficos tridimensionais em computadores são na verdade imagens bidimensionais numa tela plana que nos fornecem uma ilusão de profundidade, ou a terceira dimensão.

A forma mais simples para o cérebro perceber a terceira dimensão dá-se quando o objeto é visto com ambos os olhos ou são fornecidas a cada olho imagens diferentes do objeto, oriundas de ângulos diferentes, devido à distância de um olho a outro. O cérebro então combina essas imagens ligeiramente diferentes do objeto para produzir uma única cena tridimensional na mente. Os filmes tridimensionais onde se usam óculos com filtros de cores diferentes para cada olho se utilizam desta técnica. Essas imagens geralmente são exageradas para maior efeito dramático.

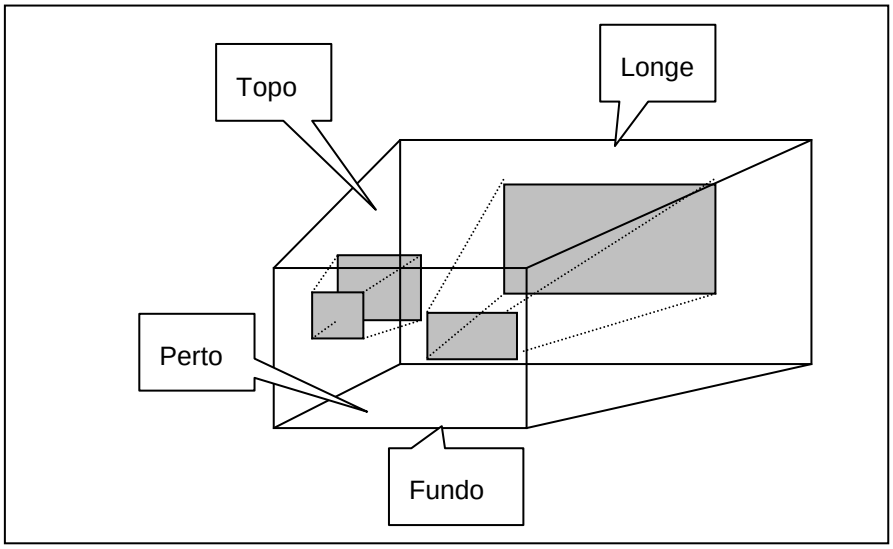
Uma tela de computador é uma imagem plana numa superfície plana, e não duas imagens de perspectivas diferentes sendo percebidas por cada olho. De fato, a maior parte do que é considerado gráfico 3D é na verdade apenas uma aproximação do 3D real. Diversas formas de se representar a impressão de profundidade são utilizadas em gráficos de computador, tais como (Woo, 1998) (Wright, 2000):

a) **Projeção** – são utilizados dois tipos básicos de projeção em gráficos tridimensionais: a projeção paralela e a projeção em perspectiva. Na projeção paralela, todos os objetos de mesma dimensão possuem tamanhos iguais na tela. Este tipo de projeção é muito utilizada, por exemplo, em CAD ou arquitetura. Já na

projeção em perspectiva os objetos mais distantes aparecem menores em relação a objetos mais próximos. O volume de visualização demonstrado na figura 1 assemelha-se a uma pirâmide com o topo cortado. Essa parte cortada, ou menor é chamada de *frustum* e é por ele que as imagens são visualizadas. Objetos mais próximos do *frustum* são mostrados com um tamanho próximo de seus originais, enquanto objetos próximos da parte de trás do volume de visualização encolhem conforme são projetados no *frustum*. Este tipo de projeção dá grande realismo aos gráficos 3D.

b) **Cores, luzes e sombras** – são utilizadas para descrever objetos sólidos. Aplicando-se uma cor a um objeto, ele destaca-se do fundo. Aplicando-se diferentes cores a um objeto, ele ganha um formato. Aplicando-se tons de cores com um sombreamento apropriado a cada lado de um objeto, não apenas dá-se a impressão de um sólido mas também de estar iluminado por uma luz em um ângulo. Projetando-se uma sombra deste objeto em outro tem-se um efeito ainda mais real.
c) **Mapeamento de texturas** – aplicar imagens bidimensionais, como uma fotografia de uma superfície real a sólidos tridimensionais, acrescenta um realismo muito grande a um custo computacional proporcionalmente pequeno em comparação com a construção de geometria tridimensional para alcançar o mesmo efeito. Ao invés de materiais planos, pode-se ter qualquer coisa do mundo real simulada de forma bastante simples com essa técnica.
d) **Neblina** – é o efeito atmosférico que dá a impressão de desaparecimento de objetos numa cena, normalmente em relação à distância do objeto ao observador. Objetos muito distantes podem até estar totalmente obscurecidos.
e) ***Blending* e transparência** – *blending* é a combinação de cores ou objetos numa cena. Variando-se como cada objeto é misturado com a cena, estes podem possuir a aparência de transparência. Este efeito também pode ser usado para dar a ilusão de reflexão.
f) ***Anti-aliasing* –** *aliasing* é o efeito visível numa tela de computador devido ao fato de uma imagem consistir de pontos (ou *pixels*) discretos. Não existe meio *pixel*. É um efeito onde uma linha não paralela ou perpendicular à orientação da tela tem a aparência de uma “escada”. Misturando-se sutilmente as cores de um objeto com as do fundo é possível contornar este problema. Esta técnica é o *anti-aliasing*.

Figura 1 – o volume de visualização, ou *frustum*, para uma projeção em perspectiva.



2.2 O TERMO RENDER

Segundo Wright (2000), o termo *render* "é o ato de tomar uma descrição geométrica de um objeto tridimensional e transformá-la em uma imagem daquele objeto na tela". Como não existe uma tradução para este significado do termo, ele é utilizado como no original. A expressão "renderização" é amplamente utilizada nas publicações sobre computação gráfica em língua portuguesa, e portanto será utilizada neste trabalho.

2.3 SISTEMAS DE COORDENADAS

2.3.1 COORDENADAS CARTESIANAS 2D

O sistema mais comum de coordenadas num plano é o sistema Cartesiano. Nele, as posições são descritas em termos de uma coordenada x e uma coordenada y, onde a coordenada x descreve a posição na direção horizontal e a coordenada y a descreve na direção vertical.

A origem do sistema Cartesiano está em x=0 e y=0. Coordenadas Cartesianas são escritas na forma de pares ordenados, entre parênteses, com a coordenada x primeiro, seguida pela coordenada y, separadas por uma vírgula. Por exemplo, a origem é representada na forma (0,0). O grande mérito do sistema cartesiano é representar posições através de eixos

perpendiculares, podendo estender estes eixos até posições positiva ou negativamente infinitas.

2.3.2 ÁREA DE CORTE

Num programa de computador o espaço para desenho é limitado pelo tamanho da sua janela, ou mesmo da resolução da tela, em caso de um programa que a ocupe totalmente. É necessário então definir-se como transformar pares ordenados de coordenadas específicos desta aplicação em coordenadas de tela. Isto é feito especificando-se a região do espaço Cartesiano que ocupa a janela, a chamada área de corte. No espaço bidimensional, a área de corte equivale aos x e y mínimos e máximos que estão dentro da janela. Outra forma de descrever a área de corte é especificar o local da origem do sistema de coordenadas em relação à tela. Como será visto no desenvolvimento do trabalho, especificar a área de corte é um passo necessário em ambas as bibliotecas gráficas a serem estudadas.

2.3.3 VIEWPORTS

Dificilmente a área de corte de uma cena é a mesma área da tela. O sistema de coordenadas então deve ser mapeado de coordenadas cartesianas lógicas para coordenadas físicas da tela, medidas em *pixels*. Este mapeamento é conhecido como *viewport*. Um *viewport* é a região da janela da aplicação utilizada para desenhar a área de corte.

2.3.4 VÉRTICE

Em 2D ou 3D, o objeto a ser desenhado é composto de várias formas mais simples denominadas primitivas. Primitivas são entidades unidimensionais ou bidimensionais, ou seja, pontos, linhas e polígonos (uma figura plana com vários lados) montadas num plano (2D) ou num espaço (3D) para criar um objeto. Cada canto de uma primitiva é um vértice. Este não é nada mais que uma coordenada num espaço 2D ou 3D.

2.3.5 COORDENADAS CARTESIANAS 3D

O sistema de coordenadas 3D não é nada mais que um sistema 2D com mais uma dimensão, a da profundidade, ou mais um eixo, perpendicular aos eixos já existentes. Este é descrito como eixo z. Observando-se um sistema de coordenadas a partir da origem, o eixo z é

aquele que vem para perto ou vai para longe do observador. A escolha da direção para onde um eixo é positivo é arbitrária. Por exemplo, na geometria descreve-se o eixo y como sendo positivo para cima, o eixo x para a direita e o eixo z aproximando-se do observador. O Microsoft Windows define, para o sistema de janelas (2D), como positivo o eixo y indo para baixo (Microsoft, 1998). Em aplicações 3D isso fica a cargo do programador e da biblioteca gráfica a ser utilizada. Na biblioteca OpenGL, tipicamente utiliza-se o eixo z positivo como aproximando-se do observador. Já na biblioteca Direct3D, utiliza-se o eixo z positivo afastando-se do mesmo.

2.4 RETAINED MODE E IMMEDIATE MODE

Existem duas abordagens para a programação de bibliotecas gráficas tridimensionais. A primeira é conhecida como *Retained Mode*. Nela, o programador fornece à biblioteca a descrição de objetos e a cena. O pacote gráfico então cria a imagem na tela. A única manipulação adicional é o envio de comandos para alterar o local e a orientação visual do usuário (também chamado *camera*) ou outros objetos na cena.

Este tipo de abordagem é típica de programas geradores de imagem, como por exemplo o 3D Studio. Em termos de programação, a estrutura construída é chamada de gráfico de cena. Segundo Wright (2000), o gráfico de cena é uma estrutura de dados (também conhecida como DAG – *directed acyclic graph*) contendo todos os objetos na cena e as relações entre estes. A Fahrenheit Scene Graph é um exemplo de biblioteca do tipo *retained mode* que estava sendo desenvolvida pela Microsoft e Silicon Graphics, e utilizaria OpenGL (uma biblioteca de nível mais baixo) para renderização. Outra biblioteca deste tipo é o Direct3D *retained mode*, objeto de pesquisa deste trabalho. Existe uma implementação de biblioteca *retained mode* para OpenGL chamada GLUT (sigla para OpenGL Utility Toolkit), disponível gratuitamente em OpenGL (2001). Algumas funções GLUT serão usadas no protótipo para maior semelhança da implementação em OpenGL com a implementação utilizando Direct3D.

A segunda abordagem para renderização 3D é chamada *immediate mode*. A maioria das bibliotecas *retained mode* utilizam-se internamente de uma biblioteca *immediate mode*

para executar a renderização. Neste modo, os modelos e ambiente não são descritos numa forma de alto nível. Ao invés disso, são dados comandos diretamente ao dispositivo que possuem um efeito imediato no seu estado e no estado dos comandos subsequentes.

Desenvolvedores que já possuem suas próprias rotinas de gerenciamento de cena, transformações ou iluminação podem facilmente adaptá-las a bibliotecas do tipo *immediate mode*, podendo assim utilizar-se de características exclusivas destas bibliotecas, como a abstração de hardware ou a multiplataforma.

2.5 RENDERING PIPELINE

Segundo Henkels (1997), “não existe uma tradução exata do termo *rendering pipeline* para o português. Uma das possíveis traduções seria fila de renderização. O *rendering pipeline* é o processo que realiza todas as funções da computação gráfica em 3 dimensões”.

Já segundo Wright (2000), a palavra *pipeline* é utilizada para descrever um processo que pode ter dois ou mais passos. Os *rendering pipeline* das bibliotecas OpenGL e Direct3D são bastante similares.

2.5.1 COMMAND BUFFER OU EXECUTE BUFFER

As diferenças são que o primeiro estágio do *pipeline* em OpenGL é chamado “OpenGL command buffer”, enquanto em Direct3D chama-se este estágio de “*execute buffer*”. Em ambos o estágio tem o mesmo significado: serve para armazenar dados da cena, como vértices e comandos, por exemplo.

2.5.2 MÓDULO DE TRANSFORMAÇÃO E ILUMINAÇÃO

O segundo estágio do *rendering pipeline* em Direct3D é chamado de módulo de transformação, e o terceiro é chamado de módulo de iluminação. Em OpenGL ambos são unidos num só estágio, o que é apenas uma diferença de demonstração, pois ambas as bibliotecas devem cumprir as mesmas exigências nestes estágios.

Estes dois módulos podem ser considerados como um estágio matematicamente intensivo onde pontos usados para descrever a geometria de um objeto são recalculados para

determinada locação e orientação do objeto. Cálculos de iluminação são também executados para indicar quão brilhantes as cores devem se apresentar em cada vértice.

É neste passo (mais especificamente, no módulo de transformação) que são determinados quais vértices da cena são visíveis. Em Direct3D, utiliza-se uma forma de identificação de superfícies ocultas camada *z-buffer*, enquanto OpenGL utiliza-se, além da técnica *z-buffer*, de outra, chamada *backface culling*.

