Manual do Geomview

Versão 1.9.5 do Geomview

para GNU/Linux/Unix

Maio de 2008

Mark Phillips et al.

Tradutor para o Português do Brasil
Jorge Barros de Abreu
http://sites.google.com/site/ficmatinf

Copyright © 1992-1998 The Geometry Center
Copyright © 1998-2006 Stuart Levy, Tamara Munzner, Mark Phillips
Copyright © 2006-2007 Claus-Justus Heine

Programa interativo de visualização tridimensional.

Introdução ao Geomview

Geomview é um programa interativo para visualizar e controlar objetos geométricos, originalmente escrito pelos membros do estado maior do Geometry Center na Universidade de Minesota (EUA), começando em 1991. O Geomview pode ser usado como um visualizador independente para objetos estáticos ou como um mecanismo de visualização para outros programas que produzem dinamicamente mudanças geométricas. Geomview roda sobre muitos tipos de computadores Unix, incluindo Linux, SGI, Sun, e HP. Geomview também executa com Cygwin. Esse manual descreve Geomview em sua versão 1.9.

Geomview é um *software* livre, disponível sob os termos da Licença Pública Geral Menor do GNU ; veja [Copiando], página 4 para detalhes.

Geomview e esse manual podem ser encontrados em http://www.geomview.org.

É permitido fazer cópias desse manual.

Se você tiver dúvidas ou comentários sobre o Geomview ou esse manual, considere inscrever-se na lista de correio eletrônico '`geomview-users`', que é um fórum no qual usuários do Geomview comunicam-se para responder outras questões e para compartilhar notícias sobre o que eles estão fazendo com o Geomview. Os autores do Geomview participam dessa lista e algumas vezes enviam respostas a questionamentos existentes. Para assinar a lista, visite a página da lista no sítio http://lists.sourceforge.net/mailman/listinfo/geomview-users.

Distribuição

Geomview é um *software* livre; isso significa que qualquer um é livre para usá-lo e livre para redistribuí-lo sob certas condições. Geomview não é de domínio público; é protegido por direitos autorais e existem restrições sobre sua distribuição, mas essas restrições são montadas de forma a permitir qualquer coisa que um bom cidadão colaborador possa querer fazer. O que não é permitido é para tentar previnir outros de compartilhamento adicional de qualquer versão do Geomview que eles possam pegar de você. As condições precisas podem ser encontradas na Licença Geral Menor do GNU que acompanha o Geomview e também aparece acompanhando essa seção.

Uma forma de acessar uma cópia do Geomview é a partir de alguém que já possua o Geomview. Você não precisa perguntar por nossa permissão para fazer isso, ou informar qualquer coisa; apenas faça a cópia. Se você tiver acesso à internet, você pode pegar a mais recente versão do Geomview em `http://www.geomview.org`.

Você também pode receber Geomview quando você compra um computador. Fabricantes de computadores estão livres para distribuir cópias sob os mesmos termos que são aplicados a qualquer pessoa. Esses termos requerem que os fabricantes forneçam a você o código completo, incluindo qualquer mudanças que eles tenham feito, e permitam a você que redistribua o Geomview recebido deles nos termos usuais da Licença Pública Geral Menor do GNU. Em outras palavras, o programa deve ser livre para você quando você o receber, não apenas livre para o fabricante.

Copiando

NOTA: Geomview é distribuído sob a LICENÇA PÚBLICA GERAL
MENOR. Para os propósitos dessa licença nós pensamos em Geomview como se ele fosse uma "biblioteca", e dos módulos externos do Geomview como "programas que se comunicam com a biblioteca". Fazemos isso porque queremos especificamente permitir que programas proprietários e módulos usem o Geomview.

LICENÇA PÚBLICA GERAL DO GNU

Version 2.1, February 1999

Licença Pública Geral Menor do GNU

This is an unofficial translation of the GNU Lesser General Public
License into Portuguese. It was not published by the Free Software Foundation,
and does not legally state the distribution terms for software that uses the
GNU LGPL–only the original English text of the GNU LGPL does that. However,
we hope that this translation will help Portuguese speakers understand the
GNU LGPL better.

Esta é uma tradução não-oficial da GNU Lesser General Public
License para o Português. Ela não é publicada pela Free Software Foundation
e não traz os termos de distribuição legal do software que usa a GNU LGPL – estes termos estão contidos apenas no texto da GNU LGPL original em inglês.
No entanto, esperamos que esta tradução ajudará no
melhor entendimento da GNU LGPL em Português.

Versão 2.1, Fevereiro de 1999

Copyright © 1991, 1999 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA [Estados Unidos da América]

É permitido a qualquer pessoa copiar e distribuir cópias
sem alterações deste documento de licença, sendo vedada, entretanto, sua modificação.

[Esta é a primeira versão da GPL Menor a ser lançada. Ela também constitui
a sucessora da Licença Pública de Biblioteca do GNU, daí o número 2.1. da versão].

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

Introdução

As licenças da maioria dos softwares são elaboradas para suprimir sua liberdade de compartilhá-los e modificá-los. As Licenças Públicas do GNU, ao contrário, têm o objetivo de assegurar sua liberdade para compartilhar e modificar softwares livres para garantir que o software seja livre para todos os seus usuários.

A presente Licença Pública Geral Menor se aplica a alguns pacotes de software especialmente designados - normalmente bibliotecas - da Free Software Foundation e de outros autores que decidam utilizá-la. Você pode utilizá-la também, mas recomendamos que antes, você analise cuidadosamente se esta licença, ou a Licença Pública Geral comum, é a melhor estratégia a ser adotada em cada caso específico, tendo como base as explicações abaixo.

Quando falamos de software livre, estamos nos referindo a liberdade de uso e não de gratuidade de preço. Nossas Licenças Públicas Gerais são elaboradas para garantir que

você tenha liberdade para distribuir cópias de software livre (cobrando por esse serviço se você assim o desejar); que você receba código-fonte ou o obtenha, se quiser; que você modifique o software e utilize partes dele em novos programas livres; e que você tenha ciência de que pode praticar estes atos.

A fim de proteger seus direitos, é necessário que façamos restrições que proíbam distribuídores de negar estes direitos a você ou de pedir que você que renuncie a eles. Essas restrições se traduzem em determinadas responsabilidades que você deverá assumir, se vier a distribuir cópias da biblioteca ou modificá-la.

Por exemplo, se você distribuir cópias da biblioteca, seja gratuitamente ou mediante um valor, terá de conceder a seus receptores todos os direitos que estamos concedendo a você. Você terá de garantir que eles, também, recebam ou possam obter o código fonte. Se você ligar outro código com a biblioteca, você deve fornecer os arquivos-objeto completos para os receptores, de modo que eles possam ligá-los novamente com a biblioteca após terem feito mudanças na biblioteca e recompilado a mesma. E você terá de exibir a eles esses termos, para que eles conheçam seus direitos.

Protegemos seus direitos através de um método que envolve dois passos: (1) estabelecemos direitos autorais sobre a biblioteca e (2) oferecemos a você esta licença, que dá a você permissão para copiar, distribuir e/ou modificar a biblioteca.

Para proteger cada distribuidor, queremos deixar bem claro que não há nenhuma garantia para a biblioteca livre. Além disso, se a biblioteca for modificada por alguém e passada adiante, os receptores devem saber que o que eles têm não é a versão original, de modo que a reputação do autor original não será afetada por problemas que possam ser introduzidos por outros.

Por fim, as patentes de software representam uma ameaça constante para a existência de qualquer programa livre. Queremos assegurar que uma empresa não possa efetivamente restringir os usuários de um programa livre por ter obtido uma licença restritiva de um titular de direitos de patente. Por isso, insistimos que qualquer licença de patente obtida para alguma versão da biblioteca seja consistente com a plena liberdade de uso, especificada nesta licença.

A maior parte dos softwares do GNU, incluindo algumas bibliotecas, está coberta pela Licença Pública Geral comum do GNU. A presente Licença Pública Geral Menor do GNU se aplica a determinadas bibliotecas designadas, sendo bastante diferente da Licença Pública Geral comum. Usamos esta licença para determinadas bibliotecas, a fim de permitir a ligação dessas bibliotecas a programas não-livres.

Quando um programa é ligado a uma biblioteca, seja estaticamente ou usando uma biblioteca compartilhada, essa combinação das duas é em termos legais uma obra combinada, uma derivação da biblioteca original. Por essa razão, a Licença Pública Geral comum somente permite essa ligação se a combinação como um todo atender a seus critérios de liberdade. A Licença Pública Geral Menor permite critérios mais flexíveis para a ligação de outros códigos à biblioteca.

Chamamos esta licença de Licença Pública Geral "Menor" porque ela faz Menos para proteger a liberdade do usuário do que a Licença Pública Geral comum. Ela também oferece a outros desenvolvedores de software livre uma Menor vantagem na competição com com programas não livres. Essas desvantagens são o motivo pelo qual usamos a Licença Pública

Geral comum para muitas bibliotecas. Por outro lado, em determinadas circunstâncias especiais, a licença Menor oferece vantagens.

Por exemplo, em raras ocasiões, pode existir uma necessidade especial de se incentivar a mais ampla utilização possível de uma determinada biblioteca, para que ela se torne um padrão de fato. Para conseguir isso, deve-se permitir que programas não-livres utilizem a biblioteca. Um caso mais freqüente ocorre quando uma biblioteca livre desempenha a mesma função de bibliotecas não-livres amplamente usadas. Nesse caso, existem poucas vantagens em restringir a biblioteca livre somente para software livre, então utilizamos a Licença Pública Geral Menor.

Em outros casos, a permissão para usar uma determinada biblioteca em programas não-livres possibilita que um maior número de pessoas use um amplo leque de softwares livres. Por exemplo, a permissão para usar a Biblioteca C do GNU permite que muito mais pessoas usem todo o sistema operacional do GNU, bem como sua variante, o sistema operacional do GNU/Linux.

Mesmo protegendo a liberdade dos usuários em menor grau, a Licença Pública Geral Menor garante ao usuário de um programa que esteja ligado à Biblioteca a liberdade e os meios para executar o programa, usando uma versão modificada da Biblioteca.

Seguem abaixo os termos e condições exatos para a cópia, distribuição e modificação. Preste muita atenção à diferença entre uma "obra baseada na biblioteca" e uma "obra que usa a biblioteca". O primeiro contém código que é derivado da biblioteca, enquanto o segundo tem de ser combinado à biblioteca para que possa ser executado.

TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO

0. O presente Contrato de Licença se aplica a qualquer biblioteca de software ou a outro programa que contenha um aviso colocado pelo titular dos direitos autorais ou outra parte autorizada, informando que ela pode ser distribuída nos termos desta Licença Pública Geral Menor (também denominada "esta Licença"). Cada licenciado doravante será denominado "você".

   Uma "biblioteca" significa uma coleção de funções de software e/ou dados preparados, de forma a serem convenientemente ligados com programas de aplicação (que usam algumas dessas funções e dados) para formar executáveis.

   O termo "Biblioteca", abaixo, refere-se a qualquer biblioteca de software ou obra que tenha sido distribuída de acordo com esses termos. Uma "obra baseada na Biblioteca" significa tanto a Biblioteca como qualquer obra derivada, nos termos da legislação autoral: isto é, uma obra contendo a Biblioteca ou parte dela, seja sem alterações ou com modificações e/ou traduzida diretamente para outra linguagem. (Doravante, o termo "modificação" inclui, sem reservas, o termo "tradução").

   O "código-fonte" de uma obra significa o formato preferencial da obra para que sejam feitas modificações na mesma. Para uma biblioteca, o código-fonte completo significa todo o código fonte para todos os módulos contidos na mesma, além de quaisquer arquivos de definição de interface associados, além dos scripts utilizados para controlar a compilação e a instalação da biblioteca.

Outras atividades que não a cópia, distribuição e modificação não são cobertas por esta Licença; elas estão fora de seu escopo. O ato de executar um programa usando a Biblioteca não tem restrições, e o resultado gerado a partir desse programa encontra-se coberto somente se seu conteúdo constituir uma obra baseada na Biblioteca (independente do uso da Biblioteca em uma ferramenta para escrevê-lo). Na verdade, isto dependerá daquilo que a Biblioteca faz e o que o programa que usa a biblioteca faz.

1. Você pode copiar e distribuir cópias sem alterações do código-fonte completo da Biblioteca ao recebê-lo, em qualquer meio ou mídia, desde que publique, ostensiva e adequadamente, um aviso de direitos autorais (ou copyright) apropriado e uma notificação sobre a exoneração de garantias; mantenha intactas as informações, avisos ou notificações referentes a esta Licença e à ausência de qualquer garantia; e distribua uma cópia desta Licença junto com a Biblioteca.

   Você poderá cobrar um valor pelo ato físico de transferir uma cópia, e você pode oferecer, se quiser, a proteção de uma garantia em troca de um valor.

2. Você pode modificar sua cópia ou cópias da Biblioteca ou qualquer parte dela, formando, assim, uma obra baseada na Biblioteca, bem como copiar e distribuir essas modificações ou obra, em conformidade com a Cláusula 1 acima, desde que atenda, ainda, a todas as seguintes condições:

   a. A obra modificada tem de ser, por si só, uma biblioteca de software.

   b. Você tem de fazer com que os arquivos modificados contenham avisos, em destaque, de que você modificou os arquivos e a data de qualquer modificação.

   c. Você tem de fazer com que a obra como um todo seja licenciada, sem nenhum custo, a todos os terceiros, de acordo com esta Licença.

   d. Se um dispositivo, na Biblioteca modificada, se referir a uma função ou a uma tabela de dados a ser fornecida por um programa de aplicação que usa esse dispositivo, outro que não um argumento transmitido quando o dispositivo é invocado, nesse caso, você terá de fazer um esforço de boa-fé para assegurar que, no caso de uma aplicação que não forneça essa função ou tabela, o dispositivo ainda assim opere, e irá realizar qualquer parte de sua finalidade que permanecer significativa.

      (Por exemplo, uma função de uma biblioteca para computar raízes quadradas tem uma finalidade que é completamente bem definida independentemente da aplicação. Por essa razão, a letra d, da Cláusula 2, exige que qualquer função ou tabela fornecida pela aplicação, usada por essa função, tem de ser opcional: se a aplicação não fornecê-la, a função de raízes quadradas deverá ainda assim computar raízes quadradas).

   Essas exigências se aplicam à obra modificada como um todo. Se partes identificáveis dessa obra não forem derivadas da Biblioteca e puderem ser consideradas razoavelmente, em si, como obras independentes e separadas, nesse caso, esta Licença e seus termos não se aplicarão a essas partes quando você distribui-las como obras separadas. Todavia, quando você distribuir essas mesmas partes como partes de um todo, que por si seja uma obra baseada na Biblioteca, a distribuição desse todo deverá ser realizada de acordo com esta Licença, cujas respectivas permissões para outros licenciados extendem-se à integralidade deste todo, dessa forma, a toda e qualquer parte, independentemente de quem a escreveu.

Assim, esta cláusula não tem a intenção de afirmar direitos ou contestar os seus direitos sobre uma obra escrita inteiramente por você; a intenção é, antes, de exercer o direito de controlar a distribuição de obras derivadas ou obras coletivas baseadas na Biblioteca.

Além disto, a simples agregação de outra obra, que não seja baseada na Biblioteca, à Biblioteca (ou a uma obra baseada na Biblioteca) em um volume de meio ou mídia de armazenamento ou distribuição, não inclui esta outra obra no âmbito desta Licença.

3. Você poderá optar por aplicar os termos da Licença Pública Geral do GNU ao invés desta Licença, para uma determinada cópia da Biblioteca. Para tanto, você deverá alterar todos os avisos ou notificações que se refiram a esta Licença, para que eles se refiram à Licença Pública Geral comum do GNU, versão 2, ao invés desta Licença. (Se uma versão mais nova do que a versão 2 da Licença Pública Geral comum do GNU tiver sido gerada, então você poderá especificar essa versão, se preferir). Não faça nenhuma outra alteração nesses avisos ou notificações.

   Uma vez que essa alteração tenha sido feita em uma determinada cópia, ela é irreversível para esta cópia, passando a Licença Pública Geral comum do GNU a ser aplicada para todas as cópias e obras derivadas subseqüentes, feitas a partir dessa cópia.

   Essa opção é útil quando você desejar copiar parte do código da Biblioteca em um programa que não seja uma biblioteca.

4. Você poderá copiar e distribuir a Biblioteca (ou uma parte ou obra derivada dela, de acordo com a Cláusula 2) em código-objeto ou formato executável, sob as Cláusulas 1 e 2 acima, desde que inclua todo o código-fonte correspondente, passível de leitura pela máquina, que deve ser distribuído sob os termos das Cláusulas 1 e 2 acima, em um meio ou mídia costumeiramente utilizado para o intercâmbio de software.

   Se a distribuição do código-objeto for feita pela oferta de acesso para cópia a partir de um local designado, então a permissão de acesso equivalente para copiar o código-fonte a partir do mesmo local atende a exigência de distribuição do código-fonte, mesmo que terceiros não sejam levados a copiar a fonte junto com o código-objeto.

5. Um programa que não contenha nenhum derivativo de qualquer parte da Biblioteca, mas que seja desenhado para operar com a Biblioteca ao ser compilado ou ligado a ela, é chamado de uma "obra que usa a Biblioteca". Essa obra, isoladamente, não é uma obra derivada da Biblioteca e, portanto, fica de fora do âmbito desta Licença.

   Entretanto, a ligação de uma "obra que usa a Biblioteca" com a Biblioteca constitui um executável que é um derivado da Biblioteca (pois contém partes da Biblioteca), e não uma "obra que usa a Biblioteca". O executável é, assim, coberto por esta Licença. A Cláusula 6 estabelece os termos para a distribuição desses executáveis.

   Quando uma "obra que usa a Biblioteca" usar material de um arquivo de cabeçalho que é parte da Biblioteca, o código-objeto para a obra poderá ser uma obra derivada da Biblioteca, mesmo que o código-fonte não o seja. Para que isto seja verdade, é especialmente importante se a obra pode ser ligada sem a Biblioteca, ou se a obra é, em si mesma, uma biblioteca. O limiar para que isto seja verdade não é definido com precisão pela lei.

   Se um arquivo-objeto usar somente parâmetros numéricos, layouts e accessors da estrutura de dados, bem como pequenas macros e pequenas funções inline (dez linhas ou menos de extensão), então o uso do arquivo-objeto não é restrito, independente de

ser ele legalmente uma obra derivada. (Executáveis contendo este código-objeto mais partes da Biblioteca continuam submetidos aos termos da Cláusula 6).

Do contrário, se a obra for um derivado da Biblioteca, você poderá distribuir o código objeto da obra sob os termos da Cláusula 6. Quaisquer executáveis contendo esta obra também se submetmem à Cláusula 6, estejam ou não diretamente ligados à Biblioteca em si.

6. Como exceção à Cláusula acima, você também pode combinar ou ligar uma "obra que usa a Biblioteca" à Biblioteca para produzir uma obra contendo partes da Biblioteca e distribuí-la de acordo com os termos de sua escolha, desde que estes termos permitam modificações na obra para uso próprio por parte do cliente e engenharia reversa para depuração dessas modificações.

   Em cada cópia da obra, você terá de colocar um aviso, em destaque, de que a Biblioteca foi usada e que ela e seu uso estão cobertos por esta Licença. Você deverá fornecer uma cópia desta Licença. Se, durante a execução, a obra exibir avisos ou notificações de direitos autorais (ou copyright), você terá de incluir, entre eles, o aviso de direitos autorais (ou copyright) referente à Biblioteca, bem como uma referência direcionando o usuário para a cópia desta Licença. Além disso, você dever tomar ao menos uma das seguintes providências:

   a. Incluir na obra todo o código-fonte da Biblioteca, passível de leitura pela máquina, incluindo quaisquer modificações que foram usadas na obra (as quais devem ser distribuídas conforme as Cláusulas 1 e 2 acima); e, se a obra for um executável ligado à Biblioteca, com toda a "obra que usa a Bilblioteca" passível de leitura pela máquina, como código-objeto e/ou código-fonte, de modo que o usuário possa modificar a biblioteca e, depois, religar para produzir um executável modificado contendo a Biblioteca modificada. (Fica entendido que o usuário que modificar o conteúdo dos arquivos de definições da Biblioteca não necessariamente será capaz de recompilar a aplicação para usar as definições modificadas).

   b. Usar um mecanismo adequado de biblioteca compartilhada para ligar com a Biblioteca. Um mecanismo adequado é aquele que (a) usa, ao tempo da execução, uma cópia da biblioteca já presente no sistema do computador do usuário, e (2) irá operar adequadamente com uma versão modificada da biblioteca, se o usuário instalar uma, desde que a versão modificada seja compatível com a interface da versão com a qual a obra foi feita.

   c. Incluir na obra uma oferta por escrito, válida por pelo menos 3 anos, oferencendo ao mesmo usuário os materiais especificados na letra "a" da Cláusula 6 acima, por um custo não superior ao custo de fazer esta distribuição.

   d. Se a distribuição da obra for feita com a permissão de acesso para copiar, a partir de um local designado, oferecer acesso equivalente para copiar os materiais acima especificados, a partir do mesmo local.

   e. Certificar-se se o usuário já recebeu uma cópia desses materiais ou de que você já enviou uma cópia a esse usuário.

   Para um executável, o formato exigido da "obra que usa a Biblioteca" deve incluir quaisquer dados e programas utilitários necessários para reprodução do executável a partir dele. Todavia, como uma exceção especial, os materiais a serem distribuídos não necessitam incluir algo que seja normalmente distribuído (tanto no formato fonte

quanto binário) com os componentes mais importantes (compilador, kernel, e assim por diante) do sistema operacional no qual executável é executado, a menos que esse componente, em si, acompanhe o executável.

Pode ocorrer que essa exigência contradiga as restrições da licença de outras bibliotecas proprietárias que normalmente não acompanham o sistema operacional. Essa contradição significa que você não pode utilizar ambas e a Biblioteca juntas em um executável distribuído por você.

7. Você pode colocar dispositivos da biblioteca que sejam uma obra baseada na Biblioteca lado-a-lado em uma única biblioteca junto com outros dispositivos de bibliotecas, desde que uma distribuição separada da obra baseada na Biblioteca e dos outros dispositivos de bibliotecas seja, de outro modo, permitida e desde que você tome uma das seguintes providências:
   a. Incluir na biblioteca combinada uma cópia dessa obra baseada na Biblioteca sem a combinação com quaisquer outros dispositivos de biblioteca. Essa cópia tem de ser distribuída de acordo com as condições das cláusulas acima.
   b. Junto com a biblioteca combinada, fornecer um aviso, em destaque, sobre o fato de que parte dela é uma obra baseada na Biblioteca, e explicando onde encontrar o formato não combinado incluso dessa mesma obra.
8. Você não poderá copiar, modificar, sublicenciar, ligar, ou distribuir a Biblioteca, exceto conforme expressamente disposto nesta Licença. Qualquer tentativa de, de outro modo, copiar, modificar, sublicenciar, ligar ou distribuir a Biblioteca é inválida, e automaticamente terminará seus direitos sob esta Licença. Todavia, terceiros que tiverem recebido cópias ou direitos de você, de acordo com esta Licença, não terão seus direitos rescindidos, enquanto estes terceiros mantiverem o seu pleno cumprimento.
9. Você não é obrigado a aceitar esta Licença, uma vez que você não a assinou. Entretanto, nada mais concede a você permissão para modificar ou distribuir a Biblioteca ou suas obras derivadas. Esses atos são proibidos por lei se você não aceitar esta Licença. Portanto, ao modificar ou distribuir a Biblioteca (ou qualquer obra baseada na Biblioteca), você manifesta sua aceitação desta Licença para fazê-lo, bem como de todos os seus termos e condições para cópia, distribuição ou modificação da Biblioteca ou obras nela baseadas.
10. A cada vez que você redistribuir a Biblioteca (ou qualquer obra nela baseada), o receptor automaticamente recebe uma licença do licenciante original para copiar, distribuir, ligar ou modificar a Biblioteca, sujeito a estes respectivos termos e condições. Você não poderá impor quaisquer restrições adicionais ao exercício, pelos receptores, dos direitos concedidos por este instrumento. Você não tem responsabilidade de promover o cumprimento desta licença por parte de terceiros.
11. Se, como resultado de uma sentença judicial ou alegação de violação de patente, ou por qualquer outro motivo (não restrito às questões de patentes), forem impostas a você condições (tanto através de mandado judicial, contrato ou qualquer outra forma) que contradigam as condições desta Licença, você não estará desobrigado quanto às condições desta Licença. Se você não puder atuar como distribuidor de modo a satisfazer simultaneamente suas obrigações sob esta Licença e quaisquer outras obrigações pertinentes, então, como conseqüência, você não poderá distribuir a Biblioteca de nenhuma forma. Por exemplo, se uma licença sob uma patente não permite a redistribuição

por parte de todos aqueles que tiverem recebido cópias, direta ou indiretamente de você, sem o pagamento de royalties, então, a única forma de cumprir tanto com esta exigência quanto com esta licença será deixar de distribuir, por completo, a Biblioteca.

Se qualquer parte desta Cláusula for considerada inválida ou não executável, sob qualquer circunstância específica, o restante da cláusula deverá continuar a ser aplicado e a cláusula, como um todo, deverá ser aplicada em outras circunstâncias.

Esta cláusula não tem a finalidade de induzir você a infringir quaisquer patentes ou direitos de propriedade, nem de contestar a validade de quaisquer reivindicações deste tipo; a única finalidade desta cláusula é proteger a integridade do sistema de distribuição do software livre, o qual é implementado mediante práticas de licenças públicas. Muitas pessoas têm feito generosas contribuições à ampla gama de software distribuído através desse sistema, confiando na aplicação consistente deste sistema; cabe ao autor/doador decidir se deseja distribuir software através de qualquer outro sistema e um licenciado não pode impor esta escolha.

Esta cláusula visa deixar absolutamente claro o que se acredita ser uma conseqüência do restante desta Licença.

12. Se a distribuição e/ou uso da Biblioteca for restrito em determinados países, tanto por patentes ou por interfaces protegidas por direito autoral, o titular original dos direitos autorais que colocar a Biblioteca sob esta Licença poderá acrescentar uma limitação geográfica de distribuição explícita excluindo esses países, de modo que a distribuição seja permitida somente nos países ou entre os países que não foram excluídos dessa forma. Nesse caso, esta Licença passa a incorporar a limitação como se esta tivesse sido escrita no corpo desta Licença

13. A Free Software Foundation [Fundação Software Livre] poderá de tempos em tempos publicar versões revisadas e/ou novasda Licença Pública Geral Menor. Essas novas versões serão semelhantes em espírito à presente versão, podendo, porém, ter diferenças nos detalhes, para tratar de novos problemas ou preocupações.

    Cada versão recebe um número distinto de versão. Se a Biblioteca especificar um número de versão desta Licença, aplicável à Biblioteca ou a "qualquer versão posterior", você terá a opção de seguir os termos e condições tanto daquela versão como de qualquer versão posterior publicada pela Free Software Foundation. Se a Biblioteca não especificar um número de licença da versão, você poderá escolher qualquer versão já publicada pela Free Software Foundation.

14. Se você desejar incorporar partes da Biblioteca em outros programas livres cujas condições de distribuição sejam incompatíveis com estas, escreva ao autor para solicitar permissão. Para software cujos direitos autorais pertencerem à Free Software Foundation, escreva à Fundação; algumas vezes, fazemos exceções nesse sentido. Nossa decisão será guiada pelos dois objetivos de preservar a condição livre de todos os derivados de nosso software livre e de promover o compartilhamento e reutilização de softwares, de modo geral.

    EXCLUSÃO DE GARANTIA

15. COMO A BIBLIOTECA É LICENCIADA SEM CUSTO, NÃO HÁ NENHUMA GARANTIA PARA A BIBLIOTECA, NO LIMITE PERMITIDO PELA LEI APLICÁVEL. EXCETO QUANDO DE OUTRA FORMA ESTABELECIDO POR ESCRITO, OS TITULARES DOS DIREITOS AUTORAIS E/OU OUTRAS PARTES

FORNECEM A BIBLIOTECA "NO ESTADO EM QUE SE ENCONTRA", SEM NENHUMA GARANTIA DE QUALQUER TIPO, TANTO EXPRESSA COMO IMPLÍCITA, INCLUINDO, DENTRE OUTRAS, AS GARANTIAS IMPLÍCITAS DE COMERCIABILIDADE E ADEQUAÇÃO PARA UMA FINALIDADE ESPECÍFICA. O RISCO INTEGRAL QUANTO À QUALIDADE E DESEMPENHO DA BIBLIOTECA É ASSUMIDO POR VOCÊ. CASO A BIBLIOTECA CONTENHA DEFEITOS, VOCÊ ARCARÁ COM OS CUSTOS DE TODOS OS SERVIÇOS, REPAROS OU CORREÇÕES NECESSÁRIAS.

16. EM NENHUMA CIRCUNSTÂNCIA, A MENOS QUE EXIGIDO PELA LEI APLICÁVEL OU ACORDADO POR ESCRITO, QUALQUER TITULAR DE DIREITOS AUTORAIS OU QUALQUER OUTRA PARTE QUE POSSA MODIFICAR E/OU REDISTRIBUIR A BIBLIOTECA, CONFORME PERMITIDO ACIMA, SERÁ RESPONSÁVEL PARA COM VOCÊ POR DANOS, INCLUINDO ENTRE OUTROS QUAISQUER DANOS GERAIS, ESPECIAIS, FORTUITOS OU EMERGENTES, ADVINDOS DO USO OU IMPOSSIBILIDADE DE USO DA BIBLIOTECA (INCLUINDO, ENTRE OUTROS, PERDA DE DADOS, DADOS SENDO GERADOS DE FORMA IMPRECISA, PERDAS SOFRIDAS POR VOCÊ OU TERCEIROS OU A IMPOSSIBILIDADE DA BIBLIOTECA DE OPERAR COM QUALQUER OUTRO SOFTWARE), MESMO QUE ESSE TITULAR, OU OUTRA PARTE, TENHA SIDO AVISADO SOBRE A POSSIBILIDADE DESSES DANOS.

FINAL DOS TERMOS E CONDIÇÕES

Como Aplicar Estes Termos para Suas Novas Bibliotecas

Se você desenvolver uma nova biblioteca e quiser que ela seja da maior utilidade possível para o público, nós recomendamos fazer dela um software livre que todos possam redistribuir e modificar. Você pode fazer isto permitindo a redistribuição sob estes termos (ou, alternativamente, sob os termos da Licença Pública Geral comum)

Para fazer isto, anexe as notificações seguintes à biblioteca. É mais seguro anexá-las ao começo de cada arquivo-fonte, de modo a transmitir do modo mais eficiente a exclusão de garantia; e cada arquivo deve ter ao menos a linha de "direitos autorais reservados" e uma indicação de onde a notificação completa se encontra.

```
uma linha para informar o nome da biblioteca e uma breve idéia do que ela faz.
Direitos Autorais Reservados (C) <ano>  nome do autor

Esta biblioteca é software livre; você pode redistribuí-la e/ou
modificá-la sob os termos da Licença Pública Geral
Menor do GNU conforme publicada pela Free Software Foundation; tanto a
versão 2.1 da Licença, ou (a seu critério) qualquer versão posterior.

Esta biblioteca é distribuído na expectativa de que seja útil,
porém, SEM NENHUMA GARANTIA; nem mesmo a garantia implícita de
COMERCIABILIDADE OU ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. Consulte a Licença Pública
Geral Menor do GNU para mais detalhes.

Você deve ter recebido uma cópia da Licença Pública Geral Menor
do GNU junto com esta biblioteca; se não, escreva para a Free Software
Foundation, Inc., no endereço 59 Temple Street, Suite 330, Boston, MA 02111-1307 USA.
```

Inclua também informações sobre como contatar você por correio eletrônico e por meio postal. Você também pode solicitar a seu empregador (se você for um programador) ou a sua instituição acadêmica, se for o caso, para assinar uma "renúncia de direitos autorais" sobre a biblioteca, se necessário. Segue um exemplo; altere os nomes:

```
A Yoyodyne Ltda., neste ato, renuncia a todos eventuais direitos autorais sobre a
biblioteca 'Frob' (uma biblioteca para ajustar fechaduras), escrita por James
Random Hacker.

<Assinatura de Ty Coon>, 1 de abril de 1990
Ty Coon, Presidente
```

Isso é tudo!

História do Desenvolvimento do Geomview

Geomview foi originalmente escrito no Geometry Center da the University of Minnesota in Minneapolis. O Geometry Center era um centro de pesquisa e educação fundado pela National Science Foundation, com a missão de promover pesquisa e comunicação de assuntos relacionados à matemática. A maioria do trabalho era direcionado ao uso de computadores para ajudar a visualização de conceitos matemáticos.

O projeto que eventualmente levou ao Geomview começou no verão de 1988 com o trabalho de Pat Hanrahan sobre um programa de visualização chamado MinneView. Pouco tempo depois Charlie Gunn começou o desenvolvimento da OOGL (Object Oriented Graphics Language) juntamente com o MinneView. Muitas pessoas comtribuiram para a OOGL e o MinneView, incluindo Stuart Levy, Mark Meuer, Tamara Munzner, Steve Anderson, Mario Lopez, Todd Kaplan.

Em 1991 o staff do Geometry Center começou o trabalho sobre uma nova e melhorada versão da OOGL, e um novo e melhorado programa de visualização, o qual eles chamaram Geomview. Naquele tempo essencialmente o único jogo no comercio para gráficos interativos em três dimensões era Silicon Graphics (SGI), de forma que Geomview foi desenvolvido inicialmente em estações de trabalho SGI, usando a IRIS GL. A primeira versão foi finalizada em Janeiro de 1992. Essa primeira versão tornou-se muito popular entre visitantes do Geometry Center, e através do ftp do Centro (isso ocorreu antes do advento da web) pessoas em outras instituições começaram a usar essa primeira versão também.

Adicionalmente a estações de trabalho SGI o Geometry Center tinha algumas poucas estações NeXT completas, de forma que após Geomview estar sendo executado em máquinas SGIs a equipe desenvolveu uma versão para NeXTStep também. Nessa época ouveram muitos milhares de pessoas usando Geomview ao redor do mundo.

Alguns anos depois a equipe portou Geomview para o X windows e a OpenGL, e eventualmente, com o desaparecimento da NeXT, a versão para NeXT seguiu o mesmo caminho.

Nessa missão de incentivar a comunicação entre pesquisadores e educadores, o Geometry Center desenvolveu um web site, www.geom.umn.edu, mais tarde em 1993. Esse foi um dos primeiros 300 sítios na web que existiram no mundo. Uma parte desse sítio web era certamente devotado ao Geomview, e ajudou a propagar o trabalho durante sua existência.

O Geometry Center fechou suas instalações de "tijolos e cimento" em Agosto de 1998 (NSF cortou suas verbas), mas o sítio web continuou a existir, e o Geomview continuou a ser popular ao redor do mundo. Em Dezembro de 1999 algum membro da equipe original do Geometry Center configurou `http://www.geomview.org` como casa permanente na web para o Geomview.

Os autores originais do Geomview, bem como vários outros volunários ao redor do mundo, estão ainda ativamente envolvidos na utilização e no desenvolvimento do Geomview.

Autores

Tamara Munzner, Stuart Levy, e Mark Phillips são os autores originais do Geomview. Celeste Fowler, Charlie Gunn, e Nathaniel Thurston também fazem contribuições sgnificativas. Daniel Krech e Scott Wisdom fizeram o NeXTStep e a adaptação do RenderMan, e Daeron Meyer e Tim Rowley fizeram a adaptação para o X windows. Muitos outros membros do

estado maior do Geometry Center, bem como muitas pessoas em muitos lugares, também contribuíram.

Mark Phillips escreveu esse manual, com ajuda substancial de Stuart Levy e Tamara Munzner. Incontáveis usuários do Geomview também foram de grande ajuda por meio da leitura do manual e indicando nossos enganos.

Plantaformas Suportadas

Geomview 1.9 pode – em princípio – compilar e executar sobre quaisquer claramente recentes sistemas operacionais semelhantes ao Unix. Especificamente, Geomview executa sobre Linux e sobre Cygwin (Cygwin emula um ambiente semelhante ao SystemV Unix environment sob o Microsoft Windows). Desafortunadamente Geomview compila com MacOS X (Darwin), mas aparentemente comunicações com Geomview por meio de pipes e sockets causam falha de segmentacão. Sinta-se livre para consertar essa falha! Veja [Contribuindo], página 169, para detalhes.

Como Pronunciar “Geomview“

A palavra ’Geomview’ é uma combinação da primeira sílaba da palavra ’geometry’, e da palavra ’view’. Os autores pronunciam Geomview como uma palavra oxítona, isto é, tonicidade na primeira sílaba.

GE-om-view

Algumas pessoas colocam a tonicidade na segunda sílaba, onde Geomview cai na palavra ’geometry’, mas os autores originais, que criaram o nome, preferem a pronúncia com tonicidade na primeira sílaba.

1 Visão Geral

O principal objetivo do Geomview é mostrar objetos cuja geometria é fornecida, permitindo controle interativo sobre detalhes tais como ponto de visão, velocidade de movimento, aparência de superfícies e linhas, e assim por diante. Geomview pode manusear qualquer número de objetos e permite controle coletivo ou separado sobre eles.

A maneira mais simples de usar Geomview é como um visualizador independente para ver e controlar objetos. Geomview pode mostrar objetos descritos em uma variedade de formatos de arquivo. Geomview é acompanhado com uma larga variedade de objetos como exemplo, e você pode criar seus próprios objetos.

Você pode também usar Geomview para manusear os dados a serem mostrados provenientes de outro programa que está sendo executado simultâneamente. Como o outro programa modifica os dados, a imagem no Geomview reflete as modificações. Programas que geram objetos e utilizam o Geomview para mostrá-los são chamados *módulos externos*. Módulos externos podem controlar quase todos os aspectos do Geomview. A idéia aqui é que muitos aspectos de visualização e partes da interação de programas geométricos são independentes do conteúdo geométrico e podem ser coletados conjuntamente em uma peça simples de programa que pode ser usada em uma larga variedade de situações. O autor de um módulo externo pode concentrar-se sobre a implementação dos algorítmos desejados e deixar os aspectos de visualização ao Geomview. Geomview é acompanhado por uma coleção de módulos externos a título de exemplo, e esse manual descreve como escrever seu próprio *módulo externo*.

Geomview é o produto de um esforço no *Geometry Center* para disponibilizar um *software* de geometria interativa que é particularmente apropriado para pesquisa matemática e educação. Em particular, Geomview pode mostrar coisas no espaço hiperbólico e no espaço esférico bem como no espaço Euclidiano.

Geomview permite múltiplos objetos e câmeras que são controlados independentemente. As câmeras fornecem controle interativo para movimento, aparências (incluindo iluminação, sombreamento, e materiais), selecionando um objeto, aresta ou nível de vértice, instantâneos em um arquivo de imagem SGI ou no formato Renderman RIB, a adição ou apagamento de objetos é possível através do controle direto do mouse, painéis de controle, e teclas de atalho via teclado.

Geomview suporta os seguintes tipos de dados simples: poliedros com vértices compartilhados (.off), quadriláteros, malhas retangulares, vetores, e ajustes em superfícies de Bezier de grau arbitrário incluindo ajustes racionais. Hierarquías de objetos podem ser construídas com listas de objetos e instâncias de objeto(s) transformado(s) por uma ou mais matrizes 4x4. Porções arbitrárias de modificações de hierarquías podem ser transmitidas por meio da criação de referências nomeadas.

Geomview pode mostrar saídas gráficas tridimensionais provenientes do Mathematica e do Maple.

2 Tutorial

Esse capítulo conduzirá você através de alguns usos elementares do Geomview. Trabalhando do começo ao fim desse capítulo de frente a um computador onde você pode tentar acompanhar os exemplos fornecidos aqui você receberá um pouco do que pode ser feito com Geomview.

Para iniciar o Geomview, coloque seu usuário e sua senha no computador e abra uma janela de shell. Uma janela de shell é uma janela na qual você pode digitar comandos Unix; o prompt na janela usualmente termina com um '%'. Na janela de shell (o cursor do mouse deve estar posicionado sobre a janela) digite o seguinte (ENTER aqui significa pressione a tecla "Enter"):

```
geomview tetra dodec ENTER
```

Esse comando inicia o Geomview e chama dois objetos exemplo, um tetraedro e um dodecaedro. Após poucos segundos três janelas irão aparecer; veja Figura 2.1.