2.5.2.1 Z-BUFFER

A técnica de *z-buffer* utiliza um espaço de memória (chamado de *buffer*) para manter informações sobre quais superfícies estão mais próximas do observador. Os valores z armazenados neste *buffer* não correspondem necessariamente ao eixo z do sistema de coordenadas utilizado; mas sim, representam a distância das superfícies para o observador. Quando a descrição da cena chega do módulo de transformação, os valores de z são comparados aos valores já armazenados no *z-buffer*. Quando o valor neste *buffer* indica que a superfície está mais próxima ao observador que aquela já presente no *buffer*, a nova superfície é então adicionada no *z-buffer* e será a superfície a ser desenhada. Caso contrário, se ela for mais distante, já foi removida do *buffer* e não será desenhada.

Segundo Trujillo (1997), "a técnica de *z-buffer* é uma das mais simples e rápidas para a remoção de faces ocultas. A precisão é em nível de pixel e trata cenas complexas de maneira eficiente. A principal desvantagem de *z-buffer* é a quantidade de memória necessária pelo *z-buffer*. Estes *buffers* devem ter um tamanho mínimo igual ao da imagem e podem ter cor de 32 bits. Um programa que executa em modo 800x600 com um *z-buffer* de 16 bits, por exemplo, requer quase um megabyte de memória apenas para o *z-buffer*".

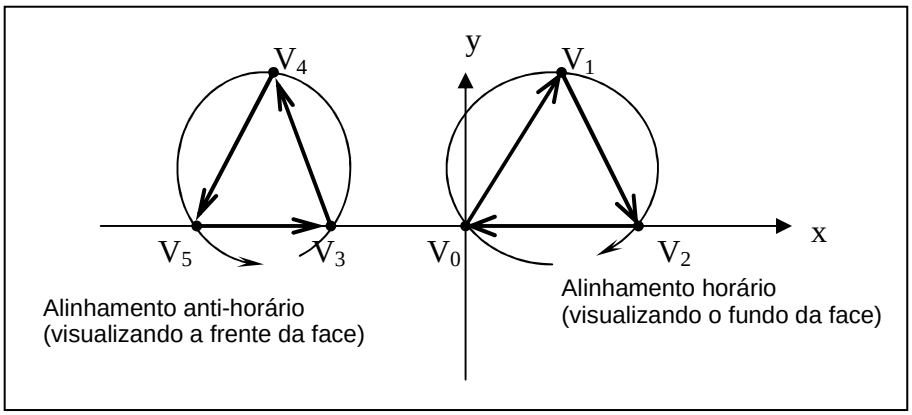
2.5.2.2 BACKFACE CULLING

Um método antes exclusivo da biblioteca OpenGL, porém agora disponível na última versão do Direct3D (não estudada neste trabalho), é a técnica de *backface culling*. *Culling* é o termo usado para descrever a técnica de eliminação de geometria que sabe-se que nunca será visualizada. O fato de não enviar essa geometria para a função de cálculo de *z-buffer* e para os módulos posteriores do *rendering pipeline* pode proporcionar incrementos significativos na velocidade de renderização. Uma das sub-técnicas do *culling* é aquela chamada *backface*

culling, que elimina as partes de trás de uma superfície. Por exemplo, uma esfera (vista de fora) é uma superfície fechada onde o interior jamais pode ser visto. Caso esta técnica não seja aplicada, mesmo que o interior nunca seja visto ele é enviado para a função de cálculo de *z-buffer*, onde será eliminado da mesma maneira.

É possível se saber qual lado é o de fora e qual é o de dentro de uma face através de uma estrutura chamada de vetor normal de face. Pelo nome é possível deduzir-se que é um vetor (ou seja, é um vértice que possui posição e uma característica adicional, a orientação), e é normal (perpendicular) a uma face. Esta face obrigatoriamente é coplanar, ou seja, pertence a um plano. O vetor normal pode estar em um ou outro lado da face, dependendo da forma como se define o lado da frente de uma face ou polígono, conforme a figura 2. Este lado é definido pela rotação das coordenadas dos vértices. Caso o sistema esteja instruído a entender como lado da frente aquele onde os vértices da face aparecem em sentido anti-horário, então o vetor normal é definido para este lado. Este sentido pode ser invertido caso necessário.

Figura 2- Definição da frente da face pela ordem de rotação dos vértices



2.5.3 MÓDULO DE RASTERIZAÇÃO

O módulo de rasterização recebe as saídas dos módulos anteriores e gera os *pixels* que formarão a cena, ou seja, ele cria a imagem colorida a partir dos dados de geometria, cor (incluída aí a iluminação) e textura.

As primeiras placas de vídeo aceleradoras 3D eram apenas rasterizadores rápidos. Elas apenas aceleravam o módulo de rasterização do *rendering pipeline*. O computador era responsável pelos módulos anteriores do *rendering pipeline*, numa implementação por software destes estágios. Atualmente, mesmo as aceleradoras 3D mais simples possuem os

módulos de transformação e iluminação implementados em hardware. O resultado é uma renderização mais rápida e detalhada, com menor utilização da CPU, podendo esta ficar livre para outras funções, como cálculos de física dos jogos, por exemplo.

2.5.4 FRAME BUFFER / SUPERFÍCIE DIRECTDRAW

A outra diferença é o último estágio do *rendering pipeline*, que em OpenGL é uma estrutura chamada *frame buffer*, enquanto em Direct3D é uma superfície DirectDraw. Ambos, porém, possuem o mesmo significado: é a memória do dispositivo gráfico, e tudo que estiver nesta memória será mostrado na tela durante a próxima atualização desta.

2.6 INTRODUÇÃO AO DIRECT3D

O Direct3D é uma interface de programação de aplicação (API) ou biblioteca, e é um subconjunto do pacote de bibliotecas DirectX. Segundo Microsoft (2001), o DirectX é “um conjunto de bibliotecas de baixo nível para a criação de jogos e outras aplicações multimídia. Inclui suporte para gráficos 2D e 3D, efeitos sonoros e música, dispositivos de entrada e suporte para aplicações em rede como jogos para vários jogadores simultâneos”. Ainda segundo Microsoft (2001), o Direct3D é “projetado para disponibilizar jogos de classe mundial e gráficos 3D interativos num computador executando o Microsoft Windows. É uma *interface* de desenho que fornece acesso independente de dispositivo a *hardware* de vídeo 3D numa forma independente de dispositivo”.

Para implementar esta independência de dispositivo utiliza-se um conceito denominado HAL (*hardware abstraction layer*, ou camada de abstração de *hardware*). A aplicação não comunica-se diretamente com o dispositivo, mas com o HAL. Se o dispositivo for capaz de atender ao comando enviado ao HAL, este o envia para o dispositivo. Caso não seja possível, o HAL pode, dependendo da aplicação, retornar um erro para a aplicação do usuário ou mesmo emular em *software* o comando solicitado. Este conceito é conhecido como HEL (*hardware emulation layer*, ou camada de emulação de *hardware*). (Henkels, 1997).

A maior vantagem do uso do Direct3D é a independência de dispositivo. Num segmento extremamente fragmentado como é o das placas de vídeo aceleradoras 3D, a necessidade de uma biblioteca que torne possível a execução de uma aplicação em uma ampla gama de dispositivos diferentes é fundamental.

O Direct3D, por ser intimamente ligado ao Microsoft Windows e exclusivo dessa plataforma, tem a capacidade de carregar diretamente recursos de programas executáveis, bem como de arquivos DLL (biblioteca de ligação dinâmica). Possui suporte também para arquivos .BMP e .PPM para serem utilizados como texturas ou decalques. Existe também um formato de descrição de cena, cuja extensão é .X . O Direct3D possui um pequeno utilitário que converte cenas ou objetos do 3D Studio diretamente para o formato .X .

O Direct3D é uma coleção de *interfaces* COM (*Component Object Model* – Modelo de Objetos Componentes), da Microsoft, que representa estruturas gráficas como malhas e faces, e estruturas de aplicação como janelas e dispositivos de visualização.

2.6.1 COMPONENT OBJECT MODEL

O COM é um modelo de programação projetado para permitir o desenvolvimento de componentes de *software* reutilizáveis e passíveis de extensões seguras. A idéia é que componentes de *software* sejam tão fáceis de instalar quanto componentes de *hardware*.

Segundo Microsoft (2001), “objetos COM são basicamente caixas-pretas que podem ser usadas por aplicações para realizar uma ou mais tarefas. Elas são mais comumente implementadas como DLLs (bibliotecas de ligação dinâmica). Como uma DLL convencional, objetos COM expõem métodos que sua aplicação pode chamar para realizar qualquer uma das tarefas suportadas. Aplicações interagem com objetos COM basicamente da mesma maneira que fazem com objetos C++, porém com algumas diferenças”.

Segundo Trujillo (1997), “O modelo COM é mais restrito. A versão COM de herança, por exemplo, é limitada em comparação à versão C++. Além disso, os objetos COM não permitem membros de dados públicos. Todas as interações com o objeto devem ser feitas através de funções-membro”.

Um componente COM é formado por um objeto e uma ou mais *interfaces* que acessam o objeto. O objeto fornece a funcionalidade real que o COM suporta, mas não pode ser acessado diretamente. Objetos COM são sempre acessados através de *interfaces*. Um único objeto pode suportar múltiplas *interfaces*.

2.6.2 AS INTERFACES DO DIRECT3D

Todas as interfaces do Direct3D *retained mode* dependem de uma interface principal: IDirect3DRM (Onde RM significa *retained mode*). Essa interface é o Direct3D *retained mode* propriamente dito. Outras interfaces do Direct3D são:

a) **IDIRECT3DRMDEVICE** – Seleciona um dispositivo para saída renderizada. Os dois tipos de dispositivos mais comuns são os dispositivos de *software* e os dispositivos de *hardware*. Dispositivos de *software* permitem às aplicações executarem em computadores não equipados com *hardware* de vídeo acelerado para 3D. Dispositivos de *hardware* estão disponíveis apenas em computadores equipados com *hardware* 3D, e permitem que o Direct3D utilize plenamente os recursos presentes no *hardware*.

b) **IDIRECT3DRMWINDEVICE** – Representa os objetos Direct3DRMDevice suportados pelo Windows.

c) **IDIRECT3DRMVIEWPORT** – É o *frustum* ou câmera. É ele que determina a posição e orientação a partir das quais a cena é visualizada.

d) **IDIRECT3DRMFRAME** – Interface de “molduras”. O Direct3D usa o termo *frame* para um “quadro de referência”, ou moldura. Todos os objetos num espaço 3D no Direct3D *retained mode* precisam de uma moldura para se posicionar ou orientar, pois não possuem essas propriedades por si mesmos.

e) **IDIRECT3DRMMESHBUILDER** – Como o nome indica, é uma *interface* construtora de malha. Ela não é uma malha, porém pode ser usada como tal. Esta *interface* possui 38 funções-membro para criar e modificar malhas. Quando acrescentados à cena como elementos visuais, eles criam e usam uma malha interna.

f) **IDIRECT3DRMMESH** – *Interface* para uso de malhas. Apesar de ser muito mais eficiente em termos de velocidade, ela não é tão fácil de usar quanto a *interface* de construção de malha.

g) **IDIRECT3DRMFACE** – Malhas são conjuntos de faces, de forma que criar uma malha significa criar faces. Dificilmente se criam faces isoladamente para serem adicionadas a um construtor de malhas, mas a *interface* e seu conjunto de funções-membro é este.

h) **IDIRECT3DRMTEXTURE** – o Direct3D tem a capacidade de carregar texturas de arquivos .BMP, .PMM, recursos de programa ou da memória e aplicá-las a faces ou

malhas, ou incorporadas diretamente a uma cena como imagem de fundo ou decalque, tudo através da *interface* IDIRECT3DRMTEXTURE.

i) **IDIRECT3DRMTEXTUREWRAP**- *interface* de invólucro de textura. Ou seja, um objeto que descreve como uma textura pode ser aplicada a uma malha. Forma do invólucro, orientação, escala e origem da textura são controladas por esta *interface*.

j) **IDIRECT3DRMMATERIAL** – Os materiais determinam o comportamento da luz sobre as faces e malhas. Esta *interface* permite o controle de três valores: intensidade da luz especular, a cor da mesma e cor da luz emitida pelo próprio objeto.

k) **IDIRECT3DRMLIGHT** – *interface* para luzes. O Direct3D suporta os tipos de luzes ambiente, pontual, direcional, paralela e *spot* (cone). Algumas dessas luzes precisam de posição e orientação, outras apenas de um desses parâmetros, e a luz ambiente não precisa de nenhum. Mas todas as luzes precisam de uma cor.

l) **IDIRECT3DRMSHADOW** – *interface* de sombras. Muito limitada, visto que as técnicas de renderização do Direct3D não consideram sombras. Para isto, deve se adicionar um objeto sombra à uma fonte de iluminação e ao objeto que irá projetar essa sombra. Outra limitação desta *interface* é o fato de as sombras só poderem ser projetadas num plano.

m) **IDIRECT3DRMANIMATION** – *interface* para animação através de quadros-chave (*keyframing*). O termo animação no Direct3D é dúbio: pode ser utilizado para descrever a movimentação de objetos na cena, como também serve para descrever a *interface* de animação por quadros-chave. Cada chave é uma translação, rotação ou fator de escala a ser aplicado a um objeto em um quadro específico da animação. O Direct3D suporte animação linear, onde o objeto se desloca de uma chave a outra utilizando-se do caminho mais curto, e animação curva, a qual evidentemente usa curvas para calcular a posição entre as chaves de um objeto, produzindo um movimento mais suave.

n) **IDIRECT3DRMANIMATIONSET** – Um conjunto de animação é um conjunto de objetos de animação, utilizado para representar cenas inteiras animadas. Normalmente a criação de um conjunto de animação dá-se pela importação de uma cena animada por um *software* de animação, como o 3D Studio. Cada objeto na cena é representado por um objeto de animação, e todos os objetos de animação compõem o conjunto de animação.

2.7 INTRODUÇÃO A OPENGL

Segundo Woo (1998), OpenGL é “uma interface de *software* para *hardware* gráfico”. Essencialmente, é uma biblioteca gráfica e de modelagem 3D extremamente portável e muito rápida. Foi desenvolvida pela Silicon Graphics, Inc. (SGI), um reconhecido líder mundial em computação gráfica e animação.

A OpenGL foi desenhada como uma *interface* independente de *hardware* para ser implementada em diversas plataformas de *hardware*. Para tal, segundo Woo (1998), nenhum comando para realizar funções de janela ou obtenção de entrada do usuário foram incluídas na OpenGL; ao invés disso, deve-se trabalhar com funções próprias do sistema particular a ser utilizado. Da mesma forma, OpenGL não fornece comandos de alto nível para descrever modelos de objetos tridimensionais. É, portanto, uma biblioteca do tipo *immediate mode*.

Entretanto é possível construir-se uma biblioteca que possa fornecer esses recursos sobre a biblioteca OpenGL. A biblioteca conhecida como GLU (OpenGL Utility Library) (OpenGL, 2001) fornece muitos recursos de modelagens, como superfícies e curvas NURBS, por exemplo. GLU é uma parte padrão de toda implementação OpenGL. Existe uma nova biblioteca, chamada GLUT (OpenGL Library Utility Toolkit) (OpenGL, 2001) que fornece ainda mais funcionalidade ao desenvolvedor, como alguns sólidos tridimensionais simples (cubo, cone, esfera), bem como entrada de usuário, de uma forma independente de plataforma. GLUT vem sendo cada vez mais utilizado em conjunto com OpenGL pela sua facilidade de uso e por já vir incluída em muitas implementações da biblioteca OpenGL.

O nome OpenGL significa, literalmente, “biblioteca gráfica aberta”, e portanto não é uma linguagem, como C ou C++. Não existe algo como um programa “escrito em OpenGL”, mas sim um programa que pode utilizar OpenGL como uma de suas bibliotecas. Da mesma forma que pode-se usar uma biblioteca do Windows para acessar um arquivo, utiliza-se OpenGL para gráficos interativos de tempo real.

Uma implementação de OpenGL pode ser uma biblioteca de *software* que cria imagens tridimensionais em resposta a chamadas de funções OpenGL ou um controlador de dispositivo (normalmente uma placa de vídeo) que faz o mesmo. Ou seja, existem dois tipos de implementação de OpenGL, uma baseada em *hardware*, outra em *software*. A implementação OpenGL que acompanha os sistemas operacionais Windows NT 3.1 em

diante, o Windows 95 em diante e o Windows 2000 é uma típica implementação de *software*. Quando instala-se uma placa de vídeo aceleradora 3D que implemente no seu controlador (*driver*) a biblioteca OpenGL, esta é uma implementação dita em *hardware*.

2.7.1 FUNCIONAMENTO DO OPENGL

OpenGL é uma biblioteca gráfica orientada a procedimento. Ao invés de descrever uma cena e como ela deve parecer-se, o programador na verdade descreve os passos necessários para alcançar certa aparência ou efeito. Estes "passos" envolvem chamadas à biblioteca OpenGL, que inclui mais de 200 comandos e funções. Estes comandos são usados para desenhar primitivas gráficas como pontos, linhas e polígonos em três dimensões. Além disso, OpenGL suporta iluminação e sombreamento, mapeamento de texturas, *blending* (combinação), transparência, animação e muitos outros efeitos especiais e capacidades. Quanto ao sombreamento, ele refere-se ao fato de um objeto ser mais iluminado no lado próximo a uma fonte de luz, enquanto o outro lado é mais escuro. Sombreamento aqui não refere-se a sombras projetadas por um objeto em outro. Embora possível de implementação de diversas formas, não existe uma função OpenGL nativa para sombras, nem recursos para as luzes projetarem sombras automaticamente.

Ao contrário do Direct3D, não existe um "formato de arquivo OpenGL" para modelos ou cenas. Estas cenas são construídas pelo programador para satisfazer suas próprias necessidades de alto nível programando-as utilizando comandos OpenGL de baixo nível. O mesmo ocorre com texturas. OpenGL não tem nenhum formato de textura nativo. Entretanto, é extremamente simples escrever um módulo em C que leia um arquivo .BMP, por exemplo.

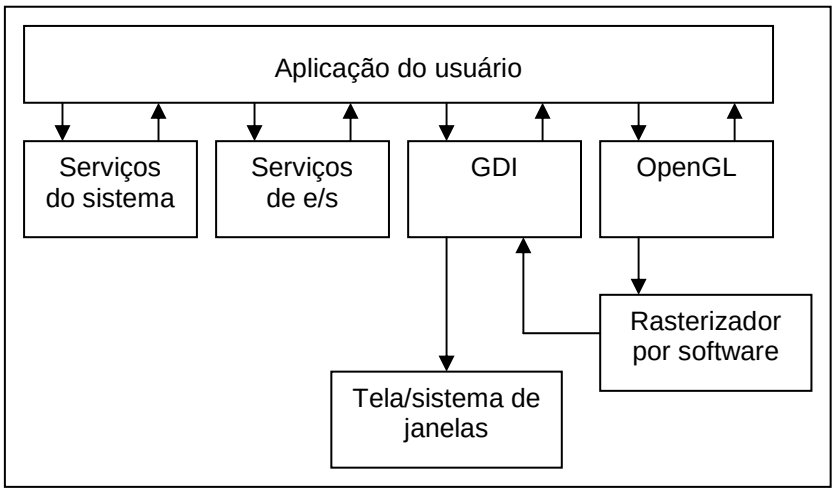
2.7.1.1 IMPLEMENTAÇÕES GENÉRICAS

Uma implementação genérica é uma implementação das funções OpenGL por software. Tecnicamente, uma implementação genérica pode funcionar em qualquer lugar onde o sistema tiver capacidade de mostrar a imagem gerada.

A figura 3 mostra onde fica a biblioteca OpenGL e a implementação genérica quando a aplicação está executando. A aplicação típica chama muitas funções, algumas criadas pelo programador, algumas fornecidas pelo sistema operacional ou pelas bibliotecas da linguagem utilizada. Aplicações Windows, por exemplo, que desejam criar saída na tela normalmente

chamam uma biblioteca do Windows chamada GDI (interface de dispositivo gráfico). A GDI contém funções para escrever texto numa janela e desenhar linhas 2D, por exemplo.

Figura 3 – Localização do OpenGL em uma aplicação típica.



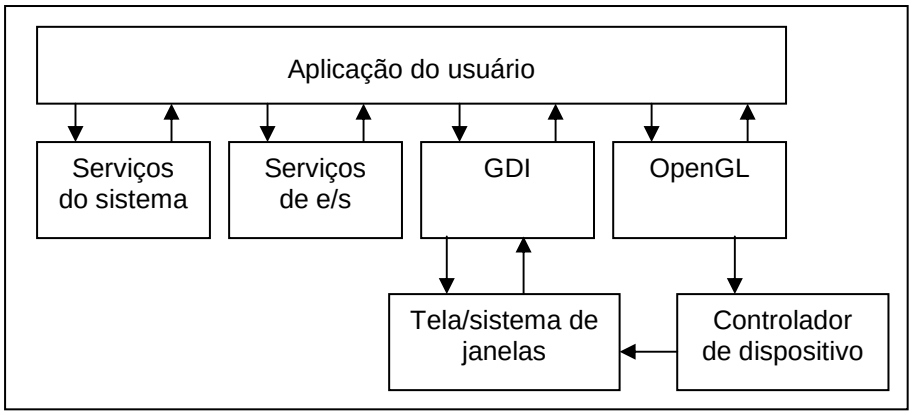
Normalmente, as placas de vídeo acompanham um controlador de dispositivo que é manipulado pela GDI para criar saída visual no monitor. Uma implementação genérica OpenGL recebe pedidos gráficos de uma aplicação e constrói uma imagem dos gráficos tridimensionais. Ela então fornece esta imagem para a GDI, que a mostra na tela. Em outros sistemas operacionais o processo é equivalente, porém trocando-se a GDI pelos serviços de desenho de tela nativos daquele sistema operacional.

Não existem muitas implementações OpenGL genéricas. A Microsoft construiu a sua e a inclui com todas as versões do Windows atualmente. A Silicon Graphics, Inc. lançou uma implementação OpenGL genérica para Windows muito superior àquela da Microsoft, especialmente em processadores baseados em tecnologia MMX. Ela não é atualmente suportada oficialmente, porém ainda é bastante utilizada. Uma outra implementação OpenGL genérica é chamada Mesa 3D e é amplamente utilizada em sistemas abertos, como o Linux e o FreeBSD. Como esta implementação tem seu código-fonte livre para modificação, ela não é uma licença OpenGL oficial. Com exceção do aspecto legal, pode-se considerá-la uma implementação OpenGL como qualquer outra.

2.7.1.2 IMPLEMENTAÇÕES EM HARDWARE

Uma implementação OpenGL em hardware normalmente é um controlador para uma placa de vídeo aceleradora 3D. A figura 4 mostra seu relacionamento com a aplicação. Nela, as chamadas à funções OpenGL são passadas para o controlador de dispositivo. Este controlador não passa sua saída para a GDI do Windows para visualização, e sim trata diretamente com o *hardware* de vídeo.

Figura 4 – Localização de uma implementação OpenGL em hardware em uma aplicação típica.



As implementações de *hardware* são normalmente descritas como implementações “aceleradas” ou “aceleradoras” nos meios comerciais e nas publicações especializadas, por serem muito mais rápidas que as implementações em *software*. O que não está demonstrado na figura 4 é que às vezes parte da funcionalidade OpenGL está ainda implementada em software como parte do controlador de dispositivo, e outros recursos e funcionalidades são passados diretamente para o *hardware*.

2.7.2 OPENGL COMO UMA MÁQUINA DE ESTADOS

Conforme dito antes, OpenGL é uma biblioteca gráfica do tipo *immediate mode*. Cada comando tem um efeito imediato baseado no estado atual de renderização. Estados de renderização são variáveis internas da biblioteca que especificam quais recursos estão ligados ou desligados, ou como eles devem ser aplicados. Por exemplo, a iluminação pode estar habilitada ou desalibitada, as texturas podem aparecer ou não, se a neblina está “ligada”, e qual sua densidade. Cada variável de estado pode ter os valores ligado ou desligado ou mesmo conter algum valor numérico. Estes estados podem ser salvos numa pilha (chamada

por isso de pilha de estados) para recuperação posterior. Podem ser salvos todos os estados ou apenas alguns deles, dependendo da necessidade do programador.

A máquina de estados OpenGL contém apenas cinco funções para configuração, salvamento e recuperação destes estados. São elas:

a) **glEnable** – Habilita algum recurso OpenGL. Na versão 1.2 da biblioteca, existem mais de 30 recursos que podem ser habilitados por esta função, como por exemplo, o uso do método de *backface culling* para otimização da cena.
b) **glDisable –** Desabilita os mesmos recursos que são habilitados por glEnable.
c) **glIsEnabled –** Testa se um recurso OpenGl está ou não ativado. Retorna um valor do tipo Glboolean, podendo ser verdadeiro ou falso.
d) **glPushAttrib –** Salva informação da máquina de estados OpenGL. Na versão atual da biblioteca existem 20 parâmetros que podem ou não ser salvos por este comando.
e) **glPopAttrib** – Restaura informação de estado salva com glPushAttrib.

2.8 A NORMA NBR 13596

A norma NBR 13596 (ABNT, 1996), que cita a si mesma como “equivalente à norma internacional ISO/IEC 9126”, define, como já foi descrito na introdução, seis grandes grupos de características: funcionalidade, confiabilidade, usabilidade, eficiência, manutenabilidade e portabilidade. Segundo a norma, qualidade de *software* é “a totalidade das características de um produto de *software*, que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas”.

Segundo Storch (2000), esta definição de qualidade levanta as seguintes questões:

a) Como definir os requisitos de qualidade do produto: conjunto de necessidades explícitas e implícitas;
b) Como medir a qualidade em uso;
c) Como relacionar as visões da qualidade de produto e processo.

“Uma das grandes dificuldades da utilização da norma NBR13596 é a especificação do método de avaliação, principalmente a definição das métricas aplicadas. Atualmente existem poucas métricas de aceitação geral para as características descritas na norma, o que permite

grupos ou organizações de normalização estabelecer seus próprios modelos de processos de avaliação e métodos para a criação e validação de métricas relacionadas com estas características. Torna-se necessário estabelecer níveis de pontuação e critérios específicos para a aplicação das características com propósito de definição e avaliação". (Storch, 2000)

3 DESENVOLVIMENTO DA APLICAÇÃO

O trabalho compõe-se de duas partes: a primeira uma comparação, teórica, entre as bibliotecas gráficas Direct3D e OpenGL. Como não existe uma norma reconhecida para comparação de produtos de software, foi utilizada a norma NBR13596 (ABNT, 1996), relativa à qualidade de software. As duas bibliotecas são submetidas simultaneamente aos critérios de qualidade da norma, sendo assim possível compará-las. A segunda parte é a implementação de um protótipo que possa demonstrar simultaneamente uma cena idêntica renderizada nas duas bibliotecas, demonstrando assim o comportamento de cada uma delas sob as mesmas condições.