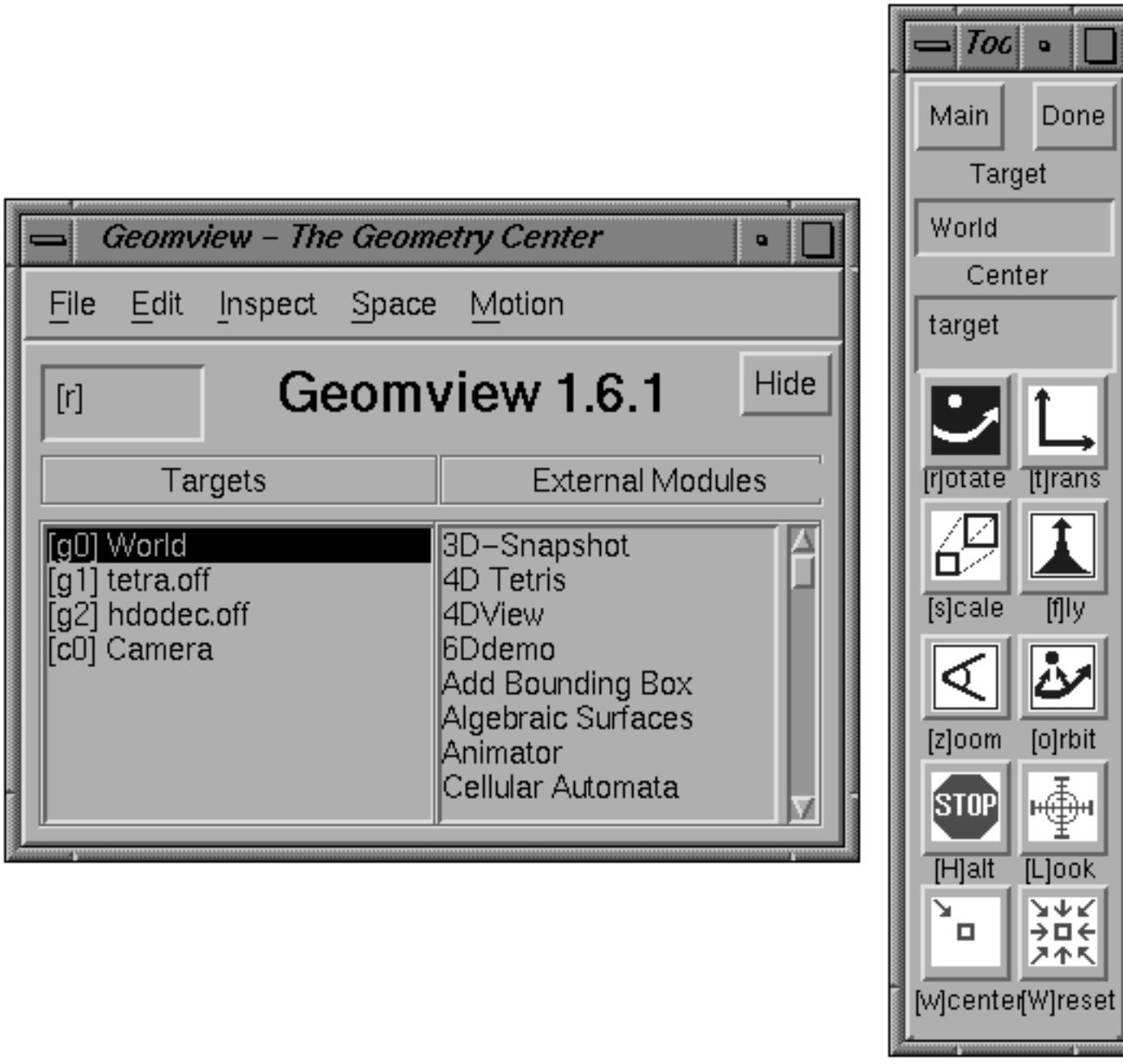


Figura 2.1: Tela Inicial do Geomview.

O painel à esquerda é o painel de controle principal do Geomview; Esse painel é chamado de painel *Main* (principal). O painel menor ao centro é o painel *Tools* (de ferramentas) e serve para selecionar diferentes tipos de movimentos. A janela do lado direito é a janela de

câmera e nessa janela você vê um tetraedro grande e um dodecaedro que está parcialmente obscurecido pelo tetraedro.

Geomview tem alguns painéis mas por padrão ele mostra somente esses três. Iremos descrever alguns aspectos desses três e alguns dos outros nesse tutorial. Você pode ler mais sobre esses e outros painéis nos capítulos adiante neste manual.

Coloque o cursor do mouse na janela de câmera e pressione e mantenha pressionado o botão esquerdo do mouse. Agora, enquanto mantém pressionado o botão, lentamente mova o mouse com movimentos pequenos. Você verá a figura rotacionar na direção na qual você mover o mouse. Se você liberar o botão do mouse enquanto move o mesmo, a figura continua girando. Para parar o movimento de rotação, mantenha o mouse sobre a figura e pressione rapidamente o botão esquerdo do mesmo.

Geomview utiliza o modelo da *esfera de vidro* para os movimentos iniciados através do mouse. Isso significa que você está supondo o objeto como estando dentro de uma esfera invisível e o cursor do mouse como sendo uma alça fora da esfera provida de uma ventosa. Quando você mantém pressionado o botão esquerdo do mouse, a ventosa da alça gruda na esfera; quando você libera o botão do mouse, a ventosa da alça libera a esfera. Movendo o mouse enquanto mantém pressionado o botão faz com que a esfera (e conceqüêntemente o objeto) mova-se na mesma direção que o mouse.

Adicionalmente para os dois sólidos que estão atualmente na tela você pode também ver duas molduras de fios em forma de caixa na janela de câmera. Essas são as "caixas associadas" dos dois objetos. Por padrão Geomview coloca uma caixa associada em torno de cada objeto que é mostrada de forma que você tenha uma idéia de o quanto grande o objeto é.

Note que quando você move o mouse em torno do tetraedro e do dodecaedro eles se movem como se fossem uma única figura. Isso ocorre porque por padrão o que você está movendo atualmente é o "World" (objeto mundo). Para mover um dos objetos individualmente em lugar de o objeto mundo como um todo, mova o cursor do mouse para o navegador de alvos (*Targets*) no painel principal (*Main*). Clique (qualquer botão) sobre a palavra *tetra*. Isso faz com que o tetraedro seja o "objeto alvo". Agora mova o cursor de volta à janela de câmera e você poderá rotacionar apenas o tetraedro.

O movimento que você aplicou até agora foi a rotação, porque esse é o modo de modo de movimento selecionado no painel de ferramentas (*Tools*). Para efetuar o movimento de translação em lugar do movimento de rotação, clique sobre o botão translação (*Translate*). Agora quando você mover o mouse na janela de câmera enquanto mantém pressionado o botão esquerdo, o tetraedro (que deve ser ainda o objeto alvo de antes) irá ser transladado na direção que você move o mouse. Note que você pode transladar o tetraedro na direção da borda da janela enquanto você mantém pressionado o botão esquerdo do mouse. Se você liberar o botão do mouse enquanto move o mesmo, o tetraedro irá continuar o movimento sozinho. O tetraedro mover-se-á ao contrário do que ocorria antes muito rapidamente de forma que é muito fácil perder o rastro de onde ele se encontra.

Se você acidentalmente perder o tetraedro através de translação para muito longe da janela de visão, você pode pegá-lo de volta através de um clique sobre o botão Centro (*Center*) no painel de ferramentas (*Tools*). Isso fará com que o tetraedro retorne para a sua posição inicial.

Clique sobre o botão Centro (*Center*) para trazer o tetraedro ao centro da janela de câmera, e então coloque-o em uma posição de forma que você possa ver completamente o dodecaedro.

Seu objeto mundo agora tem dois objetos que estão um ao lado do outro. Você pode ver o dodecaedro no meio da janela de câmera e pode ver parte do tetraedro parcialmente fora da janela de câmera. Volte para o navegador de alvos (*Targets*) no painel principal (*Main*) e clique sobre o "World" para selecionar o referido objeto mundo novamente. Agora clique sobre o botão "Olhar Para" (*Look At*) no painel de ferramentas (*Tools*). Você pode ver o dodecaedro e o tetraedro ajustando-se ao meio da janela (figura veja Figura 2.2). O botão "Olhar Para" (*Look At*) posiciona a câmera em uma posição tal que o objeto alvo fique centrado na janela.

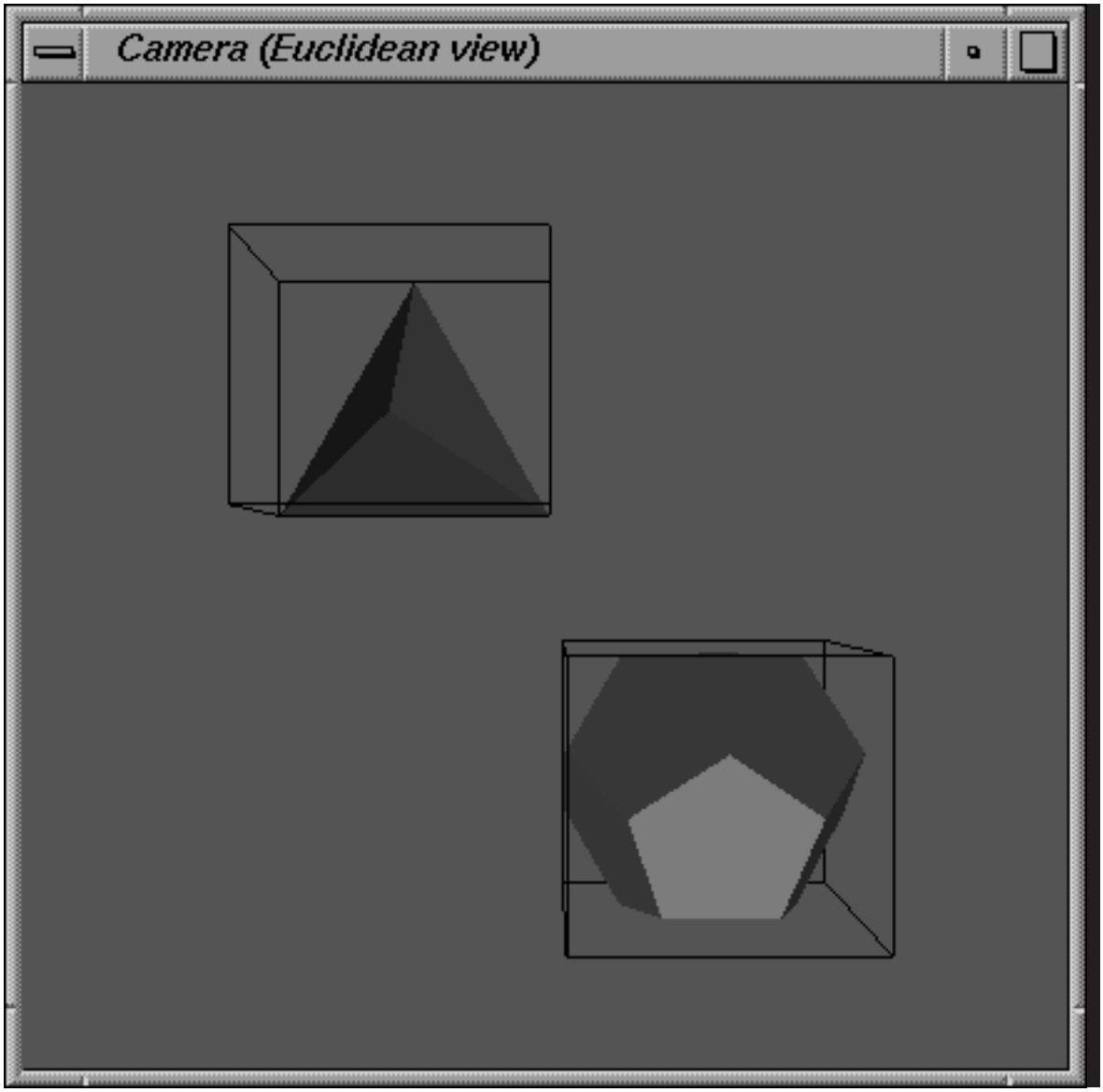


Figura 2.2: Olhando para o Objeto Mundo.

Agora coloque o cursor sobre o meio do dodecaedro e dê sobre ele um duplo clique com o botão direito do mouse. Isso significa clicar no mouse para baixo e para cima duas vezes em uma rápida sucessão. Note que o dodecaedro torna-se o objeto alvo; você pode ver isso no navegador de alvos (*Targets*) do painel principal (*Main*). Um duplo clique no botão direito do mouse sobre um objeto é outra forma de fazer esse objeto tornar-se o objeto alvo.

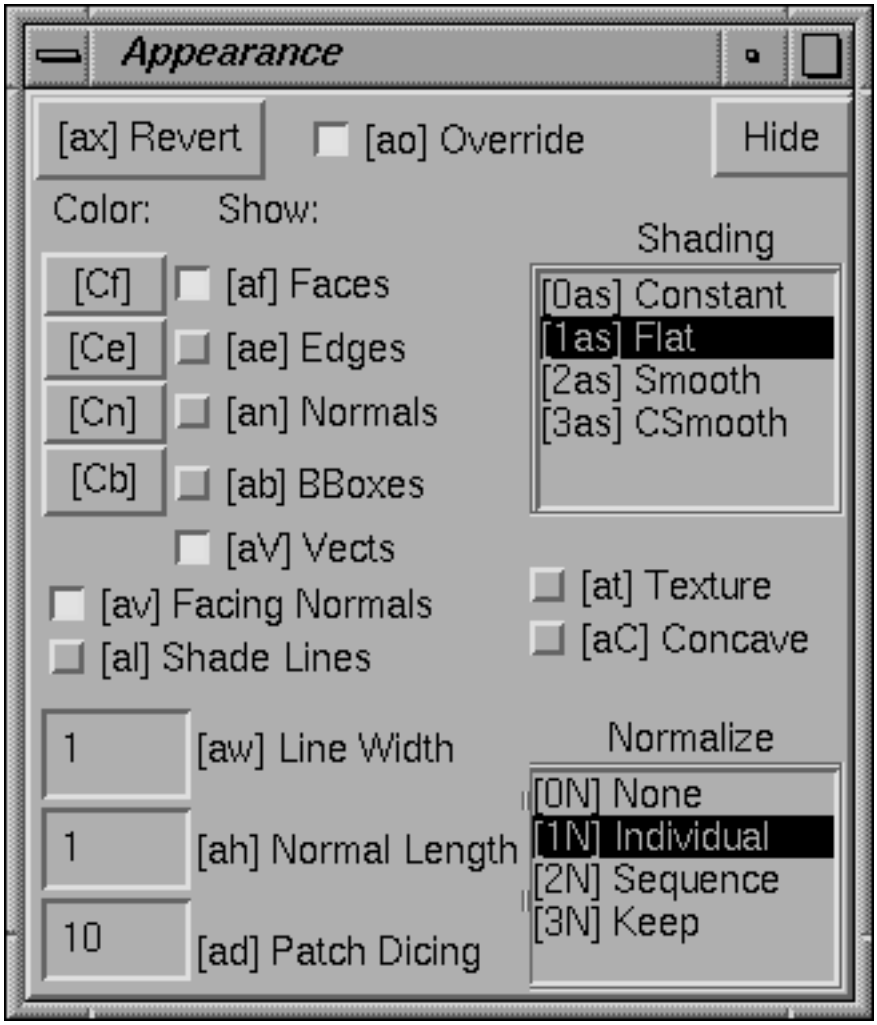


Figura 2.3: A Painel Aparência.

Vá para o menu *Inspect* no topo do painel principal (*Main*) e selecione Aparência (*Appearance*). Isso faz aparecer o painel "Aparência" (*Appearance*). Quando ele aparece, se estiver parcialmente obscurecido por outra janela do Geomview você pode movê-lo para um lado arrastando sua moldura com o botão do meio do mouse pressionado.

O painel Aparência (*Appearance*) permite a você controlar várias coisas sobre a maneira como o Geomview desenha objetos. Note os botões rotulados com *[af] Faces* e *[ae] Edges* (arestas). Clique sobre o *[ae] Edges* uma vez, e note que Geomview agora ressalta/destaca as arestas do dodecaedro. Clique sobre o *[ae] Edges* novamente e as arestas desaparecem. Clique muitas vezes e assista as arestas indo e voltando. Quando você tiver feito isso o suficiente, deixe as arestas habilitadas e clique sobre o botão *[af] Faces*. Essa ação alterna entre exibir ou não as faces. Clique sobre o botão novamente para de forma que a exibição das face fique habilitada.

Agora clique sobre o botão *[Cf] Faces* sob a palavra *COLOR*. Um painel de escolha de cores aparecerá (veja Figura 2.4).

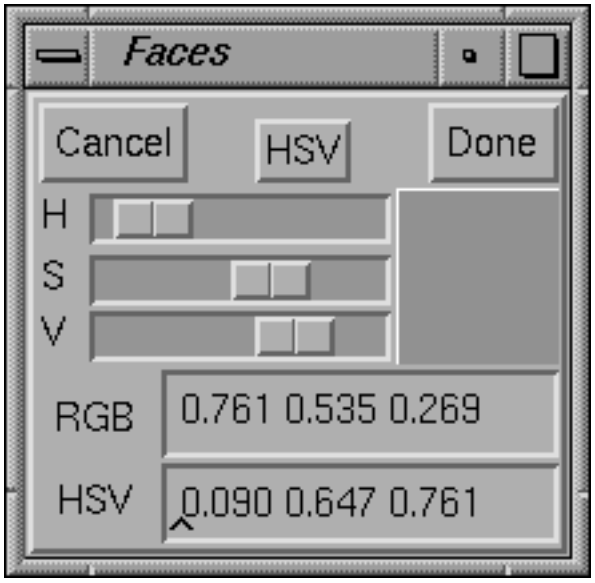


Figura 2.4: Painel de Escolha de Cores.

Note os três botões deslizantes, *H*, *S*, e *V*, controlando a matiz (*hue*), saturação, e valor (iluminação). Clicando sobre o botão *HSV* fornece um diferente conjunto de botões deslizantes, um para vermelho (*red*), outro para verde (*green*), e outro para azul (*blue*). Valores numéricos para ambos os sistemas de cores RGB e HSV podem ser vistos ou editados na parte inferior do painel. A cor inicial do dodecaedro foi especificada no arquivo '`dodec`' que você chamou quando iniciamos o Geomview. A cor que você especificou com o painel de cores sobrescreveu as cores antigas. Você pode ajustar a intensidade da cor com o botão deslizante *Intensity*. Quando você encontrar uma cor que você gosta, clique sobre o botão *Done*.

Agora coloque o cursor do mouse em algum lugar sobre sobre o fundo cinza da janela de câmera e duplo-clique no botão direito; isso seleciona "World" como objeto alvo. Clique no botão *Look At* para para olhar para o objeto mundo novamente.

Note que no painel de Aparência (*Appearance*) as escolhas dos botões se modificavam à medida que o lado esquerdo também mudava com o dodecaedro. Isso ocorre porque o painel *Appearance* sempre mostra as escolhas para o objeto alvo, que agora é o objeto mundo, o qual ainda tem suas escolhas padrão.

Clique sobre o botão *[ab] BBox* sob a palavra *Draw*. A caixa associada desaparece. agora ponha o cursor de volta na janela de câmera. No teclado, digite as teclas `a` b. Note que a caixa associada aparece novamente. `a` b é o atalho de teclado para o botão que alterna entre a exibição ou não da caixa associada; a sequência de caracteres "[ab]" aparece sobre o botão para indicar isso. A maioria dos botões do Geomview possuem atalhos de teclado que você pode usar se preferir. Isso será últil quando você estiver familiarizado com o Geomview e não quiser ter de se mover entre uma montanha de painéis.

Agora selecione o tetraedro, use qualquer das duas formas: duplo-clicando o botão direito do mouse sobre o tetraedro, ou selecionando "tetra" no navegador de alvos (*Targets*). Então clique sobre o botão *Delete* do menu *Edit* no painel principal (*Main*). O tetraedro deve desaparecer. Essa é a forma de você se livrar de um objeto.

Você pode também chamar objetos de dentro do Geomview. Clique sobre o menu *File* no painel principal (*Main*) e escolha abrir (*Open*). O painel de arquivos (*Files*) irá aparecer. Abaixo do meio desse painel, onde se lê *Path List*, temos um navegador com três linhas dentro dele; a segunda linha é um diretório com montanhas de exemplos do Geomview. Clique sobre aquela segunda linha; veja Figura 2.5. Role para baixo na lista de arquivos até você ver '`tref.off`'. Clique sobre aquela linha, e então clique sobre o botão *OK*. Um grande tubo em forma de trevo irá aparecer em sua janela. Clique sobre o botão *Hide* no painel *Files* para dispensar o painel.

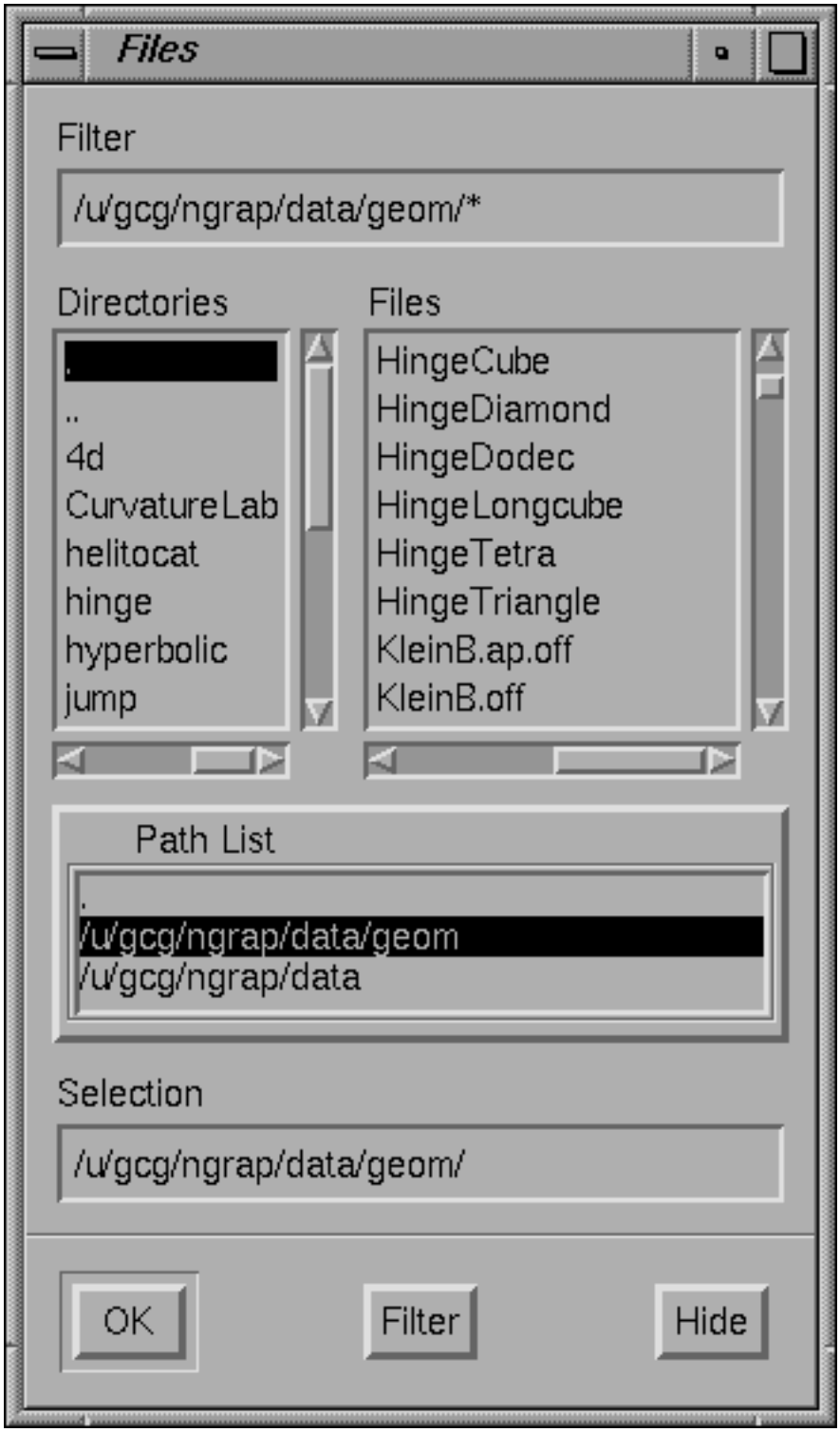


Figura 2.5: O Painel de Arquivos.

Agora clique sobre o botão *Reset* no painel de ferramentas (*Tools*). Isso fará com que todas as figuras retornem ao centro da janela de câmera. Você pode ver um dodecaedro e uma protuberância do trevo (veja Figura 2.6).

Brinque com a protuberância do trevo e o dodecaedro. Faça experiências com alguns outros botões no painel de ferramentas (*Tools*). Tente colorir o trevo com o painel de aparência (*Appearance*).

Para um tutorial sobre criar seus próprios objetos para chamá-los dentro do Geomview, veja '`doc/oogltour`' distribuido com Geomview. As coisas naquele arquivo irão ser incorporadas em futuras versões desse manual.



Figura 2.6: Trevo e Dodecaedro.

3 Interação

Esse capítulo descreve como você interage com Geomview através do mouse e do teclado.

3.1 Iniciando o Geomview

A forma usual para iniciar o Geomview é digitar *geomview ENTER* em uma janela de shell (ENTER significa pressionar a tecla "Enter"). Esse procedimento carrega o Geomview na memória do computador em uns poucos segundos; uma ou mais janelas irão aparecer e você pode começar a interagir com o Geomview imediatamente.

É também possível especificar ações para o Geomview executar no momento de iniciar fornecendo argumentos na linha de comando do shell. Veja seção 3.2 [Opcoes de Linha de Comando], página 31.

3.2 Opções de Linha de Comando

Aqui estão as opções de linha de comando que o Geomview permite:

‘`-b` *r g* `b`’ Escolhe a cor de fundo da janela de câmera para valores fornecidos de *r g* b.

‘`-c` *arquivo*’
: Interpreta os comandos GCL em *arquivo*, que pode ser o símbolo especial ‘`-`’ para a entrada padrão. Para uma descrição de GCL, veja Capítulo 7 [GCL], página 127.

‘`-c` *comando*’
: Comandos podem também serem fornecidos literalmente, como em

```
-c "(ui-panel main off)"
```

Uma vez que *comando* inclui parêntesis, que possuem significado especial para o shell, *comando* deve receber apóstrofo. Multiplas opções *-c* são permitidas.

‘`-wins` *n*’ Faz com que Geomview mostre inicialmente *n* janelas de câmera.

‘`-wpos` *largura*`,`*altura*`[@`*xmin*`,`*ymin*`]`’
: Especifica a localização inicial e o tamanho da primeira janela de câmera. Os valores para *largura*, *altura*, *xmin*, e *ymin* estão em coordenadas de tela (pixel).

‘`-M[cg][ps[un|in|in6]]` *PIPENOME*`|`*TCPPORT*’
: A opção ‘`-M`’ aceita modificadores: um sufixo ‘`g`’ espera dados geométricos (o padrão), enquanto um sufixo ‘`c`’ espera comandos GCL. Um ‘`p`’ implica que a conecção pode usar um pipe nomeado (o padrão para tudo exceto para "NeXT"), enquanto ‘`s`’ implica no uso de um "UNIX-domain socket" (o padrão em "NeXT"). Uma vez que na versão 1.9 do Geomview "Internet domain sockets" são também suportados; use ‘`sin`’ para fazer o Geomview escutar uma porta IPv4 fornecida por *TCPPORT*, ou use ‘`sin6`’ para fazer Geomview escute uma porta IPv6 (também como especificado em *TCPPORT*). ‘`sun`’ é um sinônimo para ‘`s`’, i.e. use o "Unix domain socket" com o nome *PIPENOME*. Se *PIPENOME* inicia com uma barra (’/’), então esse nome é assumido ser um caminho absoluto, de outra forma o pipe nomeado ou socket é criado sob o diretório ‘`${TMPDIR}/geomview/`’.

Escutando fluxo de comando em portas TCP pode ser um risco de segurança, como Geomview por si mesmo não toma nenhum tipo de precaução de segurança, Geomview simplesmente executa todos os comandos alimentados a ele através do socket de rede. Isso também implica entrada e saída para unidades de armazenamento locais devem ser permitidas remotamente.

Exemplos:

`-M` *`nome_de_objeto`*
: Mostra (possivelmente mudando dinamicamente) geometria enviada de programas `geomstuff` ou `togeomview`. Essa opção "-M" escuta o pipe nomeado '`/tmp/geomview/nome_de_objeto`'; você pode conseguir o mesmo efeito com os comandos de shell abaixo:

```
mkdir /tmp/geomview
mknod /tmp/geomview/nome_de_objeto p
```

(assumindo que o diretório e o pipe nomeado não existam atualmente), então executando o comando GCL:

```
(geometry nome_de_objeto  <
/tmp/geomview/nome_de_objeto)
```

(veja seção 7.2.57 [geometry], página 137)

`-Mc` *`pipenome`*
: Como '`-M`' acima, mas espera comandos GCL, em lugar de dados geométricos OOGL, na conecção.

`-Mcs nome` Lê comandos a partir do "UNIX-domain socket" nomeado. '`/tmp/geomview/nome`'

`-Mcsin 40000`
: Lê comandos a partir da porta IPv4 '`40000`'. Geomview por si mesmo não toma qualquer precaução de segurança, de forma que "-Mcsin 40000" pode ser um risco de segurança.

'`-noopengl`'
: Desabilita o uso de OpenGL para (possivelmente) conversão acelerada de hardware, mesmo que o binário do Geomview tenha suporte a OpenGL compilado internamente. "-noopengl" também desabilita o suporte a transparência e texturas na janelas de câmera. Instantâneos "RenderMan" ainda terão a transparência correta e suporte a alguma textura limitada.

'`-nopanels`'
: Inicia sem mostrar nenhum painel, somente a janelas gráficas. Painéis podem ser invocados mais tarde da forma usual com as teclas de atalho *`Px`* ou com comando `ui-panel`. Veja seção 7.2.149 [ui-panel], página 157.

'`-noinit`' Não lê nenhum arquivo de inicialização. Por padrão, Geomview lê o arquivo '`.geomview`' do sistema, seguido daqueles em '`${HOME}/.geomview`' e em '`./.geomview`'.

‘`-e modulo`’
: Inicial um módulo externo; *modulo* é o nome associado ao módulo chamado, aparecendo no painel principal no navegador de "Applications", como definido pelo comando `emodule-define`. Veja seção 7.2.40 [emodule-define], página 134.

‘`-start module args ...`’
: Como -e mas permite a você enviar argumentos para o módulo externo. "–" sinaliza o fim da lista de argumentos; o "–" pode ser omitido se for o último argumento na linha de comando do Geomview.

‘`-run coamando-shell args ...`’
: Como "-start" mas toma o caminho de arquivos do executável do módulo externo em lugar do nome do módulo. Os caminhos de arquivo de todos os diretórios de módulos conhecidos são anexados ao final do caminho de busca do UNIX quando for invocado o *comando-shell*.

3.3 Interacao Basica: O Painel Principal

Normalmente quando você invoca Geomview, três janelas aparecem: O painel principal (*Main*), o painel de ferramentas (*Tools*), e uma janela de câmera. Geomview tem muitas outras janelas mas muitas coisas podem ser realizadas com essas três de forma que por padrão as outras não aparecem. Essa seção do manual introduz alguns conceitos básicos que são usados nas seções restantes do manual e descreve o painel principal (*Main*).

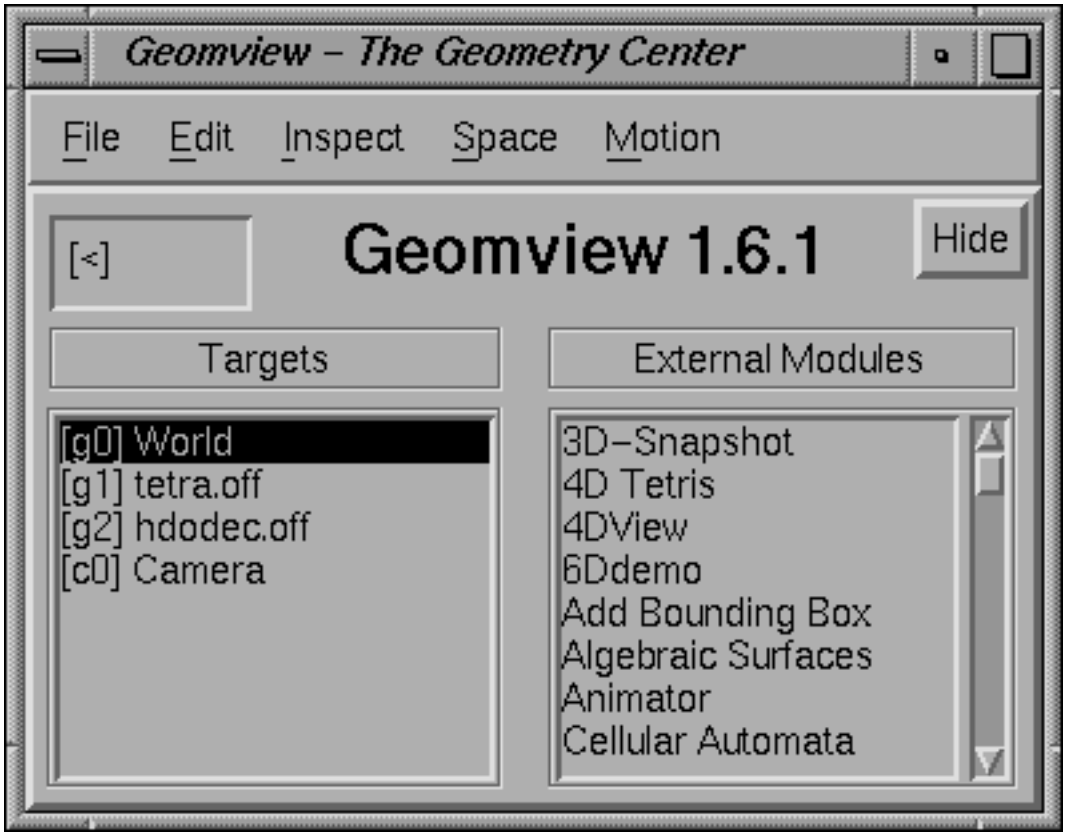


Figura 3.1: O Painel Principal

Geomview pode mostrar um número arbitrário de objetos simultaneamente. O navegador *Targets* no painel principal (*Main*) mostra uma lista de todos os objetos dos quais Geomview atualmente abertos. Esse navegador tem uma linha para cada objeto que você tiver chamado, adicionalmente algumas linhas para outros objetos. Um desses outros objetos é chamado `World` e corresponde a todos os objetos atualmente chamados, tratados como se eles fossem um objeto. A maioria das operações que você pode fazer sobre um objeto, tais como aplicar um movimento ou mudar uma cor, pode também ser feita para o objeto "World".

O navegador de alvos (*Targets*) também possui uma entrada para cada câmera. Por padrão existe somente uma câmera; é possível adicionar mais delas através da entrada *New Camera* do painel principal (*Main*) via menu *File*. Geomview trata câmeras na maioria das vezes como trata objetos geométricos. Por exemplo, você pode mover câmeras pelas proximidades e adicioná-las e apagá-las como objetos geométricos. Câmeras não são mostradas na tela como um objeto que você vê. Cada câmera tem uma janela de câmera separada que mostra a visão como vista através da lente daquela câmera. (É possível para cada câmera mostrar uma representação geométrica de outras câmeras. Veja seção 3.7 [Cameras], página 52.)

Devido ao fato de Geomview tratar câmeras e objetos geométricos muito similarmente, o termo *objeto* nessa documentação é usado para referir-se a qualquer dos dois indistintamente. Quando precisamos distinguir entre os dois tipos de objetos, usamos o termo "*geom*" para denotar um objeto geométrico e a palavra *câmera* para denotar uma câmera.

O objeto que está selecionado (luminosidade alta) no navegador "*Targets*" é chamado objeto alvo. Esse é o objeto que recebe quaisquer ações que você faz com o mouse ou com o teclado. Você pode mudar o objeto alvo selecionando uma linha diferente no navegador de alvos (*Targets*). Outro caminho de modificar o objeto alvo é colocar o cursor do mouse diretamente sobre um geom na janela de câmera e rapidamente dar um duplo clique no botão direito do mouse. Esse processo é chamado *selecionar*; o objeto selecionado torna-se o novo alvo.

Objetos do Geomview são todos conhecidos por dois nomes, ambos dos quais são mostrados no navegador de alvos (*Targets*). O primeiro nome lá fornecido, que aparece entre colchêtes ([ ]), é um nome curto atribuido pelo Geomview quando você chama o objeto. Esse nome consiste da letra 'g' para geometria e da letra 'c' para câmeras, seguindo por um número. O segundo nome é maior e mais descritivo; por padrão esse é o nome do arquivo do qual o objeto foi chamado. Os dois nomes são equivalentes no que diz respeito ao Geomview; em qualquer ponto onde você precisar especificar um nome você pode fornecer qualquer dos dois.

Para controlar um objeto, garanta que aquele objeto que você quer mover seja o objeto alvo, e coloque o cursor do mouse em uma janela de câmera. Movimentos são aplicados pressionando ou o botão esquerdo ou o botão do meio do mouse e movendo o mouse. Existem muitos modos de movimento diferentes, cada modo de movimento aplicando um diferente tipo de movimento. O navegador de modos de movimento (*MOTION MODE*) no painel principal indica o modo de movimento atual. O padrão é a rotação ("Rotate"). Você pode mudar o modo corrente de movimento selecionando um novo modo de movimento no navegador de modos de movimento (*MOTION MODE*), ou usando o painel de ferramentas (*Tools*). Para maiores informações sobre modos de movimento, veja seção 3.5 [Movimentos do Mouse], página 39.

O navegador de módulos (*Modules*) lista módulos externos do Geomview. Um módulo externo é um programa separado que interage com Geomview para extender suas funcionalidades. Para informações sobre módulos externos, veja Capítulo 6 [Modulos], página 104.

A barra de menu no topo do painel principal oferece menus para operações comuns.

Para criar novas janelas, chame novos objetos, grave os objetos ou outras informações, ou saia do Geomview, veja o menu *File*.

Para copiar ou apagar objetos, veja o menu *Edit*.

Você pode chamar qualquer painel a partir do menu *Inspect*.

O menu *Space* permite a você escolher se Geomview trabalha no modo Euclidiano, Hiperbolico ou Esférico. O modo Euclideano é usado por padrão. Para detalhes sobre a utilização do modo espaço *Hyperbolic* e do modo *Spherical*, veja Capítulo 8 [Geometrias Nao-Euclidianas], página 161.

A maioria das ações que você pode fazer através dos painéis do Geomview possuem equivalentes atalhos de teclado de forma que você pode fazer a mesma ação atavés de digitação de uma sequência de teclas no teclado. Isso é útil para usuários avançados que estão familiarizados com as capacidades do Geomview e querem trabalhar rapidamente

sem ter montanhas de painéis amontoando-se na tela. Atalhos de teclado são usualmente indicados ente colchêtes ([ ]) próximo ao item correspondente em um painel. Por exemplo, o atalho de teclado para o modo *Rotate* é 'r'; isso é indicado par "[r]" que aparece antes da palavra "Rotate" no navegador *MOTION MODE*. Para usar esse atalaho de teclado, apenas pressione a tecla `r` enquanto o cursor do mouse estiver em qualquer janela do Geomview. Não é necessário pressionar a tecla ENTER posteriormente.

Alguns atalhos de teclado consistem em mais de uma tecla. Nesses caso apenas digite as teclas uma após a outra, sem pressionar ENTER posteriormente ou entre as teclas pressionadas. Atalhos de teclado são sensíveis à caixa alta/baixa.

Muitas teclas de atalho podem ser precedidas de um parâmetro numérico. Por exemplo, digitando `ae` muda o estado do desenho de arestas, enquanto `1ae` sempre habilita o desenho de arestas.

O campo *keyboard* no canto superior esquerdo do painel principal (*Main*), imediatamente acima da palavra "Targets", ecoa o estado atual das teclas de atalho.

Para uma lista de todas as teclas de atalho, pressione a telca `?`.

3.4 Disponibilizando Objetos dentro do Geomview

Existem muitos caminhos para chamar um objeto dentro do Geomview.

No painel de arquivos (*Files*)
: Se você clicar no botão *Load* no painel principal do Geomview (*Main*), o painel de arquivos (*Files*) irá aparecer.



Figura 3.2: O Painel de Arquivos.

Esse painel permite que você selecione um arquivo a partir de uma variedade de diretórios. O topo do painel é um navegador de arquivos padrão do Motif. Abaixo deste está uma lista de diretórios no caminho de busca padrão do Geomview; clique sobre um desses para navegar entre os arquivos naquele diretório.

Para selecionar um arquivo, duplo-clique sobre o seu nome no navegador no canto superior direito, ou clique sobre o seu nome e pressione a tecla ENTER, ou ainda digite o nome do arquivo dentro da caixa de texto na parte inferior do navegador e pressione a tecla ENTER.