O protótipo também demonstra alguns itens exclusivos de uma das bibliotecas. Estes estão especificados na tecla de ajuda do mesmo.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Para cumprir-se o objetivo deste trabalho, que é o de avaliar as duas bibliotecas gráficas sob a ótica da norma NBR 13596, faz-se primeiramente necessário entender e especificar o que a norma chama de “fatores explícitos e implícitos” de qualidade.

Os fatores explícitos são externados pelo usuário final, sendo este quem define a qualidade do produto final, necessidade de melhorias ou manutenções no produto, correções ou adaptações. No caso das bibliotecas gráficas, o usuário final pode ser tanto o desenvolvedor de aplicações que se utilizam destas como o usuário das aplicações que se utilizam destas ferramentas. Este último tem condições de figurar neste conjunto devido ao fato de grande parte das aplicações comerciais (jogos, em sua maioria) oferecerem implementações em ambas as bibliotecas, permitindo assim ao usuário avaliar as diferenças entre uma e outra biblioteca. No caso do desenvolvimento deste trabalho, entretanto, a figura do avaliador é definida pelo desenvolvedor que utiliza destas bibliotecas gráficas para suas próprias aplicações.

Os fatores implícitos dizem respeito aos fatores de qualidade percebidos pelos desenvolvedores. São eles que atendem aos fatores explícitos e são responsáveis pela produção do software.

A norma NBR 13596 “não fornece subcaracterísticas e métricas, nem métodos para medição, pontuação ou julgamento” de quesitos de qualidade de software. Portanto, cada categoria de software define uma importância específica à cada característica de qualidade presente na norma. Como as bibliotecas gráficas analisadas têm por objetivo principal a produção de gráficos 3D em tempo real, as subcaracterísticas de eficiência são mais importante, por exemplo, do que subcaraterísticas como a instalação do pacote. Segundo Storch (2000), “a importância de cada característica de qualidade varia dependendo do ponto de vista considerado”.

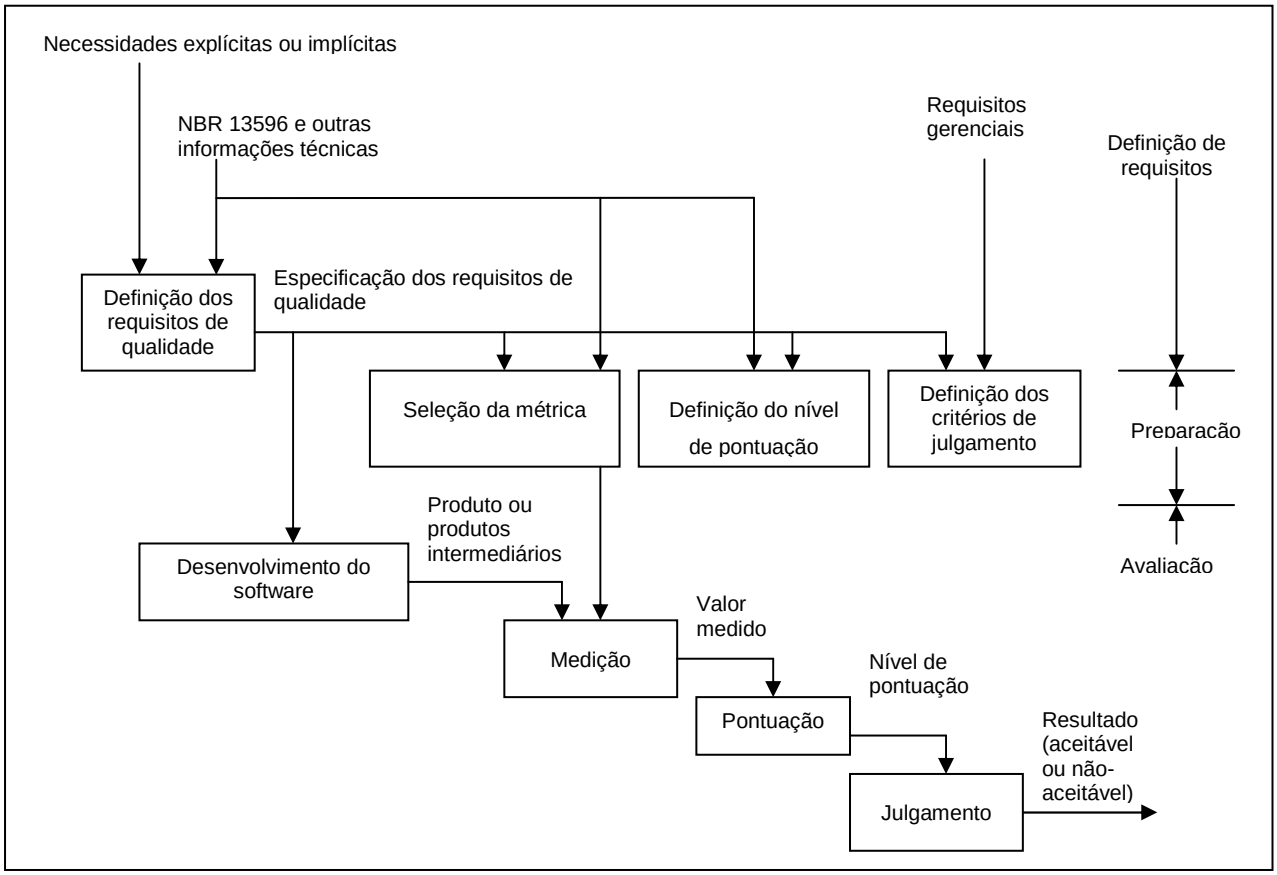
3.1.1 MÉTODO DE AVALIAÇÃO

A norma NBR 13596 (ABNT, 1996) define os seguintes estágios para avaliação de qualidade: definição dos requisitos de qualidade da avaliação, especificação e preparação para avaliação e execução da avaliação. A execução da avaliação ainda desdobra-se em três itens:

a) **Medição** – Na medição, as métricas escolhidas são aplicadas ao software. O resultado é dado em valores nas escadas das métricas.
b) **Pontuação** – No passo referente à pontuação, é determinado um nível para o valor medido. Este nível é específico de cada avaliação.
c) **Julgamento** – O julgamento é o passo final no processo de avaliação de software, no qual um conjunto de níveis pontuados é sintetizado. O resultado é uma declaração da qualidade do software. Segundo ABNT (1996), esta qualidade sintetizada pode ser “comparada com outros aspectos, tais como tempo e custo”.

A figura 5 demonstra o modelo do processo de avaliação de software segundo a norma NBR 13596.

Figura 5 – Modelo de processo de avaliação.



Os requisitos genéricos citados na norma não podem ser diretamente utilizados para uma avaliação de qualidade, portanto fazendo-se necessário um detalhamento maior das características aplicadas. O trabalho de Storch (2000) define, baseado na norma ISO/IEC 9126, portanto similar à norma NBR 13596 utilizada neste trabalho, um questionário para avaliação de qualidade de *softwares* de qualquer área. A avaliação corresponde, portanto, de uma série de perguntas, onde cada resposta a uma questão possui uma pontuação. As três respostas possíveis para determinada característica são:

a) **Atende** – a resposta soma dois pontos no total da característica;
b) **Atende parcialmente** – vale um ponto;
c) **Não atende** – não conta pontos, ou seja, 0 (zero) pontos.

Além de utilizar para avaliação a métrica e um subconjunto das perguntas contidas em Storch (2000), faz-se necessária a atribuição de um grau de importância para cada um dos grupos de características da NBR 13596. No seu trabalho, Storch (2000) define três graus de importância possíveis para uma dada categoria: baixo, médio e alto, valendo 1, 2 ou 3 pontos,

respectivamente. Para questões que não se apliquem ao contexto das bibliotecas gráficas, ou a resposta seja impossível (por exemplo, por não possuir-se o código-fonte das bibliotecas), será adicionada mais uma categoria de importância, de nome “nula”, com valor igual a zero. Uma questão com essa importância, portanto, não influencia o resultado final da avaliação.

Para o cálculo da qualidade final do produto utiliza-se a seguinte fórmula:

$$QFP = \frac{\Sigma(NQC * GIC)}{\Sigma(GIC)}$$

Onde NQC é o nível de qualidade da característica, e GIC é o grau de importância desta característica. Toma-se a soma de todas as características multiplicadas por suas importâncias e divide-se pela soma das importâncias, gerando assim o grau de qualidade do produto segundo a necessidade específica do avaliador, medida pelo grau de importância de cada característica. A NBR 13596 (ABNT, 1996) define quatro faixas de classificação, as quais são: excelente, bom, regular e insuficiente, sendo as três primeiras consideradas satisfatórias, enquanto a classificação insuficiente é considerada insatisfatória.

No gráfico dos níveis de pontuação na NBR 13596 (ABNT, 1996), cada pontuação tem valor igual na escala, onde insuficiente é a faixa de 0 até 0,25, regular é a faixa imediatamente posterior, de 0,25 até 0,50, bom é a faixa de 0,50 até 0,75 e a faixa excelente é de 0,75 até 1, sendo este o valor máximo.

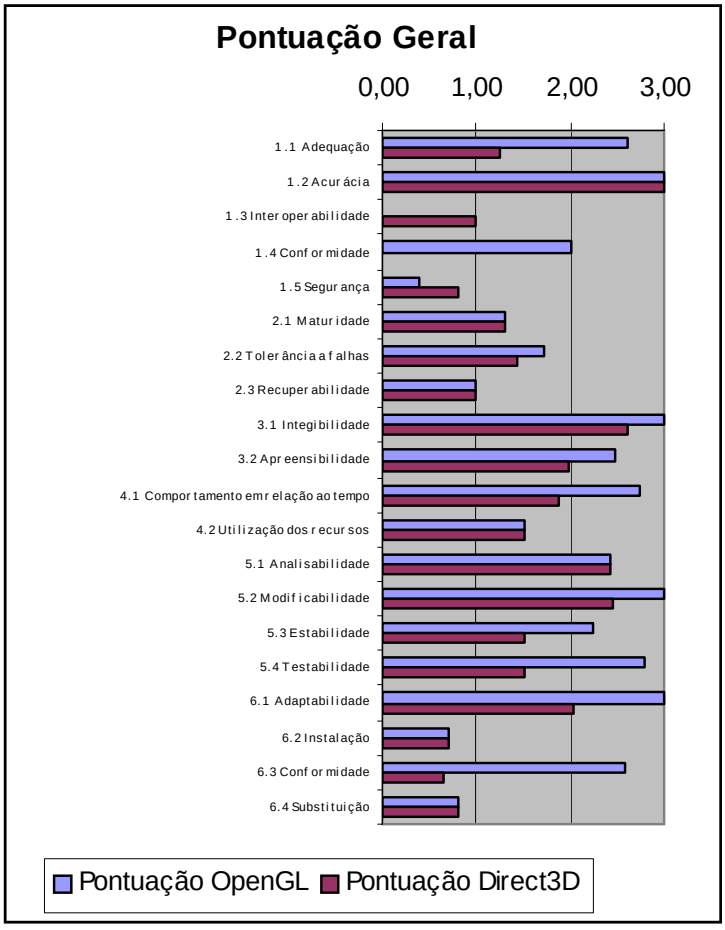
A tabela 1 contém uma descrição de cada subcaracterística, o seu peso, a pontuação máxima atingível para aquela subcaracterística e a pontuação alcançada por cada uma das bibliotecas, já aplicados os pesos. A figura 6 contém um gráfico demonstrando esta pontuação. Estas tabelas são um resumo da tabela do apêndice A.

O apêndice A contém as categorias e perguntas, seus respectivos graus de importância e a pontuação de cada biblioteca em cada pergunta. A tabela e figura supracitadas são na verdade um resumo da tabela presente no apêndice A. Foram adicionadas perguntas específicas ao contexto de bibliotecas gráficas para uma medição mais focalizada. Todas as questões incluídas pelo autor deste trabalho são relativas apenas a bibliotecas gráficas (ex.: a biblioteca dispõe de *buffers* duplos?) Esta tabela contém também comentários pertinentes relativos à algumas das perguntas, observados durante o aprendizado e utilização destas bibliotecas.

TABELA 1 – AVALIAÇÃO DAS BIBLIOTECAS

Descrição	Peso	Máximo	OpenGL	Direct3D		Pontuação OpenGL	Pontuação Direct3D
1.1 Adequação	3	156	136	65		2,62	1,25
1.2 Acurácia	3	12	12	12		3,00	3,00
1.3 Interoperabilidade	1	2	0	2		0,00	1,00
1.4 Conformidade	2	4	4	0		2,00	0,00
1.5 Segurança	1	10	4	8		0,40	0,80
2.1 Maturidade	3	16	7	7		1,31	1,31
2.2 Tolerância a falhas	3	56	32	27		1,71	1,45
2.3 Recuperabilidade	2	12	6	6		1,00	1,00
3.1 Integibilidade	3	52	52	45		3,00	2,60
3.2 Apreensibilidade	3	112	93	74		2,49	1,98
4.1 Comportamento em relação ao tempo	3	24	22	15		2,75	1,88
4.2 Utilização dos recursos	3	24	12	12		1,50	1,50
5.1 Analisabilidade	3	52	42	42		2,42	2,42
5.2 Modificabilidade	3	28	28	23		3,00	2,46
5.3 Estabilidade	3	20	15	10		2,25	1,50
5.4 Testabilidade	3	30	28	15		2,80	1,50
6.1 Adaptabilidade	3	28	28	19		3,00	2,04
6.2 Instalação	1	26	18	18		0,69	0,69
6.3 Conformidade	3	14	12	3		2,57	0,64
6.4 Substituição	1	16	13	13		0,81	0,81
Total	50	694	564	416		40,63	29,97
Percentual de Qualidade:						81,27	59,94

Figura 6 – Gráfico de pontuação das bibliotecas por subcaracterística.

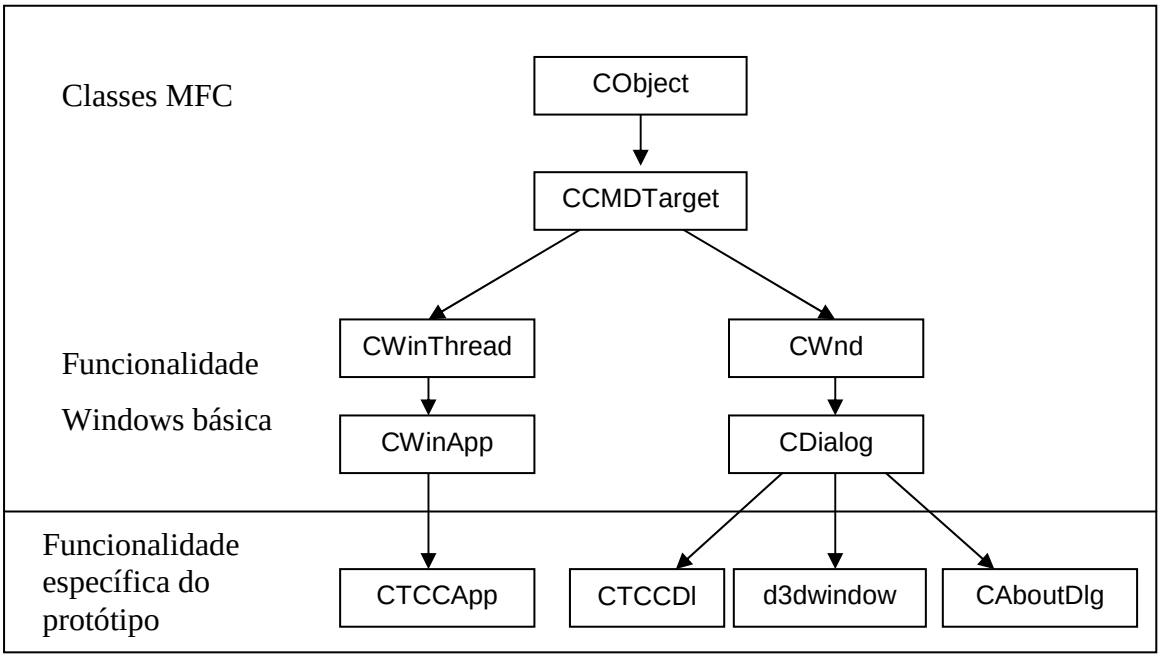


3.2 ESPECIFICAÇÃO

O protótipo foi desenvolvido para demonstrar o comportamento e as diferenças visuais da renderização de cada uma das bibliotecas gráficas. Em linhas gerais, uma cena idêntica, ou seja, os mesmos objetos, com mesmas posições de vértices, as mesmas posições e intensidade de luzes, os mesmos modelos de textura estão escritos utilizando-se código específico para OpenGL, bem como utilizando-se código específico para Direct3D. O código tenta manter-se o mais típico possível de cada biblioteca – como um programador normalmente faria caso aprendesse apenas uma das bibliotecas. Portanto, foram utilizadas ao máximo possível rotinas exclusivas de cada uma das bibliotecas, evitando rotinas genéricas que poderiam influenciar os resultados da implementação. Foi utilizada como técnica para especificação a orientação a objetos, sendo limitada porém pelas características do modelo COM, tornando a especificação conforme o método de Wirfs-Brock (1990) apenas parcialmente possível.

O protótipo utiliza-se da hierarquia de objetos MFC para sua funcionalidade. A figura 7 mostra um diagrama das classes utilizadas no protótipo.

Figura 7 – Diagrama de classes do protótipo.



A MFC é uma biblioteca de classes C++ que isola os programadores dos detalhes do Windows SDK. Segundo Trujillo (1997), a MFC é “um conjunto de objetos flexíveis de baixo nível que podem ser estendidos para servir praticamente a qualquer propósito”.

Neste esquema simplificado, a classe CObject é a superclasse da MFC. Todas as classes da MFC, bem como todas as classes desenvolvidas pelo programador são subclasses de CObject. A classe CCmdTarget é a classe base para a arquitetura de mapas de mensagens da MFC. Todas as classes que lidam com mensagens do sistema são derivadas de CCmdTarget. CWinThread representa uma tarefa de execução em uma aplicação. A tarefa principal de execução de uma aplicação é normalmente uma classe derivada de CWinApp, esta por sua vez sendo derivada de CWinThread. Instâncias adicionais da classe CWinThread possibilitam a execução de múltiplas tarefas dentro de uma dada aplicação. A classe CWnd suporta operações com janelas – de fato, toda janela no Windows é uma classe derivada de Cwnd. CDialog encapsula a funcionalidade de uma janela de caixa de diálogo.

A classe CTccApp é a classe básica do protótipo. Ela basicamente define o ponto de início de execução, e chama o diálogo representado pela classe CTccDlg, que será descrita em detalhe mais adiante. A classe CaboutDlg é a classe do diálogo “Sobre...” da aplicação. A classe d3dwindow é outra classe que implementa a funcionalidade da janela Direct3D. A janela OpenGL foi criada e inicializada com as funções da própria biblioteca, durante o próprio evento de pressionamento do botão no diálogo principal da aplicação.

3.3 IMPLEMENTAÇÃO

Neste capítulo será descrita as características do protótipo de implementação de uma cena tridimensional utilizando-se das bibliotecas gráficas Direct3D e OpenGL. Serão tratados o ambiente de desenvolvimento utilizado, o Microsoft Visual C++ 6.0, uma descrição do contexto do programa e a hierarquia de classes e de cena.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

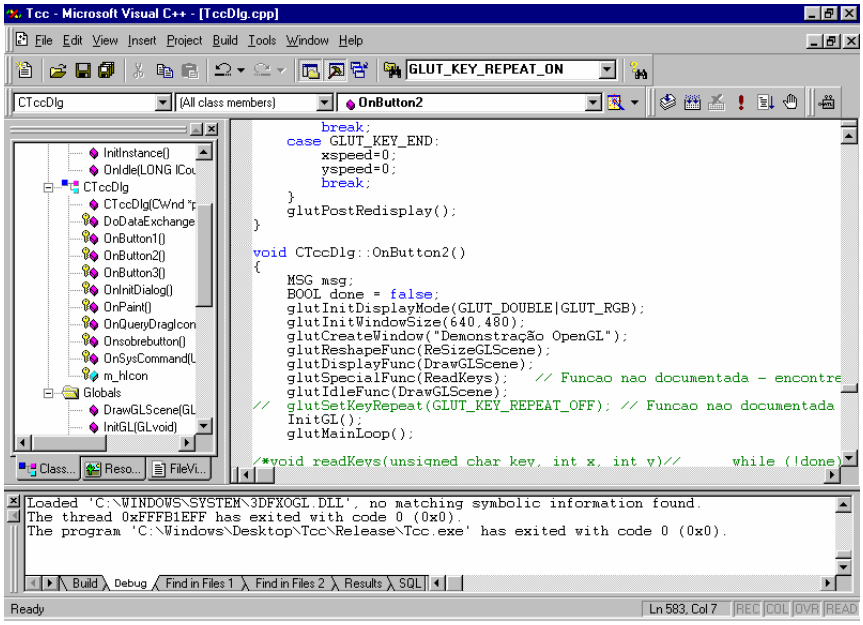
O Microsoft Visual C++ 6.0 faz parte do pacote de desenvolvimento Microsoft Visual Studio, que contempla, além da linguagem Visual C++, as linguagens Visual Basic, Visual Fortan, Visual J++, ferramentas para depuração e otimização dos programas, e ferramentas de conexão e desenvolvimento de aplicações para banco de dados, entre outros recursos. O pacote de desenvolvimento da Microsoft é uma das causas do sucesso do Microsoft Windows, com ferramentas extremamente flexíveis e poderosas, que permitem desenvolver aplicações extremamente complexas em um ambiente integrado. Diversos componentes reutilizáveis

estão disponíveis como objetos COM ou ActiveX. Podem ser utilizados assistentes que desenvolvem o “esqueleto” de uma aplicação genérica, facilitando muito o momento inicial de desenvolvimento de uma aplicação (Microsoft, 1998). A incorporação de recursos como *menus*, ícones, *bitmaps* e outros é simples, utilizando-se de constantes identificadoras universais. O fato de o ambiente ser oriundo da mesma empresa que desenvolve o sistema operacional faz com que a documentação do SDK do sistema acabe se misturando com a da própria linguagem, tornando o aprendizado de novos recursos do sistema operacional bastante rápido.

O Microsoft Visual C++ pode ser utilizado com sua interface visual ou seus elementos individuais (compilador, montador, gerenciador de biblioteca, etc) executados em linha de comando. É possível ainda a depuração de aplicativos através de uma rede. A sua tela principal pode ser vista na figura 8.

Um pacote de documentos chamado Microsoft Developer Network, contendo toda a documentação não só do Microsoft Visual Studio, mas também do SDK do Windows em suas diversas variantes, acompanha o produto. É possível, entretanto, obter-se uma assinatura, onde durante determinado período de tempo, atualizações das documentações, bibliotecas ou mesmo atualizações dos sistemas são enviadas para o usuário, tanto de forma *on-line* via *internet* como por *CD-ROM*.

Figura 8 – Aparência do Microsoft Visual C++ 6.0 durante edição.



O protótipo foi implementado na linguagem Visual C++, que é uma variante do padrão internacional da linguagem orientada a objetos C++ desenvolvido pela Microsoft. Esta linguagem utiliza uma hierarquia de objetos baseados no ambiente de janelas.

As aplicações desenvolvidas em Visual C++ podem chamar funções ou requisitar serviços do sistema operacional, basicamente, por duas formas: utilizando-se o Windows SDK (*software development kit*) ou a MFC (*Microsoft foundation classes*).

O Windows SDK está presente no Windows desde sua versão 1.0. Ele foi escrito em C, e não é orientado a objetos, portanto é extremamente difícil estender sua funcionalidade, obrigando o programador a lidar com todos os detalhes possíveis da programação Windows. Porém, pela sua extrema dificuldade de uso e aprendizado, a Microsoft desenvolveu a MFC, e o Visual C++ só mostra toda sua funcionalidade utilizando-se da MFC (Trujillo, 1997).

A classe CTccDlg é a classe responsável pelo processamento das mensagens vindas da janela principal da aplicação (figura 10). É ela a responsável pelos eventos de abertura das janelas Direct3D e OpenGL, bem como do diálogo “Sobre...” e pelo fechamento da janela. Cada pressionamento de botão gera um evento que é processado por esta classe. A função responsável pelo gerenciamento do evento gerado pelo pressionamento do botão “Abre janela OpenGL” contém as rotinas para inicialização e criação da janela OpenGL, conforme mostrado na figura 9. Da mesma forma, a função responsável pelo botão “Abre janela Direct3D” chama esta janela, que por razões de construção das bibliotecas, tem uma inicialização completamente diferente daquela da janela OpenGL.

Figura 9 – Código responsável pelo gerenciamento da janela OpenGL.

```
void CTccDlg::OnButton2()
{
	MSG msg;
	BOOL done = false;
	glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
	glutInitWindowSize(640,480);
	glutCreateWindow("Demonstração OpenGL");
	glutReshapeFunc(ReSizeGLScene);
	glutDisplayFunc(DrawGLScene);
	glutSpecialFunc(ReadKeys);	//	Funcao	nao	documentada	-
encontrei no glut.h
	glutIdleFunc(DrawGLScene);
//	glutSetKeyRepeat(GLUT_KEY_REPEAT_OFF);	//	Funcao	nao	documentada
- idem - SEM EFEITO NO WINDOWS
	InitGL();
	glutMainLoop();
}
```

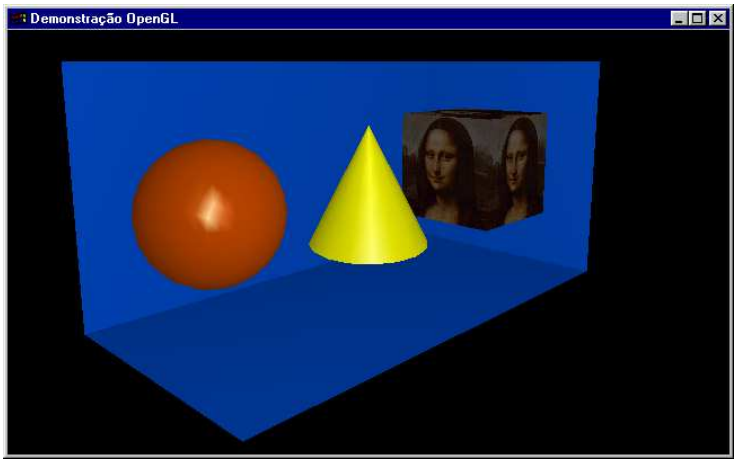
3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

A operação do protótipo é bastante simples: a interface principal, representada pelo diálogo mostrado na figura 10, contém os botões para abertura das janelas Direct3D e OpenGL. Pressionando-se um dos botões obtém-se a janela com uma cena tridimensional, podendo esta estar implementada em uma das duas bibliotecas avaliadas, dependendo evidentemente do botão pressionado, resultando na cena demonstrada na figura 11.