Se o arquivo selecionado contiver dados geométricos OOGL, esse arquivo irá ser adicionado ao navegador de alvos (*Targets*) do Geomview. Se esse arquivo contiver comandos GCL em lugar de conter dados geométricos OOGL, o arquivo será interpretado. Veja Capítulo 7 [GCL], página 127.

Quando o apinel de arquivos (*Files*) aparecer pela primeira vez, o diretório selecionado no navegador de diretórios é o diretório atual — que corresponde ao diretório a partir do qual você chamou o Geomview. O navegador de arquivos mostra *todos* os arquivos nesse diretório, incluindo os que não são arquivos do Geomview. Se você tentar chamar um arquivo que não contenha um objeto OOGL e também não contenha comandos do Geomview, o Geomview irá mostrar uma mensagem de erro.

O navegador de diretórios também lista um segundo e um terceiro diretórios adicionalmente além do diretório atual. O segundo, que termina em '`data/geom`', é o diretório de exemplos de dados do Geomview. Esse diretório contém uma grande variedade de amostras de objetos. Esse diretório também contém muitos subdiretórios. Em particular, os subdiretórios '`hyperbolic`' e o subdiretório '`spherical`' possuem amostras de objetos hiperbólicos e esféricos, respectivamente. Entradas no navegador de diretórios são vistas apenas como entradas de arquivos; para visualizar um subdiretório, clique sobre o nome do referido diretório.

O terceiro diretório mostrado no navegador de diretório, que termina em '`geom`', contém muitos subdiretórios com outros arquivos do Geomview dentro deles. Esses arquivos são usados menos frequêntemente que os outros no diretório '`data/geom`'.

Você pode mudar a lista de diretórios mostrada no navegador de diretórios do painel de arquivos (*Files*) usando o comando `set-load-path`; Veja seção 7.2.124 [set-load-path], página 151.

A tecla de atalho <:

Se você digitar < em qualquer janela do Geomview, o painel *Load* irá aparecer. Esse painel é uma pequena versão do painel de arquivos (*Files*); o painel *Load* contém um campo de texto no qual você o nome de um arquivo a ser chamado (ou um comando GCL entre parêntesis). Após digitar o nome do arquivo a ser chamado, aperte a tecla ENTER; Geomview irá chamar o arquivo como se você o tivesse chamado com o botão *Add* no painel de arquivos (*Files*). Se, após fazer surgir o pequeno painel *Load* com <, você decidir que quer usar o grande painel de arquivos (*Files*) após tudo, pressione o botão *File Browser*.

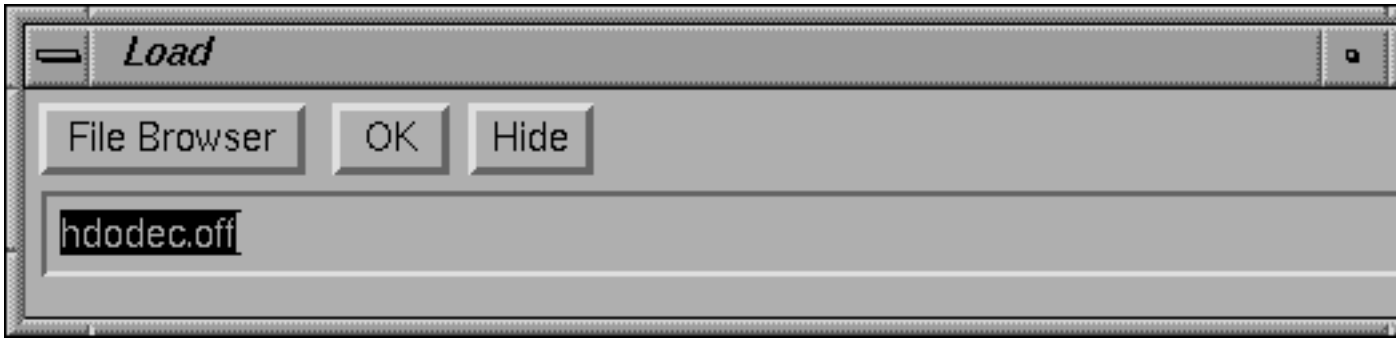


Figura 3.3: O Painel de Chamar Arquivos.

Comandos para chamar objetos geométricos:

Os comandos GCL `load`, `geometry`, `new-geometry`, e `read` permitem a você chamar um objeto dentro do Geomview; veja Capítulo 7 [GCL], página 127. Veja seção 7.2.71 [load], página 141. Veja seção 7.2.93 [new-geometry], página 145. Veja seção 7.2.111 [read], página 149.

3.5 Usando o Mouse para Controlar Objetos

Geomview permite a você controlar objetos com o mouse. Existem seis diferentes modos de movimento do mouse: rotação (*Rotate*), translação (*Translate*), vôo da câmera (*Cam Fly*), zoom da câmera (*Cam Zoom*), homotetia de um objeto geométrico (*Geom Scale*), e órbita de câmera (*Cam Orbit*). O painel de ferramentas tem um botão para cada um desses modos; para trocar os modos, clique sobre o botão correspondente. Você pode também selecionar um novo modo através do navegador de modos de movimento (*Motion Mode*) no painel principal (*Main*).

Essa seção descreve a interação básica com o mouse. Para detalhes, veja seção 3.9 [Comandos], página 60.

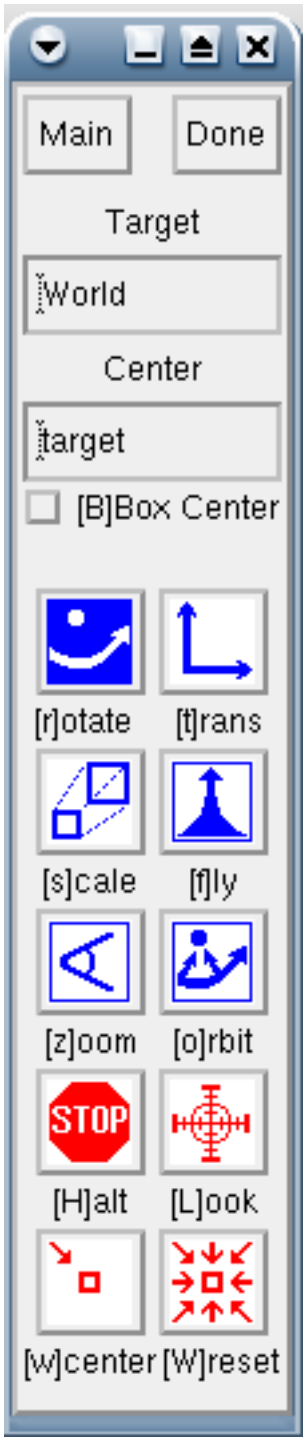


Figura 3.4: O Painel de Ferramentas.

Cada um dos modos de movimento usa um paradigma comum para como o movimento é aplicado. Em particular, cada modo de movimento depende do objeto alvo (*target*) atual e do atual objeto do centro (*center*). O objeto alvo atual e o atual objeto do centro são explicados nos parágrafos seguintes.

O objeto alvo atual é mostrado no campo *Target* no painel de ferramentas (*Tools*). Isso é o mesmo que o objeto selecionado no navegador de alvos (*Targets*) no painel principal (*Main*), e você pode mudar o alvo ou selecionando um novo objeto no navegador, digitando uma nova entrada no campo, ou selecionando um objeto na janela de câmera duplo-clicando no botão direito do mouse com o cursor sobre o objeto.

O atual objeto do centro é mostrado no campo *Center* no painel de ferramentas(*Tools*). Seu valor padrão é a palavra especial `"target"`, que significa que o objeto do centro é o objeto que estiver designado como objeto alvo. Você pode mudar o objeto atual do centro para qualquer objeto digitando seu nome no campo *Center*. A origem do objeto do centro é mantida fixa no modo rotação *Rotate* e no modo *Orbit*. Normalmente o objeto do centro é um dos objetos geométricos (geoms) existentes listados no navegador de alvos (*Targets*), o centro atual das rotações é a origem daquele sistema de coordenadas daquele objeto. É possível, todavia, selecionar um ponto arbitrário de interesse sobre um objeto como o centro. Para detalhes, veja seção 3.5.1 [Ponto de Interesse], página 44.

Isso também é possível mudando o botão *BBox Center* para escolher o centro de movimento como sendo o centro do objeto atual da caixa associada. Uma vez modificado o centro da caixa geométrica ativa associada irá tornar-se o centro do movimento, se você selecionar outro objeto, então o centro do movimento irá tornar-se o centro da caixa associada à aquele objeto. Nenhuma modificação ocorrerá quando uma câmera ou o objeto mundo (*World*) for selecionado; você tem que digitar a palavra `target` no campo *Center* para retornar ao valor padrão.

Você aplica um movimento de mouse pressionando ou o botão esquerdo ou o botão do meio do mouse com o cursor em uma janela de câmera e movendo o mouse. A maioria dos modos de movimento possui inércia (*inertia*), que significa que se você soltar o botão enquanto move o mouse, o movimento irá continuar. Para imaginar a inércia pode ser útil imaginar o cursor do mouse como sendo uma alça; quando você pressiona um botão do mouse para baixo, o mouse agarra firmemente no objeto alvo e você pode mover esse objeto. Quando você libera o botão do mouse, a alça libera o objeto. Liberando o botão do mouse enquanto move o mesmo funciona como abandonar o objeto — o objeto continua movendo-se independentemente do mouse. Inércia pode ser desligada; veja o menu de movimento (*Motion*) no painel principal (*Main*), descrito abaixo.

Geralmente, o botão esquerdo do mouse controla movimento no plano da tela, enquanto o botão médio do mouse controla movimento ao longo ou em torno da direção de avanço.

Pressionando o tecla `"shift"` enquanto arrasta com o botão esquerdo ou médio do mouse na maioria dos modos de movimento fornece movimentos de baixa velocidade , útil para ajustes finos.

Você pode selecionar qualquer ponto sobre um objeto (não apenas sua origem) como centro do movimento pressionando a tecla `"shift"` enquanto clica no botão direito do mouse; isso escolhe o ponto de interesse.

Rotate No modo rotação (*Rotate*), pressione o botão esquerdo do mouse para rotacionar o objeto alvo en torno do objeto do centro. A rotação ocorre na direção que

você move o mouse. Especificamente, o eixo de rotação passa através da orígem do objeto do centro, é paralelo ao plano de visão da câmera, e é perpendicular à direção do movimento do mouse. Quando o centro for o alvo ("target"), isso significa que o objeto alvo rotaciona em torno de sua própria orígem.

O botã do meio do mouse no modo de movimento tipo rotação (*Rotate*) rotaciona o objeto alvo em torno de um eixo perpendicular ao plano de visão.

Translate No modo translação (*Translate*), mantenha pressionado o botão esquerdo do mouse para transladar o objeto alvo na direção do movimento do mouse. O botão do meio do mouse translada o alvo ao longo de um eixo perpendicular ao plano de visualização.

No espaço Euclideano, o objeto do centro é essencialmente irrelevante para translações. Nos espaços hiperbólicos e esféricos, onde translações possuem um único eixo, esse eixo é escolhido para ir através da orígem do objeto do centro.

Cam Fly O Vôo de Câmera (*Cam Fly*) é um simulador de vôo muito simples que permite a você voar em torno da cena. *Cam Fly* trabalha através do movimento da câmera. Movimente o mouse enquanto mantém pressionado o botão esquerdo do mouse para posicionar a câmera em uma direção diferente. Para mover adiante ou para trás, mantenha pressionado o botã do meio e mova o mouse verticalmente. Os dois movimentos aqui descritos possuem inércia; tipicamente o caminho mais fácil para voar em torno de uma cena é fornecer a câmera um passo adiante pressionando o botã do meio enquanto move-se o mouse para cima, e então usar o botão esquerdo para pilotar.

Cam Fly afeta a janela de câmera onde o mouse está correntemente posicionado; *Cam Fly* ignora o objeto alvo e o objeto do centro.

Cam Orbit

O modo órbita de Câmera (*Cam Orbit*) permite a você rotacionar a câmera atual em torno do centro atual. O botão esquerdo do mouse faz essa rotação. O botão do do meio do mouse no modo *Cam Orbit* atua da mesma forma que no modo *Cam Fly*: O botão do do meio do mouse move a câmera para adiante e para trás.

Em geral *Cam Orbit* não move o objeto alvo, embora se a câmera atual for selecionada como o alvo e o centro for também o alvo, *Cam Orbit* irá apenas pivotar aquela câmera sobre sí mesma como no modo *Cam Fly*.

Cam Zoom

O modo Zoom de Câmera (*Cam Zoom*) permite a você modificar o campo atual de visão com o mouse; mantenha pressionado o botão esquerdo do mouse e mova o mouse para modificar o campo de visão. O valor numérico do campo de visão é mostrado no campo *FOV* (field of view) no painel de câmera (*Camera*).

Geom Scale

O modo *Geom Scale* permite a você ampliar ou diminuir um objeto geométrico (geom). *Geom Scale* atua sobre o objeto alvo se aquele objeto for um geom. Se o alvo for uma câmera, *Geom Scale* atual sobre o geom que foi o objeto alvo mais recentemente. Movendo o mouse enquanto mantem-se pressionado o botão esquerdo do mouse homotetiza-se o objeto ou ampliando ou reduzindo o mesmo,

dependendo da direção do movimento do mouse. o centro da transformação homotética aplicada é o objeto do centro.

Homotetia possui significado somente no espaço Euclideano; tentativas do palicar homotetia são ignoradas em outros espaços.

O modo *Geom Scale* não possui inércia.

Os botões *Stop*, *Look At*, *Center*, e *Reset* no painel de ferramentas (*Tools*) executam ações relacionadas a movimentos mas não modificam o modo atual de movimento (nota do tradutor: de rotação para translação por exemplo).

Stop O botão *Stop* faz com que cessem todos os movimentos. O botão *Stop* afeta todos os objetos em movimento, não apenas o objeto alvo. Sua tecla de atalho é `H`.

O comando de teclado `h`, que não corresponde a um botão do painel, cessa o movimento atual para o objeto alvo somente.

Look At O botão *Look At* faz com que a câmera atual seja movida para uma posição tal que a referida câmera esteja olhando para o objeto alvo, e de forma que o objeto alvo mais ou menos ajuste-se à janela.

O comando *Look At* não funciona perfeitamente em espaços não Euclideanos.

Center O botão *Center* desfaz transformação do objeto alvo, movendo o objeto alvo de volta à sua posição inicial padrão, que é onde ele estava quando você originalmente o chamou a partir do Geomview.

Reset O botão *Reset* cessa todo movimento e faz com que todos os objetos sejam movidos de vota às suas posições iniciais padrão.

O painel de ferramentas (*Tools*) possui um botão *Main*, para invocar o painel principal no caso de esse painel ter sido dispensado ou sepultado, e um botão *Done* para fechar o painel de ferramentas *Tools*.

O painel principal do menu de movimento (*Motion*) tem controles especiais que afetam como movimentos do mouse são interpretados; as modificações são também acessíveis através de comandos GCL. Veja seção 7.2.148 [ui-motion], página 156.

[ui] Inertia

Normalmente, ao mover objetos tem-se inércia: se o mouse estiver ainda se movendo quando o botão for liberado, o objeto selecionado continua a mover-se. Quando a inércia (*Inertia*) for desabilitada, objetos cessam seu movimento no momento em que você libera o mouse.

[uc] Constrain Motion

É necessário algumas vezes ter ao alcance da mão o movimento de um objeto em uma direção alinhada com um eixo coordenado: exatamente na horizontal ou na vertical. Selecionando restringir movimento (*Constrain Motion*) a interpretação de movimentos do mouse é modificada para permitir isso; arrastros de mouse aproximadamente horizontais ou aproximadamente verticais transformam-se em movimentos exatamente horizontais ou exatamente verticais. Note que o movimento é ainda ao longo dos eixos X ou Y da câmera na qual você move o mouse, não necessáriamente no sistema de coordenadas do objeto.

[uo] Own Coordinates

É necessário algumas vezes ter ao alcance da mão o movimento de objetos objetos com relação ao sistema de coordenadas onde o referido objeto foi definido, em lugar de com relação ao sistema de coordenadas da câmera através da qual esse objeto está sendo visto. Enquanto *Own Coordinates* estiver selecionado, todos os movimentos são interpretados da forma citada nesse item: arrastando o mouse para a direita no modo translação corresponde a mover o objeto em sua própria direção +X, e assim por diante. Pode ser especialmente útil conjuntamente com o botão Restringir Movimento (*Constrain Motion*).

3.5.1 Selecionando um Ponto de Interesse

É algumas vezes útil especificar um ponto em particular sobre algum objeto em uma janela do Geomview como o ponto de centro para movimentos do mouse. Você pode fazer isso segurando a tecla shift e clicando o botão direito do mouse (i.e. clique no botão direito uma vez enquanto mantém pressionada a tecla shift do teclado) com o cursor sobre o ponto desejado. Esse ponto torna-se então o *ponto de interesse*. O ponto de interesse deve estar sobre um objeto existente.

Selecionado um ponto de interesse simplifica o exame de uma pequena porção de um grande objeto. Mantendo a tecla shift pressionada e clicando sobre o ponto de interesse com o botão direito do mouse, e selecionando o modo órbita (*Orbit*). Use o botã do meio do mouse para aproximar, e o esquerdo para orbitar o ponto, examinando a região de diferentes direções.

Quando você tiver selecionado um ponto de interesse, o atual objeto do centro é modificado para um objeto chamado "CENTER", que é um objeto invisível localizado no ponto de interesse. Adicionalmente, movimentos de mouse para a janela na qual você fez a seleção são ajustados de forma que o ponto de interesse acompanhe o mouse.

Você pode mudar o ponto de interesse a qualquer momento selecionando um novo ponto de interesse shift-clicando o botão direito do mouse novamente. Você pode cancelar o ponto de interesse completamente shift-clicando o botão direito do mouse com o cursor no plano de fundo (i.e. não sobre qualquer objeto). Isso modifica o objeto do centro de volta a seu valor padrão, "target".

O objeto que possui o nome de "CENTER", que serve como um objeto do centro para o ponto de interesse, é um tipo especial de geom chamado "alien". Esse "Alien" não aparece no navegador de alvos (*Targets*). Por padrão esse objeto "Alien" não tem geometria associada e concequêntemente é invisível. Você pode, Todavia, explicitamente fornecer ao Alien alguma geometria usando um comando GCL, fazendo com que o Alien aparece no navegador de alvos. Use o comando `geometry` para fazer isso: `(geometry CENTER` *`geometry`*`)`, onde *geometry* é qualquer geometria válida. Por exemplo, `(geometry CENTER { < xyz.vect })` faz com que o arquivo '`xyz.vect`', que é um dos arquivos de exemplo padronizados distribuídos com Geomview, seja usado na geometria para CENTER. Veja seção 7.2.57 [geometry], página 137.

O que acontece internamente quando você seleciona um ponto de interesse é que o centro é ajustado para o objeto chamado CENTER, e aquele objeto é posicionado no ponto de interesse. Adicionalmente, para que movimentos de mouse acompanhem o ponto de interesse, o atual comprimento focal da câmera é escolhido para ser a distância da câmera até o ponto de interesse. Você pode realizar isso via GCL com os seguintes comandos:

```
(if (real-id CENTER) nil (new-alien CENTER {}))
(ui-center CENTER)
(transform-set CENTER universe universe translate x y z)
(merge camera cam-id { focus d })
```

onde `(x,y,z)` são as coordenadas (universe) do ponto de interesse, e *d* é a distância daquele ponto à atual câmera, *cam-id*. O primeiro comando acima cria o "alien" CENTER se esse Alien não existir ainda.

3.6 Modificando a Forma de Ver as Coisas

Geomview utiliza uma hierarquia de aparências para controlar o caminho através do qual olha-se coisas. Uma aparência (*appearance*) é uma especificação de informação sobre como alguma coisa pode ser desenhada. Isso pode incluir muitas características como cor, brilho, propriedades do material, etc. Aparências trabalham de uma maneira hierárquica: se uma certa propriedade de aparência, por exemplo cor de face, não for especificada em uma aparência particular de algum objeto, esse bojeto é desenhado usando aquele propriedade de uma hierarquia superior. Se ambas as aparências de hierarquias, a atual e a superior, especificam uma propriedade, a hierarquia atual tem precedência a menos que a hierarquia superior não for escolhida para sobrescrevê-la.

Todo geom no Geomview tem uma apaência associada. Existe também uma aparência associada ao geom do tipo "World", que comporta-se como hierarquia superior a cada aparência individual de qualquer geom. Finalmente, existe uma aparência básica global, que é a hierarquia superior da aparência "World".

A aparência básica especifica valores razoáveis para toda informação de aparência, e por padrão nenhuma outra aparência especifica qualquer coisa, o que significa que as aparências erdam seus valores a a partir da aparência básica. Isso significa que por padrão todos os objetos são desenhados usando a aparência básica.

Se você modificar uma certa propriedade de aparência de um geom, aquela propriedade é usada na construção daquele geom. A aparência de hierarquia superior é usada para quaisquer propriedades que você não explicitamente escolheu.

Geomview possui três painéis que levam você a modificar aparências.

3.6.1 The Appearance Panel

O painel de Aparência (*Appearance*) leva você a modificar as mais comuns propriedades de aparência do objeto alvo.

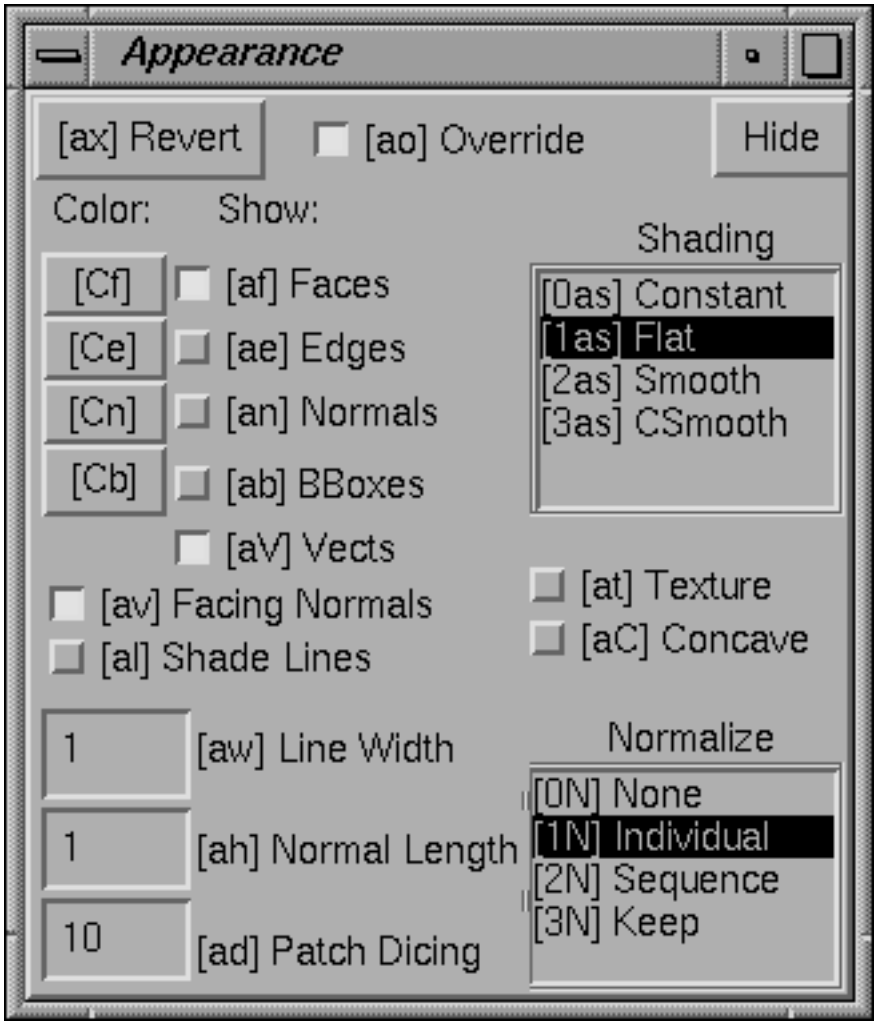


Figura 3.5: O Painel de Aparência.

Se o alvo for um geom individual, então modificaçãoes que você fizer no painel de aparências aplicam-se a aparência daquele objeto geométrico. Se o alvo for o "World", então as modificações no painel de aparências aplicam-se à aparênca do objeto atual *e* a aparência de todos os outros objetos individuais. (Usuários acham que esse comportamento é mais desejável que ter as modificações somente aplicadas a aparência do "World".) Se o alvo for uma câmera, então as modificações do painel de aparência aplica-se ao geom que ocupou mais recentemente a posição de alvo.

Os cinco botões próximos ao canto superior esquerdo sob a palavra *Show* (mostrar) controlam que partes do geom alvo são desenhadas.

Faces Esse botão especifica se faces são desenhadas.

Edges Esse botão especifica se arestas são desenhadas.

Normals Esse botão especifica se vetores normais à superfície são desenhados.

BBox Esse botão especifica se a caixa associada é desenhada.

Vects Esse botão especifica se objetos VECT são desenhados. VECTs são um tipo de objeto OOGL que representa pontos e segmentos de reta em espaço tridimencional; os VECTs são distintos de arestas ou de outros tipos de objetos, e é desejável algumas vezes ter controle separado sobre se eles são desenhados.

Os quatro botões sob *Color* rotulados *Faces*, Arestas (*Edges*), Normais (*Normals*), e *BBox* permitem a você especificar a cor do aspecto correspondente de um geom alvo. Clicando sobre um deles faz com que apareça um painel de escolha de cores.

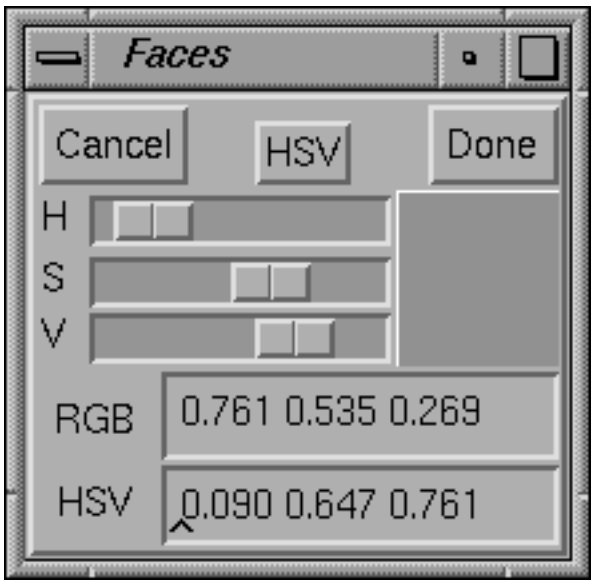


Figura 3.6: Painel de Escolha de Cor.

Esse painel oferece dois conjuntos de botões deslizantes: H(ue) S(Saturation) V(alue), ou R(ed) G(reen) B(lue), cada um no intervalo real fechado de 0 a 1. O quadrado mostra a cor atual, que é fornecida numericamente em ambos os sitemas HSV e RGB nas caixas de texto correspondentes.

No sistema de cores HSV, hue H mostra vermelho em 0, verde em .333, azul em .667, e volta ao vermelho em 1.0. Saturação fornece a fração de branco misturada na cor, de 0 para cinza puro a 1 para a cor pura. Valor fornece o brilho, de 0 para preto a 1 para brilho máximo.

Pressionado o botão *RGB* ou o botão *HSV* ao centro do topo do painel alterna os botões deslizantes para outro sistema de cores. Você pode ajustar cores ou através dos botões deslizantes, ou através de digitação nas caixas de texto RGB ou HSV.

Clique *OK* para aceitar as cores que você tiver escolhido, ou *Cancel* para reter as escolhas anteriores de cor.

O navegador *SHADING* permite a você especificar o modelo de sombreamento que Geomview utilizará para desenhar o geom alvo.

`Constant` Toda face do objeto é desenhada com uma cor constante que não depende da localização da face, nem da câmera, nem também das fontes de luz. Se o objeto não contiver cores por face ou por vértice, a cor difusa de aparência do objeto é usada. Se o objeto contiver cores por face, essas cores serão usadas. Se o objeto contiver cores por vértice, cada face é pintada usando a cor de seu primeiro vértice.

`Flat` Cada face do objeto é desenhada com uma cor que depende da localização relaiva da face, da câmera, e das fontes de luz. A cor é constante ao longo da face mas pode mudar conforme muda a face, a câmera, ou o movimento das luzes.

`Smooth` Cada face do objeto é desenhada com cores lisamente interpoladas baseadas nos vetores normais em cada vértice. Se o objeto não contiver normais por vértice, "Smooth" tem o mesmo efeito que o modo de sombreamento "Flat". Se o objeto tiver normais por vértice rasoáveis, o efeito é alisamento sobre as arestas e entre as faces.

`CSmooth` Cada face do objeto é desenhada com as cores exatamente especificada(s), independente de iluminação, orientação, e propriedades de material. Se o objeto for definido com cores por vértice, as cores serão interpoladas lisamente ao longo da face; de outra forma o efeito é o mesmo que ocorre no estilo de sombreamento "Constant".

`VCflat` Uma combinação de `CSmooth` e sombreamento `Flat`. Dessa forma o sombreamento é constante em cada face, de acordo com a orientação relativa das fontes de luz, a câmera e a superfície normal da face.

O botão *Facing Normals* no painel de aparência (*Appearance*) indica se Geomview pode arranjar aqueles vetores normais ou se não pode sempre conforme a visualização. Se um vetor normal direciona-se afastando-se do visualizador a cor da face correspondente ou vértice correspondente usualmente é mais forte do que é desejado. Geomview pode evitar isso através do uso da normal oposta em cálculos de sombreamento. Esse comportamento é o padrão. Usando *Facing Normals* podemos fornecer estranha suavidade pesada ou estranhos efeitos de sombreamento de luz, embora, proximo ao horizonte de um objeto distantemente liso facetado. Pressione esse botão para usar as normais fornecidas com o objeto.

Os três campos no canto inferior esquerdo do painel de aparência (*Appearance*) são:

Line Width
A espessura, em pixels, para linhas desenhadas pelo Geomview.

Normal Length
Isso é atualmente um fator de homotetia; quando vetores normais forem desenhados, Geomview desenha-os de forma que tenham um comprimento que é seu comprimento natural vezes esse número.

Patch Dicing
Geomview desenha retalhos de Bezier primeiro convertendo-so em malhas. Esse parâmetro especifica a resolução da malha: se *Patch Dicing* for n, então uma malha n por n é usada para desenhar cada retalho de Bezier. Se *Patch Dicing* for 1, a resolução reverte-se para um valor padrão interno.

O botão *Revert* no painel de aparência (*Appearance*) desfaz todas as escolhas na aparência do alvo. Isso faz com que o geom alvo herde todas as suas propriedades de aparência de seus pais.

O botão do painel de aparência (*Appearance*) determina se controles de aparência devem sobrescrever escolhas feitas nos objetos em si mesmos – por exemplos, escolhendo a cor de face irá afetar todas as faces de objetos com faces multicoloridas. De outra forma, controles de aparência somente fornecem escolhas que não forem especificadas nos objetos em si mesmos não especificarem. Por padrão, *Override* está habilitado. Esse botão aplica-se a todos os objetos, e a todos os painéis relacionados a aparência.

Normalização é um tipo de homotetia; Geomview pode alterar o tamanho de um objeto proporcionalmente de forma que esse objeto se ajuste dentro de uma certa região. O objetivo principal da normalização é permitir a você facilmente visualizar um objeto sem ter que se preocupar com o tamanho do mesmo. Estamos substituindo gradualmente o recurso de normalização do Geomview por recursos mais robustos de posicionamento de câmera. Em geral, o melhor caminho de garantir que você está vendo tudo de um objeto é usar o botão *Look At* do painel de ferramentas (*Tools*). A normalização pode ser completamente substituída por esse botão do painel de ferramentas e por outros recursos em uma versão futura do Geomview.

Normalização é uma propriedade que aplica-se a cada geom separadamente. o navegador *NORMALIZE GEOMETRY* afeta a propriedade de normalização do geom alvo. Se o geom alvo for "World", a normalização afetará todos os geoms.

None Sem normalização.

Individual Normaliza o geom atual para ajustar-se dentro de uma esfera unitária.

Sequence Assemelha-se a "Individual", exceto quando um objeto está mudando. Então, "Individual" com muita precisão ajusta a caixa associada em torno do objeto quando esse objeto modifica-se e normaliza-se adequadamente, enquanto "Sequence" normaliza a união de todas as variantes do objeto e normaliza adequadamente.

Keep Mantem a transformação de normalização inalterada quando o objeto modifica-se. *Keep* pode ser útil para aplicar a normalização "Individual" ou a normalização "Sequence" à primeira versão de um objeto que se modifica para trazer esse mesmo objeto ao campo de visão.

3.6.2 O Painel de Materiais

O painel de materiais (*Materials*) controla as propriedades materiais das superfícies. O painel de materiais trabalha com o objeto alvo da mesma forma que o painel de aparência (*Appearance*) faz.

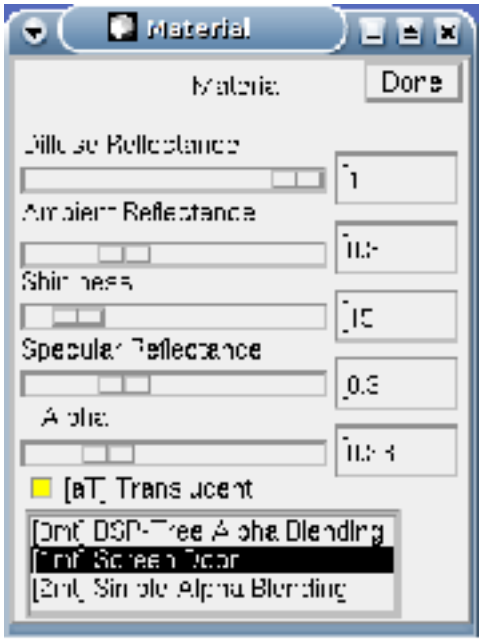


Figura 3.7: O Painel de Materiais.

Translucent

Esse botão determina se a capacidade de ser translúcido está habilitada. Geomview suporta três diferentes formas da capacidade de ser translúcido (translucência):

Alpha-blending com BSP-tree depth-sorting

Esse é o nível mais acurado de preferência de visualização imediata, mas consome vastos montantes de tempo de computação e memória. No modo simples de translucência objetos são mostrados corretamente em relação a si mesmos; todavia multiplos objetos translúcidos podem aparecer na ordem inadequada na tela. A noção *objeto* significa aqui: geometria de nível mais alto como mostrado no navegador de alvo de geometria.

Screen Door Translucency

Se a máquina suporta OpenGL então existe suporte para tipo de translucência por mascaramento de saída (completamente) de pixels transparentes por meio de uma mascara de ponteamento. Essa forma é atualmente muito experimental, e o resultado é de certa forma o ideal, mas funciona e é rápido.

Alphe-blending sem depth-sorting

Esse é a antiga forma de fazer uma translucência rápida e com muitas falhas. Essa é rápida, mas os resultados são completamente incorretos.

Quando a transparência estiver habilitada, um instantâneo RenderMan irá conter a informação alfa, um renderizador obediente pode então gerar figuras de alta qualidade, incluindo a translucência correta.

Alpha O botão deslizante determina a opacidade/transparência quando a transparência estiver habilitada. 0 (zero) significa totalmente transparente, 1 significa totalmente opaco.

Diffuse Reflectance
Esse botão deslizante controla a reflectância difusa de uma superfície. Isso tem a ver com o quanto a superfície dispersa a luz que reflete.

Shininess Esse botão deslizante controla o quanto brilhante a superfície é. Esse botão determina o tamanho de destaques especulares sobre a superfície. Valores pequenos fornecem à superfície uma aparência sombria.

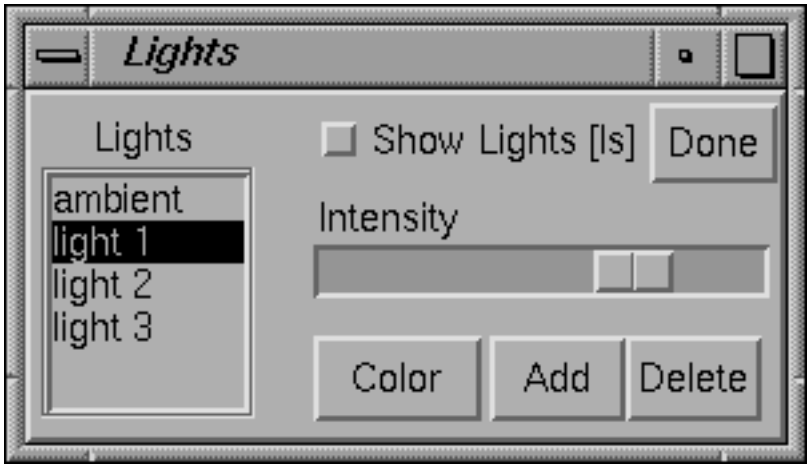
Ambient Reflectance
Esse botão deslizante controla o quanto da luz ambiente uma superfície reflete.

Specular Reflectance
Esse botão deslizante controla a reflectância especular de uma superfície. Isso tem a ver com como a superfície reflete diretamente os raios luminosos. Valores maiores fornecem destaques de brilho especular.

Done Esse botão dispensa o painel de materiais (*Materials*).

3.6.3 O Painel de Luzes

O painel de luzes (*Lights*) controla o número, a posição, e a cor das fontes de luz usadas no sombreamento.



O Painel de Iluminação.

O painel de iluminação (*Lighting*) é diferente do painel de aparência (*Appearance*) e dos painéis de material (*Material*) no sentido de que o painel de iluminação sempre trabalha com a aparência base. Isso ocorre pelo fato de fazer sentido usar o mesmo conjunto de luzes para desenhar todos os objetos na sua cena.

LIGHTS O navegador de luzes (*LIGHTS*) mostra a luz atualmente selecionada. Modificações são feitas usando os outros meios de escolha sobre esse painel aplicado a essa luz. Existe sempre pelo menos uma luz, a luz ambiente.

Intensity Esse botão deslizante controla a intensidade da luz atual.

Color Esse botão faz surgir um modificador de corres que lhe permite selecionar a cor da luz que atualmente ilumina o objeto.

Add Esse botão adiciona uma luz.

Delete Esse botão exclui a luz atual.

Show Lights
Esse botão permite a você ver e modificar a posição das fontes de iluminação em uma janela de câmera. Cada luz é movida ao longo de um cilindro que é suposto para lembrar a você um raio de luz. Quando você clica sobre o botão *Show Lights* Geomview entra no modo "light edit", durante o qual você pode rotacionar a iluminação atual mantendo pressionado o botão esquerdo do mouse e movendo o mouse. Iluminação colocada dessa forma estão infinitamente distantes, de forma que o que você está modificando é a posição angular. Clique sobre o botão *Show Lights* novamente para retornar ao modo anterior de movimento e para sair movendo os raios de luz.

Done Esse botão dispensa o painel de iluminação (*Lighting*).

Os painéis do Geomview *Appearance*, *Materials*, e *Lighting* são construídos para permitir a você fazer facilmente a maioria das coisas relacionadas a aparência que você pode querer fazer. A hierarquia de aparência que Geomview suporta internamente, todavia, é muito complexa e existem certas operações que você não pode fazer com os painéis. A linguagem de comandos do Geomview (GCL) fornece suporte completo a operações sobre aparência. Em particular, o comando `merge-baseap` pode ser usado para modificar a aparência de base (a qual, exceto para iluminação, não pode ser modificada através de painéis do Geomview). O comando `merge-ap` pode ser usado para modificar a aparência de um geom individualmente. Aparências podem também serem especificadas nos arquivos OOGL; para detalhes veja, seção 4.1.10 [Aparencias], página 69. seção 7.2.81 [merge-baseap], página 143. seção 7.2.79 [merge-ap], página 142.

3.7 Cameras

Uma câmera no Geomview é o objeto que corresponde a uma janela de câmera. Por padrão existe somente uma câmera, mas é possível ter tantas quantas você quiser. Você pode controlar certos aspectos do objeto visível atualmente na janela de câmera arrastando em cada janela de câmera via o painel *Cameras*.

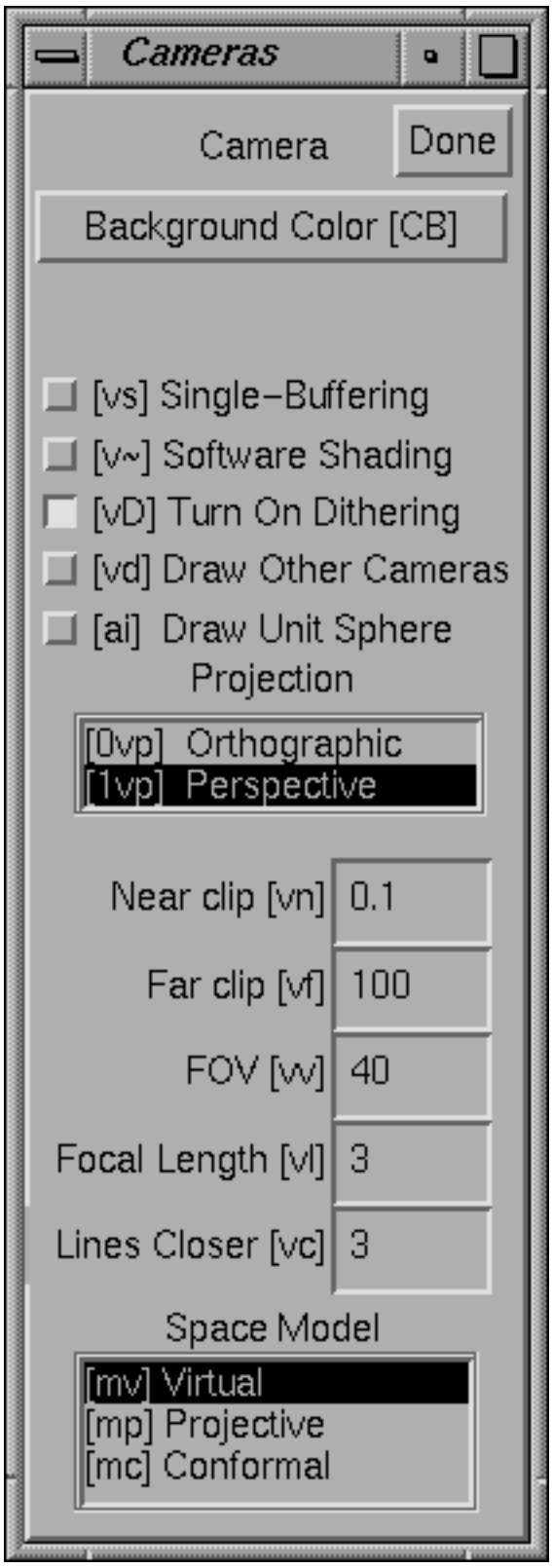


Figura 3.8: O Painel de Câmera.

Se o objeto alvo for uma câmera, o painel de câmeras (*Cameras*) afeta essa câmera. Se o objeto alvo não for uma câmera, o painel de câmeras (*Cameras*) afeta a câmera atual (*current camera*). A câmera atual é a câmera da janela que o cursor do mouse está nela, ou estava mais recentemente se o cursor não estiver em uma janela de câmera. Dessa forma, se você usa teclas de atalho para as ações no painel de câmeras (*Cameras*) enquanto o cursor estiver em uma janela de câmera, as ações aplicam-se a aquela câmera, a menos que você tenha explicitamente selecionado outra câmera.

Para criar novas janelas de câmera, use a tecla de atalho `v+`, ou veja o menu arquivo (*File*) no painel principal (*Main*).

Single-Buffering
: Normalmente, janelas do Geomview são armazenadas em áreas de memória duplas (*double-buffered*): Geomview desenha a figura seguinte em uma janela escondida, então alterna as áreas de memória para fazer essa janela completamente visível em algum momento. Sobre muitos sistemas operacionais, a memória para a área de memória escondida vem da apropriação indevida de metade dos bits de cada pixel de tela, reduzindo a resolução de cor. Quando a opção área de memória simples (single-buffering) for habilitada, as películas de tela de cada cena está sendo desenhada, mas você pode pegar imagens suaves com redução de granularidade em artefatos para mistrura de cores. Área de memória simples é possível se Geomview for compilado com GL ou com OpenGL, mas não com gráficos desenhados com recursos únicos e exclusivos do X.

Dither
: Muitos monitores oferecem menos que 24 bits por pixel (8 bits para cada vermelho, verde, e azul) convencionalmente suficiente para mostrar gradações de cor simplesmente. Quando tenta mostrar uma cor não acuradamente disponível no monitor, Geomview normalmente mistura cores (*dithers*), modificando as cores do pixel para algumas vezes mais brilhante, algumas vezes mais escuro que o valor desejado, de forma que a cor disponível sobre uma área é uma melhor aproximação para a cor verdadeira que um pixel simples pode ter. Efetivamente perdas de resolução espacial são para ganhar resolução de cor. Esse compartamento não é sempre desejável, todavia. Desabilitando *Dither* fornece menor granularidade, mas a precisão de cores é menor, das imagens.

Software Shading
: Esse botão controla se Geomview faz cálculos de sombreamento via software. O padrão é permitir que o hardware manuseie esses cálculos, e no espaço Euclideano esse caminho é o melhor sempre porque é o caminho mais rápido. No espaço hiperbólico e também no espaço esférico, todavia, os cálculos de sombreamento que o hardware faz são incorretos. Clique sobre esse botão para habilitar a forma correta mas lenta do cálculo de sombreamento via software.

Background Color
: Esse botão faz surgir um modificador de cores que você pode usar para escolher a cor de fundo da janela da câmera.

PROJECTION
: Esse navegador permite a você selecionar entre projeção perspectiva ou projeção ortográfica para essa câmera.

Near clip Essa caixa de texto determina a distância nas coordenadas do objeto mundo do próximo plano de corte a partir do ponto de visão. Deve ser um número positivo.

Far clip Essa caixa de texto determina a distância nas coordenadas do objeto mundo do plano de corte mais distante a partir do ponto de visão. Deve ser um número positivo e em geral deve ser maior que o valor de *Near clip*.

FOV Essa caixa de texto é o campo de visão da câmera, medido em sua menor direção. No modo perspectiva, essa caixa corresponde a um ângulo em graus. No modo ortográfico, essa caixa de texto corresponde ao tamanho linear do campo de visão. Esse número pode ser modificado com o mouse no modo *Cam Zoom*.

Focal Length
A distância focal pretende pretende sugerir a distância da câmera a um plano imaginário de interesse. Seu valor é usado quando alternamos entre as visualizações ortográfica e perspectiva (e durante visualização stereo), de forma a preservar o tamanho aparente de objetos sendo desonesto quanto à distância focal da câmera. Distância focal também afeta a interpolação de movimentos de translação efetuados usando o mouse. A velocidade do movimento para adiante (nos modos translação, vôo e no modo orbital) é proporcional à distância focal; e objetos desonestos quanto à distância focal da câmera translacionam lateralmente na mesma razão que o cursor do mouse. Finalmente, no modo de projeção N-Dimencional, câmeras são substituídas de volta através da distância focal a partir da projeção tridimensional da origem do objeto mundo.

Lines Closer
Esse número tem a ver com o caminho pelo qual as linhas são desenhadas. Normalmente a área de armazenamento temporário do algorítmo que controla as coordenadas do eixo z podem confundir-se na hora de desenhar linhas que localizam-se exatamente sobre superfícies (tais como as arestas de um objeto); devido a erros de arredondamento de máquina, algumas vezes as linhas parecem estar em frente à superfície e algumas vezes elas parecem estar por detrás da superfície. O valor *Lines Closer* é um fator de correção — Geomview modifica sutilmente todas as linhas que o algorítmo da área de armazanamento temporária desenha fechando para a câmera através desse fator. O número deve ser um inteiro pequeno; tente 5 ou 10. O valor zero (0) desabilita esse recurso completamente. Escolhendo valores grandes tornará as linhas visíveis mesmo quando elas devam ser escondidas.

SPACE MODEL
Essa opção determina o modelo usado para desenhar o objeto mundo. É mais útil em espaços hiperbólicos e esféricos. Você provavelmente não necessitará tocar esse navegador se você permanecer no espaço Euclideano. Para mais informação sobre esses modelos, veja Capítulo 8 [Geometrias Nao-Euclidianas], página 161.

Virtual Esse é o modelo padrão e representa a visualização natural de dentro do espaço.

Projective Corresponde ao modelo projetivo do espaço hiperbólico e do espaço esférico. Geoms movem-se obedecendo as isometrias do espaço, e câmeras movem-se através de movimentos Euclideanos. Por padrão em modelos projetivos, a esfera unitária Euclidiana é desenhada. No espaço hiperbólico essa esfera localiza-se no infinito. No espaço Euclideano o modelo projetivo é o mesmo que o modelo virtual exceto que a esferea é desenhada por padrão.

Conformal

Corresponde ao modelo conformal do espaço hiperbólico e do espaço esférico. Geoms movem-se obedecendo as isometrias do espaco, e câmeras move-se através de movimentos Euclideanos. No espaço Euclidiano, o modelo conformal equivale a inverter tudo na esfera unitária.

Draw Sphere

Essa opção controla se Geomview desenha a esfera unitária ou não. Por padrão a esfera unitária aparece no modelo projetivo e no modelo conformal. No espaço hiperbólico a esfera é colocada no infinito. No espaço esférico corresponde à esfera equatorial.

Done Esse botão dispensa o painel de *Cameras*.

3.8 Gravando Seu Trabalho

O painel *Save* do Geomview permite a você armazenar objetos do Geomview e outra informação em arquivos que você pode recuperar dentro do Geomview ou a partir de outros programas.

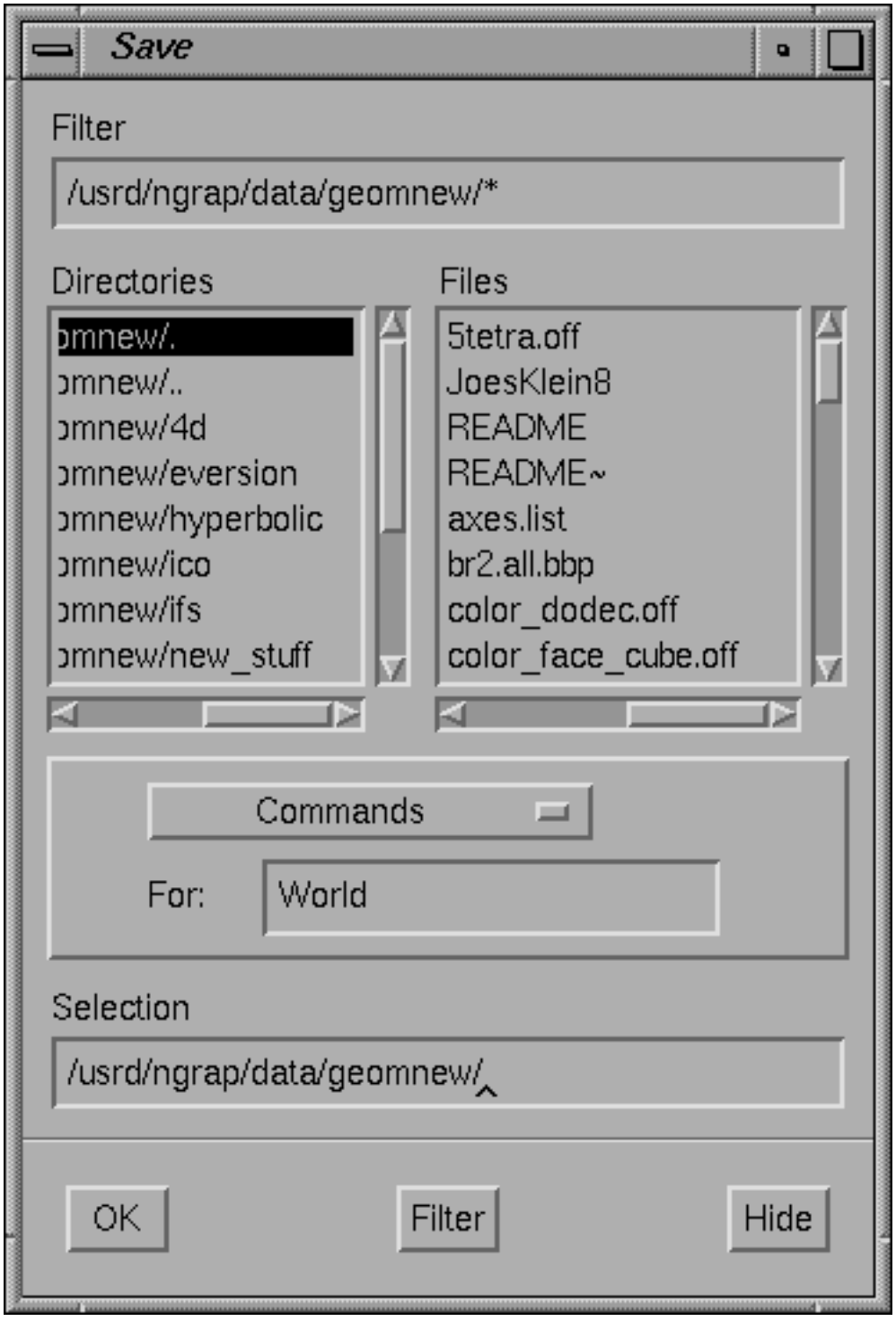


Figura 3.9: O Painel Gravar.

Para usar o painel *Save* você seleciona o formato desejado no navegador perto da palavra *Save*, informe o nome do objeto que você quer gravar no campo de texto perto da palavra *for*, e informe o nome do local no qual você deseja gravar o arquivo no campo longo de texto perto da palavra *in*. Você pode ou pressionar ENTER ou clicar sobre o botão *OK*. Quando o arquivo tiver sido gravado no local desejado, o painel *Save* desaparecerá. Caso você queira dispensar o painel *Save* sem gravar um arquivo, clique no botão *Cancel*.

Caso você especifique '-' como nome de arquivo, Geomview mostra o arquivo na saída padrão, i.e. na janela de shell a partir da qual você invocou o Geomview.

Os possíveis formatos são fornecidos abaixo. O tipo de objeto que pode ser escrito com cada formato é fornecido entre parêntesis.

Commands (qualquer objeto)
: A opção "Commands" escreve um arquivo de comandos GCL contendo todas as informações sobre o objeto. Chamando o arquivo posteriormente irá restaurar o objeto bem como todas as outras informações sobre o referido objeto, tais como aparência, transformações, etc.

Geometry alone (geom)
: A opção "Geometry alone" escreve um arquivo OOGL contendo apenas a geometria do objeto.

Geometry [in world] (geom)
: A opção "Geometry [in world]" escreve um arquivo OOGL contendo a geometria do objeto, transformado sob a transformação atual do Geomview para esse objeto. Use a opção "Geometry [in world]" caso você tenha movido o objeto de sua posição inicial e queira a nova posição em relação ao objeto mundo.

Geometry [in universe] (geom)
: A opção "Geometry [in universe]" grava um arquivo OOGL contendo apenas a geometria do geom, transformado sob transformações sofridas pelo objeto e também sob transformações sofridas pelo objeto mundo.

RMan [->tiff] (camera)
: A opção "RMan [->tiff]" escreve um arquivo que quando for renderizado cria uma imagem tiff. Transparência e textura (a mais recente somente até certo ponto) estará disponível.

RMan [->frame] (camera)
: A opção "RMan [->frame]" escreve um arquivo RenderMan que quando for rederizado faz com que uma imagem apareça em uma janela na tela. Transparência e textura (a mais recente somente até certo ponto) estará disponível.

SGI snapshot (camera)
: A opção "SGI snapshot" escreve um arquivo de varredura SGI. Uma campainha toca quando o instantâneo for completado. Somente disponível em sistemas SGI.

PPM GLX-offscreen snapshot (camera)
: Renderiza uma cena completa novamente dentro da memória off-screen; GLX fornece os meios para usar um Pixmap como área de renderização. A vantagem de renderizar dentro da memória *off*-screen em relação a pegar um instantâneo

de tela é que a janela de câmera não precisa ser mapeada e também não precisa aparece na hora do intantâneo é realizado. De forma que com o intantâneo off-screen se pode seguramente colocar a janela de câmera no formato de ícone (mas não fechá-la!), ativar a proteção de tela e ir dormir enquanto algum script avança as cenas e grava os instantâneos.

PPM Screen snapshot (camera)

Grava os instantâneos a partir da janela fornecida e grava no formato de imagem PPM. Se você especificar uma sequência de caracteres começando com um barra vertical (|) como nome de arquivo, isso é interpretado com um comando de redirecionamento do shell para o qual os dados do PPM deverão ser canalizados, como em '`| pnmtotiff > snap.tiff`' ou em '`| convert -geometry 50% ppm:-snap.gif`'.

Instantâneos de tela PPM estão somente disponíveis com GL e open GL, não com gráficos X somente. A janela pode ocupar inteiramente tela. Geomview irá garantir que não haja outras janelas reproduzindo-a enquanto o instantâneo é gravado. É provavelmente melhor usar instantâneos GLX-*off-screen*, como acima exposto.

PPM software snapshot (camera)

Escreve um instantâneo da janela atualmente visualizada, como uma imagem PPM, apra o arquivo fornecido. O nome do arquivo pode ser um comando do shell Bourne precedido por uma barra vertical (|), da mesma forma que com o instântâneo de tela PPM (PPM screen snapshot). O instantâneo de software, apesar disso, é produzido através do uso de software renderizador interno (relacionado ao renderizador do sistema X-window). Não é importante se a janela está visível ou não, e é independente do GL ou do OpenGL. Também essa opção não suporta alguns recursos, tais como mapeamento de textura.

Postscript snapshot (camera)

Escreve um instantâneo Postscript da visão da câmera. O instantâneo é feito através da decomposição da cena em linhas e polígonos, ordenando por intensidade, e gerando linhas no formato Postscript e polígonos para cada uma. Vantagens desse processo sobre o processo baseado em pixes do instantâneo de imagens: a resolução é muito alta, de forma que arestas parecem na posição correta mesmo em impressoras de alta resolução, ou imagens de resolução comparável são tipicamente muito mais compactas. Desvantagens: ordenação por intensidade fornece bons resultados em algumas cenas, mas pode ser grandemente ruim como no algorítmo de remoção de cenas ocultas em outras cenas. Também, Postscript não oferece sombreamento interpolado linear, somente sombreamento linear simples para cada faceta.

Camera (camera)

Escreve um arquivo OOGL de uma câmera.

Transform [to world] (qualquerobjeto)

Escreve um arquivo de transformação OOGL fornecendo transformação do Geomview para o objeto.

Transform [to universe] (qualquerobjeto)
: Escreve um arquivo de transformação OOGL fornecendo uma transformação que é a composição de transformação do Geomview para o objeto e a transformação sofrida pelo objeto mundo.

Window (camera)
: Escreve um arquivo de janela OOGL para uma câmera.

Panels
: Escreve um arquivo GCL contendo comandos que gravam o estado de todos os painéis do Geomview. Chamando esse arquivo posteriormente irá restaurar as posições de todos os painéis.

3.9 O Painel de Comandos

O painel de comandos (*Commands*) permite a você digitar comandos GCL. Quando você pressionar ENTER, Geomview interpreta o comando e imprime qualquer saída resultante ou uma mensagens de erro para a saída padrão. Você pode editar o texto e pressionar ENTER tantas vezes quantas você quiser, em geral, mesmo que você pressione (ENTER) com o cursor no painel de comandos (*Commands*), Geomview tenta interpretar se aquele texto que você tiver digitado no campo de texto como um comando.



Figura 3.10: O Painel de Comandos.

3.10 Atalhos de Teclado

A maioria das ações que você pode fazer através dos painéis do Geomview possui as equivalentes teclas de atalho de forma que você pode fazer a mesma ação digitando uma sequência de teclas usando o teclado. Isso é útil para usuários avançados que forem familiarizados com as capacidades do Geomview e que queiram trabalhar rapidamente sem ter montes de

painéis confundindo a tela. Teclas de atalho comumente são indicadas entre colchêtes ([ ]) próximo ao item correspondente em um painel. Por exemplo, a tecla de atalho para o modo de rotação (*Rotate*) é 'r'; isso é indicado por "[r]" aparecendo antes da palavra "Rotate" no navegador *MOTION MODE*. Para utilizar essa tecla de atalho apenas pressione a tecla `r` enquanto o cursor do mouse estiver sobre qualquer janela do Geomview. Você não precisa pressionar a tecla ENTER ou teclas de SPACE.

Alguns atalhos consistem de mais que uma tecla. Nesses casos apenas digite as teclas indicadas uma após a outra e na ordem indicada, sem ENTER após cada tecla. Atalhos de teclado são sensíveis à caixa alta/baixa. Você pode cancelar atalhos compostos por várias teclas que você tiver iniciado a digitação digitando qualquer tecla inválida como de atalho, por exemplo a barra de espaço.

Comandos de teclado aplicam-se enquanto o cursor estiver em qualquer janela de câmera e na maioria dos painéis de controle.

Muitas teclas de atalho permitem argumentos numéricos que você digitar como um prefixo a tecla(s) de comando. Por exemplo, a tecla de atalho para *Near clip* no painel de câmera é `v n`. Para escolher o próximo plano de corte para '`0.5`', digite `0.5vn`. Comandos que não recebem um prefixo numérico modificam ou mudam para zero o valor atual.

A maioria dos comando permitem um dos seguintes prefixos de seleção. Se nenhum objeto for fornecido o comando aplica-se ao objeto alvo.

`g`	objeto geométrico mundo ("world")
`g#`	#'ésimo objeto geométrico ("geom")
`g*`	Todos os objetos geométricos ("geoms")
`c`	câmera atual
`c#`	#'ésima câmera
`c*`	Todas as câmeras

Por exemplo, `g4af` significa modifique a face desenhada do objeto *g4*.

Simplesmente digitando um prefixo de seleção, como `g4`, não seleciona um objeto ainda; isso somente acontece quando um comando, como `ae`, segue o prefixo. Para selecionar um objeto como alvo sem fazer nada mais para isso, use o comando `p`. Então `g3p` seleciona o objeto g3.

O campo de texto no canto superior esquerdo faz o painel principal (*Main*) mostrar o estado da tecla de atalho atual.

Para adição de teclas de atalho ao painel de comandos, existe também um atalho para selecionar um objeto alvo: digite o nome curto do objeto seguido por `p`. Por exemplo, para selecionar o objeto *g3*, digite `g 3 p`. Essa forma funciona somente com nomes curtos — aqueles que aparecem entre colchêtes ([ ]) no navegador de alvos (*Targets*) do painel principal *Main*.

Abaixo encontra-se um sumário de todas as teclas de atalho.

Desenho

`af`	Faces
`ae`	Arestas

	`an`	Normais
	`ab`	Caixas Associadas
	`aV`	Vetores
Sombreamento		
	`0as`	Constante
	`1as`	Monótono
	`2as`	Linear
	`3as`	Linear, não iluminado
	`aT`	permite transparência
	`at`	mapeamento de textura
Outro		
	`av`	vira as normais pelo avesso: sempre visualizador de face
	`#aw`	Espessura da linha (em pixels)
	`aC`	manuseia polígonos côncavos
	`#vc`	fechador de arestas que não pertencem a faces (tente 5-100)
Cor		
	`Cf`	faces
	`Ce`	arestas
	`Cn`	normais
	`Cb`	caixas associadas
	`CB`	fundo
Movimentos		
	`r`	rotação
	`t`	translação
	`z`	modificações proporcionais de tamanho (FOV - campo de visão)
	`f`	vôo
	`o`	orbita
	`s`	homotetia
	`w`	recentralizar o alvo
	`W`	recentralizar tudo
	`h`	pare
	`H`	pare tudo
	`@`	selecione o centro do movimento (e.g. `g 3 @`)

	L	Olhe para objeto
Visualizando		
	0vp	visão Ortográfica
	1vp	visão em perspectiva
	vd	Habilite outras câmeras de visão
	#vv	campo de visão
	#vn	próxima distância de corte
	#vf	distância de corte afastada
	v+	adicione nova câmera
	vx	cursor ligado/desligado
	vb	face de trás de um polígono separada habilitada/desabilitada
	#vl	distância focal
	v~	Sombreamento feito via software ligada/desligada
Painéis		
	Pm	Principal
	Pa	Aparência
	Pl	Iluminação
	Po	Obscuro
	Pt	Ferramentas
	Pc	Câmeras
	PC	Comandos
	Pf	arquivos
	Ps	Salvar
	P-	ler comandos pelo tty
	PA	Créditos ("about")
Luzes		
	ls	mostrar luzes
	le	editar luzes
Espaço		
	me	Euclidiano
	mh	Hiperbólico
	ms	Esférico
Modelo		

mv	Virtual
mp	Projetivo
mc	Conformal

Outro

0N	normalização: nenhuma
1N	normalização: individual
2N all	normalização: todos
ui	movimento: Inércial
uc	movimento: Contração para o eixo
uo	movimento: coordenadas próprias do objeto
<	
Pf	carregar arquivo geométrico ou de comandos
dd	apagar objeto alvo
>	
Ps	grave o estado atual em um arquivo
TV	mudar para o modo NTSC
p	selecionar como objeto alvo (e.g. *g 3 p*) Sem preixo, seleciona o objeto sob o cursor do mouse (da mesma forma que duplo-clicando o botão direito do mouse)

4 Formatos dos Arquivos da OOGL

Os objetos que você pode carregar dentro do Geomview são chamados objetos OOGL. OOGL significa "Object Oriented Graphics Library" (biblioteca gráfica orientada a objetos); é a biblioteca sobre a qual Geomview é construído.

Existem muitos tipos diferentes de objetos OOGL. Esse capítulo fornece descrições sintáticas de formatos de arquivo para objetos OOGL.

Exemplos da maioria dos tipos de arquivo podem ser encontrados no diretório '`data/geom`' do Geomview.

4.1 Convenções

4.1.1 Sintaxe Comum a Todos os Formatos de Arquivo da OOGL

A maioria dos formatos de arquivo de objeto OOGL são do formato livre ASCII — qualquer quantidade de espaços em branco (caracteres não imprimíveis, tabulações, caractere de nova linha) pode aparecer entre os sinalizadores (números, palavras chave, etc.). Paradas de linha são na maioria das vezes sempre insignificantes, com algumas excessões devidamente ressaltadas. Comentários começam com # e continuam até o fim da linha; esses comentários são permitidos em qualquer lugar onde um caractere de nova linha for permitido também.

Formatos binários são também definidos para muitos objetos; Veja seção 4.1.8 [Formato binario], página 67, e as descrições individuais do objeto.

Objetos típicos OOGL começam com uma palavra chave designando o tipo de objeto, possivelmente com modificadores indicando a presença de informações de cor, etc. Em alguns formatos a palavra chave é opcional, por questões de compatibilidade com formatos de arquivo definidos em outros lugares. O tipos de objeto é então determinado por suposição sobre o sufixo do arquivo (se houver) ou pelos dados em si mesmos.

Palavras chave são sensíveis à caixa alta/baixa. Algumas palavras chave possuem letras de prefixo adicionais indicando a presença de cor ou outros dados; nesse caso a ordem dos prefixos é importante, e.g. `CNMESH` é significativo mas `NCMESH` é inválido.

4.1.2 Nomes de Arquivo

Quando objetos OOGL são lidos de arquivos localizados em disco, a biblioteca OOGL usa o sufixo do arquivo para supor o tipo de arquivo.

Se o sufixo for desconhecido, ou estiver ausênte (e.g. para um objeto sendo lido a partir da saída de um outro comando diretamente, ou lido diretamente de dentro de um outro objeto OOGL), todos os tipos conhecidos de objeto são tentados por sua vez até que se aceite os dados como válidos.

4.1.3 Vértices

Muitos objetos compartilham um estilo comum de representar vértices com opções de superfície normal de vértice e cor. Todos os vértices dentro de um objeto possuem o mesmo formato, especificado pela palavra chave no cabeçalho.

Todos os dados para um vértice estão agrupados juntos (em oposição a e.g. fornecimento de coordenadas para todos os vértices, em seguida cores para todos os vértices, e assim por diante).

A sintaxe é

‘*x y z*’ (coordenadas do vértice tridimensionais em ponto flutuante) ou

‘*x y z w*’ (coordenadas do vértice tetradimensionais em ponto flutuante)

opcionalmente seguida por

‘*nx ny nz*’
(superfície normalizada tridimensional se presente)

opcionalmente seguida por

‘*r g* b a’ (componente quádrupla em ponto flutuante se presente, cada componente no intervalo 0..1. A componente *a* (alfa) representa a opacidade: 0 transparente, 1 opaco.)
opcionalmente seguida por

‘*s t*’

‘ou’

‘*s t u*’

(duas ou três valores coordenados de textura).

Valores são separados por espaços em branco, e quebras de linha são imateriais.

Letras na palavra chave de cabeçalho do objeto devem aparecer numa ordem específica; isto é a ordem reversa na qual os dados são fornecidos para cada vértice. Então vértices de objeto do tipo ‘CN4OFF’ possuem primeiramente as componentes quadridimensionais de posicionamento no espaço, a seguir as componentes normais tridimensionais, finalmente as componentes quadridimensionais de cor. Você pode modificar a ordem dos dados modificando o cabeçalho de palavra chave; um ‘NCOFF’ é apenas não reconhecido.

4.1.4 Vértices *N*-dimensionais

Muitos objetos compartilham um estilo comum de representação de vértice com opcionais de superfície normal e cor. Todos os vértices dentro de um objeto possuem o mesmo formato, especificado pela palavra chave do cabeçalho.

Todos os dados para um vértice estão agrupados juntos (em oposição a e.g. fornecendo coordenadas para todos os vértices, a seguir cores para todos os vértices, e assim por diante).

A sintaxe para vértices *N*-dimensionais ($N > 3$) é

‘*x[1] x[2] x[3] x[4] ...*’
(*N* coordenadas do vértice em ponto flutuante) ou

‘*x[0] x[1] x[2] x[3] x[4] ...*’
(*(N+1)* coordenadas do vértice em ponto flutuante, se o modificador 4 tiver sido especificado na linha de cabeçalho do objeto)

Note, todavia, que objetos *N*-dimensionais internamente sempre possuem pontos *(N+1)*-dimensionais; a primeira componente *x[0]* – se presente no arquivo do objeto file – é usada como divisor homogêneo. Isso é diferente do caso tridimensional comum onde o modificador 4 gera um objeto quadridimensional onde a componente quadridimensional implicitamente é escolhida para 1.

Componentes de cor usualmente podem ser especificadas da mesma forma que para vértices tridimensionais, Veja seção 4.1.3 [Vertices], página 65, enquanto especificando componentes de cor em normais não faz sentido.

4.1.5 Direções de superfícies normais

Geomview utiliza vetores normais para determinar como um objeto é compartilhado. A direção da normal é importante nesse cálculo.

Quando normais forem fornecidos com um objeto, a direção da normal é determinada pelos dados fornecidos.

Quando normais não forem fornecidos com o objeto, Geomview calcula vetores normais automaticamente; nesse caso normais apontam para adiante do lado do qual os vértices aparecem na ordem anti-horária.

Sobre superfícies paramétricas (retalhos de Bezier), o vetor normal no ponto P(u,v) está na direção dP/du multiplicado vetorialmente por dP/dv.

4.1.6 Matrizes de transformação

Alguns objetos incorporam matrizes quadradas reais de ordem 4 para transformações sobre objetos homogêneos. Essas matrizes atuam através de multiplicação à direita de vetores. Dessa forma, se p for um vetor linha de 4 elementos representando coordenadas homogêneas de um ponto no objeto OOGL, e A for a matrix 4x4, então o ponto resultante da transformação é p' = p A. Essa convenção matricial é comum em computação gráfica; é a matriz transposta daquela muitas vezes usada em matemática, onde pontos são vetores coluna multiplicados à direita pelas matrizes.

Dessa forma para transformações Euclideanas, as componentes para translação aparecem na quarta linha (os últimos quatro elementos) da matriz A. A última coluna da matriz A (quarto, oitavo, décimo segundo e décimo sexto elementos) são tipicamente 0, 0, 0, e 1 respectivamente.

4.1.7 Matrizes de transformação ND

No contexto do espaço *N*-dimensional ($N > 3$) alguns objetos incorporam *(N+1)*x*(N+1)* matrizes reais para transformações sobre objetos homogêneos. Essas matrizes atuam através de multiplicação à direita de vetores. Dessa forma, se p for um vetor linha de *(N+1)*-elementos representando coordenadas homogêneas de um ponto no objeto OOGL, e A é a matriz quadrada de ordem *(N+1)*, então o ponto resultante da transformação é p' = p A.

Note que (a excessão de matrizes de transformação 4x4, veja seção 4.1.6 [Matrizes de transformacao], página 67) a componente homogênea é localizada nos elementos com índice **zero**, de forma que componentes de transformação para transformações Euclideanas aparecem na **zero**-ésima linha (primeiros *(N+1)* elementos). A primeira coluna da matriz A (a coluna com índice zero) é tipicamente 1, 0, . . . , 0.

4.1.8 Formato binário

Muitos objetos OOGL aceitam formatos o formato de arquivo binário bem como o formato de arquivo ASCII. Esses arquivos começam com a indicação usual (e.g. `CQUAD`) seguida pela palavra `BINARY`. Dados binários iniciam-se no byte imediatamente seguinte ao primeiro caractere de nova linha após a palavra `BINARY`. Espaços em branco e um simples comentário podem atrapalhar, e.g.

```
OFF BINARY # binary-format "OFF" data follows
```

Dados binários compreendem inteiros de 32 bits e os inteiros em ponto flutuante de 32 bits conforme definido pela IEEE (IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985 - Institute of Electrical and Electronics Engineers), ambos os formatos seguem à forma de ordenação "big-endian" (i.e., com o byte mais significativo em primeiro lugar). Esse é o formato nativo para tipos de dado int e tipos de dado float em sistemas do tipo Sun-3, Sun-4, e Irises, além de muitos outros sistemas.

Formatos de dados binários assemelham-se aos formatos correspondentes em ASCII, com tipos de dado "int" e tipos de dado "float" nos lugares que você está acostumado a encontrar. Existe algumas excessões todavia, especificamente dos formatos d arquivo `QUAD`, `OFF` e `COMMENT`. Detalhes são fornecidos no arquivo individual das descrições do formato específico. Veja em seção 4.2.1 [QUAD], página 76, seção 4.2.5 [OFF], página 80, e em Veja seção 4.2.14 [COMMENT], página 90.

Objetos OOGL binários podem ser livremente misturados em fluxos de objetos ASCII:

```
LIST
{ = MESH BINARY
... dados binários de uma malha aqui ...
}
{ = QUAD
1 0 0   0 0 1   0 1 0  0 1 0
}
```

Note que dados ASCII continuam seguindo imediatamente o último byte dos dados binários.

Naturalmente, é impossível incluir comentários dentro de um objeto em formato binário OOGL, todavia comentários podem aparecer no cabeçalho antes do início dos dados binários.

4.1.9 Referências a Objetos Embutidos e a Objetos Externos

Alguns tipos de objeto OOGL (`LIST`, `INST`) permitem referências a outros objetos OOGL, que podem aparecer literalmente no fluxo de dados, serem chamados a partir de arquivos em disco, ou serem comunicados a partir de outro lugar via objetos nomeados. Comandos GCL também aceitam objetos geométricos através desses meios citados.

A sintaxe genérica é

```
 <oogl-object>  ::=
[ "{" ]
    [ "define" symbolname ]
    [ ["="] object-keyword ...
 | "<" filename
 | ":" symbolname ]
[ "}" ]
```

onde os itens entre aspas duplas são sequências de caracteres literais (que aparecem sem as aspas), os itens entre colchêtes ([]) são opcionais, e a barra vertical (|) denota alternativas. Chaves, quando estiverem presentes, apenas indicam coincidência; o par mais externo de chaves é geralmente requerido quando o objeto estiver em um contexto maior, e.g. quando for parte de um objeto maior ou incluído em um fluxo de comando do Geomview.

Por exemplo, cada uma das três linhas seguintes:

```
{ define fred   QUAD 1 0 0  0 0 1  0 1 0  1 0 0 }

{ define fred = QUAD 1 0 0  0 0 1  0 1 0  1 0 0 }

{ appearance { +edge } LIST { < "file1" } { : fred } }

VECT 1 2 0   2 0   0 0 0   1 1 2
```

é um bojeto OOGL válido. O último exemplo é válido somente quando estiver delimitado precisamente e residindo precisamente no seu próprio arquivo em disco.

A construção com `":"` permite referência a símbolos, criados com `define`. Um valor inicial de um símbolo é um objeto nulo. Quando um símbolo for (re)definido, todas as referências a esse símbolo são automaticamente modificadas.

A construção `"define` NOME" permite definir um simbolo global para um objeto especificado. Se "NOME" referir-se a um objeto que já existia anteriormente, então o antigo objeto é descartado e substituído pela nova definição. Veja seção 7.2.111 [read], página 149. seção 7.2.59 [hdefine], página 137.

A construção `"<"` faz com que um arquivo em disco seja lido. Note que isso não é um mecanismo textual genérico de uso da diretiva de programação "include"; um objeto OOGL completo deve aparecer no arquivo referenciado na construção `"<"`.

Arquivos lidos usando `"<"` são procurados primeiramente no diretório do arquivo que é refrenciado em `"<"`, se existir; se essa busca falhar, o caminho normal de busca (veja seção 7.2.72 [load-path], página 141) é usado. A busca padrão olha primeiro no diretório atual, a seguir nos diretórios de dados do Geomview.

Ressaltando, espaços em branco e caracteres indicadores de final de linha são insignificantes, e comentários de "#" podem aparecer em qualquer lugar.

4.1.10 Aparências

Objetos geométricos podem ter informações de aparência associada, especificando sombreamento, brilho, cor, visualização wire-frame vs. shaded-surface, e assim por diante. Aparências são erdadas através de hierarquías de objetos, e.g. anexando uma aparência a uma lista (`LIST`) significa que a aparência é aplicada a todos os membros de `LIST`.

Algumas propriedades relacionada a aparência são relegadas a subestruturas tais como "material", "lighting" e "texture". Seja cuidadoso para notar quais propriedade pertencem a qual estrutura. Qualquer objeto geométrico pode ser precedido por uma definição de aparência como no sguinte exemplo:

```
{
   appearance { +edge }
   LIST { < "arquivo1" } { QUAD 1 0 0  0 0 1  0 1 0  1 0 0 }
}
```

Aparências são também objetos OOGL no seu próprio direito e pode ser fornecido nomes simbólicos e referência a elas. Veja (seção 4.1.9 [Referencias], página 68) e em seção 4.3.1 [Objetos de aparencia], página 91.

Texture Mapping
: Existe uma seção separada com relação a definição de texturas: (seção 4.1.11 [Mapeamento de Textura], página 74).

Transparency
: Objetos de renderização translúcida não são suportados por todos os desenhadores que funcionam no Geomview. O renderizador OpenGL tem suporte limitado para isso: objetos de nível mais alto (i.e. aqueles que aparecem no navegador de objetos do painel principal (veja seção 3.3 [Interacao Basica], página 33) são renderizados corretamente por meio de alpha-blending). Também, os instantâneos do RenderMan irão incluir valores de opacidade.

Aqui está um exemplo de estrutura de aparência incluindo valores para todos atributos. A ordem dos atributos não é importante. Como usual, espaços em branco ssão irrelevates. Atributos Booleanos podem ser precedidos por "+" ou "-" para torná-los habilitados ou desabilitados; "+" é assumido se somente o nome do atributo aparecer. Outros valores esperados de atributo.