Figura 10 – Diálogo principal do protótipo de aplicação.

Sua operação dá-se da seguinte forma: as teclas de direção rotacionam a cena, permitindo sua visualização por diferentes ângulos. As PAGE UP e PAGE DOWN afastam ou aproximam a cena, respectivamente. Caso a cena ainda esteja girando, a tecla END interrompe imediatamente o movimento. A tecla HOME (apenas OpenGL) muda o filtro de minificação da textura do cubo. Este tipo de filtro é utilizado quando a textura está sendo demonstrada na tela (no protótipo, em uma das faces do cubo) em um tamanho menor do que o tamanho real (em *pixels*) da figura. A diferença é dramática quando a face ou grupo de faces que mostra uma textura está quase “deitada” em relação ao visualizador, como por exemplo a face superior do cubo. A tecla F1 abre uma pequena caixa de diálogo com as opções adicionais disponíveis para cada biblioteca.

Figura 11 – Cena do protótipo.



3.4 RESULTADOS E DISCUSSÃO

Não só na avaliação de qualidade baseada na norma NBR 13596 (ABNT, 1996) e no questionário baseado em (Storch, 2000) como também na demonstração exibida na forma de protótipo de aplicação, ficou clara a qualidade superior da biblioteca OpenGL. Entretanto, a versão mais atual da biblioteca Direct3D tornou-se completamente diferente da versão analisada. A biblioteca do tipo *retained mode* que estava sendo desenvolvida pela Silicon Graphics, Inc. e pela Microsoft, chamada Fahrenheit Scene Graph, e que utilizaria o OpenGL como biblioteca *immediate mode* para sua implementação sofreu uma mudança drástica de planos (Wright, 2000). Primeiramente, decidiu-se que não seria usado OpenGL como biblioteca *immediate mode*, mas seria desenvolvida uma outra biblioteca deste tipo do zero. Então a Silicon Graphics, Inc. decidiu não mais portar a biblioteca para outras plataformas além do Windows. O resultado foi que a biblioteca *immediate mode* desenvolvida para o Fahrenheit Scene Graph tornou-se a última versão do Direct3D, a funcionalidade de uma biblioteca *retained mode* foi diluída em parte na biblioteca Direct3D e parte numa nova biblioteca, chamada Direct3DX. O fato da nova versão ter sido desenvolvido pela mesma equipe que originalmente desenvolveu OpenGL tornou esta biblioteca bastante parecida com OpenGL. Com a inclusão de facilidades de uma biblioteca *retained mode*, espera-se que a nova versão do Direct3D não só equipare como até supere OpenGL em alguns quesitos.

Entretanto, a implementação na forma de uma máquina de estados de OpenGL torna-a bastante intuitiva, não só nas linguagens C e C++, para as quais ela foi elaborada, como também para praticamente todas as linguagens de programação capazes de oferecer uma

extensão OpenGL. Direct3D, ao contrário, só está disponível para as linguagens C++ e Visual Basic. Para acessá-las de outras linguagens, faz-se necessário que estas possam chamar funções e interfaces C++, uma adaptação pouco usual. Além disso, a implementação de classes COM de Direct3D mostrou-se extremamente complexa para implementações simples. Mesmo o fato de esta biblioteca possuir um *retained mode* não livra o programador de detalhes excessivos da programação Windows, e este excesso de detalhes não mostrou uma maior funcionalidade do que as funções de alto nível da biblioteca GLUT disponível para todas as implementações de OpenGL.

Este trabalho foi implementado e testado em um microcomputador equipado com um dispositivo de *hardware* acelerado para 3D, o que tornou a execução do protótipo bastante rápida, conseguindo uma taxa de quadros por segundo de cerca de mais de 200fps (*frames per second*, ou quadros por segundo) na implementação OpenGL e cerca de 90fps na implementação Direct3D. Salienta-se que o monitor utilizado durante o desenvolvimento do trabalho tem uma capacidade máxima de 75fps, o que faz com quadros renderizados pelo dispositivo de vídeo jamais sejam mostrados neste monitor. Estes resultados são exclusivos da máquina utilizada para o desenvolvimento e demonstração deste trabalho. Esta máquina é composta por um processador Intel Celeron 333 mhz, com 64mb de memória *RAM*, equipado com uma 3dfx Voodoo 4 com 64 mb de memória *VRAM* conectada num *slot PCI*. Qualquer outra configuração terá resultados provavelmente muito diferentes.

Uma das dificuldades no estudo de bibliotecas gráficas é a literatura. Até a data de edição deste trabalho não foi encontrado um livro sequer em língua portuguesa sobre OpenGL. Em língua inglesa a documentação é satisfatória para a biblioteca. Direct3D é conhecida por não possuir boa literatura. Até a edição deste trabalho, o livro sobre DirectX da Microsoft Press, a editora dos guias oficiais da Microsoft sequer cobria o assunto Direct3D, nem sequer existiam livros específicos do assunto desta editora. Foi encontrado apenas um livro sobre o assunto em língua portuguesa, cuja qualidade, infelizmente, deixa a desejar.

O trabalho de Storch (2000) foi de extremo valor para a comparação entre as bibliotecas gráficas. Entretanto, apesar de declarar-se apto a avaliar qualquer tipo de *software*, demonstrou-se bastante genérico em alguns pontos onde poderia ter sido mais específico, enquanto em outros poderia ter sido mais genérico, como por exemplo, tendo perguntas

demais sobre impressora e *drives* de disquete, quando há muitos anos estas funções tornaram-se obrigações do sistema operacional.

O trabalho de Henkels (1997) foi também de grande auxílio, principalmente porque a versão analisada da biblioteca Direct3D neste trabalho é aquela utilizada em seu trabalho.

4 CONCLUSÕES

O trabalho cumpriu os seu objetivos, que eram os de exercer um estudo sobre as bibliotecas gráficas e exercer uma comparação entre elas.

As bibliotecas gráficas permitem ao desenvolvedor abstrair-se de grande parte dos detalhes de programação de *hardware* 3D, bem como permitem que aplicações que fazem uso destas possam executar em ambientes sem *hardware* especializado para tal. Elas também permitem ao desenvolvedor abstrair-se da matemática envolvida nas operações de transformações geométricas tanto quanto nas complexas operações de texturização, iluminação e sombreamento, entre outras.

A ferramenta Microsoft Visual C++ demonstrou-se adequada para a construção do protótipo, porém demonstrando certa complexidade não presente em produtos concorrentes, como o Borland C++. Sua estrita integração com o Windows auxiliou no desenvolvimento da aplicação. Seu nível de depuração, entretanto, poderia ser maior, ao menos avisando o programador sobre recursos e serviços requeridos do sistema e não devolvidos a ele.

Os tópicos de ajuda cobertos pelo produto Microsoft Developer Network (Microsoft, 1998) abrangem todos os detalhes da programação Windows, sendo essencial para qualquer programador C++ que deseje aprofundar-se nesta plataforma. De pouca importância foi o livro de Holzner (1999), já que o Microsoft Developer Network supre todas as necessidades em termos de ajuda para o Microsoft Visual C++.

O protótipo cumpriu seu objetivo de mostrar uma cena idêntica em ambas as bibliotecas, demonstrando assim as semelhanças e diferenças entre elas.

Por tratar-se de um protótipo, ele possui certa instabilidade quando do uso simultâneo com outras aplicações que fazem uso do *hardware* 3D disponível no equipamento. Outra limitação é a pequena quantidade de elementos comparados entre as bibliotecas, e a capacidade de testar as diversas versões de cada uma das bibliotecas. Seria desejável também um módulo independente de biblioteca, capaz de carregar arquivos de malha oriundos de *softwares* de modelagem, como o 3D Studio, e demonstrá-los em ambas as bibliotecas. Como OpenGL é uma biblioteca do tipo *immediate mode*, isso foge ao escopo do trabalho.

A principal contribuição deste trabalho para os desenvolvedores que desejam optar por uma das bibliotecas gráficas é a avaliação de qualidade, entretanto não esquecendo que esta avaliação só ocorreu com uma versão de Direct3D que não reflete o estado-da-arte nesta tecnologia simplesmente por não encontrar-se literatura disponível. Um desenvolvedor desejando por optar por uma das bibliotecas certamente terá este item em mente quando da opção de uma das bibliotecas.

4.1 EXTENSÕES

A computação gráfica é uma área bastante ampla, e portanto existe muita coisa a ser estudada e desenvolvida. Algumas sugestões para trabalhos:

a) um protótipo que converta arquivos do 3D Studio (ou qualquer outro aplicativo de modelagem) em código OpenGL ou Direct3D.
b) um jogo, dando a possibilidade do usuário escolher entre utilizar OpenGL ou Direct3D.
c) uma aplicação que possa converter código fonte OpenGL para Direct3D ou vice-versa.

APÊNDICE A – COMPARAÇÃO ENTRE AS BIBLIOTECAS

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
1 Funcionalidade	Capacidade do software satisfazer quaisquer funções adequando estados e necessidades implicadas quando usados sob condições específicas						
1.1 Adequação	Atributos do software que evidenciam a presença de um conjunto de funções e sua apropriação para as tarefas especificadas.						
	1.1.1	O software possui todas as funções mencionadas no produto?	3	3	2	2	
	1.1.2	O software dispõe de todas as funções necessárias para sua execução?		3	1	0	Direct3D é extremamente dependente do DirectX e do Windows como um todo. OpenGL tem as bibliotecas GLU e GLUT que suprem a maioria das funções independente de plataforma.
	1.1.3	O software dispõe de funções para processamento de rotinas específicas?		3	2	2	
	1.1.4	O software dispõe de funções para mapeamento de texturas?		3	2	2	
	1.1.5	O software dispõe de funções para mapeamento de texturas uni e bidimensionais?		3	2	1	Texturas 1D podem economizar geometria. Direct3D não as suporta. OpenGL requer que as texturas 2D sejam quadradas em potências de 2.
	1.1.6	O software dispõe de funções para mapeamento de texturas tridimensionais?		2	2	0	OpenGL disponibiliza, porém ainda não funciona na maioria das implementações. Além disso, o consumo de memória ainda é impraticável.
	1.1.7	O software dispõe de funções para geração automática de coordenadas de textura?		3	2	1	Direct3D possui alguns tipos de invólucro de textura, mas são fixos às formas mais comuns (cilindro, esfera e plano).
	1.1.8	O software dispõe de funções para mapeamento de múltiplas texturas (mipmapping)?		3	2	0	
	1.1.9	O software dispõe de funções para geração automática de múltiplas texturas?		3	2	0	OpenGL pode, através de uma textura de alta definição, gerar texturas menores e utilizá-las quando necessário.
	1.1.10	O software dispõe de funções para geração de luzes pontuais (omni)?		2	0	2	Direct3D possui luzes omni, porém evita-se usar devido ao seu alto custo de processamento.
	1.1.11	O software dispõe de funções para projeção de sombras em superfícies planas?		2	1	2	Direct3D tem uma confusa interface chamada Direct3DRMShadow. Não é possível especificar valores de translucência para esta sombra, entretanto.
	1.1.12	O software dispõe de funções para projeção de sombras em superfícies não planas?		2	2	0	OpenGL pode alcançar esta característica através do uso de transparência combinado com buffer de exclusão (stencil). Direct3D não implementa nenhum destes recursos no retained mode.
	1.1.13	O software dispõe de funções para especificação da espessura das linhas desenhadas?		1	2	0	

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	1.1.14	O software dispõe de funções para aplicação de um padrão de preenchimento nas linhas desenhadas?		1	2	0	
	1.1.15	O software dispõe de funções para desenho de polígonos com mais de três lados?		2	2	1	Todos os polígonos com mais de três lados são internamente transformados em triângulos em Direct3D. Em OpenGL isso depende da implementação.
	1.1.16	O software dispõe de função para desenho de polígonos convexos?		2	2	0	
	1.1.17	O software dispõe de funções para desenho de polígonos com furos?		2	2	0	OpenGL requer que os vértices que definem o furo sejam descritos em ordem horária inversa aos vértices que descrevem a parte externa do polígono.
	1.1.18	O software dispõe de funções para desenho de sólidos?		2	2	0	OpenGL possui esta capacidade através da extensão GLUT que agora acompanha todas as implementações.
	1.1.19	O software dispõe de métodos de remoção de superfícies ocultas?		3	2	1	Direct3D utiliza apenas o método de z-buffer. OpenGL, além deste, tem o mecanismo de backface culling. Nenhuma das duas bibliotecas dispõe de funções de eliminação de objetos não contidos no frustum mas presentes na cena.
	1.1.20	O software não possui limitação de quantidade de luzes na cena?		3	0	2	A implementação Microsoft de OpenGL pode utilizar até 8 luzes.
	1.1.21	O software possui a capacidade de carregar cenas de arquivos modelados por aplicações comerciais?		2	0	2	Direct3D possui uma aplicação chamada conv3ds, que converte arquivos do 3D Studio para o formato nativo do Direct3D, que está começando a ser suportado por aplicações de modelagem comerciais.
	1.1.22	O software dispõe de estruturas para armazenamento de elementos de cena para permitir otimização pelo hardware?		3	2	0	OpenGL disponibiliza diversas estruturas para otimização, entre elas display lists, vertex arrays e texture objects.
	1.1.23	O software dispõe de funções para o uso de transparência e neblina?		3	2	1	Direct3D pode apenas definir uma cor do objeto com totalmente transparente. Graus intermediários não são possíveis.
	1.1.24	O software dispõe de buffers duplos?		3	2	2	Buffers duplos são essenciais para uma animação passar sem chamar a atenção do observador para o processo de redesenho da janela.
	1.1.25	O software dispõe de buffers estereovisuais?		1	1	0	A maioria dos dispositivos de aceleração 3D não suporta este recurso.