Um "*" prefixado a um atributo, e.g. "*+edge" ou "*linewidth 2" ou "material { *diffuse 1 1 .25 }", seleciona a situação atual para sobrescrever ("override") para aquele atributo.

```
appearance {
  +face                # (Faz) desenho de faces dos polígonos.  Habili-
                       # tado por padrão.
  -edge                # (Não faz) desenho de arestas de polígonos.
  +vect                # (Faz) desenho de vetores (VECTs).  Habilitado
                       # por padrão.

  transparent screendoor
                       # (Habilita) transparência. Habilitação de
                       # transparência.
                       # não (necessariamente) resulta em cores corretas
                       # no Geomview, mas os valores de alfa são usados
                       # em instantâneos RenderMan.
                       # As palavras chave permitidas são ‘‘screendoor’’
                       # (mascarando pixels de saída por meio de um
                       # modelo de ponteamento), ‘‘blending’’ para
                       # alpha-blending ( harmonização de alfa)
                       # com BSP-tree (árvore BSP) particionando o
                       # espaço e ordenando com precisão
                       # (lento) e ‘‘naive’’ para harmonização de
                       # alfa sem mesmo ordenar com precisão, não
                       # para falar sobre particionamento de espaço.
                       # Omitindo a palavra chave o padrão será
                       # harmonização de alfa com árvore
                       # particionando o espaço e ordenando com
                       # precisão.
  -normal              # (Faz) desenho de vetores normais a uma
                       # superfície.
```

```
normscale 1          # ... com comprimento 1.0 em coordenadas do
                     # objeto.

+evert               # faz a inversão de normais de polígono onde
                     # necessário de forma que sempre tenha
                     # câmera naquela face

+texturing           # (Habilita) mapeamento de textura
+linear              # (Habilita) média linear de elementos de textura
                     # mais fechados

+mipmap              # (Habilita) mapeamento mip de textura
+mipinterp           # (Habilita) mapeamento mip linear

-backcull            # (Não faz) descarte da orientação de faces
                     # no sentido anti-horário

-concave             # (Não faz) presunção e manuseio de polí-
                     # gonos côncavos

-shadelines        # (Não faz) linhas de sombreamento como se elas esti-
                     # vessem iluminando cilindros
                     # Esses quatro (mipinterp, backcull, concave,
                     # shadelines) são somente efetivos onde o
                     # sistema gráfico suportá-los, a saber em GL e
                     # Open GL.

-keepcolor         # Normalmente, quando informação de cor N-D posi-
                     # cional está habilitada como
                     # com o comando do geomview (ND-color ...), todas
                     # as cores de objetos são afetadas.  Mas, objetos
                     # com o atributo "+keepcolor" são imunes a infor-
                     # mações de cor N-D.

shading smooth       # ou ‘‘shading constant’’ ou ‘‘shading flat’’ ou
                     # ou ‘‘shading csmooth’’ ou ‘‘shading vcflat’’.
                     # smooth = sombreamento de Gouraud, flat = facetado,
                     # csmooth = linearmente interpolado mas não
                     # iluminado, vcflat = sombreamento monótono, mas
                     # cores linearmente interpoladas.

linewidth 1          # linhas, pontos, e arestas são da largura de
                     # 1 pixel.

patchdice 10 10      # subdivide retalhos de Bezier esmiuçadamente em
                     # u e v
```

```
material {          # Aqui está uma definição material;
                    # pode também ser lido de um arquivo como em
                    #  ‘‘material < arquivo.mat’’

    ka  1.0         # coeficiente de reflexão ambiente.
    ambient .3 .5 .3 # cor do ambiente (em componentes de vermelho,
                    # verde, azul). A contribuição ambiente para a to-
                    # nalidade é o produto de ka, a cor do ambiente,
                    # pela cor da luz ambiente.

    kd  0.8         # coeficiente de reflexão difusa.
    diffuse .9 1 .4 # cor difusa.
                      # (No modo ‘‘shading constant’’ (sombreamento
                      # constante), a superfície
                      # é colorida com a cor difusa.)

    ks 1.0          # coeficiente de reflexão especular.
    specular 1 1 1  # cor especular (destacada).
    shininess  25   # expoente especular; grandes valores fornecem
                    # destaques falsos.

    backdiffuse .7 .5 0 # cor da face de trás para superfícies com
                    # dois lados. Se definido, esse campo determina
                    # a cor difusa para o lado de trás de uma super-
                    # fície. É implementada através de sombreamento
                    # via software, e por sombreamento via hardware
                    # sob sistemas GL que suportam iluminação de
                    # dois lados, e sob Open GL.

    alpha   1.0     # opacidade; 0 = transparente (invisível),
                    # 1 = opaco. Ignorado quando a transparência
                    # estiver desabilitada.

    edgecolor   1 1 0  # cor de linha & aresta

    normalcolor 0 0 0  # cor para vetores normais a uma superfície
}

lighting {          # Modelo de iluminação

    ambient  .3 .3 .3  # luz ambiente

    replacelights   # ‘‘Use somente as seguintes luzes para
                    # iluminar os objetos sob essa
                    # aparência.’’
                    # Sem "replacelights" (substituição de luzes),
                    # quaisquer luzes listadas são adicionadas
```

```
                    # a esses na cena.

                    # Agora uma coleção exemplo de luzes:
        light {
            color  1 .7 .6      # cor da luz
            position  1 0 .5 0  # posição da luz [distant light]
                                # fornecida em coordenadas homogêneas.
                                # Com a quarta componente = 0,
                                # isso significa uma luz vindo da
                                # direção (1,0,.5).
        }

        light {                         # Outra luz.
            color 1 1 1
            position  0 0 .5 1  # luz na posição finita ...
            location camera     # especificada em coordenadas de câmera.
                                # (Uma vez que a câmera olha adiante -Z,
                                # esse exemplo coloca a luz
                                # .5 unidade atrás do olho.)
            # Possível palavra chave de localização ("location"):
            #  global   posição da luz global em coordenadas de objeto mundo
            #             (bem, universo). Esse é o padrão se nenhuma
            #             localização for especificada.
            #  camera   posição da câmera em coordenadas do sistemade
            #             câmera
            #  local    posição local em coordenadas do sistema onde
            #                   a aparência foi definida
        }
    }                   # fim do modelo de iluminação
    texture {
          clamp st              # ou ‘‘s’’ ou ‘‘t’’ ou ‘‘none’’
          file lump.tiff        # arquivo fornecendo imagem de mapa de
                                # textura
          alphafile mask.pgm.Z  # arquivo fornecendo imagem de máscara
                                # de transparência
          apply blend           # ou ‘‘modulate’’ ou ‘‘decal’’
          transform  1 0 0 0    # superfície (s,t,0,1) * tfm -> coor-
                     0 1 0 0    # denadas de textura
                     0 0 1 0
                    .5 0 0 1

          background 1 0 0 1    # relevante para ‘‘apply blend’’
    }
}                   # fim de aparência
```

Existem regras para herança de atributos de aparência quando muitas aparência são impostas em diferentes níveis na hierarquía.

Por exemplo, Geomview instala uma aparência de parada de segurança que fornece valores padrão para a maioria dos parâmetros; seu painel de controle instala outras aparências que fornecem novos valores para uns poucos atributos; a geometria fornecida pelo usuário pode também conter aparências.

A regra geral é que a aparência dos filhos (aquelas fechadas para as primitivas geométricas) vencem. Adicionalmente, controles de aparência com situação atual em sobrescrever ("override") (e.g. `*+face ou material { *diffuse 1 1 0 }`) vencem sobre as outras que estiverem sem o atributo de sobrescrever.

Os controles de aparência do Geomview usam o recurso de sobrescrever "override" de forma a ser efetivo mesmo se objetos fornecidos pelo usuário contiverem suas próprias escolhas de aparência. Todavia, Se um objeto fornecido pelo usuário contiver um campo de aparência com o recurso de sobrescrever ativado, esse campo de aparência será imune aos controles do Geomview.

4.1.11 Mapeamento de Textura

Alguns programas que trabalham sem contato direto com o usuário suportam objetos mapeados em textura, atualmente apenas o OpenGL e a interface RenderMan no momento dessa escrita. Existe também alguns recursos com a interface RMan quando em uso um canal alfa na imagem de textura. Aqueles programas que não trabalham diretamente com o usuário cujo suporte a textura não existe silenciosamente ignoram tentativas de uso de mapeamento de textura. Uma textura é especificada como parte de uma estrutura de aparência (Veja seção 4.1.10 [Aparencias], página 69). Resumidamente, se fornece uma imagem de textura (Veja também seção 4.3.2 [image], página 91), a qual é considerada contida em um quadrado no espaço parametrizado `(s,t)` no intervalo 0 <= s <= 1, 0 <= t <= 1. Então se fornece uma primitiva geométrica, com cada vértice acompanhado com as coordenadas de textura `(s,t)`. Se a texturização estiver habilitada, a porção apropriada da imagem de textura é colada sobre cada face do objeto texturizado.

Não existe (atualmente) nenhuma provisão para herança de parte de uma estrutura de textura; se a palavra chave `texture` é mencionada em uma aparência, essa mensão suplanta qualquer outra especificação de textura.

O atributo de aparência `texturing` controla se texturas são usadas; não existe perda de performace tendo campos "texture `{ ... }`" definidos quando a utilização de texturas está desabilitada.

Os campos de textura disponíveis são:

```
clamp   none  -ou-  s  -ou-  t  -ou-  st
  Determina o significado de coordenadas de textura fora do intervalo
  0..1. Com clamp none, o padrão, coordenadas são
  interpretadas modulo 1, então (s,t) = (1.25,0), (.25,0), e (-.75,0)
  referem-se todos ao mesmo ponto no espaço das texturas.  Com
  clamp s ou clamp t ou clamp st, individualmente
  ou ambos de s-coordenadas ou t-coordenadas menor que 0 ou
  maior que 1 são remapeadas para 1 ou para 0, respectivamente.

image { <especificação de imagem> (seção 4.3.2 [image], página 91) }
  Especifica a atual imagem de textura. Imagens de textura podem ter
```

```
  1, 2, 3 ou 4 canais:
    1 canal:  luminância
    2 canais: luminância e alfa (opcidade:0 transparente, 1 opaco)
    3 canais: dados RGB
    4 canais: dados RGBA

  Veja seção 4.3.2 [image], página 91, para a definição atual de
  objetos de imagem. O canal usado por alfa é somente interpretado
  como máscara: onde a mácara é zero, pixels são simplesmente
  omitidos. Uma exceção é o caso onde apply
  é igual a modulate e a translucência está habilitada:
  nesse caso o valor de alfa resultante é o resultado da
  multiplicação da cor da superfície pelo valor de alfa do
  canal alfa da textura.

file      nomearquivo
alphafile nomearquivo
  Isso é considerado obsoleto, e somente mantido por compatibilidade,
  o moderno caminho é usar o novo objeto imagem OOGL. Veja
  seção 4.3.2 [image], página 91.  O material documentado aqui pode
  ainda funcionar apesar disso

  Especifica arquivos de imagem contendo a textura.

  A palavra chave file especifica um arquivo com informações de cor
  ou de brilho; alphafile se presente, especifica uma máscara de
  transparência ("alpha"); onde a máscara for zero, pixels simplesmente
  são omitidos.  Muitos formatos de arquivo de imagem estão
  disponíveis; o tipo de arquivo deve ser indicado pelos últmos poucos
  caracteres do nome do arquivo:
    .ppm ou .ppm.Z ou .ppm.gz  24-bit 3-color imagem no formato PPM
    .pgm ou .pgm.Z ou .pgm.gz  8-bit tons de cinza imagem no formato PGM
    .sgi ou .sgi.Z ou .sgi.gz  8-bit, 24-bit, ou 32-bit imagem SGI
    .tiff                      8-bit ou 24-bit imagem TIFF
    .gif                       imagem GIF
  Para esse recurso trabalhar, alguns programas devem estar disponíveis
  no caminho de busca do Geomview:
    zcat  para arquivo .Z
    gzip  para arquivos .gz
    tifftopnm para arquivos .tiff
    giftoppm para arquivos .gif

  Se uma imagem alphafile for fornecida, essa imagem deve ser do mesmo
  tamanho que a imagem file.

  Objetos imagem fornecem um caminho mais flexível para especificar da-
dos de
```

textura. Veja *seção 4.3.2 [image], página 91.*

```
apply   modulate  -ou-  blend  -ou-  decal
  Indica como a imagem de textura é aplicada à superfície.
  Aqui a "surface color" (cor da superfície) significa a cor que a
  superfície pode ter   na ausência de mapeamento de textura.

  Com modulate, o padrão, a cor de textura (ou iluminação,
  se texturizado por meio de uma imagem de escala de cinza) é multiplicada
  pela cor da superfície.

  Com blend, textura harmoniza-se entre cor de fundo (background)
  e a cor da superfície.  O parâmetro file deve especificar uma
  imagem de escala de cinza.  Onde a imagem de textura é 0, a cor da
  superfície permanece inalterada; onde for 1, a superfície é
  colorida na cor dada por background; e cor é interpolada para
  valores imediatos.

  Com decal, o parâmetro file deve especificar uma
  imagem de tres cores.  Se um parâmetro alphafile estiver presente,
  seus valores interpolam-se entre a cor da superfície (onde alpha=0)
  e a cor de textura (onde alpha=1).  Iluminação não afeta a
  cor de textura no modo decal; efetivamente a textura é
  tonalidademente constante.

background  R G B A
  Especifica uma cor com 4 componentes, com números R, G, B, e A em
  ponto flutuante normalmente no intervalo 0..1, usados quando
  apply blend for selecionado.

transform matriz-de-transformação
  Espera uma lista de 16 números, ou um dos outros caminhos de
  representar uma transformação (: nomecabeçalho ou
  < nomearquivo). A matriz 4x4 de transformação é
  aplicada a coordenadas de textura, no sentido de um vetor linha de
  4 componentes (s,t,0,1) multiplicado à esquerda pela matriz, para
  produzir novas coordenadas (s',t') às quais atualmente indexam a
  textura.
```

4.2 Formatos de Arquivo de Objeto

4.2.1 QUAD: coleção de quadriláteros

O sufixo convencional para um arquivo `QUAD` é '`.quad`'.

A sintaxe do arquivo é

```
[C][N][4]QUAD  -ou-  [C][N][4]POLY    # Palavra chave
vértice  vértice  vértice  vértice  # vértices 4-D para algum N
```

```
vértice  vértice  vértice  vértice
...
```

A palavra chave inicial é `[C][N][4]QUAD` ou `[C][N][4]POLY`, onde o prefixos opcionais `C` e `N` indicam que cada vértice inclui cores e retas normais respectivamente. Isto é, esses arquivos iniciam-se com uma das palavras

`QUAD CQUAD NQUAD CNQUAD POLY CPOLY NPOLY CNPOLY`

(mas não com `NCQUAD` ou `NCPOLY`). `QUAD` e `POLY` são sinônimos; ambas as formas são permitidas apenas por compatibilidade com ChapReyes.

Seguindo a palavra chave está um número arbitrário de grupos de quatro vértices, cada grupo descrevendo um quadrilátero. Veja a sintaxe de vértice acima. O objeto termina no caractere de fim de arquivo, ou com uma chave fechada se incorporado dentro de uma referência de objeto (veja acima).

Um formato de arquivo `QUAD BINARY` é aceito; veja seção 4.1.8 [Formato binario], página 67. A primeira palavra de dados binários deve ser um inteiro de 32 bits fornecendo o número de quads no objeto; seguindo esse inteiro encontra-se uma série de inteiros em ponto flutuante de 32 bits, arranjados apenas como no formato ASCII.

4.2.2 MESH: Malha retangularmente conectada

O sufixo convencional para um arquivo `MESH` é '`.mesh`'.

A sintaxe do arquivo é

```
[U][C][N][Z][4][u][v][n]MESH # Palavra chave
[Ndim]                       # Dimensão do espaço, presente
                              # somente se nMESH
Nu Nv                        # dimensões da grade da malha
                             # Nu*Nv vértices, no formato especificado
                             # pela palavra chave inicial
vértice(u=0,v=0)  vértice(1,0)  ... vértice(Nu-1,0)
vértice(0,1) ...    vértice(Nu-1,1)
...
vértice(0,Nv-1) ... vértice(Nu-1,Nv-1)
```

A palavra chave é `[U][C][N][Z][4][u][v][n]MESH`. Os caracteres opcionais prefixados significam:

'`U`' Cada vértice inclue uma textura de 3 parâmetros de espaço de componente. As primeiras duas componentes são parâmetros usuais de textura `S` e `T` para quele vértice; o terceiro pode ser especificado como zero.

'`C`' Cada vértice (veja Vertices acima) inclui uma cor de quatro componentes.

'`N`' Cada vértice inclui um vetor normal à superfície.

'`Z`' Dos valores dos 3 eixos coordenados (x , y e z) de vértice somente a componente Z está presente; X e Y são omitidos, e assumidos ambos como sendo iguais às coordenadas de malha (u,v) de forma que X varia de 0 .. (Nu-1), Y varia de 0 .. (Nv-1) onde Nu e Nv são as dimenssões de malha – veja acima.

'`4`' Vértices são quadridimensionais, cada vértice consiste em 4 valores em ponto flutuante. `Z` e `4` não podem ambos estarem presentes ao mesmo tempo.

'u' A malha é ajustada na direção u, de forma que o (0,v)'ésimo vértice está conectado ao (*Nu*-1,v)'ésimo para todo v.

'v' A malha é ajustada na direção v, de forma que o (u,0)'esimo vértice está conectado ao (u,*Nv*-1)'ésimo para todo u. Dessa forma uma malha u-ajustada ou v-ajustada é topologicamente um cilindro, enquanto uma malha uv-ajustada é um toro.

'n' Especifica uma malha cujos vértices existem em um espaço de dimensão mais alta. A dimensão segue a palavra chave "MESH". Cada vértice então tem *Ndim* componentes.

Note que a ordem dos caracteres do prefixo é significativa; uma malha colorida, u-ajustada é uma `CuMESH` não uma `uCMESH`.

Seguindo o cabeçalho da malha estão os inteiros *Nu* e *Nv*, as simensões da malha.

Então segue-se *Nu***Nv* vértices, cada um desses vértices na forma fornecida através do cabeçalho. Eles aparecem na sequência v-crescente, i.e. se chamarmos cada vértice de (u,v) então os vértices aparecerão na ordem

```
(0,0) (1,0) (2,0) (3,0) ...  (Nu-1,0)
(0,1) (1,1) (2,1) (3,1) ...  (Nu-1,1)
...
(0,Nv-1) ...  (Nu-1,Nv-1)
```

O formato `MESH BINARY` é aceito; Veja seção 4.1.8 [Formato binario], página 67. Os valores de *Nu* e *Nv* são inteiros de 32-bit; todos os outros valores são números em ponto flutuante de 32-bit.

4.2.3 BBOX: Caixas associada simples

Esse é um objeto-brinquedo muito simples: Toma 2 vértices e desenha um (hiper-) cubo que é a caixa associada dos dois vértices.

Sintaxe:

```
BBOX
x[0] y[0] z[0]
x[1] y[1] z[1]
```

ou

```
4BBOX
x[0] y[0] z[0] w[0]
x[1] y[1] z[1] w[1]
```

ou

```
nBBOX
Ndim # > 3
x[0] y[0] z[0] w[0] ...
x[1] y[1] z[1] w[1] ...
```

ou

```
4nBBOX
Ndim # > 3
d[0] x[0] y[0] z[0] w[0] ...
```

```
d[0] x[1] y[1] z[1] w[1] ...
```

Não existe formato binário BBOX. O modificador `4` tem diferentes significados dependendo da dimensão da caixa associada: `4BBOX` significa que as 4 componentes dos vértices constroem uma caixa associada tetradimensional. Usando `4` em conjunção com `n` – `4nBBOX` *NDim* – significa que os vértices especificados no arquivo possuem *NDim+1* componentes, mas a componente no índice 0 é o divisor homogêneo (em oposição ao caso comum tridimensional onde o divisor homogêneo pode ser `w` – a terceira – componente).

4.2.4 Superfícies de Bezier

O sufixo de arquivo convencional para arquivos de superfície de Bezier é '`.bbp`' ou '`.bez`'. O arquivo com qualquer dos dois sufixos pode conter qualquer dos dois tipos de parte de superfície de Bezier.

Sintaxe:

```
  [ST]BBP -ou- [C]BEZ<Nu><Nv><Nd>[_ST]
# Nu, Nv são direções nos eixos u e v
# graus de polinômios variam 1..6
# Nd = dimensão: 3->3-D, 4->4-D (racional)
# (O sinal '<' e o sinal '>' não aparecem na entrada.)
# Nu,Nv,Nd são cada um dígito decimal simples.
# a forma BBP implica Nu=Nv=Nd=3 de forma que BBP = BEZ333.

# Qualquer número de partes de superfície de Bezier segue o cabeçalho
# (Nu+1)*(Nv+1) pontos de controle do pedaço da sperfície de Bezier
# cada 3 ou 4 números em ponto flutuante conforme o cabeçalho
  vertex(u=0,v=0)  vertex(1,0) ... vertex(Nu,0)
  vertex(0,1)    ... vertex(Nu,1)
  ...
  vertex(0,Nv)    ... vertex(Nu,Nv)

# coordenadas de textura ST se mencionado no cabeçalho
  S(u=0,v=0) T(0,0) S(0,Nv) T(0,Nv)
  S(Nu,0) T(Nu,0) S(Nu,Nv) T(Nu,Nv)

# número em ponto flutuante com 4 componentes no intervalo (0..1) de
# cores R G B A para cada canto se mencionado no cabeçalho
  RGBA(0,0)   RGBA(0,Nv)
  RGBA(Nu,0)  RGBA(Nu,Nv)
```

Esses formatos representam coleções de partes de superfícies de Bezier, de graus maiores que 6, e com vértices 3-D ou 4-D (racionais).

A palavra chave de cabeçalho pode assumir as formas `[ST]BBP` ou `[C]BEZ<Nu><Nv><Nd>[_ST]` (os símbolos '`<`' e '`>`' não são parte da palara chave.

O prefixo `ST` sobre `BBP`, ou o sufixo `_ST` sobre `BEZuvn`, indicam que cada pedaço de superfície inclui quatro pares de pontos com coordenadas em ponto flutuante no espaço de textura, um em cada canto do pedaço.

O prefixo `C` sobre `BEZuvn` indica um pedaço colorido, incluindo quatro conjuntos de quatro componentes com os números que especificam as cores em ponto flutuante (vermelho, verde, azul, e alfa) no intervalo 0..1, uma cor para cada canto.

Nu e *Nv*, cada um é um simples dígito no intervalo 1..6, são os graus do polinômio do pedaço nas direções u e v respectivamente.

Nd é o número de componentes no vértice de cada pedaço, e deve ser ou `3` para 3-D ou `4` para coordenadas homogêneas, isto é, pedaços racionais.

Pedaços `BBP` são pedaços bicúbicos com vértices tridimensionais, de forma que `BBP` = `BEZ333` e `STBBP` = `BEZ333_ST`.

Qualquer número de pedaços segue o cabeçalho. Cada pedaço compreende uma série de vértices do pedaço, seguido por coordenadas opcionais de textura (s,t), seguidas por cores opcionais no formato (r,g,b,a).

Cada pedaço tem (*Nu*+1)*(*Nv*+1) vértices na ordem v-crescente, de forma que se designarmos um vértice através de seus índices (u, v) de controle de ponto a ordem é

```
(0,0) (1,0) (2,0) ...  (Nu,0)
(0,1) (1,1) (2,1) ...  (Nu,1)
...
(0,Nv)             ...  (Nu,Nv)
```

com cada vértice contendo ou 3 ou 4 números em ponto flutuantes como especificado pelo cabeçalho.

Se o cabeçalho chama por coordenadas ST, quatro pares de números em ponto flutuante seguem: o espaço de coordenadas de textura para (0,0), (*Nu*,0), (0,*Nv*), e os cantos (*Nu*,*Nv*) do pedaço, respectivamente.

Se o cabeçalho chama por cores, segue quatro grupos de quatro componentes (vermelho, verde, azul, alfa) d cores em ponto flutuante, um para cada canto do pedaço.

A série de pedaços termina em um caractere de fim de arquivo, ou com uma chave fechada se incorporado em uma referência de objeto.

4.2.5 Arquivos do Tipo OFF

O sufixo convencional para arquivos `OFF` é '`.off`'.

Sintaxe:

```
[ST][C][N][4][n]OFF # Palavra chave do cabeçalho
[Ndim] # Dimensão do espaço dos vértices, presente somente se nOFF
# estiver também presente
NVértices  NFaces  NArestas   # NArestas não é usado nem checado

x[0]  y[0]  z[0] # Vértices, possivelmente com normais,
# cores, e/ou coordenadas de textura, nessa ordem,
# se os prefixo N, C, ST
# estiverem presentes.
# Se 4OFF, cada vértice possui 4 componentes,
# incluindo uma componente final homogênea.
# Se nOFF, cada vértice possui Ndim componentes.
# Se 4nOFF, cada vértice possui Ndim+1 componentes.
```

```
...
x[NVértices-1]  y[NVértices-1]  z[NVértices-1]

     # Faces
     # Nv = # vértices na referida face
     # v[0] ... v[Nv-1]: índices dos vértices
     # no intervalo 0..NVértices-1
Nv  v[0] v[1] ... v[Nv-1]  colorspec
...
     # colorspec colar contínuo v[Nv-1]
     # até o aparecer um caractere de fim de linha; pode ser de 0 a 4
     # números
     # nenhum: padrão
     # inteiro: índice do mapa de cores
     # 3 ou 4 inteiros: valores RGB[A] no intervalo 0..255
# 3 ou 4 números em ponto flutuante: valores RGB[A] no intervalo 0..1
```

Arquivos `OFF` (nome para "object file format" formato de arquivo de objeto) representa coleções de polígonos planos com vértices possivelmente compartilhados, um caminho conveniente para descrever poliedros. Os polígonos podem ser côncavos mas não existe suporte para polígonos contendo buracos.

Um arquivo `OFF` pode começar com a palavra chave `OFF`; isso é recomendado mas também é opcional, muitos arquivos existentes precisam dessa palavra chave.

Três inteiros ASCII seguem a palavra chave `OFF`: *NVértices*, *NFaces*, e *NArestas*. Esses são o número de vértices, faces, e arestas, respectivamente. Atualmente o software não utiliza nem verifica *NArestas*; ele não precisa ser correto mas deve estar presente.

As coordenadas do vértice seguem: dimensão * *Nvértices* valores em ponto flutuante. Esses valores em ponto flutuante estão implicitamente numerados de 0 a *NVértices*-1. A dimensão é ou 3 (o padrão) ou 4 (especificado pelo caractere chave `4` diretamente antes da palavra chave `OFF`).

Seguindo esses acima citados estão as descrições das faces, tipicamente escritos com uma linha por face. Cada linha tem a forma

```
N  Vert1 Vert2 ... VertN  [cor]
```

Aqui *N* é o número de vértices sobre a considerada face, e *Vert1* a *VertN* são índices dentro da lista de vértices (no intervalo 0..*NVértices*-1).

O modificador opcional *cor* no final da linha acima pode tomar várias formas. Caracteres de fim de linha são significativos nesse ponto: a descrição *cor* inicia-se após *VertN* e termina com o caractere de fim de linha (ou próximo caractere cerquilha # representativo de comentário). Uma *cor* pode ser:

ausência de caractere
: a cor padrão

um inteiro
: índice dentro "do" mapa de cores; veja abaixo

três ou quatro inteiros
: valores de RGB e possivelmente alfa no intervalo 0..255

três ou quatro números em ponto flutuante
: valores RGB e possivelmente alfa no intervalo 0..1

Para o caso de um inteiro, o mapa de cores é lido diretamente do arquivo '`cmap.fmap`' em sua forma atual no diretório '`data`' do Geomview. Algum melhor mecanismo para fornecer um mapa de cor será fornecido provavelmente algum dia.

O significado de "cor padrão" varia. Se nenhuma face do objeto tem uma cor, tudo recebe como herança a cor material padrão do ambiente. Se alguma mas não todas as faces possuem cores, o padrão é cinza (R,G,B,A=.666).

Um formato `[ST][C][N][n]OFF BINARY` é aceito; veja seção 4.1.8 [Formato binario], página 67. Esse formato assemelha-se ao formato ASCII em quase tudo que você poderia esperar, com inteiros de 32-bit para todos os contadores e índices de vértice e números em ponto flutuante de 32-bit para posições de vértice (e coordenadas de textura ou cores de vértice ou retas normais se algum dos formatos `COFF`/`NOFF`/`CNOFF`/`STCNOFF`/etc. estiver presente).

Exceção: cada um dos índices de face do vértice são seguidos por um inteiro indicando quantas componentes de cor o acompanham. Componentes de cor de face devem ser números em ponto flutuante, não valores inteiros. Dessa forma uma face triangular pouco colorida pode ser representada como

```
int int int int int
3   17   5   9   0
```

enquanto a mesma face colorida com vermelho pode ser

```
int int int int int float float float float
 3  17   5   9   4   1.0   0.0   0.0   1.0
```

4.2.6 Arquivos do Tipo VECT

O sufixo convencional para arquivos `VECT` é '`.vect`'.

Sintaxe:

```
[4]VECT
NLinhaspoligonais  NVértices  NCores

Nv[0] ... Nv[NLinhaspoligonais-1]     # número de vértices
                                          # em cada linha poligonal

Nc[0] ... Nc[NLinhaspoligonais-1]     # número de cores fornecido
                                          # em cada linha poligonal

Vert[0] ... Vert[NVertices-1]  # Todos os vértices
                                          # (3*NVertices números em
                                          # ponto flutuante)

Color[0] ... Color[NCores-1]  # Todas as cores
                                          # (4*NCores números em ponto
                                          # flutuante, RGBA)
```

Objetos do tipo `VECT` representam listas de linhas poligonais (sequência de caracteres que representam segmentos de reta, possivelmente fechados). Uma linha poligonal degenerada pode ser usada para representar um ponto.

Um arquivo `VECT` inicia-se com a palavra chave `VECT` ou com `4VECT` e três inteiros: *NLinhas*, *NVértices*, e *NCores*. Aqui *NLinhas* é o numero de linhas poligonais no arquivo, *NVértices* o número total de vertices vértices, e *NCores* o número de cores como explanado abaixo.

A seguir vem *NLinhas* que são inteiros de **16-bit**

Nv[0] Nv[1] Nv[2] . . . Nv[NLinhas-1]

fornecendo o número de vértices em cada linha poligonal. Um número negativo indica uma linha poligonal fechada; 1 denota ponto composto por um pixel simples. O somatório (dos valores absolutos) de *Nv[i]* deve ser igual a *NVértices*.

A seguir vem *NLinhas* que são formadas por inteiros de **16-bit** *Nc[i]*: o número de cores em cada linha poligonal. Normalmente um dos três valores abaixo:

0 Nenhuma cor é especificada para esta linha poligonal. Seu desenho na mesma cor que a linha poligonal anterior.

1 Uma cor simples é especificada. A linha poligonal completa é desenhada nessa cor.

abs(*Nv[i]*) Cada vértice tem uma cor. Ou cada segmento é desenhado na correspondente cor, ou as cores são linearmente interpoladas ao longo dos segmentos de reta, dependendo da implementação.

A seguir vem *NVértices* grupos de 3 ou 4 números em ponto flutuante: as coordenadas de todos os vértices. Se a palavra chave for *4VECT* então existirão 4 valores por vértice. O primeiro grupo abs(*Nv[0]*) forma a primeira linha poligonal, o grupo seguinte abs(*Nv[1]*) forma a segunda e assim por diante.

Finalmente *NCores* grupos de 4 números em ponto flutuante fornecendo valores de vermelho, verde, azul e alfa (opacidade). O primeiro grupo *Nc[0]* aplica-se à primeira linha poligonal, e assim por diante.

Um formato *VECT BINARY* é aceito; veja seção 4.1.8 [Formato binario], página 67. O formato binário segue exatamente o formato ASCII, com inteiros de 32-bit Big-Endian onde aparecem os números inteiros comuns, e com inteiros de **16-bit** Big-Endian onde aparecem inteiros de 16-bit; números em ponto flutuante de 32-bit Big-Endian onde aparecem os valores reais. **NOTA REALMENTE GRANDE:** Os contadores de vértice *Nv[i]* e os contadores de cor *Nc[i]* são inteiros de **16-bit** Big-Endian.

4.2.7 Arquivos do Tipo SKEL

Arquivos do Tipo `SKEL` representam coleções de pontos e linhas poligonais, com vértices compartilhados. O sufixo convencional para arquivos `SKEL` é '`.skel`'.

Sintaxe:

```
[C][4][n]SKEL
[NDim]                    # Dimensão dos vértices, presente somente
    # se nSKEL também estiver presente
NVértices  NLinhaspoligonais
```

```
x[0]  y[0]  z[0]  # Vértices
    # se 4SKEL, cada vértice terá 4 componentes
    # se nSKEL, each vértice terá NDim componentes
                                  # se C[4][n]SKEL coordenadas de
                                  # vértice são seguidas por uma
                                  # especificação de cor RGBA
...
x[NVértices-1]  y[NVértices-1]  z[NVértices-1]

                        # linhas poligonais
                        # Nv = vértices sobre essa linha poligonal
                        # (1 = ponto)
                        # v[0] ... v[Nv-1]: índices de vértice
                        #               no intervalo 0..NVértices-1
Nv  v[0] v[1] ... v[Nv-1]  [colorspec]
...
                        # colorspec continua adiante de v[Nv-1]
                        # atém encontrar um fim de linha; pode ser
                        # vazia, ou 3 ou 4 números.
                        # vazia: cor padrão
# 3 ou 4 números em ponto flutuante: valores RGB[A] no intervalo 0..1
```

A sintaxe é semelhante à sintaxe dos arquivos `OFF`, com uma tabela de vértices seguidos de uma sequência de descrições de linhas poligonais, cada descrição referindo-se a vértices através de índices na tabela. Cada linha poligonal tem uma cor opcional.

Para objetos `nSKEL`, cada vértice tem *NDim* componentes. Para objetos `4nSKEL`, cada vértice tem *NDim+1* componentes; a componente final é o divisor homogêneo.

Um formato *[4][n]SKEL BINARY* é aceito; veja seção 4.1.8 [Formato binario], página 67. Esse formato assemelha-se ao formato ASCII na maioria das formas que você pode esperar, com inteiros de 32-bit para todos os contadores e índices de vértice e números em ponto flutuante de 32-bit para posições de vértice.

Exceção: cada índice do vértice de linhas poligonais é seguido de um inteiro indicando quantas componentes de cor o acompanham. Componentes de cor de linhas poligonais devem ser inteiros em ponto flutuante, não valores inteiros. Dessa forma uma pouco colorida linha poligonal com 3 vértices pode ser representada como

```
int int int int int
3   17   5   9   0
```

enquanto a mesma linha poligonal colorida em vermelho pode ser

```
int int int int int float float float float
 3  17   5   9   4   1.0   0.0   0.0   1.0
```

4.2.8 SPHERE Files

O sufixo convencional para arquivos `SPHERE` é '`.sph`'.

```
[ST][E|H|S]SPHERE # Palavra chave
# coordenadas de textura geradas automaticamente, somente permitido
```

```
# com objetos STSPHERE [SINUSOIDAL|CYLINDRICAL|RECTANGULAR|
# STEREOGRAPHIC|ONEFACE]
# os próximos quatro campos são requeridos
Radius
Xcenter Ycenter Zcenter
```

A palavra chave é `[ST][E|H|S]SPHERE`. Os caracteres prefixados opcionais significam:

'`ST`' A esfera é desenhada com coordenadas de textura geradas automaticamente. Veja abaixo.

'`E`' Assume-se que a esfera encontra-se dentro do espaço Euclidiano.

'`H`' Assume-se que a esfera encontra-se dentro de um espaço Hiperbólico. Veja Capítulo 8 [Geometrias Nao-Euclidianas], página 161.

'`S`' Assume-se que a esfera encontra-se dentro de um espaço esférico. Veja Capítulo 8 [Geometrias Nao-Euclidianas], página 161.

Objetos do tipo esfera são desenhados usando malhas que são retangulares em um sistema de coordenadas polares, com o plano equatorial paralelo ao plano `x,y`. Sua suavidade superficial, e o tempo gasto para desenhá-la, depende da escolha dos cortes quadrados, 10x10 por padrão. Para Geomview, o painel de aparência, o comando de teclado `<N>ad`, ou um atributo de aparência `dice nu nv` escolhe isso.

Coordenadas de textura são geradas para objetos `STSPHERE`; a palavra chave seguindo a palavra chave inicial `STSPHERE` define o caminho para fazer isso. A palavra chave que segue a papavra chave inicial segue as convenções do script perl `mktxmesh` que acompanha o diretório *Orrery* na árvore de diretórios instalados que acompanha o Geomview.

SINUSOIDAL
: projeção sinusoidal de área equivalente

CYLINDRICAL
: projeção cilídrica: s é a longitude, t é a latitude

RECTANGULAR
: projeção retangular: s é a longitude, t é `sin(latitude)` (i.e. a coordenada `z` no sistema de coordenadas da esfera)

STEREOGRAPHIC
: projeção estereográfica a partir do pólo sul (`z=-1`)

ONEFACE
: visão ortográfica extendida do hemisfério `+y` sobre ambos, espelhando

4.2.9 Arquivos do Tipo INST

O sufixo convencional para um arquivo `INST` é '`.inst`'.

Não existe formato INST BINARY.

Um `INST` aplica uma transformação 4x4 (ou *(N+1)*x*(N+1)* no contexto de ND-viewing) a outro objeto OOGL. Um arquivo do tipo `INST` inicia-se com `INST` seguido de as seções mostradas adiante que podem vir em qualquer ordem:

`geom` *`oogl-object`*

especifica o objeto OOGL a ser instanciado. Veja seção 4.1.9 [Referencias], página 68, para a sintaxe de um *oogl-object*. palavra chave `unit` é um sinônimo para `geom`.

`transform` `["{"] transformação 4x4 ["}"]`

especifica uma matriz de transformação simples. Ou a matriz pode aparecer literalmente como 16 números, ou a matriz pode aparecer como uma referência a um objeto "transform", i.e.

`"<" arquivo-contendo-matriz-4x4`

ou

`":" símbolo-representando-transformação-de-objeto`

Outra forma de especificar a transformação é

`transforms`
`objeto-oogl`

O *objeto-oogl* deve ser um objeto `TLIST` (lista de transformações), ou uma `LIST` cujos membros dessa lista são objetos `TLIST` mais recentes. Com efeito, a palavra chave `transforms` toma uma coleção de matrizes 4x4 e replica o objeto `geom`, fazendo uma cópia para cada matriz 4x4.

Se nem a palavra chave `transform` nem a palavra chave `transforms` aparecerem, nenhuma transformação é aplicada (atualmente a identidade é aplicada). Você pode usar isso para, e.g., empacotar uma aparência em torno de um objeto externo fornecido, através de uma LIST de membros simples pode-se fazer isso mais eficientemente.

Veja seção 4.1.6 [Matrizes de transformacao], página 67, para o formato de matriz.

A cópia de um objeto geométrico simples por meio de um objeto `TLIST` (veja '`transforms`' acima) pode ser útil para transformar coordenadas de textura através de outra lista de transformações; essa lista pode ser especificada por

`txtransforms`
`objeto-TLIST`

O número de transoformações de textura deve coincidir com o número de transformações geométricas. O objeto `SPHERE` (Veja seção 4.2.8 [SPHERE], página 84) utiliza essa técnica para gerar um esfera completamente texturizada fora de alguma fração de uma esfera (usualmente um octante).

Uma transformação *(N+1)*-dimensional pode ser especificada por

`ntransform ["{"] N+1 N+1 (N+1)x(N+1) floats ["}"]`

A linha acima fornece uma matriz de transformação *N+1*-dimensional. Ou a matriz pode aparecer literalmente como *(N+1)*x*(N+1)* números, ou pode existir uma referência a um objeto '`ntransform`', i.e.

`"<" arquivo-contendo-matriz-`*`(N+1)`*`x`*`(N+1)`*

ou

`":" símbolo-representando-objeto-ntransform`

Veja seção 4.1.7 [Matrizes de transformacao ND], página 67, para o formato de matriz.

Mais dois campos INST são aceitos: `location` e `origin`.

Note que `location` bem como `origin` são ignorados se esse objeto `INST` realiza uma `ntransform`. Também, se a visualização ND está ativada (comando `ND-axes`, veja Capítulo 7 [GCL], página 127) então objetos `INST` com `origin` diferente de `local` não irão ser desenhados, embora o material `location` possa trabalhar (ou não).

```
location [global ou camera ou ndc ou screen ou local]
```

Normalmente um INST especifica uma posição relativa a seu objeto pai; o campo `location` permite colocar um objeto em qualquer lugar.

- `location global` anexa o objeto ao ambiente do sistema de coordenadas global (também conhecido como "universe") – o mesmo dos objetos mundo do Geomview, das geometrias alienígenas, e câmeras são colocados.
- `location camera` coloca o pai do objeto como sendo uma câmera. (Dessa forma se houverem multiplas visualizações, essas visualizações podem aparecer em uma diferente posição espacial em cada visualização.) O centro de visão da camera está em volta da parte negativa do seu eixo z; X positivo está à direita e adiante, Y positivo está acima e adiante. Normalmente as unidades do espaço da câmera são as mesmas das coordenadas globais. Quando uma câmera é colocada de volta à sua posição inicial, a origem global está localizada em (0,0,-3.0).
- `location ndc` coloca os parentes do objeto no cubo unitário normalizado no qual a projeção da câmera (perspectiva ou ortográfica) mapeia o objeto mundo visível. X, Y, e Z estão no intervalo de -1 a +1, com Z = -1 estando mais próximo e Z = +1 adiante do plano de corte, e X e Y nas direções à direita e adiante e acima e adiante respectivamente. Dessa forma alguma coisa como

  ```
  INST  transform  1 0 0 0  0 1 0 0  0 0 1 0  -.9 -.9 -.999 1
        location ndc
        geom < label.vect
  ```

  cola `label.vect` dentro do canto infeiror esquerdo de cada janela, e em frente de tudo mais que estiver próximo, assumindo o conteúdo `label.vect` como localizado no quadrante positivo do plano XY. É tentado usar -1 em lugar de como a componente Z da posição, mas o -1 pode colocar o objeto apenas nas proximidades em lugar de muito perto do plano de corte e tornar o objeto (parcialmente) invisível, devido a algum erro de cálculo com números em ponto flutuante.
- `location screen` coloca o objeto objeto em coordenadas de tela. O intervalo de Z é ainda de -1 a +1 como para coordenadas ndc; X e Y são medidos em pixels, e a posição de (0,0) localiza-se no canto *inferior esquerdo* da janela, avançado para a direita e adiante e para cima e adiante.

`location local` é o padrão; o objeto está posicionado relativametne a seus genitores.

```
origin [global ou camera ou ndc ou screen ou local] x y z
```

O campo `origin` reposiciona o conteúdo da INST de forma que o local da origem seja o ponto especificado do sistema de coordenadas fornecido. A menos que `location` seja especificado, essa opção não muda a orientação, somente a escolha da origem. Ambas as opções `location` e `origin` podem ser usadas juntas.

Então por exemplo

```
{ INST
  location screen
```

```
  origin ndc 0 0 -.99
  geom { < xyz.vect }
  transform { 100 0 0 0  0 100 0 0  0 0 -.009 0   0 0 0 1 }
}
```

coloca a origem de xyz.vect no centro da janela, apenas transformando o plano de corte mais próximo. O comprimento da unidade das arestas X e Y são ajustados proporcionalmente para ter o comprimento de apenas 100 unidades de tela – pixels – , independentemente do tamanho da janela.

4.2.9.1 Exemplos INST

Aqui estão alguns exemplos de arquivos `INST`

```
INST
     unit < xyz.vect
     transform {
        1 0 0 0
        0 1 0 0
        0 0 1 0
        1 3 0 1
     }

{ appearance { +edge  material { edgecolor 1 1 0 } }
    INST geom < mysurface.quad }

{INST transform {: T} geom {<dodec.off}}

{ INST
     transforms
         { LIST
      { < some-matrices.prj }
      { < others.prj }
      { TLIST <still more of them> }

         }
     geom
         { # material copiado a partir de todas as matrizes acima
       ...
         }
}
```

O exemplo adiante assemelham-se ao exemplo de `origin` na seção acima, mas fazem com que as arestas X e Y sejam 1/4 do tamanho da janela (1/4, não 1/2, uma vez que o intervalo das coordenadas ndc de X e Y estejam de -1 a +1).

```
{ INST
  location ndc
  geom { < xyz.vect }
  transform { .5 0 0 0  0 .5 0 0  0 0 -.009 0   0 0 -.99 1 }
}
```

4.2.10 Arquivos do Tipo LIST

O sufixo convencional para um arquivo `LIST` é '`.list`'.

Uma lista de objetos OOGL

Sintaxe:

```
LIST
    oogl-object
    oogl-object
    ...
```

Note que não existe separação explícita entre os objetos oogl, de forma que eles podem ser colocados entre chaves (`{ }`) por questões de clareza. Da mesma forma não existe um marcador explícito para o final da lista; a menos que o arquivo apareça sozinho em um arquivo de disco, a construção completa pode também ser empacotada entre chaves, como em:

```
{ LIST { QUAD ... } { < xyz.quad } }
```

Uma `LIST` vazia, i.e. `{ LIST }`, é válida, e é o caminho mais fácil para criar um objeto vazio. Por exemplo, para remover uma definição de símbolo você pode escrever

```
{ define algumsímbolo  { LIST } }
```

4.2.11 Arquivos do Tipo TLIST

O sufixo convencional para um arquivo `TLIST` é '`.grp`' ("grupo") ou '`.prj`' (matrizes "projetivas").

Coleções de matrizes 4x4, usadas na seção `transforms` de/e objeto `INST`.

Sintaxe:

```
TLIST # palavra chave
<matriz 4x4 (16 números em ponto flutuante)>
... # qualquer número de matrizes 4x4
transform {             # referência a um objeto de transformação
<objeto de transformação (pode ser um controlador)>
}
tlist {                 # TLIST aninhada
<objeto TLIST OOGL (pode ser um controlador)>
}
```

Objetos `TLIST` são usados somente dentro de cláusulas `transforms` de um objeto `INST`. Objetos `TLIST` fazem com que os objetos `geom` `INST`s sejam instanciados uma vez sob cada uma das transformações na `TLIST`. O efeito é como aquele de um `LIST` de `INST`s cada uma com uma transformação simples, e todas as transformações referindo-se ao mesmo objeto, mas é mais eficiente.

`TLIST`s podem ser aninhadas: efetivamente isto significa que todas as transformações em cada objeto `TLIST` aninhado são multiplicadas (à esquerda) através das transformações no objeto `TLIST` mais externo.

Esteja informado de que uma `TLIST` é um tipo de geom, distinto de um objeto `transform`. Alguns contextos esperam um geom, alguns contextos esperam um `transform`. Por exemplo em

```
INST transform { : meuT } geom { ... }
```

meuT deve ser um objeto transform, que pode ter sido criado com a GCL

```
(read transform { define meuT 1 0 0 1 ... })
```

enquanto em

```
INST transforms { : meusTs } geom { ... }
ou
INST transforms { LIST {: meusTs} {< more.prj} } geom { ... }
```

meusTs deve ser um objeto geom, definido e.g. com

```
(read geometry { define meusTs { TLIST 1 0 0 1 ... } })
```

O formato `TLIST BINARY` é aceito. Dados binários iniciam-se com um inteiro de 32-bit fornecendo o números de transformações, seguido por aquele número de matrizes 4x4 no formato em número de ponto flutuante de 32-bit. A ordenação dos elementos da matriz é a mesma do formato ASCII.

4.2.12 Arquivos do Tipo GROUP

Esse formato é obsoleto, mas ainda é aceito. It combined the functions of `INST` and `TLIST`, taking a series of transformations and a single Geom (`unit`) objeto, and replicating the objeto under each transformation.

```
GROUP ... < matrices > ... unit { oogl-object }
```

is still accepted and effectively translated into

```
INST
transforms { TLIST ... <matrices> ... }
unit { oogl-object }
```

4.2.13 Arquivos do Tipo DISCGRP

Esse formato é para grupos discretos, tais como aparecem na teoria dos coletores (manifolds) ou em modelos de simetria. Esse formato tem sua própria página de manual. Veja discgrp(5).

4.2.14 Objetos do Tipo COMMENT

O objeto COMMENT é um mecanismo para codificar dados arbitrários dentro de um objeto OOGL. O objeto COMMENT pode ser usado para manter trilhas de dados ou para passagem de dados de retorno e passar dados adiante entre módulos externos.

Sintaxe:

```
COMMENT                 # palavra chave

nome tipo   # nome individual e especificador de tipo
{ ... }             # dados arbitrários
```

Os dados, que devem ser contidos dentro de chaves, podem incluir qualquer coisa exceto chaves não balanceadas (aberta mas sem o fechamento correspondente ou fechada mas sem a abertura correspondente). O campo *tipo* pode ser usado para identificar dados de interesse de um programa em particular apesar de nomear convenções.

Objetos `COMMENT` são intendidos como estando associados com outros objetos apesar da inclusão em um objeto `LIST`. (Veja seção 4.2.10 [LIST], página 89.) A sintaxe da

cerquilha "#" OOGL de comentário não é suficiente para troca de dados uma vez que esses comentários são removidos quando um objeto OOGL é lido dentro do Geomview. O objeto `COMMENT` é preservado quando chamado dentro do Geomview e é mantido intacto na saída.

Aqui está um exemplo associando uma localização na internet a uma peça de geometria:

```
{ LIST
 { < Tetrahedron}
 {COMMENT GCHomepage HREF { http://www.geomview.org/ }}
}
```

Um formato de `COMMENT` binário é aceito. Esse formato binário de comentário não é consistente com outros formatos binários OOGL. Veja seção 4.1.8 [Formato binario], página 67. O `nome` e o `tipo` são seguidos por

```
N Byte1 Byte2 ... ByteN
```

Em lugar de dados contidos dentro de chaves.

4.3 Objetos nao-geometricos

A sintaxe desses objetos é fornecida na forma usada em seção 4.1.9 [Referencias], página 68, onde itens "entre aspas duplas" devem aparecer literalmente mas sem aspas duplas, itens entre colchêtes ([ ]) são opcionais, e a barra vertical "|" separa escolhas alternativas.

4.3.1 Objetos de aparencia

Aparências são objetos OOGL com características próprias, que simplesmente significam que é possível fornecer a eles nomes simbólicos (Veja seção 4.1.9 [Referencias], página 68). Existem outras seções manuseado detalhes de aparência. Veja seção 4.1.10 [Aparencias], página 69.

4.3.2 Objetos de imagem

Objetos de imagem são usados para especificar dados de pixmap ou para texturas (Veja seção 4.1.11 [Mapeamento de Textura], página 74), ou para imagens de fundo de câmeras (Veja seção 7.2.19 [camera], página 131).

Ao mesmo tempo em que imagens são escritas elas são também comprimidas de 1 a 4 canais, um canal fornece um número simples no intervalo que vai de 0 a `maxval` para cada ponto de imagem (pixel); e `maxval` é colocado em 255. A interpretação dos dados de imagem dependente do número de canais é como segue:

#Canais	**No. do Canal**	**Interpretação**
1	1	escala de cinza ou dados de luminância
2	1	escala de cinza ou dados de luminância
	2	canal alfa (0: transparente, `maxval`: opaco)
3	1	canal vermelho
	2	canal verde
	3	canal azul
4	1	canal vermelho
	2	canal verde
	3	canal azul

4 canal alfa (0: transparente, `maxval`: opaco)

Dados de imagem podem ser especificados inline (embutidos dentro do fluxo de dados atual) ou via referências de arquivos; em ambos os casos os dados são lidos e interpretados ao mesmo tempo que o objeto de imagem é passado. *Essa forma é diferente da antiga (e desatualizada) forma de especificação de textura de imagem, onde os dados de imagem em uma mídia podem eventualmente serem re-lidos pelo Geomview.*

A sintaxe geral de objetos de imagem é como segue:

```
<image> ::=
  [ "{" ]              (abertura de chave, geralmente precisam informar
                        o fim do objeto de forma clara.)
  [ "image" ]           (palavra chave opcional; desnecessária se o tipo
                        é determinado pelo contexto, o que
                        usualmente acontece.)
  [ "define" <nome> ]
                        (define uma imagem chamada <nome>, escolhendo
                        seus valores a partir do material adiante)
  |
      "<" <nomedearquivo>   (significando: leia a imagem de contida
                        em nomedearquivo)
  |
      ":" <nome>        (significando: use a variável nome,
                        definida em algum lugar; se a variável não
                        for definida em algum lugar a imagem
                        é tida como vazia)
  |
                        (material atual de definição de imagem; dados
                        da imagem obrigatóriamente vêm por ultimo,
                        após a definição da largura e da altura
                        e do número de canais)

      "width"           (largura da imagem, detectado automaticamente
                        a partir dos dados da imagem se possível)

      "height"          (altura da imagem, detectado automaticamente
                        a partir dos dados da imagem se possível)

      "channels"        (número de canais, detectado automaticamente
                        a partir dos dados da imagem e a partir das
                        especificações data descrita
                        adiante, se possível)

      "maxval"          (não suportado, obrigatoriamente deve ser 255 se
                        especificado)

      "data MASCDEST [FILTER] [{] < NOMEDOARQUIVO [}]"
      "data MASCDEST [FILTER] TAMAN_IMAGEM [{][\n]DADOS_LIT_IMAGEM[}]"
```

```
                    (dados de imagem ou externos ou embutidos,
                    veja abaixo para uma descrição detalhada do
                    significado de MASCDEST e FILTER. Uma imagem
                    pode -e tem, em geral- multiplas seções de
                    dados.)

    [ "}" ]         (fechamento correspondente da chave)
```

Detalhes relativos à especificação dos dados de uma imagem:

'MASCDEST'

Esse é um campo-bit descrevendo onde os dados da imagem especificada devem ser colocados no pixmap de destino. O campo-bit é especificado por meio de um inteiro em um dos formatos conhecidos (decimal, octal, hexadecimal). Os canais dos dados fonte são sempre enumerados consecutivamente. Se, e.g. 'NOMEDOARQUIVO' ou 'DADOS_LIT_IMAGEM' especificam um imagem (provavelmente RGB ...) de três canais e 'MASCDEST' for igual a 0xD (i.e. o primeiro bit é 0), então o terceiro canal do pixmap fonte pode ser substituído no quarto canal do objeto imagem de destino (o canal alfa), o segundo canal pode determinar o valor de destino 'azul' e o primeiro canal da fonte determina o valor de destino correspondente ao 'vermelho'.

O número de canais dos dados fontes sempre tem que coincidir com o número de bits especificado como 'MASCDEST'. **Exceção:** se o pixmap fonte possui somente um canal, então o número de canais dos dados fonte pode ser usado para preencher qualquer número de canais de destino; todos os canais especificados em 'MASCDEST' são preenchidos com os dados do canal simples do pixmap fonte.

Geomview conhece as seguintes constantes simbólicas, que podem ser usadas em lugar de especificar o campo-bit 'MASCDEST' numericamente:

LUMINANCE
: o mesmo que '1', '0x1', '\01'

LUMINANCE_ALPHA
: o mesmo que '3', '0x3', '\03'

RGB
: o mesmo que '7', '0x7', '\07'

RGBA
: o mesmo que '15', '0xf', '\017'

ALPHA
: dependendo do contexto: o número absoluto de canais deve obrigatoriamente ser conhecido; i.e. 'data ALPHA ...' deve obrigatoriamente ser colocado antes de alguma coisa de forma a determinar o número de canais da imagem, e.g.

```
...
data RGB ...
data ALPHA
...
```

é válido, mas

```
<nenhumoutrocanalouespecificaçãodedadosdeimagem>
data ALPHA ...
```

```
<todo o resto ...>
```

não é válido, porque Geomview não tem meios de determinar o canal de destino a partir do contexto.

`AUTO` Dados de imagem no formato PGM é interpretado como canal simples em escala de cinza, dados RGB PNM como dados de imagem RGB. `AUTO` não pode trabalhar com dados de imagem no formato '`raw`'.

'`FILTER`' A especificação '`FILTER`' é opcional. se for omitida, então Geomview tenta determinar o tipode imagem usando o sufixo de '`NOMEDOARQUIVO`'. Se ão houver sufixo ou o sufixo for desconhecido, ou para dados embutidos de imagem, Geomview está apto a auto-detectar o formato do arquivo de imagem SGI (por razões históricas . . .) e formatos de imagem NetPBM (por razões práticas). A auto-detecção de formatos NetPBM incluem o novo formato de imagem *PAM* que permite (em meio a um monte de outras coisas) armazenar um canal alfa juntamente com os dados de luminânica ou de RGB. Da mesma forma, a saída final de qualquer dos filtros especificados devem ou ser no formato de arquivo de imagem SGI, ou especificar uma imagem PAM, PNM ou PGM. Se o formato de arquivo de imagem não puder ser determinado por ou pelo sufixo do nome de arquivo ou pela especificação de filtro ou pela auto-detecção de dados SGI ou NetPBM, então Geomview assume que os dados sejam `"raw"`. Veja abaixo.

Os filtros de descompressão podem ser deduzidos ou de um dos formatos de imagem conhecidos ou de um especificador de filtro explícito, e.g. o seguinte é válido:

```
data LUMINANCE raw.gzip { < arquivoemtonsdecinzagzipado }
```

A linha acima deve ser equivalente a

```
data LUMINANCE raw { < arquivoemtonsdecinza },
```

fornece dados descomprimidos realizados através de dados de canal simples, com o primeiro pixel correspondendo ao canto inferior esquerdo (devido ao formato '`raw`' de imagem, veja abaixo).

Geomview tem conhecimento interno dos seguintes filtros/sufixos:

Descompressão de Dados

'`z`'

'`gz`'

'`gzip`' os dados são direcionados por '`gzip -dc`'

'`bz2`'

'`bzip2`' os dados são direcionados por '`bzip2 -dc`'

Formatos de Imagem

'`tiff`'

'`tif`' Formato de imagem `TIFF`. Somente suportado se o executável `tifftopnm` puder ser executado no caminho de execução atual.

'`png`' Formato de imagem `PNG`. Somente suportado se o executável `pngtopnm` puder ser executado no caminho de execução atual.

'`jpg`'

'`jpeg`' Formato de imagem `JPEG`. Somente suportado se o executável `jpegtopnm` puder ser executado no caminho de execução atual.

'`gif`' Formato de imagem `GIF` image file format. Somente suportado se o executável `giftoppm` puder ser executado no caminho de execução atual.

'`raw`' Dados de imagem em Raw; o número de canais deve coincidir com o número de bits informado em '`MASCDEST`'. Pixels são especificados com 1 byte por canal. Os pixels são organizados em linhas como em '`liminance[-alpha]`' ou em amostras '`RGB[A]`'. O pixel mais à esquerda é o primeiro pixel em cada linha de dados, a linha de dados mais acima deve vir primeiramente (isso é apenas o mesmo que a convenção de NetPBM, os sistemas de coordenadas de imagem têm sua origem no canto superior esquerdo, da forma usual).

Filtros Explicitamente Especificados

Se nenhum dos sufixos especificados acima coincidirem, então o sufixo/filtro é interpretado como um filtro de programa externo; o programa do filtro externo deve ler de `STDIN` (da entrada padrão) e escrever para `STDOUT` (a saída padrão). A saída deve ou ser no formato de imagem SGI, ou no formatos de image PNM ou PGM. De outra forma os dados de saída são interpretados como dados de imagem no formato raw (veja acima).

Alguma coisa como o seguinte pode trabalhar, garantindo que o programa '`${HOME}/bin/bububfilter`' exista, seja executável e faça alguma coísa útil:

```
...
data RGB "${HOME}/bin/bububfilter.bzip2" 7
{ # dados binários seguem
bububub
}
...
```

Note que – previamente fornecendo os dados para '`bububfilter`' – Geomview irá tentar descompactar o material com '`bzip2 -dc`'.

Omitindo dados de imagem: Normalmente, o número de canais de imagem é determinado automaticamente a partir das especificações dos `dados` de imagem; se a especificação de imagem carrega um número explícito de canais via palavra chave `channels` que excede o número de canais encontrado nas especificações de `dados`, ou se a união de todas as

especificações 'MASCDEST' possuem buracos, então omitindo luminância e canais RGB são initializados para 0, e um canal alfa omitido é inicializado para maxval, i.e. omitindo os dados do canal alfa para uma imagem RGBA é apenas o mesmo que definir uma imagem RGB.

4.3.3 Objetos de Transformação

Onde uma matriz simples 4x4 é esperada – como no campo de transform INST, a transformação da câmera camtoworld e os comandos do Geomview xform* – use um objeto de transformação.

Note que uma transformação é diferente de uma TLIST, que é um tipo de geometria. TLISTs podem conter uma ou mais transformações 4x4; objetos "transform" devem ter exa-tamente uma.

Por que temos ambos ("transform" e TLIST)? Em muitos lugares – e.g. posicionamento de câmeras – é somente significativo ter uma transofrmação simples. Usando um tipo separado de objeto reforça isso.

A sintaxe para um objeto de transformação "transform" é

```
<transform> ::=
  [ "{" ]              (Abertura de chave, geralmente necessário para
                        tornar o fim do objeto claramente explicitado.)

   [ "transform" ]     (palavra chave opcional; desnecessária se o tipo
                        for determinado pelo contexto, o que comumente
                        ocorre.)
   [ "define" <nome> ]
                       (define uma transformação chamada <nome>,
                       escolhendo seus valores do material adiante)

      <dezesseis números em ponto flutuante>
                       (interpretado como uma transformação homogênea 4x4
       fornecida linha por linha, intencionalmente aplicada a um
                        vetor linha multiplicado à ESQUERDA, de forma que
                        e.g. translações euclidianas apareçam na linha
                        inferior)
   |
      "<" <nomedearquivo>  (significando: leia a transformação a partir
                        daquele arquivo)
   |
      ":" <nome>       (significando: use a variável <nome>,
                         definida em algum lugar; se não definido o valor
                         inicial é a transformação identidade)

  [ "}" ]              (fechamento de chave correspondente)
```

O conjunto pode ser colocado entre { chaves }. As chaves não são essenciais se exatamente um dos itens acima estiver presente, então e.g. um array 4x4 de números em ponto flutuante independente não precisa necessariamente ter chaves.

Alguns exemplos, em contextos onde eles podem ser usados:

```
# Exemplo 1: Um comando GCL para definir uma transformação
# chamada "fred"

(read transform { transform  define fred
         1 0 0 0
         0 1 0 0
         0 0 1 0
        -3 0 1 1
    }
)
# Exemplo 2:  Um objeto câmera usando a transformação
# "fred" para posicionamento de câmera
# Fornecida a definição acima, isso coloca a câmera em
# (-3, 0, 1), olhando na direção -Z.

{ camera
        halfyfield 1
        aspect 1.33
        camtoworld { : fred }
}
```

4.3.4 Objetos ND-Transform

Onde – no contexto de visualização NDimensional – uma matriz simples (N+1)x(N+1) é esperada – como no campo `INST ntransform`, ou a `ND-xform*` (veja comandos Capítulo 7 [GCL], página 127) – use um objeto `ntransform`.

`ntransform` são matrizes de transformação *NLinhas* x *NColunas* onde usualmente *NLinhas* = *N+1* no contexto de objetos *N*-dimensionais e visualização. A componente homogênea de uma `ntransform` situa-se na coluna zero (em oposição a objetos `transform` comuns onde a componente homogênea situa-se na coluna três). Objetos `ntransform` trabalham sobre pontos de qualquer dimensão: se um ponto é para ser transformado através de um objeto `ntransform` e a dimensão do ponto não coincide com o número de linhas do objeto `ntransform`, então ou o ponto está implicitamente preenchido com zeros para coincidir com *NLinhas* ou a matriz está implicitamente preenchida com unidades abaixo de sua diagonal principal (e zeros em todas as outras posições) de forma que a matriz irá trabalhar como a matriz identidade sobre as dimensões excedentes do ponto de entrada.

A sintaxe para um objeto `ntransform` é

```
<ntransform> ::=
  [ "{" ]              (abertura de chave, geralmente necessária para
                        tornar o fim do objeto claramente explicitado.)

    [ "ntransform" ]    (palavra chave opcional; desnecessária se o tipo
                        for determinado pelo contexto, o que
                        comumente ocorre.)

    [ "define" <nome> ]
                        (define uma transformação chamada <nome>,
```

```
                        escolhendo seus valores do material adiante)

      NLinhas NColunas
                        (número de linhas e colunas da matriz,
                        tipicamente N+1 N+1, mas qualquer dimensão
                        é possível)
      <NLinhas x NColunas números em ponto flutuante>
                        (interpretados como uma transformação homogênea
                        NLinhas x NColunas fornecida linha por linha,
                        pretensamente a ser aplicada a um vetor linha
                        multiplicado à ESQUERDA, de forma que e.g.
                        translações Euclideanas aparecem na linha mais
                        acima -- em oposição a objetos de
                        transformação comuns onde as
                        translações aparecem na linha mais inferior)
    |
       "<" <nomedearquivo>  (significando: leia a transformação daquele
                        arquivo)
    |
       ":" <nome>       (significando: use a variável <nome>,
                        definida em algum lugar; se não for definida o
                        valor inicial é a transformação identidade)

  [ "}" ]             (correspondente fechamento de chave)
```

O conjunto deve ser delimitado entre { chaves }. Chaves não são necessariamente essenciais, de forma que e.g. dois inteiros – *NLinhas NColunas* – seguidos por um array composto de *NLinhas* x *NColunas* números em ponto flutuante independentes pode mas não precisa ter chaves.

Alguns exemplos, em contextos onde eles possivelmente podem ser usados:

```
# Exemplo 1: Um comando GCL para definir uma transformação 6x6 chamada
# "fred", uma mera translação por meio do vetor -3 0 1 1 0. Essa
# transformação é significativa para um espaço pentadimensional, com uma componente
# homogênea um índice zero.

(read ntransform { ntransform  define fred
        6 6
        1 -3 0 1 1 0
        0  1 0 0 0 0
        0  0 1 0 0 0
        0  0 0 1 0 0
        0  0 0 0 1 0
        0  0 0 0 0 1
    }
)

# Exemplo 2: Escolhe o ND-xform de um objeto -- um geométrico ou um grupo de
```

```
# cameras. Fornecendo a definição acima, o ND-xform escolhido coloca o ob-
jeto em (-3 0 1 1
# 0) no espaço pentadimensional.

(ND-xform-set focus : fred)

# ou

(ND-xform-set g1 : fred)
```

4.3.5 câmera

Um objeto câmera especifica as seguintes propriedades de uma câmera:

posição e orientação
: especificada por ou uma transformação camera para o objeto mundo ou uma transformação de objeto mundo para a câmera; essa transformação não inclui a projeção, de forma que essa transformação é tipicamente apenas uma composição de translação e rotação. Especificado como um objeto transform, tipicamente uma matriz 4x4.

"focus" - distância focal
: Ao invés de sugerir uma distância típica da câmera ao objeto de interesse; usada para o posicionamento padrão da câmera (a câmera é colocada em (X,Y,Z) = (0,0,focus) quando resetada) e para ajustar o campo de visão quando alternando entre os modos de visualização ortográfico e perspectivo.

razão de aspecto da janela
: A verdadeira razão de aspecto no sentido <Xsize>/<Ysize>. Essa razão de aspecto normalmente pode concordar com a razão de aspecto da janela de camera. Geomview normalmente ajusta a razão de aspecto de suas câmeras para coincidir com suas janelas associadas.

distância de plano de corte próximo e distante
: Note que ambas as distâncias devem ser estritamente maiores que zero. Razões de distância <distante>/<próximo> muito grandes fazem com que a área de armazenamento temporário do eixo Z comporte-se de forma péssima; parte de um objeto pode ser visível mesmo se estiver em algum lugar mais distante que outro.

campo de visão
: Especificado em uma das duas formas a seguir.

  ‘`fov`’ é o campo de visão – em graus se perpectivo, ou distância linear se ortográfico – na *menor* direção.

  ‘`halfyfield`’
  : é metade do campo projetado no eixo Y, em coordenadas do objeto mundo (não angulo!), em unidades de distância a partir da câmera. Para uma câmera em perpectiva, halfyfield é relacionado ao campo angular:

    halfyfield = tan(Y_axis_angular_field / 2)

enquanto para uma câmera em visão ortográfica temos simplesmente:

halfyfield = Y_axis_linear_field / 2

Essa definição singular de visualização é (a) maneira fácil de fazer cálculos que a envolvam e (b) é bem definida em ambas as formas de visualização ortográfica e perspectiva.

cor de fundo

Argumentavelmente (de forma discutível) não é uma propriedade de uma câmera, mas da cena (do cenário). Todavia, como não existe o objeto "cordefundo" no OOGL, e a cor de fundo pode não ser uma propriedade do equipamento de desenho, essa propriedade pode ser especificada aqui. No momento em que esse manual está sendo escrito, todavia, a interface gráfica de usuário (GUI) sempre sobrescreve a cor de fundo com suas próprias escolhas.

imagem de fundo

Pelas mesmas razões mostradas acima,somente a referida GUI não sobrescreve essa opção. A imagem é centralizada em "Normalized-Device-Coordinates"/NDC `(0,0,-1)`; a imagem não é redimensonada, apenas desenhada por trás de tudo mais como é. Veja seção 4.3.2 [image], página 91.

A sintaxe para uma câmera é:

```
<nomecamera> ::=

   [ "camera" ]                 (palavra chave opcional)
    [ "{" ]                    (abertura de chave, geralmente requerida)
       [ "define" <nome> ]

       "<" <nomearquivo>
     |
       ":" <nome>
     |
                               (ou qualquer número dos seguintes,
                                em qualquer ordem...)

       "perspective"  {"0" | "1"}            (padrão 1)
                                        (de outra forma "ortographic")

       "stereo"       {"0" | "1"}            (padrão 0)
                                        (de outra forma mono)

       "worldtocam" <transformação>        (veja a sintaxe de transfor-
                                        mação acima)

       "camtoworld" <transformação>
                               (nenhum ponto na especificação de ambos
                                camtoworld e worldtocam; se é
                                restingido para ser o inverso de
```

```
        "halfyfield" <meio-campo-linear-Y-em-unidade-de-distância>
                              (o padrão é tan 40/2 graus)
                              Nota de tradução:40 é o valor padrão
                              do campo de visão (fov).

        "fov"           (campo de visão angular se "perspective",
                         campo de visão linear em caso contrário.
                         Medido em qualquer direção é menor,
                         dando a razão aspecto.  Quando a razão de
                         aspecto muda -- e.g. quando uma janela é
                         reajustada -- "fov" é preservado.)

        "frameaspect" <razão-de-aspecto>    (X/Y) (padrão 1.333)

        "near"  <distância-de-corte-próxima>        (padrão 0.1)

        "far"   <distância-de-corte-afastada>     (padrão 10.0)

        "focus" <distância-focal>                 (padrão 3.0)

        "bgcolor" <cor RGB(A) em ponto flutuanter>
                                                (padrão 1/3 1/3 1/3 1)

        "bgimage" { <especificação de imagem> }
                                          (padrão nenhuma imagem de fundo)


    [ "}" ]                         (correspondente fechamento de chave)
```

4.3.6 window

Um objeto window especifica tamanho, posição, e outras informações relacionadas ao sistema de janelas sobre uma janela de forma que essas informações sejam independentes do dispositivo.

A sintaxe de um objeto window é:

```
nomejanela ::=

[ "window" ] (palavra chave opcional)
  [ "{" ] (abertura de chave, requerida na maioria das vezes)

     (qualquer dos seguintes, em qualquer ordem)

"size"  <tamanhox> <tamanhoy>
(tamanho da janela)

"position"  <xmin> <xmax> <ymin> <ymax>
```

```
(posição & tamanho)


"noborder"
(especifica se a janela pode
 ter ou não uma borda)

"pixelaspect"  <aspecto>
    (especifica a razão de aspecto real visualizada
     de um pixel nessa janela no sentido
     tamanhox/tamanhoy, normalmente 1.0.
     Para hardware stereo o qual corta o
     display verticalmente por um fator de 2,
     ‘‘pixelaspect 0.5’’ pode funcionar.
     O valor é usado no cálculo da
     projeção de uma câmera associada a
     essa janela.)

  [ "}" ] (correspondente fechamento de chave)
```

Objetos window são usados em `janelas` do Geomview e em comandos do painel de interação/interface com o usuário (`ui-panel`) para escolher propriedades padronizadas para futuras janelas ou mudar as mesmas propriedades padronizadas de uma janela existente. Veeja seção 7.2.156 [window], página 159. Veja também seção 7.2.149 [ui-panel], página 157.

5 Customização: arquivos '.geomview'

Quando Geomview é inciado, ele chama e executa comando em um arquivo de inicialização dependente do sistema operacional chamado '.geomview'. Esse arquivo está no subdiretório 'data' do diretório de distribuição do Geomview e contém comandos GCL para configurar Geomview de uma forma comum para todos os usuários no sistema.

A seguir, Geomview procura o arquivo '~/.geomview' ('~' corresponde a seu diretório base). Você pode usar esse arquivo para configurar seu próprio comportamento padrão do Geomview de forma que ele se ajuste às suas preferências.

Após a leitura de '~/.geomview', Geomview procura por um arquivo chamado '.geomview' no diretório atual. Se tal arquivo existir Geomview lê esse arquivo, a menos que esse arquivo seja o mesmo que '~/.geomview' (o que pode ser o caso se você estiver executando o Geomview a partir de seu diretório base). Você pode usar o '.geomview' do diretório atual para criar uma personalização no Geomview específica para um certo projeto.

Você pode usar arquivos '.geomview' para controlar todos os tipos de coisa no Geomview. Eles podem conter quaisquer declarações GCL válidas. Especialmente útil é o comando ui-panel que controla a localização inicial dos painéis do Geomview. Para um exemplo veja o arquivo dependente do sistema '.geomview' mencionado acima. Veja Capítulo 7 [GCL], página 127. Veja seção 7.2.149 [ui-panel], página 157.

Uma boa idéia colocar juntos todos os comandos que você coloca em um arquivo '.geomview' em uma declaração progn com o objetivo de fazer com que Geomview execute todos de uma só vez. Caso você não faça isso Geomview possivelmente pode executar esses comando sequêncialmente sobre os primeiros poucos ciclos de atualização após a incialização.

Para modificar, e.g. a política de focalização da janela de câmera de forma que ela selecione a política de focalização do gerenciador de janela (em lugar de ser ativada quando o cursor do mouse cruza a janela), você pode colocar o seguinte no seu arquivo '~/.geomview':

```
(progn
  (ui-cam-focus focus-change)
  ... # other stuff
)
```

Você pode colocar qualquer comando GCL válido nos seus arquivos '.geomview', Veja Capítulo 7 [GCL], página 127. Veja seção 7.2.106 [progn], página 148. Veja seção 7.2.141 [ui-cam-focus], página 155.

6 Módulos Externos

Um módulo externo é um programa que interage com Geomview. Um módulo comunica-se com Geomview através de GCL e pode controlar qualquer aspecto do Geomview que você possa controlar através da interface de usuário do Geomview.

Em muitos casos um módulo externo é um programa especializado que imprementa algum algorítmo matemático que cria um objeto geométrico que modifica aparência à medida que o algorítmo progride. O módulo informa ao Geomview da nova aparência do objeto a cada passo, de forma que o objeto aparenta evoluir com o tempo na janela do Geomview. Dessa modo Geomview serve como um *ferramenta de exibição* para o módulo.

Um módulo externo pode ser interativo. Esse módulo pode responder a eventos de mouse e a eventos de teclado que ocorram em uma janela do Geomview, dessa forma extendendo a capacidade do Geomview propriamente dito.

6.1 Como Módulos Externos Interagem com o Geomview

Módulos externos aparecem no navegador de Módulos (*Modules*) no painel principal (*Main*) do Geomview. Para executar um módulo, clique no botão esquerdo do mouse sobre a entrada do módulo no navegador. Enquanto o módulo estiver sendo executado, uma linha adicional para aquele módulo irá aparecer no navegador. Essa linha inicia-se com um número entre colchêtes, que indica o número de *instância* do módulo. (Para alguns módulos faz sentido ter mais de uma instância do módulo sendo executado ao mesmo tempo.) Você pode encerrar um módulo externo através de um clique sobre sua entrada vermelha de instância.

Por padrão quando Geomview inicia, mostra todos os módulos que tiverem sido instalados no seu sistema.

Para instruções sobre intalação de algum módulo no seu sistema de forma que esse módulo apareça no navegador de módulos (*Modules*) toda vez que Geomview estiver sendo executado por alguém no seu sistema, veja seção 6.7 [Module Installation], página 125.

Quando Geomview chama um módulo externo, cria pipes conectados às saídas e às entradas padrão do modulo. (Pipes são como arquivos exceto que eles são usados para comunicação entre programas em lugar de armazenar coisas em um disco.) Geomview interpreta qualquer coisa que o modulo escreve em sua saída padrão como um comando GCL. Da mesma forma, Se um módulo externo requisita qualquer dado do Geomview, Geomview escreve aquele dado para a entrada padrão do módulo. Dessa forma tudo que um módulo tem de fazer com o objetivo de comunicar-se com Geomview é escrever comandos para sua saída padrão e (opcionalmente) receber dados de sua entrada padrão. Note que isso significa que o módulo não pode usar a entrada padrão e a saída padrão para comunicar-se com o usuário. Se um módulo precisar comunicar-se com o usuário isso pode ser feito ou através de um painel de controle nele próprio ou em caso contrário através de respostas a certos eventos que esse módulo encontrar como saída vinda do Geomview.

6.2 Exemplo 1: Módulo Externo Simples

Essa seção fornece um módulo externo extremamente simples que mostra uma malha oscilando. Para experimentar esse exemplo, faça uma cópia do arquivo '`example1.c`' (esse arquivo é distribuído com Geomview no subdiretio '`doc`') em seu diretório e compile '`example1.c`' com o comando

```
cc -o example1 example1.c -lm
```

A seguir coloque a linha

```
(emodule-define "Example 1" "./example1")
```

em um arquivo chamado '`.geomview`' no seu diretório atual. A seguir chame o Geomview; É importante que você compile o programa exemplo, crie o arquivo '`.geomview`' e chame o Geomview a partir do mesmo diretório. Você pode ver "Example 1" no navegador de módulos (*Modules*) do painel principal (*Main*) do Geomview; clique sobre essa entrada no navegador para iniciar o módulo. Uma superfície deve aparecer na sua janela de câmera e deve estar oscilando. Você pode parar o módulo clicando sobre a linha "[1] Example 1" no navegador de módulos (*Modules*).

```
/*
 * example1.c: oscillating mesh
 *
 * This example module is distributed with the Geomview manual.
 * If you are not reading this in the manual, see the "External
 * Modules" chapter of the manual for more details.
 *
 * This module creates an oscillating mesh.
 */

#include <math.h>
#include <stdio.h>

/* F is the function that we plot
 */
float F(x,y,t)
     float x,y,t;
{
  float r = sqrt(x*x+y*y);
  return(sin(r + t)*sqrt(r));
}

main(argc, argv)
     char **argv;
{
  int xdim, ydim;
  float xmin, xmax, ymin, ymax, dx, dy, t, dt;

  xmin = ymin = -5;             /* Set x and y           */
  xmax = ymax = 5;              /*    plot ranges        */
  xdim = ydim = 24;             /* Set x and y resolution */
  dt = 0.1;                     /* Time increment is 0.1  */

  /* Geomview setup.  We begin by sending the command
   *            (geometry example { : foo})
   * to Geomview.  This tells Geomview to create a geom called
```

```
   * "example" which is an instance of the handle "foo".
   */
  printf("(geometry example { : foo })\n");
  fflush(stdout);

  /* Loop until killed.
   */
  for (t=0; ; t+=dt) {
    UpdateMesh(xmin, xmax, ymin, ymax, xdim, ydim, t);
  }
}

/* UpdateMesh sends one mesh iteration to Geomview.  This consists of
 * a command of the form
 *    (read geometry { define foo
 *       MESH
 *       ...
 *    })
 * where ... is the actual data of the mesh.  This command tells
 * Geomview to make the value of the handle "foo" be the specified
 * mesh.
 */
UpdateMesh(xmin, xmax, ymin, ymax, xdim, ydim, t)
     float xmin, xmax, ymin, ymax, t;
     int xdim, ydim;
{
  int i,j;
  float x,y, dx,dy;

  dx = (xmax-xmin)/(xdim-1);
  dy = (ymax-ymin)/(ydim-1);

  printf("(read geometry { define foo \n");
  printf("MESH\n");
  printf("%1d %1d\n", xdim, ydim);
  for (j=0, y = ymin; j<ydim; ++j, y += dy) {
    for (i=0, x = xmin; i<xdim; ++i, x += dx) {
      printf("%f %f %f\t", x, y, F(x,y,t));
    }
    printf("\n");
  }
  printf("})\n");
  fflush(stdout);
}
```

O módulo inicia-se definindo uma função `F(x,y,t)` que especifica uma superfície variando com o tempo. O propósito do módulo é animar essa superfície com o passar do tempo.

O programa principal começa definindo algumas variáveis que especificam os parâmetros com os quais a função é para ser mostrada.

O próximo pedaço de código no programa principal envia a seguinte linha para a saída padrão

```
(geometry example { : algumacoisa })
```

Isso diz ao Geomview para criar um geom chamado `example` que é uma instância do controlador `algumacoisa`. *Controladores* são uma parte do formato de arquivo OOGL que permite a você nomear uma peça do objeto geométrico cujo valor pode ser especificado em seu lugar (e nesse caso atualizado muitas vezes); para maiores informações sobre controladores, veja Capítulo 4 [Formatos dos Arquivos da OOGL], página 65. Nesse caso, `example` é o título através do qual o usuário irá ver o objeto no navegador de objeto do Geomview, e `foo` é o nome interno do controlador para o qual o objeto aponta.

Nós então fazemos `fflush(stdout)` para garantir que Geomview receba esse comando imediatamente. Em geral, uma vez que pipes podem ser colocados em uma área de memória temporária de armazenamento, um módulo externo deve fazer isso sempre que esse módulo externo tenha que garantir que Geomview tenha atualmente recebido tudo que tenha sido mostrado na saída.

A última coisa no programa principal é um ciclo infinito que circula através de chamadas ao procedimento `UpdateMesh` com incremento dos valores de `t`. `UpdateMesh` envia ao Geomview um comando da forma

```
(read geometry { define algumacoisa
MESH
24 24
...
})
```

onde `...` é uma longa lista de números. Esse comando diz ao Geomview fazer os valores do controlador `algumacoisa` ser a malha especificada. Tão breve quanto Geomview receba esse comando, o geom sendo mostrado muda para refletir o novo objeto geométrico.

A malha é fornecida no formato de uma OOGL MESH. Esse formato começa com a palavra chave `MESH`. A seguir temos dois números que fornecem as dimensões x e y da malha; nesse caso elas são ambas 24. Essa linha é seguida por 24 linhas, cada linha contendo 24 trios de números. Cada um desses trios representa um ponto sobre superfície. Então finalmente existe uma linha com `"})"` nesse formato que termina o `"{"` que iniciou a declaração `define` e o `"("` que iniciou o comando. Para mais detalhes sobre o formato de dados MESH, veja seção 4.2.2 [MESH], página 77.