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	1.1.26	O software dispõe de buffer de profundidade?		3	2	1	Direct3D possui um buffer de profundidade, porém seu funcionamento é automático para cálculo de z-buffer. Não pode ser modificado para outras aplicações.
	1.1.27	O software dispõe de buffer de acumulação?		2	2	0	Este buffer pode ser utilizado para efeitos de antialias de cena inteira, bem como para borra por movimento (motion blur), porém é extremamente lento. Ele pode ainda ser utilizado para mistura de imagens.
	1.1.28	O software dispõe de buffer de restrição (stencil)?		3	2	0	
	1.1.29	O software dispõe de buffer de seleção?		3	2	2	
	1.1.30	O software dispõe de funções para o desenho de linhas e superfícies do tipo spline?		3	2	0	
	1.1.31	O software dispõe de funções para o desenho de superfícies NURBS?		3	2	0	O uso de superfícies NURBS facilita imensamente a construção de superfícies complexas.
	1.1.32	O software dispõe de capacidade de renderização em múltiplos processadores?		1	2	0	Depende da implementação (OpenGL).
1.2 Acurácia	Atributos do software que evidenciam a geração de resultados ou efeitos corretos conforme os resultados.						
	1.2.1	O software é preciso na execução das funções?	3	3	2	2	
	1.2.2	O software é preciso nos resultados?		3	2	2	
1.3 Interoperabilidade	Atributos do software que evidenciam sua capacidade de interagir com sistemas especificados.						
	1.3.1	O software tem capacidade para processamento multiusuário?	1	0			
	1.3.2	O software tem capacidade para operação em redes?		1	0	2	Como parte do DirectX, o DirectPlay é parte do pacote e supre as funções de rede.
	1.3.3	O software tem restrições quanto ao número de estações trabalhando ao mesmo tempo?		0			
1.4 Conformidade	Atributos do software que fazem com que ele esteja de acordo com as normas, convenções ou regulamentações previstas em leis e descrições similares, relacionadas à aplicação.						
	1.4.1	O software é conciso às leis vigentes?	2	2	2	0	Apesar de ambas as bibliotecas serem destinadas à gráficos tridimensionais de tempo real, Direct3D não pode ser utilizado em aplicações de tempo real de missão crítica (tráfego aéreo, monitoramento de pacientes, etc.) devido ao fato de o próprio Windows ser proibido por contrato de executar tais funções. Já OpenGL depende do sistema operacional permitir essa habilidade ou não.
1.5 Segurança	Atributos do software que evidenciam sua capacidade de evitar o acesso não autorizado, acidental ou deliberado, a programas e dados.						
	1.5.1	O software informa ao usuário a necessidade da realização de backups na instalação de novas versões?	1	1	1	2	Depende da plataforma (OpenGL)

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	1.5.2	O software dispõe de uma rotina interna de backup?		1	1	2	Depende da plataforma (OpenGL)
	1.5.3	O software permite a compactação de backups?		1	0	0	
	1.5.4	O software dispõe de segurança de acesso através de senhas?		0			
	1.5.5	O software dispõe de uma rotina interna de restore?		1	1	2	Depende da plataforma (OpenGL)
	1.5.6	O software possui controle de versão na restauração?		1	1	2	Depende da plataforma (OpenGL)
2 Confiabilidade	A capacidade do software manter seu desempenho quando usado sob condições específicas.						
2.1 Maturidade	Atributos do software que evidenciam a frequência de falhas por defeitos no software.						
	2.1.1	O software tem capacidade de continuar executando na ocorrência de erros de execução?	3	3	0	0	Os programas que utilizam as bibliotecas costumam ter erros irrecuperáveis.
	2.1.2	O software tem capacidade de continuar executando na ocorrência de erros do usuário?		3	2	2	Existem rotinas de tratamento em ambas as bibliotecas.
	2.1.3	O software tem capacidade de garantir a integridade dos dados na ocorrência de erros em execução?		1	0	0	
	2.1.4	O software tem capacidade de garantir a integridade dos dados na ocorrência de queda de energia durante o processamento de atualização dos dados?		1	1	1	Depende da aplicação implementar ou não estas funções.
2.2 Tolerância a falhas	Atributos do software que evidenciam sua capacidade em manter um nível de desempenho especificado nos casos de falhas no software ou de violação nas interfaces especificadas.						
	2.2.1	O software possui advertência de erros cometidos pelo usuário?	3	2	2	2	As duas bibliotecas podem retornar erros ou levantar exceções.
	2.2.2	O software possui advertência quando ocorre erros na recuperação de arquivos?		3	1	2	OpenGL não tem rotinas próprias de tratamento de arquivos. Direct3D possui para carga ou salvamento de texturas, malhas ou cenas.
	2.2.3	O software garante advertência quando ocorrem erros de acesso ao software?		3	2	2	
	2.2.4	O software controla preenchimento de campos?		0			
	2.2.5	O software tem capacidade de controlar o preenchimento incorreto dos campos?		0			
	2.2.6	O software tem capacidade de verificar se as informações cadastradas estão corretas?		2	2	1	Entende-se como parâmetros de funções. OpenGL tem no nome de cada função os valores e tipos aceitos pela mesma.
	2.2.7	O software tem capacidade de evitar inclusão de dados existentes?		0			
	2.2.8	O software garante a integridade dos dados no caso de erros de execução?		3	0	0	Novamente enfatiza-se que travamentos costumam ser irrecuperáveis em ambas as bibliotecas, provavelmente por alterarem estados do hardware 3D.
	2.2.9	O software possui advertência de erros de configuração da impressora?		0			

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	2.2.10	O software permite mudar o modelo padrão de configuração de software?		3	2	2	
	2.2.11	O software tem capacidade de voltar ao estado anterior após parada anormal da máquina?		3	0	0	
	2.2.12	O software mantém a integridade quando ocorrem paradas anormais?		3	0	0	
	2.2.13	O software tem capacidade de informar ao usuário, a situação dos dados após paradas anormais?		3	1	0	Em algumas plataformas, OpenGL pode informar o programa.
	2.2.14	O software tem capacidade de continuar o processamento com grande volume de dados?		3	2	1	As taxas de atualização de cena, medidas em quadros por segundo (FPS), são normalmente maiores em OpenGL, porém depende muito do controlador do hardware 3d.
	2.2.15	O software tem capacidade de suportar usuários simultâneos?		0			
2.3 Recuperabilidade	Atributos do software que evidenciam sua capacidade de restabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e no tempo e esforço para tal.						
	2.3.1	O software tem capacidade de alterar senhas incorretas?	2	0			
	2.3.2	O software possui arquivo temporário para evitar a perda de dados no caso de desligamento do equipamento sem salvar as últimas alterações?		0			
	2.3.3	O software tem capacidade de recuperar dados excluídos no caso de erro de execução?		0			
	2.3.4	Em caso de erros de execução ocorrem erros de informação?		3	0	0	As aplicações falhantes costumam prejudicar todo o sistema (no Windows).
	2.3.5	O software permite recuperar dados excluídos em caso de erros de software?		3	2	2	Caso implementem-se as rotinas de tratamento de erros e exceções presentes em ambas as bibliotecas.
	2.3.6	O software tem capacidade de recuperar dados excluídos?		0			
	2.3.7	A rotina de recuperação do software é fácil de ser usada?		0			
	2.3.8	A recuperação dos dados ocorre de forma rápida?		0			
3 Usabilidade	A capacidade do software em ser fácil de usar e satisfazer o usuário, quando usado sob condições específicas.						
3.1 Integibilidade	Atributos do software que evidenciam o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade.						
	3.1.1	O software possui autodemonstração?	3	2	2	2	Diversos programas-exemplo acompanham ambas as bibliotecas.
	3.1.2	A autodemonstração permite compreender como o software funciona?		1	2	2	
	3.1.3	A autdemonstração é facilmente acessada?		1	2	2	
	3.1.4	O software possui tutorial on-line?		3	2	2	
	3.1.5	Após a instalação, o software oferece a possibilidade de usar o tutorial?		3	2	2	
	3.1.6	As telas do software são autoinstrutivas, permitindo ao usuário visualizar com facilidade qual sua função?		3	2	2	

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	3.1.7	As telas do mesmo nível possuem o mesmo padrão?		3	2	2	
	3.1.8	Os itens de menus possuem termos lógicos de fácil compreensão?		2	2	2	
	3.1.9	Os itens de menus são padronizados, possuindo sempre o mesmo significado?		1	2	2	
	3.1.10	A ordem de apresentação dos menus segue uma lógica?		1	2	2	
	3.1.11	Os submenus mantém a mesma lógica dos menus?		1	2	2	
	3.1.12	O usuário precisa ter profundo conhecimento da área para utilizar o software?		1	2	2	
	3.1.13	Caso o usuário seja leigo, o software fornece as informações adequadas para sua perfeita utilização?		0			As bibliotecas são destinadas a programadores experientes. Faz-se necessário o domínio de uma linguagem de programação e de conceitos de computação gráfica.
	3.1.14	A teoria que embasa o software é explicada com documentação impressa?		3	2	0	Não existe documentação oficial impressa para Direct3D. Todo o pacote é fornecido pela internet. OpenGL possui o guia oficial disponível como livro.
	3.1.15	Os jargões técnicos são explicados no manual do usuário?		1	2	1	A documentação é precária para Direct3D. Alguns termos podem confundir o desenvolvedor. (ex: a palavra FRAME descreve tanto um quadro de animação quanto as molduras que todos os objetos precisam ter em Direct3D.
3.2 Apreensibilidade	Atributos do software que evidenciam o esforço do usuário para aprender sua aplicação.						
	3.2.1	O software possui manual de instalação?	3	1	2	2	
	3.2.2	O software possui manual de operação?		3	2	2	
	3.2.3	O manual de instalação possui índice analítico?		1	2	0	
	3.2.4	O manual de instalação possui índice remissivo?		1	2	2	O índice remissivo pode ser considerada a busca por palavras.
	3.2.5	O manual de operação possui índice analítico?		3	2	2	Como o manual para Direct3D é completamente on line, a guia "conteúdo" da documentação pode ser considerada índice analítico.
	3.2.6	O manual de operação possui índice remissivo?		3	2	2	
	3.2.7	O manual de operação do software possui índice de figuras?		1	0	0	
	3.2.8	O manual de operação do software inclui os termos técnicos utilizados?		1	2	1	O duplo sentido da palavra "Frame" em Direct3D é confuso e não explicado na sua documentação.
	3.2.9	Existe uma hierarquia de manuais de acordo com o nível de conhecimento do usuário?		0			
	3.2.10	Os termos são usados com o mesmo significado durante todo o processamento?		3	2	0	

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	3.2.11	Todas as funções do software estão explicadas no manual?		3	2	0	
	3.2.12	Todos os passos para a instalação do software estão claramente apresentados?		1	1	2	A instalação de OpenGL em algumas plataformas pode ser sofrível.
	3.2.13	A configuração de hardware mínima para a instalação está claramente definida?		2	0	0	
	3.2.14	Os possíveis erros de instalação são apresentados claramente no manual?		2	1	0	Depende da implementação (OpenGL).
	3.2.15	O manual apresenta exemplos de como utilizar o software?		3	1	1	
	3.2.16	O manual apresenta todos os erros que podem ocorrer no software?		3	2	1	
	3.2.17	Os textos dos manuais são corretamente escritos?		3	2	2	
	3.2.18	O manual do usuário explica as convenções de estilo utilizadas durante o documento (negrito, itálico, etc.)?		1	2	2	
	3.2.19	O volume de texto está de acordo com a quantidade de informações obtidas?		3	2	2	
	3.2.20	O software possui ajuda on-line?		3	1	2	Depende da implementação (OpenGL).
	3.2.21	O software possui ajuda on-line de como corrigir os erros cometidos pelo usuário?		3	2	2	
	3.2.22	A ajuda on-line possui um índice?		3	2	2	
	3.2.23	Os termos utilizados tem o mesmo significado em todo arquivo de ajuda?		3	2	1	
	3.2.24	As mensagens de ajuda apresentam uma explicação para todos os itens relacionados à utilização do software?		1	2	2	
	3.2.25	A linguagem utilizada na mensagem de ajuda é facilmente entendida pelo usuário?		3	1	1	
	3.2.26	O software mostra como o usuário deve navegar pelo arquivo de ajuda?		1	1	1	
	3.2.27	As mensagens de orientação estão padronizadas?		1	2	2	
3.3 Operacionalidade	Atributos do software que evidenciam o esforço do usuário para sua operação e controle da sua operação.						
	3.3.1	Os comandos utilizados tem a mesma finalidade em todas as funções do software?	0				
	3.3.2	Os comandos do software estão de acordo com os padrões existentes (F1, DEL, ESC, ENTER)?					
	3.3.3	É possível prever o que existe dentro de cada opção do menu?					
	3.3.4	Existe padronização de teclas de função para todo o software?					
	3.3.5	O software possui atalhos para os usuários mais experientes?					
4 Eficiência	Os recursos usados por um software contido no sistema para alcançar a performance requerida sob condições específicas.						
4.1 Comportamento em relação ao tempo	Atributos do software que evidenciam seu tempo de resposta, tempo de processamento e velocidade na execução de suas funções.						
	4.1.1	O tempo necessário para instalação do software é satisfatório?	3	1	2	1	Direct3D tem um arquivo de cerca de 140 megabytes na versão mais recente.