Esse módulo pode ser escrito sem o uso de controladores escrevendo comandos da forma

```
(geometry example {
MESH
24 24
...
})
```

Nesse primeiro tempo Geomview recebe um comando e dessa forma pode criar um geom chamado `example` com os dados fornecidos da `MESH`. Subsequentemente comandos `(geometry example ...)` podem fazer com que Geomview substitua a geometria do geom `example` com os novos dados da `MESH`. Se realizado dessa forma pode não ser necessário enviar o comando inicial `(geometry example { : foo })` como dito acima. A técnica do controlador é útil, todavia, porque essa técnica pode ser usada em situações mais gerais onde um controlador representa somente parte de um complexo geom, permitindo a um módulo externo substituir somente aquela parte sem ter que retransmitir o geom completo. Para mais informações sobre controladores, veja Capítulo 7 [GCL], página 127. Veja seção 4.1.9 [Referencias], página 68. Veja seção 7.2.59 [hdefine], página 137. Veja seção 7.2.111 [read], página 149.

O módulo entra em ciclos através de chamadas a `UpdateMesh` que fornece como saída comandos da forma acima um após o outro tão rápido quanto possível. O ciclo continua indefinidamente; o módulo irá terminar quando o usuário fizer isso clicando sobre sua linha de instância no navegador de módulos (*Modules*), ou em caso contrário quando Geomview for encerrado.

Algumas vezes quando você encerra o módulo clicando sobre sua entrada de instância no navegador de módulos (*Modules*), Geomview irá encerrá-lo enquanto o módulo está no meio de um comando ao Geomview. Geomview irá então receber somente uma peça de um comando e irá mostrar uma crítica mas inofensiva mensagem de erro sobre isso. uando um módulo tiver um painel de interface com o usuário o usuário pode usar um botão `"Quit"` para fornecer uma forma mais elegante para o usuário encerrar o módulo. Veja o exemplo seguinte.

Você pode usar esse módulo em uma janela de shell sem o Geomview para ver os comando que o módulo mostra como saída. Você irá ter que encerrá-lo com *`ctrl-C`* para que ele pare.

6.3 Exemplo 2: Módulo Externo Simples Usando o Painel de Controle FORMS

Nota de tradução: Este exemplo é muito antigo e não compila nas máquinas atuais. Até o momento de digitação destas linhas de tradução ele não tinha sido atualizado permanecendo portanto inoperante.

Essa seção fornece uma nova versão do módulo acima — uma versão que inclue uma interface de usuário para controlar a velocidade da oscilação. Usamos a biblioteca FORMS criada por Mark Overmars para o painel de controle. A biblioteca FORMS é uma coleção de ferramentas para interface de usuário de domínio público para IRISes.

Para experimentar esse exemplo, faça uma cópia do arquivo '`example2.c`' (distribuído com Geomview no subdiretório '`doc`') no seu diretório e compile-o com o comando

```
cc -I/u/gcg/ngrap/include -o example2 example2.c \
  -L/u/gcg/ngrap/lib/sgi -lforms -lfm_s -lgl_s -lm
```

Você pode substituir a sequência de caracteres '`/u/gcg/ngrap`' acima com o nome do caminho do diretório de distribuição do Geomview em seu sistema. (A biblioteca forms é distribuída com Geomview e as opções `-I` e `-L` acima dizem ao compilador onde encontrá-las.)

Então coloque a linha

```
(emodule-define "Example 2" "./example2")
```

em um arquivo chamado '`.geomview`' no diretório e chame Geomview do mesmo diretório. Clique sobre a entrada "Example 2" no navegador de módulos (*Modules*) para chamar o módulo. Um pequeno painel de controle deverá aparecer. Você pode então controlar a velocidade da oscilação da malha movendo o botão deslizante.

```
/*
 * example2.c: oscillating mesh with FORMS control panel
 *
 * This example module is distributed with the Geomview manual.
 * If you are not reading this in the manual, see the "External
 * Modules" chapter of the manual for an explanation.
 *
 * This module creates an oscillating mesh and has a FORMS control
 * panel that lets you change the speed of the oscillation with a
 * slider.
 */

#include <math.h>
#include <stdio.h>
#include <sys/time.h>           /* for struct timeval below */

#include "forms.h"              /* for FORMS library */

FL_FORM *OurForm;
FL_OBJECT *VelocitySlider;
float dt;

/* F is the function that we plot
 */
float F(x,y,t)
     float x,y,t;
{
  float r = sqrt(x*x+y*y);
  return(sin(r + t)*sqrt(r));
}

/* SetVelocity is the slider callback procedure; FORMS calls this
 * when the user moves the slider bar.
 */
void SetVelocity(FL_OBJECT *obj, long val)
{
  dt = fl_get_slider_value(VelocitySlider);
}

/* Quit is the "Quit" botão callback procedure; FORMS calls this
 * when the user cliques the "Quit" botão.
```

```
 */
void Quit(FL_OBJECT *obj, long val)
{
  exit(0);
}

/* create_form_OurForm() creates the FORMS panel by calling a bunch of
 * procedures in the FORMS library.  This code was generated
 * automatically by the FORMS designer program; normally this code
 * would be in a separate file which you would not edit by hand.  For
 * simplicity of this example, however, we include this code here.
 */
create_form_OurForm()
{
  FL_OBJECT *obj;
  FL_FORM *form;
  OurForm = form = fl_bgn_form(FL_NO_BOX,380.0,120.0);
  obj = fl_add_box(FL_UP_BOX,0.0,0.0,380.0,120.0,"");
  VelocitySlider = obj = fl_add_valslider(FL_HOR_SLIDER,20.0,30.0,
                                          340.0,40.0,"Velocity");
    fl_set_object_lsize(obj,FL_LARGE_FONT);
    fl_set_object_align(obj,FL_ALIGN_TOP);
    fl_set_call_back(obj,SetVelocity,0);
  obj = fl_add_button(FL_NORMAL_BUTTON,290.0,75.0,70.0,35.0,"Quit");
    fl_set_object_lsize(obj,FL_LARGE_FONT);
    fl_set_call_back(obj,Quit,0);
  fl_end_form();
}

main(argc, argv)
     char **argv;
{
  int xdim, ydim;
  float xmin, xmax, ymin, ymax, dx, dy, t;
  int fdmask;
  static struct timeval timeout = {0, 200000};

  xmin = ymin = -5;             /* Set x and y            */
  xmax = ymax = 5;              /*    plot ranges         */
  xdim = ydim = 24;             /* Set x and y resolution */
  dt = 0.1;                     /* Time increment is 0.1  */

  /* Forms panel setup.
   */
  foreground();
  create_form_OurForm();
  fl_set_slider_bounds(VelocitySlider, 0.0, 1.0);
```

```
  fl_set_slider_value(VelocitySlider, dt);
  fl_show_form(OurForm, FL_PLACE_SIZE, TRUE, "Example 2");


  /* Geomview setup.
   */
  printf("(geometry example { : foo })\n");
  fflush(stdout);

  /* Loop until killed.
   */
  for (t=0; ; t+=dt) {
    fdmask = (1 << fileno(stdin)) | (1 << qgetfd());
    select(qgetfd()+1, &fdmask, NULL, NULL, &timeout);
    fl_check_forms();
    UpdateMesh(xmin, xmax, ymin, ymax, xdim, ydim, t);
  }
}

/* UpdateMesh sends one mesh iteration to Geomview
 */
UpdateMesh(xmin, xmax, ymin, ymax, xdim, ydim, t)
     float xmin, xmax, ymin, ymax, t;
     int xdim, ydim;
{
  int i,j;
  float x,y, dx,dy;

  dx = (xmax-xmin)/(xdim-1);
  dy = (ymax-ymin)/(ydim-1);

  printf("(read geometry { define foo \n");
  printf("MESH\n");
  printf("%1d %1d\n", xdim, ydim);
  for (j=0, y = ymin; j<ydim; ++j, y += dy) {
    for (i=0, x = xmin; i<xdim; ++i, x += dx) {
      printf("%f %f %f\t", x, y, F(x,y,t));
    }
    printf("\n");
  }
  printf("})\n");
  fflush(stdout);
}
```

O Código inicia-se pela inclusão de alguns arquivos de cabeçalho necassários para o evento cíclico e para a biblioteca FORMS. O código então declara variáveis globais para manter um ponteiro para o objeto FORMS do tipo botão deslizante e a velocidade `dt`. Essas

duas variáveis são do tipo global porque elas são necessárias no procedimento de retorno do botão deslizante chamado `SetVelocity`, que faz todas as chamadas a cada vez que o usuário move a barra do botão deslizante. `SetVelocity` atualiza o valor da variável `dt` com o novo valor do botão deslizante.

`Quit` é o procedimento de chamada de retorno para o botão *Quit*; esse botão fornece um caminho elegante para o usuário encerrar o programa.

O procedimento `create_panel` chama um conjunto de procedimentos da biblioteca FORMS para ajustar o painel de controle com o botão deslizante e o botão simples. Para mais informação sobre o uso de FORMS para criar painéis de interface veja a documentação de FORMS. Em particular, FORMS vem com um desenhista de painel gráfico que conduz você a desenhar seus painéis de forma interativa e a gerar códigos como o código em `create_panel`.

Esse programa principal exemplo é similar ao exemplo anterior, mas inclui codificação extra para lidar com ajustes e gerenciamento o painel FORMS.

Para ajustar o painel chamamos o procedimento GL `foreground` para fazer com que o processo execute em primeiro plano. Por padrão programas GL executam em segundo plano, e por várias razões módulos externos que usam FORMS (que é baseado em GL) precisa executar em primeiro plano. Chamamos então `create_panel` para criar o painel e `fl_set_slider_value` para ajustar o valor inicial do botão deslizante. A chamada a `fl_show_form` faz com que o painel apareça na tela.

As primeiras três linhas do ciclo principal, iniciam-se com

```
fdmask = (1 << fileno(stdin)) | (1 << qgetfd());
```

monitora e conduz os eventos no painel. A chamada a `select` impõe uma espera a cada passagem pelo ciclo principal. Essa chamada retorna ou após uma espera de 1/5 de segundo ou quando o evento GL seguinte ocorrer, ou quando dados aparecerem na entrada padrão, o que acontecer primeiro. A variável `timeout` especifica a quantidade de tempo a esperar nessa chamada; o primeiro número (0 nesse exemplo) fornece o número de segundos, e o segundo número (200000 nesse exemplo) fornece o número de microsegundos. Finalmente, `fl_check_forms()` procura e executa quaisquer eventos FORMS que tenham ocorridos; nesse caso isso significa uma chamada a `SetVelocity` se o usuário iver movido o botão deslizante ou uma chamada a `Quit` se o usuário tiver clicado no botão *Quit*.

O propósito da espera no ciclo é para prevenir que o programa use excessivamente o tempo da CPU executando seu ciclo pincipal quando não hoverem eventos a serem processados. Isso não é tão crucial nesse exemplo, e de fato pode tornar a animação mais lenta até certo ponto, mas em geral com módulos extrnos que possuirem eventos cíclicos essa espera é importante para do something like this because otherwise the module will needlessly take que os ciplos da CPU sigam adiante para outros programamas que estejam sendo executados (tais como Geomview!) mesmo quando estes não estiverem fazendo nada.

A última linha do ciclo principal ensse exemplo, a chamada a `UpdateMesh`, é a mesma que no exemplo anterior.

6.4 A Biblioteca XForms

XForms é uma interface de usuário ou conjunto de ferramentas muito fácil de encontrar e relativamente simples para X11. Muitos módulos externos do Geomview, incluindo os

exemplos nesse manual, usam XForms para criar e gerenciar painéis de controle. XForms está disponível no sítio `http://www.nongnu.org/xforms/`. XForms é free-ware. Se você quiser pode usar qualquer outro conjunto de ferramentas como interface ao invés de XForms em um módulo externo. Escolhemos FORMS pelo fato de omesmo ser livre e relativamente simples.

Existe um pacote completo baseado na ferramenta autoconf '`gvemod-xforms-example`' disponível no sítio do Geomview Sourceforge.NET.

6.5 Exemplo 3: Módulo Externo com Comunicação Bi-Direcional

Os dois módulos exemplo anteriores simplesmente enviam comandos ao Geomview sem receber nada de volta do Geomview. Esta seção descreve um módulo que se comuica em ambas as direções. Existe dois tipos de comunicação que podem ir do Geomview para um módulo externo. Esse exemplo mostra comunicação *sem sincronismo* — o módulo precisa estar apto a responder a qualquer momento a expressões que o Geomview possa emitir as quais informam ao módulo de alguma modificação de estado dentro do Geomview.

(O outro tipo de comunicação é *com sincronismo*, onde um módulo envia uma requisição ao Geomview sobre alguma peça de informação e espera por uma resposta venha de volta antes de fazer qualquer coisa a mais. O principal comando GCL para requisitar informação desse tipo é seção 7.2.158 [write], página 159. Esse módulo exemplo não faz nada com comunicação sincronizada.)

Na comnicação sem sincronismo, Geomview envia expressões que são essencialmente ecos de comandos GCL. O módulo externo envia ao Geomview um comando expressando interesse em um certo comando, e então toda vez que Geomview executar o referido comando, o módulo recebe uma cópia dele. O envio de informação ocorre independentemente de que envia o comando ao Geomview; a requisição pode ser resultado do usuário fazendo alguma coisa com o painel do Geomview, ou pode vir de outro módulo ou de um arquivo que o Geomview leu. É dessa forma que um módulo descobre informações e age sobre coisas que ocorrem no Geomview.

Esse exemplo usa a biblioteca em lisp da OOGL para analisar e agir sobre as expressões que Geomview escreve para a entrada padrão do módulo. Essa biblioteca faz atualmente parte do Geomview propriamente dito — escrevemos a biblioteca no processo de implementação da GCL. Essa biblioteca lisp da OOGL também é conveniente para ser usada em módulos externos que devem entender um subconjunto da GCL — especificamente, aqueles comandos que o módulo tem interesse expresso.

Esse exemplo mostra como um módulo pode receber eventos de seleção de usuário, i.e. quando o usuário clicar com o botão direito do mouse com o cursor sobre um geom em uma janela de câmera do Geomview. Quando isso ocorrer Geomview gera uma chamada interna a um procedimento chamado `pick`; o argumento para o procedimento fornece informação sobre a seleção, tal como o objeto que foi selecionado, as coordenadas co ponto selecionado, etc. Se um módulo externo tiver expressado interesse em chamadas ao procdimento `pick`, então sempre que o procedimento `pick` for chamado Geomview irá ecoar a chamada à entrada padrão do módulo que manifestou o interesse. O módulo que recebe o echo pode então fazer o que desejar caom a informação do procedimento pick.

Esse módulo é o mesmo que o módulo *Nose* que vem com o Geomview. Seu propósito é ilustrar processos de seleção. Qualquer coisa que você selecionar sobre um geom por meio de um clique do botão direito do mouse sobre esse geom, o módulo desenha uma pequena caixa na localização onde você tiver clicado. De forma geral a caixa é amarela. Caso você selecione um vértice, a caixa é da cor magenta. Se você selecionar um ponto sobre uma aresta de um objeto, o módulo irá também ressaltar a aresta desenhando caixas da cor ciano em suas extremidades e desenhar uma linha amarela ao lonfo da aresta.

Note que para esse módulo fazer alguma coisa você deve ter um geom carregado no Geomview e você deve cicar com o botão direito do mouse com o cursor sobre uma parte do geom.

```
/*
 * example3.c: external module with bi-directional communication
 *
 * This example module is distributed with the Geomview manual.
 * If you are not reading this in the manual, see the "External
 * Modules" chapter of the manual for an explanation.
 *
 * This module is the same as the "Nose" program that is distributed
 * with Geomview.  It illustrates how a module can find out about
 * and respond to user pick events in Geomview.  It draws a little box
 * at the point where a pick occurrs.  The box is yellow if it is not
 * at a vertex, and magenta if it is on a vertex.  If it is on an edge,
 * the program also marks the edge.
 *
 * To compile:
 *
 *   cc -I/u/gcg/ngrap/include -g -o example3 example3.c \
 *      -L/u/gcg/ngrap/lib/sgi -loogl -lm
 *
 * You should replace "/u/gcg/ngrap" above with the pathname of the
 * Geomview distribution directory on your system.
 */

#include <stdio.h>
#include "lisp.h"               /* We use the OOGL lisp library */
#include "pickfunc.h"           /* for PICKFUNC below */
#include "3d.h"                 /* for 3d geometry library */

/* boxstring gives the OOGL data to define the little box that
 * we draw at the pick point.  NOTE:  It is very important to
 * have a newline at the end of the OFF objeto in this string.
 */
char boxstring[] = "\
INST\n\
transform\n\
.04 0 0 0\n\
```

```
0 .04 0 0\n\
0 0 .04 0\n\
0 0 0 1\n\
geom\n\
OFF\n\
8 6 12\n\
\n\
-.5 -.5 -.5     # 0   \n\
.5 -.5 -.5      # 1   \n\
.5  .5 -.5      # 2   \n\
-.5  .5 -.5     # 3   \n\
-.5 -.5  .5     # 4   \n\
.5 -.5  .5      # 5   \n\
.5  .5  .5      # 6   \n\
-.5  .5  .5     # 7   \n\
\n\
4 0 1 2 3\n\
4 4 5 6 7\n\
4 2 3 7 6\n\
4 0 1 5 4\n\
4 0 4 7 3\n\
4 1 2 6 5\n";

progn()
{
  printf("(progn\n");
}

endprogn()
{
  printf(")\n");
  fflush(stdout);
}

Initialize()
{
  extern LObject *Lpick();  /* This is defined by PICKFUNC below but must */
       /* be used in the following LDefun() call */
  LInit();
  LDefun("pick", Lpick, NULL);

  progn(); {
    /* Define handle "littlebox" for use later
     */
    printf("(read geometry { define littlebox { %s }})\n", boxstring);

    /* Express interest in pick events; see Geomview manual for explanation.
```

```
     */
    printf("(interest (pick world * * * * nil nil nil nil nil))\n");

    /* Define "pick" objeto, initially the empty list (= null objeto).
     * We replace this later upon receiving a pick event.
     */
    printf("(geometry \"pick\" { LIST } )\n");

    /* Make the "pick" objeto be non-pickable.
     */
    printf("(pickable \"pick\" no)\n");

    /* Turn off normalization, so that our pick objeto will appear in the
     * right place.
     */
    printf("(normalization \"pick\" none)\n");

    /* Don't draw the pick objeto's bounding box.
     */
    printf("(bbox-draw \"pick\" off)\n");

  } endprogn();
}

/* The following is a macro call that defines a procedure called
 * Lpick().  The reason for doing this in a macro is that that macro
 * encapsulates a lot of necessary stuff that would be the same for
 * this procedure in any program.  If you write a Geomview module that
 * wants to know about user pick events you can just copy this macro
 * call and change the body to suit your needs; the body is the last
 * argument to the macro and is delimited by curly braces.
 *
 * The first argument to the macro is the name of the procedure to
 * be defined, "Lpick".
 *
 * The next two arguments are numbers which specify the sizes that
 * certain arrays inside the body of the procedure should have.
 * These arrays are used for storing the face and path information
 * of the picked objeto.  In this module we don't care about this
 * information so we declare them to have length 1, the minimum
 * allowed.
 *
 * The last argument is a block of code to be executed when the module
 * receives a pick event.  In this body you can refer to certain local
 * variables that hold information about the pick.  For details see
 * Example 3 in the Extenal Modules chapter of the Geomview manual.
 */
```

```
PICKFUNC(Lpick, 1, 1,
{
  handle_pick(pn>0, &point, vn>0, &vertex, en>0, edge);
},
/* version for picking Nd-objects (not documented here) */)

handle_pick(picked, p, vert, v, edge, e)
     int picked;                /* was something actually picked?     */
     int vert;                  /* was the pick near a vertex?        */
     int edge;                  /* was the pick near an edge?         */
     HPoint3 *p;                /* coords of pick point               */
     HPoint3 *v;                /* coords of picked vértice            */
     HPoint3 e[2];              /* coords of endpoints of picked edge */
{
  Normalize(&e[0]);             /* Normalize makes 4th coord 1.0 */
  Normalize(&e[1]);
  Normalize(p);
  progn(); {
    if (!picked) {
      printf("(geometry \"pick\" { LIST } )\n");
    } else {
      /*
       * Put the box in place, and color it magenta if it's on a vértice,
       * yellow if not.
       */
      printf("(xform-set pick { 1 0 0 0  0 1 0 0  0 0 1 0  %g %g %g 1 })\n",
             p->x, p->y, p->z);
      printf("(geometry \"pick\"\n");
      if (vert) printf("{ appearance { material { diffuse 1 0 1 } }\n");
      else printf("{ appearance { material { diffuse 1 1 0 } }\n");
      printf("  { LIST { :littlebox }\n");

      /*
       * If it's on an edge and not a vertex, mark the edge
       * with cyan boxes at the endpoins and a black line
       * along the edge.
       */
      if (edge && !vert) {
        e[0].x -= p->x; e[0].y -= p->y; e[0].z -= p->z;
        e[1].x -= p->x; e[1].y -= p->y; e[1].z -= p->z;
        printf("{ appearance { material { diffuse 0 1 1 } }\n\
  LIST\n\
   { INST transform 1 0 0 0 0 1 0 0 0 0 1 0 %f %f %f 1 geom :littlebox }\n\
   { INST transform 1 0 0 0 0 1 0 0 0 0 1 0 %f %f %f 1 geom :littlebox }\n\
   { VECT\n\
          1 2 1\n\
          2\n\
```

```
            1\n\
            %f %f %f\n\
            %f %f %f\n\
            1 1 0 1\n\
     }\n\
    }\n",
                  e[0].x, e[0].y, e[0].z,
                  e[1].x, e[1].y, e[1].z,
                  e[0].x, e[0].y, e[0].z,
                  e[1].x, e[1].y, e[1].z);
        }
        printf("    }\n  }\n)\n");
      }

    } endprogn();

}

Normalize(HPoint3 *p)
{
  if (p->w != 0) {
    p->x /= p->w;
    p->y /= p->w;
    p->z /= p->w;
    p->w = 1;
  }
}

main()
{
  Lake *lake;
  LObject *lit, *val;
  extern char *getenv();

  Initialize();

  lake = LakeDefine(stdin, stdout, NULL);
  while (!feof(stdin)) {

    /* Parse next lisp expression from stdin.
     */
    lit = LSexpr(lake);

    /* Evaluate that expression; this is where Lpick() gets called.
     */
    val = LEval(lit);
```

```
        /* Free the two expressions from above.
         */
        LFree(lit);
        LFree(val);
      }
    }
```

The code begins by defining procedures `progn()` and `endprogn()` which begin and end a Geomview `progn` group. The purpose do Geomview `progn` command is to group commands together and cause Geomview to execute them all at once, without refreshing any graphics windows until the end. It is a good idea to group blocks of commands that a module sends to Geomview like this so that the user sees their cumulative effect all at once.

Procedure `Initialize()` does various things needed at program startup time. It initializes the lisp library by calling `LInit()`. Any program that uses the lisp library should call this once before calling any other lisp library functions. It then calls `LDefun` to tell the library about our `pick` procedure, which is defined further down with a call to the `PICKFUNC` macro. Then it sends a bunch of setup commands to Geomview, grouped in a `progn` block. This includes defining a handle called `littlebox` that stores the geometry da little box. Next it sends the command

```
(interest (pick world * * * * nil nil nil nil nil))
```

which tells Geomview to notify us when a pick event happens.

The syntax of this `interest` statement merece some explanation. In general `interest` takes one argument which is a (parenthesized) expression representing a Geomview function call. It especifica a type of call that the module is interested in knowing about. The arguments can be any particular argument values, ou the special symbols `*` or `nil`. For example, the first argument in the `pick` expression above is `world`. This means that the module is interested in calls to `pick` where the first argument, which especifica the coordinate system, is `world`. A `*` is like a wild-card; it means that the module is interested in calls where the corresponding argument has any value. The word `nil` is like `*`, except that the argument's value is not reported to the module. This is useful for cutting down on the amount of data that must be transmitted in cases where there are arguments that the module doesn't care about.

The second, third, fourth, and fifth arguments to the `pick` command give the name, pick point coordenadas, coordenadas do vértice, and edge coordenadas of a pick event. We specify these by `*`'s above. The remaining five arguments to the `pick` command give other information about the pick event that we do not care about in this module, so we specify these with `nil`'s. For the details dos arguments to `pick`, Veja Capítulo 7 [GCL], página 127.

The `geometry` statement defines a geom called `pick` that is initially an empty list, specified as `{ LIST }` ; this is the best way of specifying a null geom. The module will replace this with something useful by sending Geomview another `geometry` command when the user picks something. Next we arrange for the `pick` objeto to be non-pickable, and turn normalization off for it so that Geomview will display it in the size and location where we put it, rather than resizing and relocating it to fit into the unit cube.

The next function in the file, `Lpick`, is defined with a strange looking call to a macro called `PICKFUNC`, defined in the header file '`pickfunc.h`'. This is the function for handling pick events. The reason we provide a macro for this is that that macro encapsulates a lot

of necessary stuff that would be the same for the pick-handling function in any program. If you write a Geomview module that wants to know about user pick events you can just copy this macro call and change it to suit yours needs.

In general the syntax for `PICKFUNC` is

```
PICKFUNC(name, block, NDblock)
```

where *name* is the name do procedure to be defined, in this case `Lpick`. The next argument, *block*, is a block of code to be executed when a pick event occurs. If *block* contains a return statement, then the returned value must be a pointer to a Lisp-objeto, that is of type `LObject *`. The last argument has the same functionality as the *block* argument, but is only invoked when picking objetos in a higher dimensional world.

`PICKFUNC` declares certain local variables in the body do procedure. When the module receives a `(pick ...)` statement from Geomview, the procedure assigns values to these variables based on the information in the `pick` call (variables corresponding to `nil`'s in the `(interest (pick ...))` are not given values).

There is also a second variant da `PICKFUNC` macro with a slightly different syntax:

```
DEFPICKFUNC(helpstr, coordsys, id,
  point, pn, vertex, vn, edge, en, face, fn, ppath, ppn,
  vi, ei, ein, fi,
  body, NDbody)
```

`DEFPICKFUNC` can be used as well as `PICKFUNC`, there is no functional differene with the exception that the name da C-function is tied to `Lpick` when using `DEFPICKFUNC` and that the `(help pick)` GCL-command (veja seção 7.2.61 [help], página 138) would respond with echoing *helpstr*.

The table below lists all variables defined in `PICKFUNC` In the context of ND-viewing `float` variants dos arguments apply: the *body* execution block sees the `HPoint3` variables, and the *NDbody* block sees only flat one-dimensional arrays of `float`-type.

In the ND-viewing context the co-ordinates passed to the pick function are still the 3-dimensional co-ordinates da câmera view-port where the pick occurred, but padded with zeroes on transformed back to the co-ordinate system specified by the second argument do `pick` command.

`char *coordsys;`
: A string specifying the coordinate system in which coordenadas are given. In this example, this will always be `world` because do `interest` call above.

`char *id;` A string specifying the name do picked geom.

`HPoint3 point; int pn;`
`float *point; int pn;`
: `point` is an `HPoint3` structure giving the coordenadas of the picked point. `HPoint3` is a homogeneous point coordinate representation equivalent to an array of 4 floats. `pn` tells how many coordenadas have been written into this array; it will always be either `0`, `4` ou greater than `4`. If it is greater than `4`, then the *NDbody* instruction block is invoked and in this case `point` is a flat array of `pn` many `float`s. A value of zero means no point was picked, i.e. the user clicado the botão direito do mouse while the cursor was not pointing at a geom. In this case the ordinary *block* 3d instruction block is executed.

`HPoint3 vertex; int vn;`
`float *vertex; int vn;`
: `vertex` is an `HPoint3` structure giving the coordenadas of the vértice selecionado, if the pick point was near a vértice. `vn` tells how many coordenadas have been written into this array; it will always be either `0` ou greater equal `4`. A value of zero means the pick point was not near a vértice. In the context of ND-viewing `vertex` will be an array of `vn` `float`s and `vn` will be equal to `pn`.

`HPoint3 edge[2]; int en;`
`float *edge; int en;`
: `edge` is an array of two `HPoint3` structures giving the coordenadas do endpoints da picked edge, if the pick point was near an edge. `en` tells how many coordenadas have been written into this array; it will always be `0` ou greater equal `8`. A value of zero means the pick point was not near an edge. In the context of ND-viewing `edge` will be a flat one-dimensional array of `en` many `float`s: the first `pn` `float`s define the first vértice, and the second `pn` many `float`s define the second vértice; `en` will be two times `pn`.

In this example module, the remaining variables will never be given values because their values in the `interest` statement were specified as `nil`.

`HPoint3 face[]; int fn;`
`float *face; int fn;`
: `face` is a variable length array of *fn* `HPoint3`'s. `face` gives the coordenadas dos vértices da picked face. `fn` tells how many coordenadas have been written into this array; it will always be either `0` ou a multiple of `pn`. A value of zero means the pick point was not near a face. In the context of ND-viewing `face` is a flat one-dimensional array of `fn` many floats of which each vértice occupies `pn` many componentes.

`int ppath[]; int ppn;`
: `ppath` is an array of *maxpathlen* `int`'s. `ppath` gives the path through the OOGL heirarchy to the picked primitive. `pn` tells how many integers have been written into this array; it will be at most *maxpathlen*. A path of `{3,1,2}`, for example, means that the picked primitive is "subobjeto number 2 of subobjeto number 1 of objeto 3 in the world".

`int vi;`
: `vi` gives the index do vértice selecionado in the picked primitive, if the pick point was near a vértice.

`int ei[2]; int ein`
: The `ei` array gives the indices dos endpoints da picked edge, if the pick point was near a vértice. `ein` tells how many integers were written into this array. It will always be either 0 ou 2; a value of 0 means the pick point was not near an edge.

`int fi;`
: `fi` gives the index da picked face in the picked primitive, if the pick point was near a face.

The `handle_pick` procedure actually does the work of dealing with the pick event. It begins by normalizing the homogeneous coordenadas passed in as arguments so that we can

assume the fourth coordinate is 1. It then sends GCL commands to define the `pick` objeto to be whatever is appropriate for the kind of pick recieved. Veja Capítulo 4 [Formatos dos Arquivos da OOGL], página 65, and veja Capítulo 7 [GCL], página 127, for an explanation do format dos data in these commands.

The main program, at the bottom do file, first calls `Initialize()`. Next, the call to `LakeDefine` defines the `Lake` that the lisp library will use. A `Lake` is a structure that the lisp library uses internally as a type of communiation vehicle. (It is like a unix stream but more general, hence the name.) This call to `LakeDefine` defines a `Lake` structure for doing I/O with `stdin` and `stdout`. The third argument to `LakeDefine` should be `NULL` for external modules (it is used by Geomview). Finally, the program enters its main loop which parses and evaluates expressions from standard input.

6.6 Example 4: Simple Tcl/Tk Module Demonstrating Picking

It's not necessary to write a Geomview module in C. The only requirement of an external module is that it send GCL commands to its standard output and expect responses (if any) on its standard input. An external module can be written in C, perl, tcl/tk, ou pretty much anything.

As an example, assuming you have Tcl/Tk version 4.0 ou later, here's an external module with a simple GUI which demonstrates interaction with geomview. This manual doesn't discuss the Tcl/Tk language; see the good book on the subjeto by its originator John Ousterhout, published by Addison-Wesley, titled *Tcl and the Tk Toolkit*.

The '`#!`' on the script's first line causes the system to interpret the script using the Tcl/Tk '`wish`' program; you might have to change its first line if that's in some location other than /usr/local/bin/wish4.0. Or, you could define it as a module using

```
(emodule-define  "Pick Demo"  "wish pickdemo.tcl")
```

in which case '`wish`' could be anywhere on the UNIX search path.

```
#! /usr/local/bin/wish4.0

# We use "fileevent" below to have "readsomething" be called whenever
# data is available from standard input, i.e. when geomview has sent us
# something.  It promises to include a trailing newline, so we can use
# "gets" to read the geomview response, then parse its nested parentheses
# into tcl-friendly {} braces.

proc readsomething {} {
  if {[gets stdin line] < 0} {
        puts stderr "EOF on input, exiting..."
        exit
  }
  regsub -all {\(} $line "\{" line
  regsub -all {\)} $line "\}" line
  # Strip outermost set of braces
  set stuff [lindex $line 0]
  # Invoke handler for whichever command we got.  Could add others here,
```

```
  # if we asked geomview for other kinds of data as well.
  switch [lindex $stuff 0] {
        pick     {handlepick $stuff}
        rawevent {handlekey $stuff}
  }
}

# Fields of a "pick" response, from geomview manual:
#     (pick COORDSYS GEOMID G V E F P VI EI FI)
#          The pick command is executed internally in response to pick
#          events (right mouse double clique).
#
#          COORDSYS = coordinate system in which coordenadas of the following
#              arguments are specified.   This can be:
#               world: world coord sys
#               self:  coord sys of the picked geom (GEOMID)
#               primitive: coord sys of the actual primitive within
#                   the picked geom where the pick occurred.
#          GEOMID = id of picked geom
#          G = picked point (actual intersection of pick ray with objeto)
#          V = picked vertex, if any
#          E = picked edge, if any
#          F = picked face
#          P = path to picked primitive [0 ou more]
#          VI = index of picked vértice in primitive
#          EI = list of indices of endpoints of picked edge, if any
#          FI = index of picked face

# Report when user picked something.
#
proc handlepick {pick} {
  global nameof selvert seledge order
  set obj [lindex $pick 2]
  set xyzw [lindex $pick 3]
  set fv [lindex $pick 6]
  set vi [lindex $pick 8]
  set ei [lindex $pick 9]
  set fi [lindex $pick 10]

  # Report result, converting 4-component homogeneous point into 3-space point.
  set w [lindex $xyzw 3]
  set x [expr [lindex $xyzw 0]/$w]
  set y [expr [lindex $xyzw 1]/$w]
  set z [expr [lindex $xyzw 2]/$w]
  set s "$x $y $z "
  if {$vi >= 0} {
        set s "$s  vertex #$vi"
```

```
  }
  if {$ei != {}} {
        set s "$s  edge [lindex $ei 0]-[lindex $ei 1]"
  }
  if {$fi != -1} {
        set s "$s  face #$fi ([expr [llength $fv]/3]-gon)"
  }
  msg $s
}


# Having asked for notification of these raw events, we report when
# the user pressed these keys in the geomview graphics windows.

proc handlekey {event} {
  global lastincr
  switch [lindex $event 1] {
    32 {msg "Pressed space bar"}
     8 {msg "Pressed backspace key"}
  }
}


#
# Display a message on the control panel, and on the terminal where geomview
# was started.  We use ``puts stderr ...'' rather than simply ``puts ...'',
# since Geomview interprets anything we send to standard output
# as a GCL command!
#
proc msg {str} {
  global msgtext
  puts stderr $str
  set msgtext $str
  update
}

# Load objeto from file
proc loadobject {fname} {
  if {$fname != ""} {
        puts "(geometry thing < $fname)"
        # Be sure to flush output to ensure geomview receives this now!
        flush stdout
  }
}


# Build simple "user interface"
```

```
# The message area could be a simple label rather than an entry box,
# but we want to be able to use X selection to copy text from it.
# The default mouse bindings do that automatically.

entry .msg -textvariable msgtext -width 45
pack .msg

frame .f

  label .f.l -text "File to load:"
  pack .f.l -side left

  entry .f.ent -textvariable fname
  pack .f.ent -side left -expand true -fill x
  bind .f.ent <Return> { loadobject $fname }

pack .f


# End UI definition.


# Call "readsomething" when data arrives from geomview.

fileevent stdin readable {readsomething}

# Geomview initialization

puts {
        (interest (pick primitive))
        (interest (rawevent 32)) # Be notified when user presses space
        (interest (rawevent 8)) # ou backspace keys.
        (geometry thing < hdodec.off)
        (normalization world none)
}
# Flush to ensure geomview receives this.
flush stdout

wm title . {Sample external module}

msg "Click right mouse in graphics window"
```

6.7 Module Installation

Essa seção diz como instalar um módulo externo de forma que você possa invocá-lo dentro do Geomview. Existem duas maneiras de instalar um módulo: você pode installar um

módulo *privado* de forma que o módulo esteja disponível somente para você mesmo sempre que executar o Geomview, ou você pode instalar um módulo de *sistema* de forma que o módulo esteja disponível para todos os usuários do seu sistema sempre que eles executarem o Geomview.

6.7.1 Private Module Installation

O comando `emodule-define` providencia que um módulo apareça no navegador de módulos (*Modules*) do Geomview. O comando recebe dois arumentos que são sequências de caractere; o primeiro é o nome que irá aparecer no navegador de módulos (*Modules*). O segundo é o comando de shell para executar o módulo; esse nome de shell pode incluir argumentos (veja seção 7.2.40 [emodule-define], página 134). Geomview executa esse comando em um subshell quando você dá um clique sobre a entrada do módulo no navegador. Por exemplo

```
(emodule-define "Foo" "/u/home/modules/foo -x")
```

adiciona uma linha rotulada "Foo" ao navegador de módulos (*Modules*) o qual faz com que o comando "/u/home/modules/foo -x" seja executado quando selecionado.

Você pode colocar comandos `emodule-define` no seu arquivo '`~/.geomview`' para providenciar que certos módulos estejam disponíveis sempre que você execute Geomview; veja Capítulo 5 [Customizacao], página 103. Você pode também executar comandos `emodule-define` a partir do painel de comandos (*Commands*) para adicionar um módulo a uma cópia do Geomview que já esteja sendo executada.

Existem muitos outros comandos GCL para controlar as entradas no navegador de módulos (*Modules*); para detalhes, veja Capítulo 7 [GCL], página 127.

6.7.2 System Module Installation

Para instalar um módulo de forma que esse módulo esteja disponível para todos os usuários do Geomview faça o seguinte

1. Crie um arquivo chamado '`.geomview-`*`módulo`*' '*`módulo`*' é o nome do módulo. Esse arquivo deve conter uma linha simples que é um comando `emodule-define` para aquele módulo:

   ```
   (emodule-define "Novo Módulo" "novomodulo")
   ```

   O primeiro argumento, `"Novo Módulo"` acima, é a sequência de caracteres que irá aparecer no navegador de módulo (*Modules*). A segunda sequência de caracteres, `"novomodulo"` acima, é o comando de Bourne shell para invocar o módulo. Esse comando de shell pode incluir argumentos, e você pode assumir que o módulo encontra-se localizado em $PATH que representa os locais de busca do shell.

2. Coloque uma cópia do '`.geomview-`*`módulo`*' e o executável do módulo propriamente dito no diretório '`/usr/local/libexec/geomview`'.

Após esses passos, o novo módulo deve aparecer, em ordem alfabética, no navegador de módulos (*Modules*) do Geomview do painel principal (*Main*) da próxima vez que o Geomview for inicializado. A razão desse trabalho é que quando Geomview for invocado processará todos os arquivos '`.geomview-*`' em seu diretório de '`módulos`'. Geomview também lembra o caminho desse diretório e coloca aquele caminho em $PATH do shell no qual Geomview invoca tal módulo.

7 GCL: a Linguagem de Comandos do Geomview

GCL tem a sintaxe do lisp – i.e. uma expressão da forma (f a b . . .) significa informar os vallores de a, b, . . . para a função f. GCL é muito limitado e GCL não é de forma alguma uma implementação de lisp. GCL é simplesmente uma linguagem para expressar comandos que são executados na ordem fornecida, ao contrário de uma linguagem de programação. GCL não suporta variável ou definição de função.

GCL é a linguagem que Geomview entende para arquivos que chama bem como para comunicação com outros programas. Para executar um comando GCL interativamente, você pode trazer o painel de comandos (*Commands*) que trás sua digitação em um comando; Geomview executa o comando quando você pressiona a tecla ENTER. A saída de tais comandos é mostrada na saída padrão. Alternativamente, você pode invocar Geomview com `geomview -c -` que faz com que o Geomview leia comandos GCL a partir da entrada padrão.

Funções GCL retornam um valor, e você pode concaenar chamadas de função de forma que outras funções usem esse valor retornado. Por exemplo

```
(f (g a b))
```

avalia `(g a b)` e então avalia `(f x)` onde `x` é o resultado retornado por `(g a b)`. Geomview mantém esses valores de retorno internamente normalmente não fornece saída alguma com os resultados guardados. Para mostrar um valor de retorno esse valor de retorno deve ser fornecido à função `echo`. Por exemplo a função `geomview-version` retorna uma sequência de caracteres representado a versão do Geomview que está sendo executada, e

```
(echo (geomview-version))
```

mostra essa sequência de caracteres.

Muitas função simplesmente retornam `t` (de "true" - verdadeiro) caso tenham sido executadas como esperado ou `nil` (nenhum) em caso de falha; esse é o caso se a documentação para a função não fornece o retorno esperado. Esses são os símbolos do lisp para verdadeiro e falso, respectivamente. (Eles correspondem às variáveis definidas em C `Lt` e `Lnil` que você verá se olhar no código fonte do Geomview ou em algum dos módulos externos.)