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	4.1.2	O tempo necessário para inicializar o software é satisfatório?		2	2	2	
	4.1.3	O tempo necessário para fechar o software é satisfatório?		2	1	1	
	4.1.4	O tempo de resposta é adequado em relação ao volume de dados envolvidos?		3	2	1	
	4.1.5	O tempo de resposta é adequado à complexidade das funções do software?		3	2	1	
	4.1.6	O tempo de resposta para a realização das consultas é satisfatório?		1	2	2	
	4.1.7	O tempo necessário para a realização de backups é satisfatório?		0			
4.2 Utilização dos recursos	Atributos do software que evidenciam a quantidade de recursos usados e a duração de seu uso na execução de suas funções.						
	4.2.1	Os recursos do equipamento exigidos pelo software são adequados a complexidade das funções?	3	3	0	0	Ambas permitem renderização por software, apesar de isto tornar a utilização praticamente inviável.
	4.2.2	O acesso a disco no software está de acordo com a complexidade das funções durante a configuração do equipamento usado?		3	2	2	
	4.2.3	O software é coerente na utilização de memória expandida?		3	1	1	
	4.2.4	O software faz muito acesso a disco?		3	1	1	Aplicações 3D costumam ter muitos arquivos de texturas, que podem ter tamanhos imensos, por isso essas aplicações costumam fazer muito acesso a disco.
	4.2.5	O software permite sua operação durante a impressão de documentos?		0			Obsoleto. Isso é controlado pelo sistema operacional.
5 Manutenabilidade	Os recursos necessários para fazer modificações específicas no software.						
5.1 Analisabilidade	Atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas.						
	5.1.1	O software contém uma saída para cada entrada?	3	2	2	2	
	5.1.2	O software contém funções com objetivos específicos?		3	2	2	
	5.1.3	O software utiliza padrão para nomes de identificadores?		3	2	2	
	5.1.4	O software utiliza nomes significativos e concisão para os indicadores?		3	2	1	
	5.1.5	O software utiliza somente variáveis locais, evita as globais?		2	0	1	
	5.1.6	O software possui funções com objetivos específicos?		0			Repetida
	5.1.7	O software possui as decisões comentadas?		0			
	5.1.8	O software possui as variáveis são descritas por comentários?		0			
	5.1.9	O software possui todos os desvios comentados?		0			
	5.1.10	O software possui toda a programação em linguagem de máquina comentada?		0			

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	5.1.11	O nome de todas as variáveis do software são exclusivos?		2	1	1	
	5.1.12	As variáveis são usadas apenas de uma única forma?		0			
	5.1.13	As variáveis globais são usadas consistentemente em relação à unidade e tipos?		2	2	2	
	5.1.14	Todas as variáveis são inicializadas antes do uso?		2	0	1	
	5.1.15	Todos os valores de default são descritos?		3	2	1	
	5.1.16	O software tem capacidade de verificar as entradas?		1	2	2	
	5.1.17	As variáveis locais são definidas?		0			
	5.1.18	O software evita o código automodificável?		2	2	0	Programas Direct3D costumam gravar recursos em seus próprios arquivos executáveis.
	5.1.19	O software verifica possíveis conflitos ou combinações legais de entradas?		1	2	2	
5.2 Modificabilidade	Atributos do software que evidenciam o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais.						
	5.2.1	O software possui notação padronizada para descrever interfaces?	3	3	2	2	
	5.2.2	O software possui notação padronizada para descrever estrutura de dados?		3	2	2	Em Direct3D, a interface D3DFINDDEVICERESULT, por exemplo, possui um campo que precisa do tamanho desta própria estrutura, o que indica uma falta de padronização dos próprios tipos de dados da biblioteca.
	5.2.3	O software permite alterações para acomodar um novo protocolo de comunicação?		1	2	0	Direct3D é totalmente atrelado ao modelo COM, e portanto, não passível de alteração do protocolo de comunicação entre os objetos. OpenGL pode utilizar qualquer forma de comunicação presente na linguagem utilizada.
	5.2.4	O software permite modificações para ser usado em uma máquina diferente?		3	2	1	Apenas em máquinas utilizando o Windows.
	5.2.5	O software permite alterações para adicionar um novo driver?		1	2	2	Ambas as bibliotecas isolam o programa final do hardware. Considera-se aqui driver como controlador de dispositivo.
	5.2.6	O software possui documentação técnica legível?		3	2	2	
	5.2.7	O software possui dicionário de dados bem estruturado para facilitar a modificação?		0			
5.3 Estabilidade	Atributos do software que evidenciam o risco de efeitos inesperados, ocasionados por modificações.						
	5.3.1	O software tem capacidade de evitar a necessidade de manutenção na ocorrência de erros?	3	1	1	1	

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	5.3.2	O software tem capacidade de evitar a atualização de versões frequentes?		3	2	0	Cada aplicação que utilizar uma nova versão de Direct3D precisa de uma instalação completa do DirectX, e novas versões do DirectX são bastante frequentes. OpenGL já tem a versão mais recente em praticamente todas as implementações.
	5.3.3	O software consegue evitar erros após a manutenção do mesmo?		2	2	1	Algumas aplicações não funcionam corretamente após a substituição da versão do DirectX.
	5.3.4	O software tem capacidade de executar a manutenção com rapidez?		1	1	1	Depende da plataforma (OpenGL).
	5.3.5	O software utiliza técnicas de encapsulamento da informação?		3	1	2	Todas as interfaces Direct3D são objetos COM totalmente encapsulados. OpenGL oferece encapsulamento em suas estruturas internas, mas o código do programador não necessariamente utiliza esta técnica.
5.4 Testabilidade	Atributos do software que evidenciam o esforço necessário para validar o software especificado.						
	5.4.1	O software possui uma base de demonstração para realização de testes?	3	3	2	1	OpenGL possui um conjunto de testes de conformidade para saber se uma implementação é ou não compatível.
	5.4.2	O software tem capacidade de executar automaticamente os testes para a validação das modificações?		3	2	1	Os testes de conformidade OpenGL vêm incluídos com a licença fornecida pela Silicon Graphics, Inc. e com a maioria de suas implementações.
	5.4.3	O software possui um guia de testes?		3	2	1	Direct3D possui um utilitário chamado Direct3D caps viewer, que reporta capacidades do atual dispositivo.
	5.4.4	O software possui documentação de testes e configuração de software?		2	1	1	
	5.4.5	O software especifica fundamentos para cada caso de teste?		1	2	1	
	5.4.6	O software especifica uma descrição dos resultados esperados em cada teste?		3	2	1	Os testes de conformidade OpenGL tem respostas fixas para uma implementação ser considerada compatível com OpenGL. Como existe apenas um fornecedor de implementações Direct3D, isso não faz-se necessário.
6 Portabilidade	Capacidade de transferir o software para outros ambientes.						
6.1 Adaptabilidade	Atributos do software que evidenciam sua capacidade de ser adaptado a ambientes diferentes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado.						
	6.1.1	O software pode ser facilmente modificado para atender às necessidades do usuário?	3	1	2	2	
	6.1.2	O software possui versão para utilizar em rede?		0			

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
	6.1.3	O software tem capacidade para operar em ambientes diferentes?		3	2	0	O Direct3D só opera em algumas versões do Windows. O OpenGL está disponível para praticamente todos os sistemas operacionais multiprogramáveis.
	6.1.4	O software pode ser facilmente modificado para atender as alterações sugeridas pelo usuário?		3	2	1	Um dispositivo Direct3D não pode ter seu tamanho alterado durante a execução. Faz-se necessário destruir o dispositivo e criar um novo com as novas dimensões desejadas (por exemplo, quando do redimensionamento de uma janela).
	6.1.5	O software possui rotinas para configuração de drivers?		1	2	2	Considera-se aqui drivers como controladores de dispositivo.
	6.1.6	O software permite adicionar funções com facilidade?		3	2	2	
	6.1.7	O software permite deletar funções com facilidade?		3	2	2	
6.2 Instalação	Atributos do software que evidenciam o esforço necessário para sua instalação em um ambiente						
	6.2.1	O software possui um programa de instalação?	1	3	2	2	OpenGL geralmente já vem instalado ou acompanha drivers de vídeo. Toda nova versão do DirectX deve ser instalada por completo.
	6.2.2	Os comandos utilizados durante a instalação são de fácil entendimento?		1	2	2	
	6.2.3	O software possui ajuda para instalação?		1	1	2	Depende da plataforma (OpenGL).
	6.2.4	O software possui uma ordem lógica na sequência dos discos de instalação?		0			
	6.2.5	O software possui alguma indicação no andamento da instalação?		1	1	2	Depende da plataforma (OpenGL).
	6.2.6	O software possui uma demonstração do software enquanto o usuário instala o mesmo?		1	0	0	
	6.2.7	O software faz a instalação sem a intervenção do usuário?		1	2	0	Depende da plataforma (OpenGL).
	6.2.8	O software tem capacidade de realizar instalação compactada?		1	0	0	
	6.2.9	O software informa o usuário da necessidade de fazer backup antes de fazer a instalação da nova versão?		1	1	1	Em OpenGL, depende da plataforma. O DirectX sempre substitui a versão anterior automaticamente, mas faz backup.
	6.2.10	O software informa o usuário das alterações que serão realizadas na configuração do equipamento?		1	1	1	
	6.2.11	O software verifica se há espaço disponível para a instalação?		1	2	2	
	6.2.12	O software é autosugestivo nas opções de instalações?		1	2	2	

Característica/ Subcaracterística		Descrição / Pergunta	Peso cat.	Peso perg.	Res. OpenGL	Res. Direct3D	Comentários
6.3 Conformidade	Atributos do software que o tornam consoante com padrões ou convenções relacionadas à portabilidade.						
	6.3.1	O software tem capacidade de ser utilizado em diferentes tipos de hardware e com diferentes configurações?	3	3	2	1	Direct3D pode ser utilizado com diversas placas aceleradoras de vídeo 3D diferentes, porém restringe-se à plataforma intel operando sob o Windows. OpenGL está disponível para praticamente todas as plataformas modernas.
	6.3.2	O software tem capacidade de ser utilizado independente da versão de sistema operacional existente?		3	2	0	Direct3D só opera em ambiente Windows, e só recentemente a biblioteca foi disponibilizada para a linha NT de sistemas operacionais. OpenGL é código independente de sistema operacional. Apenas partes específicas da aplicação que lidem com o sistema devem ser modificadas.
	6.3.3	O software possui interface ODBC?		1	0	0	Embora seria interessante este tipo de interface para a carga de recursos (por exemplo, texturas) de um banco de dados, esta função não está diretamente disponível nas bibliotecas.
6.4 Substituição	Atributos do software que evidenciam sua capacidade e esforço necessário para substituir um outro software, no ambiente estabelecido para este outro software.						
	6.4.1	O software tem capacidade de ser substituído por novas versões e continuar utilizando a mesma base de dados?	1	2	2	2	Aplicações escritas para uma versão da biblioteca podem rodar em ambientes com versões mais novas das mesmas. Entretanto, Direct3D apresentou algum problema com este recurso em testes.
	6.4.2	O software tem capacidade de continuar funcionando sem sofrer modificações quando da troca de ambiente?		3	1	1	Depende de como foi escrita a aplicação. As bibliotecas oferecem esta capacidade.
	6.4.3	O software tem condições de executar todas as funções necessárias?		3	2	2	Caso uma aceleração em hardware não esteja disponível, ambas as bibliotecas podem utilizar uma renderização por software.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR13596:** tecnologia da informação – avaliação de produto de software – características de qualidade e diretrizes para o seu uso. Rio de Janeiro, 1996.

HENKELS, Tarcisio. **Desenvolvimento de aplicações gráficas utilizando DirectDraw e Direct3D**. Blumenau, 1997. Monografia (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

HOLZNER, Steven. **Programando visual c++ 6 em tempo recorde**, São Paulo: Makron Books, 1999.

MICROSOFT. **DirectX 8.0**, [S.l.], fev. [2001]. Disponível em: <http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/directx/dx8_c/hh/directx8_c/_introducing_directx_8.0.htm>. Acesso em: 7 mar. 2001.

MICROSOFT. **Microsoft Developer Network**, Redmond: Microsoft, 1998.

OPENGL GROUP. **Overview of OpenGL**, [S.l.], fev [2001]. Disponível em: <http://www.opengl.org/developers/about/overview.html >. Acesso em: 7 mar. 2001.

STORCH, Mirian Mirdes. **Proposta de avaliacão da qualidade de produtos de software utilizando a norma ISO/IEC 9126**. Blumenau, 2000. Monografia (Bacharelado em Ciências da Computação) Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

TRUJILLO, Stan. **Tudo o que você precisa para programar Direct3d**. Rio de Janeiro: Ciência Moderna, 1997.

WIRFS-BROCK, Rebecca. **Designing object-oriented software**. [S.l.]: Prentice Hall, 1990.

WOO, Mason. **OpenGL programming guide**. Berkeley: Addison-wesley, 1998.

WRIGHT, Richard S.; WRIGHT, Richard S. Jr.; SWEET, Michael. **OpenGL superbible**. [S.l.]: Waite Group Press, 2000.