Nas descrições ds comandos acima muitas refer6encias são feitas a formatos "OOGL" formats. OOGL é a linguagem de descrição dos dados que Geomview utiliza para descrever objetos geométricos, câmeras, aparências, e outos objetos básicos. Para detalhes dos formatos OOGL, veja Capítulo 4 [Formatos dos Arquivos da OOGL], página 65. (Ou equivalentemente, veja a página de manual oogl(5), distributed with Geomview in the file /share/man/man5/oogl.5gv.

Os comandos GCL e tipos de argumentos são listados abaixo. A maioria da documentação nessa seção do manual está disponível dentro do Geomview via comandos `?` e `??`. O comando `(? comando)` faz com que Geomview mostre na tela um sumário de uma linha da sintaxe de *comando*, e `(?? comando)` mostra na tela uma explanação de o que *comando* faz. Você pode incluir o caractere coringa `*` no *comando* para mostrar informações para um frupo de comandos coincidindo com um modelo. Por exemplo, `(?? *emodule*)` irá mostrar todas as informações sobre todos os comandos contendo a sequência de caracteres `emodule`. `(? *)` irá mostrar um lista curta de todos os comandos.

7.1 Conventions Used In Describing Argument Types

Os seguintes símbolos são usados para descrever tipos de argumentos na documentação para funções GCL.

aparência
: é uma especificação de aparência OOGL.

cam-id
: é uma *id*entificação que refre-se a uma câmera.

câmera
: é uma especificação de câmera do OOGL.

geom-id
: é uma *id*entificação que se refere a um objeto geométrico.

geometry
: é uma especificação de objeto geométrico do OOGL.

id
: é uma sequência de caracteres que nomeia um objeto geométrico ou câmera. Como aqueles que você cria, os valores permitidos são:

  `World, world, worldgeom, g0`
  : a coleção de todos os objetos geométricos

  `target`
  : objeto alvo selecionado (câmera ou objeto geométrico)

  `center`
  : objeto central do movimento selecionado

  `targetcam`
  : a última câmera alvo selecionada

  `targetgeom`
  : o último objeto geométrico alvo selecionado

  `focus`
  : câmera onde o cursor está (ou mais recentemente esteve)

  `allgeoms`
  : todos os objetos geométricos

  `allcams`
  : todas as câmeras

  `default, defaultcam, prototype`
  : câmeras futuras que herdarão as escolhas padronizadas

  As seguintes *id*s são usadas para nomear sistemas de coordenadas, e.g. em comandos `pick` e `write`:

  `World, world, worldgeom, g0`
  : o objeto mundo, dentro do qual todos os objetos geométricos vivem.

  `universe`
  : o universo, no qual o no qual o objeto mundo, as luzes e as câmeras vivem. Transformações world2cam das câmeras podem melhor serem chamadas universe2cam, etc.

  `self`
  : "esse objeto do Geomview". Transforma de um objeto para `si mesmo` é a identidade; escrevendo seu objeto geométrico fornece o bojeto em si mesmo sem executar nenhuma transformação; pontos selecionados aparecem nas coordenadas do objeto.

  `primitive`
  : (para objetos selecionados (`pick`) somente) Pontos selecionados aparecem no sistema de coordenadas da primitiva de menor nível do OOGL.

Um nome também é uma `id`entificação aceitável. Fornecimento de nomes é feito único pela anexação de números se necessário (i.e. `foo<2>`). Todo objeto geométrico é também chamado g[n] e toda câmera é também chamada c[n] (`g0` é sempre o objeto geométrico mundo - worldgeom): esse nome é usado como um prefixo a comandos de teclado e pode também ser usado como uma identificação GCL. Números são usados após um objeto ser deletado. Ambos os nomes são mostrados no navegador de Objeto.

declaração
: representa uma chamada de função. Chamadas a funções possuem a forma `(func arg1 arg2 ... )`, onde `func` é o nome da funçào e `arg1`, `arg2`, `...` são os argumentos.

transformação
: é uma matriz de transformação OOGL 4x4.

ntransform
: é uma matriz de transformação OOGL (N+1)x(N+1).

janela
: é uma especificação de janela do OOGL.

7.2 GCL Reference Guide

Nota do tradutor: os termos “expressão lambda" e “expressão S" são específicos da linguagem de programação Lisp.

7.2.1 !

`!`
: é um sinônimo de `shell`. Veja seção 7.2.129 [shell], página 152.

7.2.2 <

`(< EXPR1 EXPR2)`
: Retorna t se EXPR1 for menor que EXPR2. EXPR1 e EXPR2 devem ser ou ambas inteiras ou número em ponto flutuante, ou ambas sequências de caractere.

7.2.3 =

`(= EXPR1 EXPR2)`
: Retorna t se EXPR1 for igual a EXPR2. EXPR1 e EXPR2 devem ser ou ambas inteiras ou número em ponto flutuante, ou ambas sequências de caractere.

7.2.4 >

`(> EXPR1 EXPR2)`
: Retorna t se EXPR1 for maior que EXPR2. EXPR1 e EXPR2 devem ser ou ambas inteiras ou número em ponto flutuante, ou ambas sequências de caractere.

7.2.5 *

`(* EXPR1 EXPR2)`
: Multiplica *EXPR1* por *EXPR2* e retorna o resultado.

7.2.6 /

`(/ EXPR1 EXPR2)`
: Divide *EXPR1* por *EXPR2* e retorna o resultado.

7.2.7 +

`(+ EXPR1 EXPR2)`
: Adiciona *EXPR1* a *EXPR2* e retorna o resultado.

7.2.8 -

`(- EXPR1 EXPR2)`
: Subtrai *EXPR2* da *EXPR1* e retorna o resultado.

7.2.9 ?

`(? [comando])`
: Fornece sumário de uso em uma linha para `comando`. Comando pode incluir `*`s como caracteres coringa; veja também seção 7.2.83 [morehelp], página 143. Ajuda de comando em uma linha; lista nomes somente se multiplos comandos coincidirem. `?` é um sinônimo para seção 7.2.61 [help], página 138

7.2.10 ??

`(?? comando)`
: `comando` pode incluir coringas `*`. Mostra mais informação que `(? comando)`. `??` é um sinônimo para seção 7.2.83 [morehelp], página 143.

7.2.11 |

`|`
: `|` ém um sinônimo para `emodule-run`.

7.2.12 all

`(all geometry)`
: retorna uma lista de nomes de todos os objetos geometry. Use e.g. '`(echo (all geometry))`' para mostrar tal lista.

`(all camera)`
: retorna uma lista de nomes de todas as câmeras.

`(all emodule defined)`
: retorna uma lista de todos os módulos externos definidos.

`(all emodule running)`
: retorna uma lista de todos módulos externos em execução.

7.2.13 and

`(and EXPR1 EXPR2)`
: Avalia `EXPR1` e `EXPR2` e retorna `t` se ambas retornarem não-`nil`, de outra forma retorna `nil`.

7.2.14 ap-override

`(ap-override [on|off])`

Seleciona se controles de aparência devem sobrescrever ajustres dos próprios objetos. Habilitado por padrão. Sem argumentos, retorna o ajuste atual.

7.2.15 backcolor

`(backcolor CAM-ID R G B)`

Ajusta a cor de fundo da CAM-ID; R G B são números entre 0 e 1.

7.2.16 background-image

`(background-image CAM-ID [NOMEDOARQUIVO])`

Use a imagem fornecida como fundo da câmera CAM-ID (a qual deve ser uma câmera real, não pode ser ou `default` ou `allcams`). A imagem é centralizada na área da janela. Trabalha somente com gráficos GL e OpenGL. Use `""` como nome de arquivo para remover o fundo. Sem argumentos, retorna o nome da imagem que é usada atualmente como fundo da janela, ou `""`. Qualquer tipo de arquivo aceitável como textura é permitido, e.g. .ppm.gz, .sgi, etc.

7.2.17 bbox-color

`(bbox-color GEOM-ID R G B)`

Ajusta a cor da caixa associada do GEOM-ID; R G B são números entre 0 e 1.

7.2.18 bbox-draw

`(bbox-draw GEOM-ID [yes|no])`

Diz se a caixa associada do GEOM-ID deve ser desenhada; a escolha padrão é `yes` se o segundo argumento for omitido.

7.2.19 camera

`(camera CAM-ID [CAMERA])`

Especifica dados para *CAM-ID*; *CAMERA* é uma sequência de caracteres fornecendo uma especificação de câmera OOGL. Se nenhuma *CAM-ID* de câmera existir, essa *CAM-ID* é criada; nesse caso, o segundo argumento é opcional,e se omitido, uma câmera padrão é usada. Veja seção 7.2.91 [new-camera], página 145.

7.2.20 camera-draw

`(camera-draw CAM-ID [yes|no])`

Diz se câmeras devem ou não serem desenhadas em CAM-ID; `yes` se omitido.

7.2.21 camera-prop

`(camera-prop { geometry object } [projective])`

Especifica o objeto a ser mostrado quando desenhando outras câmeras. Por padrão, esse objeto é desenhado com sua origem na camera, e com a câmera olhando adiante do eixo -Z do objeto. Com a palavra chave `projective`, a

projeção da visão da câmera é também aplicada ao objeto; isso coloca o Z=-1 e o Z=+1 do objeto perto e distante dos planos de corte, com a área de visão -1<={X,Y}<=+1. Exemplo: (camera-prop { < cube } projective)

7.2.22 camera-reset

`(camera-reset CAM-ID)`
Ajusta CAM-ID para seu valor padrão.

7.2.23 car

`(car LISTA)`
retorna o primeiro elemento de LISTA.

7.2.24 cdr

`(cdr LISTA)`
retorna a lista obtida removendo o primeiro elemento de LISTA.

7.2.25 clock

`(clock)` Retorna a hora atual, em segundos, como mostrado por esse relógio do fluxo. Veja seção 7.2.121 [set-clock], página 150. Veja seção 7.2.131 [sleep-until], página 152.

7.2.26 command

`(command ARQUIVOENTRADA [ARQUIVOSAIDA])`
Lê comandos de ARQUIVOENTRADA; envia respostas correspondentes (e.g. qualquer coisa escrita para nome de arquivo -) para ARQUIVOSAIDA, stdout por padrão.

7.2.27 cons

`(cons EXPR LISTA)`
Retorna a lista obtida adicionando *EXPR* como primeiro elemento da *LISTA*. Note que o segundo argumento tem de ser uma lista.

7.2.28 copy

`(copy [ID] [nome])`
Copia um objeto ou câmera. Se ID não for especificado, esse ID é assumido como sendo targetgeom. If nome não for especificado, esse nome é assumido como sendo o mesmo nome de ID.

7.2.29 cursor-still

`(cursor-still [INT])`
Ajusta o número de microssegundos para os quais o cursor não deve mover-se para registrar como fixo. Se INT não for especificado, o valor irá ser ajustado para o valor padrão.

7.2.30 cursor-twitch

`(cursor-twitch [INT])`

Ajusta a distância na qual o cursos não deve mover-se (em x ou y) para registrar como fixo. Se INT não for especificado, o valor irá ser ajustado para o valor padrão.

7.2.31 defun

`(defun NOME (ARG1 ...) [DOCSTRING] EXPR1 ...)`

Define uma chamada expressão lambda, isto é: define *NOME* para avaliar para a expressão lambda `(lambda (ARG1 ...) (EXPR1 ...))` quando chamada como uma função. Também, instala *DOCSTRING* como resposta para os comandos `(help NOME)` e `(morehelp NOME)`. Note que *DOCSTRING* não precisa conter a sinopse do comando, essa sinopse é gerada automaticamente. *EXPR1* não pode ser uma sequência de caracteres se `DOCSTRING` for omitida; *EXPR1* deve ser interpretada como a sequência de caracteres documento. O valor de retorno da `(defun ...)` é o nome da função. Funções podem ser recursivas e podem modificar a si mesmas. é posível redefinir funções internas, nesse caso a definição antiga está ainda disponível so o nome `-builtin-OLDNAME-`. Valores de argumento podem ser alterados por `setq`; a nova associação é descartada após avaliação de surrounding`defun`-body. As palavras especiais `&optional` e `&rest` possuem o mesmo significado que a expressão lambda anonymous, veja nas referência adiante. Veja seção 7.2.68 [lambda], página 140. Veja seção 7.2.127 [setq], página 151. Veja seção 7.2.69 [let], página 140.

7.2.32 delete

`(delete ID)`

Apaga objeto ou câmera especificado em ID.

7.2.33 dice

`(dice GEOM-ID N)`

Divide qualquer ajustes Bezier dentro de *GEOM-ID* em malhas de medida NxN; o padrão para N é 10. Veja também o atributo de aparência seção 4.1.10 [Aparencias], página 69, o qual torna esse comando obsoleto.

7.2.34 dimension

`(dimension [N])`

Ajusta ou lê a dimensão do espaço para visão N-dimensional. (Uma vez que cálculos forem concluídos usando coordenadas homogêneas, isso significa que matrizes são (N+1)x(N+1).) Sem argumentos, retorna a dimensão atual, ou 0 se a visualização N-dimensional não puder ser habilitada.

7.2.35 dither

`(dither CAM-ID {on|off|toggle})`

Alterna entre mistura de cores ligada e desligada em CAM-ID.

7.2.36 draw

`(draw CAM-ID)`

Desenha a visão em CAM-ID, se a visão precisar ser redesenhada. Veja seção 7.2.113 [redraw], página 149.

7.2.37 dump-handles

`(dump-handles)`

Descarrega a lista dos controladores ativos atualmente para a saída padrão. Essa função é pensada para uso em depuração interna somente.

7.2.38 echo

`(echo ...)`

Escreve os dados fornecidos para o arquivo especial `-`. Sequências de caracteres são escritas literalmente; expressões em lisp são avaliadas e seus valores escritos. Se recebido de um programa externo, `echo` envia para a entrada do programa. De outra forma escreve para a própria saída padrão do Geomview (tipicamente o terminal).

7.2.39 emodule-clear

`(emodule-clear)`

Limpa o navegador de aplicação do Geomview (módulo externo).

7.2.40 emodule-define

`(emodule-define NOME COMANDO-SHELL ...)`

Define um módulo externo chamado NOME, o qual então aparece no navegador de módulos externos. A sequência de caracteres COMANDO-SHELL é um comando shell UNIX que chama o módulo. Veja seção 7.2.44 [emodule-run], página 135 para discurssão sobre módulos externos.

7.2.41 emodule-defined

`(emodule-defined nomemodulo)`

Se o nome do módulo externo for conhecido, retorna o nome do programa chamado quando nomemodulo está executando como sequência de caracteres entre aspas duplas; de outra forma retorna nil. `(echo (emodule-defined nome))` mostra na tela a sequência de caracteres.

7.2.42 emodule-isrunning

`(emodule-isrunning NOME)`

Retorna Lt se o módulo externo NOME estiver rodando, ou Lnil se o módulo externo não estiver rodando. NOME é pesquisado nos nomes como eles aparecem no navegador de módulos e nos comandos de shell usados para executar o módulo externo (não incluindo os argumentos).

7.2.43 emodule-path

`(emodule-path)`

Retorna o caminho de busca atual para módulos externos. Nota: para ver agora o valor retornado por essa função você deve envolver emodule-path em uma chamada ao comando shell echo: `(echo (emodule-path))`. Veja seção 7.2.123 [set-emodule-path], página 151.

7.2.44 emodule-run

`(emodule-run COMANDO-SHELL ARGS...)`

Executa o COMANDO-SHELL fornecido (uma sequência de caracteres contendo um comando shell UNIX) como um módulo externo. A saída padrão do módulo é interpretado como comandos do geomview; respostas (escrita para nomearquivo -) são enviadas para a entrada padrão do módulo. O comando shell é interpretado por /bin/sh, de forma que redirecionamento de E/S pode ser usada: um programa que pergunta ao usuário por entradas a partir do terminal poderá vir a ser executado com:

```
(emodule-run  seuprograma  <&2)
```

Caso já não tenha sido ajustada, a variável de ambiente $MACHTYPE é ajustada para o nome do tipo da máquina. Conecções de entrada e saída para o geomview são liberadas automaticamente quando o comando shell encerra. Clicando sobre um prorama que está sendo executado na entrada do navegador de módulos envia o sinal SIGHUP ao programa. Para que esse recurso funcione, programas devem evitar executar em segundo plano; os programas usando as bibliotecas FORMS ou GL devem chamar a função foreground() antes da primeira chamada a FORMS ou winopen(). Veja seção 7.2.40 [emodule-define], página 134. Veja seção 7.2.46 [emodule-start], página 135.

7.2.45 emodule-sort

`(emodule-sort)`

Ordena os módulos alfabéticamente no navegador de aplicação.

7.2.46 emodule-start

`(emodule-start NOME)`

Inicia o módulo externo NOME, definido por emodule-define. Equivalente a clicar na entrada correspondente no navegador de módulos.

7.2.47 emodule-transmit

`(emodule-transmit NOME LISTA)`

Coloca LISTA na entrada padrão do módulo externo NOME. NOME é pesquisado nos nomes dos módulos na forma em que esses nomes aparecem no navegador de Módulos Externos e a seguir nos comandos shell usados para executar os módulos externos. Não faz nada se o módulo NOME nõ estiver sendo executado.

7.2.48 escale

`(escale GEOM-ID FACTOR)`

O mesmo que scale mas multiplicado por exp(scale). Obsoleto.

7.2.49 eval

`(eval EXPR)`

Avalia uma expressão lisp. Se *EXPR* é uma expressão-S não avaliada como retornado pelo comando `(apóstrofo ...)` então o efeito irá ser ser como se tivesse chamado a expressão sem apóstrofo diretamente. A eval também torna possível avaliar expressões-S construídas via `car`, `cdr` e `cons`. Veja seção 7.2.23 [car], página 132. Veja seção 7.2.24 [cdr], página 132. Veja seção 7.2.27 [cons], página 132.

7.2.50 event-keys

`(event-keys {on|off})`

Alterna entre eventos de teclado on ou off para habilitar/desabilitar teclas de atalho.

7.2.51 event-mode

`(event-mode SEQ_CARAC_MODOS)`

Ajusta o modo de um evento do mouse (movimento); SEQ_CARAC_MODOS deve ser um entre as seguintes sequências de caractere:

1. `"[r] Rotate"`
2. `"[t] Translate"`
3. `"[z] Cam Zoom"`
4. `"[s] Geom Scale"`
5. `"[f] Cam Fly"`
6. `"[o] Cam Orbit"`
7. `"[le] Edit Lights"`

Início de nota do tradutor:

Alguns termos não são de significado tão óbvio.

1. `"[r] Rotate"` Rotação
2. `"[t] Translate"` Translação
3. `"[z] Cam Zoom"` Aproximação ou afastamento de câmera
4. `"[s] Geom Scale"` Homotetia
5. `"[f] Cam Fly"` Voo de câmera
6. `"[o] Cam Orbit"` rbita de câmera
7. `"[le] Edit Lights"` Editar luzes

Fim de nota do tradutor.

7.2.52 event-pick

`(event-pick {on|off})`
Alterna entre selecionar on ou off.

7.2.53 evert

`(evert GEOM-ID [yes|no])`
Ajusta o estado normal de eversão (nota do tradutor: ver do outro lado de alguma superfície sobre a direção do vetor normal à mesma superfície) de GEOM-ID. Se o segundo argumento for omitido, inverte o estado de eversão.

7.2.54 exit

`(exit)` Encerra o geomview.

7.2.55 ezoom

`(ezoom GEOM-ID FATOR)`
O mesmo que zoom mas multiplica por exp(zoom). Obsoleto.

7.2.56 freeze

`(freeze CAM-ID)`
Congela CAM-ID; desenho nessa janela de camera é desligado até que a imagem dessa câmera seja redesenhada com `(redraw CAM-ID)`, após o redesenho da imagem da câmera alterações na imagem voltam a ser permitidas.

7.2.57 geometry

`(geometry GEOM-ID [GEOMETRIA])`
Especifica a geometria para GEOM-ID. GEOMETRIA é uma sequência de caracteres fornecendo a especificação de um objeto geométrico OOGL. Se nenhum objeto chamado GEOM-ID existir, esse objeto inesistente será criado; nesse caso o argumento GEOMETRIA é opcional, e se omitido, o novo objeto chamado GEOM-ID é retornado sendo um objeto geométrico vazio.

7.2.58 geomview-version

`(geomview-version)`
Retorna uma sequência de caracteres representado a versão do geomview que está executando.

7.2.59 hdefine

`(hdefine geometria|camera|janela|aparência|imagem|transformação|ntransform nome valor)`
Ajusta o valor de um controlador de tipo fornecido.

```
(hdefine  <tipo>  <nome>  <valor>)
```

is generally equivalent to

```
(read <tipo>  { define <nome> <valor> })
```

exeto que a atribuição é desfeita ao final da execução de hdefine, (possivelmente não em todos os lugares se dentro de uma declaração condicional), enquanto oe `read ... define` realiza a atribuição tão rapidamente quanto o texto é lido. Veja seção 4.1.9 [Referencias], página 68. Veja seção 7.2.111 [read], página 149. Veja seção 7.2.60 [hdelete], página 138.

7.2.60 hdelete

`(hdelete [geometria|camera|janela|aparência|imagem|transform|ntransform] nome)`

Apaga o controlador fornecido. Note que o controlador não irá atualmente ser apagado no caso de existir ainda outros objetos fazendo referência ao controlador, mas uma vez que os objetos que fazem referência ao maipulador apagado forem fechados, o controlador irá também ser automaticamente mandado embora. O objeto que faz referência ao controlador (se existir algum) irá somente ser deletado se não existirem outras referências para esse objeto.

Se o opcional primeiro argumento for omitido, então o primeiro controlador que coincidir com *nome* irá ser apagado, independentemente do tipo de objeto ao qual esse controlador estiver anexado. Não é um erro chamar essa função com um controlador inexistente, mas é uma erro chamar essa função com o nome de um controlador não global, i.e. um que não tenha sido criado por `(hdefine ...)` ou `(read ... { define ...})`. Veja seção 4.1.9 [Referencias], página 68. Veja seção 7.2.111 [read], página 149. Veja seção 7.2.59 [hdefine], página 137.

7.2.61 help

`(help [comando])`

O comando pode incluir `*`s como caracteres coringa; veja também seção 7.2.9 [help-synonym], página 130 Ajuda de comando em uma linha; lista nomes somente se multiplos comandos coincidirem.

7.2.62 hmodel

`(hmodel CAMID {virtual|projective|conformal})`

Ajusta o modelo usado para mostrar o objeto geométrico nessa câmera. Veja seção 7.2.134 [space], página 153.

7.2.63 hsphere-draw

`(hsphere-draw CAMID [yes|no])`

Informa se é para desenhar ou não a esfera unitária: a esfera no infinito no espaço hiperbólico, e uma esfera de referência nos espaços Euclidiano esférico. Se o segundo argumento for omitido, `yes` é assumido.

7.2.64 if

`(if TEST EXPR1 [EXPR2])`

Avalia TEST; se TEST retornar um valor não-nil (nota do tradutor:não nulo), retorna o valor de EXPR1. Se TEST retornar nil, retorna o value de EXPR2 se EXPR2 estiver presente, de outra forma retorna nil.

7.2.65 inhibit-warning

`(inhibit-warning STRING)`

Inibe alertas de inibição do geomview de mostrar uma mensagem de alerta determinada por STRING. Atualmente não existe mensgenns de alerta na qual o comando inhibit-warning seja aplicado, de forma que esse comando é pouco útil.

7.2.66 input-translator

`(input-translator "#prefix_string" "comando-shell")`

Define um programa externo de tradução para tipos especiais de entrada. Quando perguntado se é para lê um arquivo especial que começa com a sequência de caracteres especificada, geomview chama comando-shell com entrada padrão obtida a partir do arquivo especificado. É esperado que comando-shell emita dados geométricos na linguagem OOGL para sua saída padrão. Nessa implementação, somente prefixos iniciando com # são reconhecidos. Muito útil em situações como

```
(input-translator "#VRML" "vrml2oogl")
```

7.2.67 interest

`(interest (COMANDO [args]))`

Permite a você expressar interesse em um comando. Quando geomview vier a executar o comando de interesse futuramente o comando de interesse será ecoado para o sistema de comunicação do qual o comando interest for originário. *COMANDO* pode ser qualquer comando. Args especifica restrições sobre os valores dos argumentos; se args estiver presente no comando interest, geomview irá somente ecoar chamadas para o comando no qual os argumentos coincidirem com aquele fornecido no comando interest. Dois valores especiais de argumento podem aparecer na lista de argumentos. `*` que coincide com qualquer valor. `nil` que coincide com qualquer valor mas suprime o retorno daquele valor; seus valores são reportados como `nil`.

O propósito do comando interest é permitir a módulos externos encontrar coisas acontecendo dentro do geomview. Por exemplo, um módulo interessado em saber quando um geom chamado `foo` é apagado pode usar o comando interest da seguinte forma `(interest (delete foo))` e iria receber a sequência de caracteres `(delete foo)` quando foo fosse apagado.

Destacando um caso especial do uso do comando interest. Para a maioria dos módulos interessados em selecionar eventos o comando `(interest (pick world))` é suficiente. O comando `(interest (pick world))` faz com que geomview envie uma sequência de caracteres da forma `(pick world ...)` toda

vez que um evento de seleção (duplo clique botão direito do mounse). Veja o comando seção 7.2.99 [pick], página 146 para detalhes.

7.2.68 lambda

`(lambda (ARG1 ...) EXPR1 ... EXPRN)`

Uma expressão lambda é como uma função. Para "chamar" uma expressão lambda, a expressão lambda tem de ser chamada como uma função: `((lambda (arg) (+ 1 arg)) 2)`. Nesse exemplo, o valor completo da expressão deve ser 3. Em geral, o valor da chamada irá ser o valor de *EXPRN*. A primeira lista serve para definir os parâmetros formais. A expressão lambda propriamente dita é apenas uma lista, iniciando-se com a palavra chava lambda, seguida por muitas listas entre aspas duplas. Veja seção 7.2.31 [defun], página 133. Veja seção 7.2.127 [setq], página 151. Veja seção 7.2.69 [let], página 140. Note que a lista argumento pode conter as palavras chaves especiais

`&optional`
: fornecimento de valores aos identificadores seguintes é opcional, seus valores padrão irão ser `nil`

`&rest`
: todos os argumentos excedentes irão ser coletados em uma lista, e essa lista irá ser atribuída ao argumento seginte, da seguinte forma:

```
((lambda (&rest rest) (echo rest)) a b c d)
```

A saída irá ser `(a b c d)`.

7.2.69 let

`(let ARGUMENTS EXPR1 ... EXPRN)`

Gera uma expressão lambda a partir de *EXRP1 ... EXPRN*, com a associação de argumento descrita por *ARGUMENTS*. *ARGUMENTS* corresponde a uma lista de símbolos (associada a `nil` por padrão) ou a listas da forma `(ARG VALUE)` onde *ARG* é um símbolo e não avaliado e *VALUE* é uma expressão-S que é primeiramente avaliada, a seguir seu valor é associa a *ARG*. A expressão completa avalia para o valor de *EXPRN*, a última expressão no corpo da declaração. A lista de argumento deve ester presente, mas pode ser vazia; no último caso a declaração `(let () ...)` é equivalente a um `(progn ...)`. Veja seção 7.2.68 [lambda], página 140. Veja seção 7.2.31 [defun], página 133. Veja seção 7.2.127 [setq], página 151.

7.2.70 lines-closer

`(lines-closer CAM-ID DIST)`

Desenha linhas (incluindo arestas) próximo à câmera do polígonos a uma distância DIST / 10^5 do intervalo contido na área de armazenamento temporário de memória que controla as coordenadas do eixo Z. DIST = 3.0 por padrão. Se DIST for muito pequena, uma linha aproxumada sobre uma superfície pode ser pontilhada ou invisível, dependendo do ponto de vista. Se DIST for muito grande, linhas podem aparecer em frente das superfícies que elas atualmente aproximam por trás. Bons valores para DIST variam com a

cena, ponto de visão, e distâncias entre planos de corte próximo e distante. Esse recurso é um remendo, mas pode ser de grande ajuda.

7.2.71 load

`(load filename [command|geometry|camera])`

Chama o arquivo fornecido dentro do geomview. O segundo argumento opcional especifica o tipo de dado que o referido arquivo chamado contém, o qual pode ser ou `command` (comandos do geomview), `geometry` (dados geométrico no formato OOGL), ou `camera` (definição de câmera no formato OOGL). se omitido, é tentado deduzir o tipo de conteúdo do arquivo. Carregando dados geométricos cria um novo objeto visível; carregando uma câmera abre uma nova janela; chamado um arquivo contendo comando executa o referido comando.

7.2.72 load-path

`(load-path)`

Retorna o atual caminho de busca para arquivos contendo comandos, objetos geométricos, etc. Nota: para ver o atual valor retornado por essa função você deve empacotar esse comando em uma chamada a echo: `(echo (load-path))`. Veja seção 7.2.124 [set-load-path], página 151.

7.2.73 look

`(look [objetoID] [cameraID])`

Rotaciona a referida câmera - cameraID - de forma que aponte em direção ao centro da caixa associada ao referido objeto - objetoID (ou a origem nos espaços hiperbolico ou esférico). No espaço Eucidiano, move a câmera para além ou para trás até que o objeto apareça tão grande quanto possível enquanto sendo inteiramente visível. Equivalente a

```
progn (
(look-toward [objetoID] [cameraID] {center | origin})
[(look-encompass [objetoID] [cameraID])]
)
```

Se objetoID não for especificado, esse objeto á assumido como sendo o objeto mundo. Se cameraID não for especificado, essa câmera é assumido como sendo a targetcam.

7.2.74 look-encompass

`(look-encompass [objetoID] [cameraID])`

Move cameraID para trás ou para adiante até que seu campo de visão alcance objetoID. Essa rotina trabalha somente no espaço Euclidiano. Se objetoID não for especificado, esse objeto á assumido como sendo o objeto mundo. Se cameraID não for especificado, essa câmera é assumido como sendo a targetcam. Veja seção 7.2.75 [look-encompass-size], página 142.

7.2.75 look-encompass-size

`(look-encompass-size [ver-fração razão-corte margem-proxima margem-distante])`

Ajusta/retorna parâmetros usados por (look-encompass). ver-fração é a porção da janela de câmera preenchida pelo objeto, razão-corte é a razão máxima permitida entre os planos de corte próximo e afastado. O plano de corte próximo é 1/magem-proxima vezes mais perto que a aresta mais próxima do objeto, e o plano de corte distante é margem-distante vezes mais adiante. Retorna a lista dos valores atuais. Valores padronizados: .75 100 0.1 4.0

7.2.76 look-recenter

`(look-recenter [objetoID] [cameraID])`

Translada e rotaciona a câmera de forma que essa câmera esteja olhando na direção-z (no sistema de coordenadas de objetoID) no centro da caixa associada a objetoID (ou a origem do sistema de coordenadas no espaço não Euclidiano). No espaço Euclidiano, a câmera é também movida para tão perto quanto possível do objeto de forma a permitir que o objeto seja inteiramente visível. Também garante que o eixos y do objetoID e de cameraID sejam paralelos.

7.2.77 look-toward

`(look-toward [objetoID] [cameraID] [origin | center])`

Rotaciona a câmera especificada de forma a apontar para adiante da origem do sistema de coordenadas do objeto, ou do centro da caixa associada ao objeto (no espaço não Euclidiano, a origem irá ser usada automaticamente). O objetoID padrão é o objeto mundo, a câmera padrão é targetcam, a localização padrão é para a qual é apontada a câmera é adiante do centro da caixa associada a objetoID.

7.2.78 merge

`(merge {window|camera} CAM-ID { WINDOW ou CAMERA ... })`

Modifica a janela ou a câmera fornecido, mudando apenas a propriedade especificada no último argumento. E.g.

```
(merge camera Camera { far 20 })
```

ajusta o plano de corte afastado para 20 permanecendo os outros atributos inalterados.

7.2.79 merge-ap

`(merge-ap GEOM-ID APARÊNCIA)`

Mescla em algumas características de aparência para GEOM-ID. Parâmetros de aparência incluem cor de linha e de superfície, estilo de sombreamento, espeçura de linha, e iluminação.

7.2.80 merge-base-ap

`(merge-base-ap APARÊNCIA)`

`merge-base-ap` é um sinônimo para seção 7.2.81 [merge-baseap], página 143.

7.2.81 merge-baseap

`(merge-baseap APARÊNCIA)`

Mescla em algumas características de aparência para a base padão de aparência (aplicada a todo geom antes de sua própria aparência). Iluminação está tipicamente incluída na base de aparência.

7.2.82 mod

`(mod EXPR1 EXPR2)`

Divide *EXPR1* por *EXPR2* e retorna o resto.

7.2.83 morehelp

`(morehelp comando)`

`comando` pode incluir coringas `*`. Mostra na tela mais informação que seção 7.2.61 [help], página 138.

7.2.84 name-object

`(name-object ID NOME)`

Atribui um novo NOME (uma sequência de caracteres) a ID. Um número é adicionado no final se NOME já estiver sendo usado (por exemplo, `foo -> foo<2>`). O novo nome, possivelmente com número anexado no final, pode ser usado como id de objeto posteriormente.

7.2.85 ND-axes

`(ND-axes CAMID [NOMEGRUPO [Xindex Yindex Zindex [Windex]]])`

No nosso modelo para visualização N-Dimensional (habilitado por (dimension)), objetos no espaço N-dimensional são visualizados por N-dimensional *grupos de câmera*. Cada janela real de câmera pertence a algum grupo de câmeras, e mostra & controla um subspaço projetado eixo-alinhado 3-D do espaço N-dimensional visto pelo seu grupo. Movendo uma câmera em um grupo afeta todos os outros membros do grupo.

O comando ND-axes configura tudo isso. O comando ND-axes especifica uma associação de câmera a um grupo, e o ajuste dos eixos do espaço N-dimensional os quais tornam-se os eixos X, Y, e Z da câmera. Eixos são especificados por seus índices, de 1 a N para um espaço N-dimensional. O grupo NOMEGRUPO é implicitamente criado se não for previamente conhecido.

Em princípio é possível mapear a componente homogênea de uma conformação de 4 pontos para algum outro índice; isso poderia ser realizado especificando 0 para um dos `Xindex`, `Yindex` ou `Zindex` e fornecendo a `Windex` algum valor positivo. Isso provavelmente não é útil pelo fato de Geomview não suportar geometria não-Euclidianas para dimensões mais altas.

Para ler uma configuração de câmeras, use `(echo (ND-axes CAMID))`. O valor de retorno é uma vetor fixo de 4 inteiros, o último dos quais deve ser 0.

7.2.86 ND-color

`(ND-color CAMID`
[(([ID] (x1 x2 . . . xN) v r g b a v r g b a . . .) ((x1 . . . xN) v r g b a v r g b a . . .) . . .)]) Especifica uma função, aplicada a cada vértice N-dimencional, o qual determina as cores dos objetos N-dimensionais como mostrado na câmera CAMID. Cada função de cor é definida por um vetor (no sistema de coordenadas do objeto geométrico ID) [x1 . . . xN] e por uma sequência de quíntuplas valor (v)/cor(r g b a), ordenados por ordem crescente de v. O produto interno v = P.[x] é linearmente interpolado nessa tabela para fornecer uma cor. Se ID for omitido, o vetor (xi) é assumido em coordenadas de universo. O comando ND-color especifica uma lista de tais funções; cada vértice é colorido por seu somatório (então e.g. intensidade verde pode indicar projeção ao longo de um eixoenquanto vermelho indica outro eixo. Uma lista vazia, como em (ND-color CAMID ()), suprime o colorido. Sem segundo argumento, (ND-color CAMID) retorna a lista de cor-função de coloração. Mesmo quando coloração está habilitada, objetos acompanhados com o atributo de aparência `keepcolor` são mostrados em suas cores naturais.

7.2.87 ND-xform

`(ND-xform OBJID [ntransform { idim odim ... }])`
Concatena a fornecida transformação-ND com a atual transformação-ND do objeto (aplica a transformação-ND para objeto ID, como oposiçaõ a simplesmente ajustar sua transformação-ND). Note que ND-transforms possuem suas coordenadas homogêneas iniciando-se no índice 0, enquanto transformações 3D possuem seu início no índice 3.

7.2.88 ND-xform-get

`(ND-xform-get ID [from-ID])`
Retorna a transformação N-D do objeto fornecido no sistema de coordenadas do from-ID (o padrão é `universe`), no sentido <ponto-em-ID-coords> * Transform = <ponto-em-from-ID-coords>. Note que ND-transforms possuem suas coordenadas homogêneas iniciando-se no índice 0, enquanto transformações 3D possuem seu início no índice 3.

7.2.89 ND-xform-set

`(ND-xform-set OBJID [ntransform { idim odim ... }])`
Ajusta a transformação N-D do objeto fornecido. Na dimensão N, a transformação é uma matriz (N+1)x(N+1), de forma que naquele caso idim e odim são esperados serem ambos iguais a (N+1). Note que todas as câmeras em um camera-grupo possuem a mesma transformação N-D. Note que ND-transforms possuem suas coordenadas homogêneas iniciando-se no índice 0, enquanto transformações 3D possuem seu início no índice 3.

7.2.90 new-alien

`(new-alien name [GEOMETRIA])`

Cria um novo alien (objeto geométrico fora de "objeto mundo") com o nome fornecido (uma sequência de caracteres). *GEOMETRIA* é uma sequência de caracteres fornecendo um especificação de objeto geométrico OOGL. Se *GEOMETRIA* for omitido, o novo alien é fornecido como sendo um objeto geométrico vazio. Se um objeto como o nome fornecido já existir, ao novo alien é dado um novo nome. Os feixes luminosos que são usados para mover em volta das luzes são um exemplo de aliens. Eles são desenhados mas não controláveis pelo caminho normal que os objetos comuns são: eles não aparecem no navegador de objetos e o usuário não pode movê-lo com o modo de movimento normal.

7.2.91 new-camera

`(new-camera name [CAMERA])`

Cria uma nova câmera com o nome fornecido (uma sequência de caracteres). Se uma câmera com o nome fornecido já existir, ao novo objeto é fornecido um nome único. Se *CAMERA* for omitido uma câmera padrão é usada.

7.2.92 new-center

`(new-center [id])`

Cessa o movimento de id, a seguir ajusta a transformação de id para a identidade. O id padrão é target. Também, se o id for uma câmera, chama (look-recenter World id). A função principal da chamada a (look-recenter) é colocar a câmera de forma que esteja apontando paralelamente ao eixo z na direção do centro do objeto mundo.

7.2.93 new-geometry

`(new-geometry name [GEOMETRY])`

Cria um novo geom com o nome fornecido (uma sequência de caracteres). *GEOMETRY* é uma sequência de caracteres fornecendo a especificação de um objeto geométrico OOGL. Se *GEOMETRY* for omitido, o novo objeto é fornecido como sendo um objeto geométrico vazio. Se um objeto com o nome fornecido já existir, ao novo objeto é fornecido um nome único.

7.2.94 new-reset

`(new-reset)`

Equivalente a `(progn (new-center ALLGEOMS)(new-center ALLCAMS))`.

7.2.95 NeXT

`(NeXT)` Retorna `t` se executando sobre um NeXT, `nil` se não estiver rodando sobre um NeXT. Uma relíquia de do ano da estação de trabalho NeXT.

7.2.96 normalization

`(normalization GEOM-ID {each|none|all|keep})`

Ajusta a situação atual da normalização de GEOM-ID.

`none` suprime toda normalização.

`each` normaliza a caixa associada ao objeto para ajustar-se dentro da esfera unitária, com o centro de sua caixa associada transladado para a origem. A caixa é alterada proprocionalmente de forma que sua diagonal maior, sqrt((xmax-xmin)^2 + (ymax-ymin)^2 + (zmax-zmin)^2), seja 2.

`all` assemelha-se a `each`, exceto quando um objeto está mudando (e.g. quando seu objeto geométrico está sendo modificado por um programa externo). Então, `each` precisamente ajusta a caixa associada em torno do objeto quando esse objeto muda e faz as normalizações de acordo com as mudanças, enquanto `all` normaliza a união de todas as variantes do objeto e normaliza conforme essa união.

`keep` mantém a transformação de normalização atual intocada quando o objeto muda. Essa intocabilidade pode ser útil para aplicar a normalização `each` ou `all` à primeira versão de um ojeto que está mudando de forma a trazer esse objeto para o campo de visão, a seguir alternar para `keep`.

7.2.97 not

`(not EXPR)`

Avalia `EXPR`; se `EXPR` retornar um valor não-`nil`, o valor de retorno do comando é nil, se `EXPR` retornar `nil`, o valor de retorno do comando é `t`.

7.2.98 or

`(or EXPR1 EXPR2)`

Avalia `EXPR1`; se `EXPR1` retorna não-`nil`, o valor de retorno do comando é o valor de `EXPR1`, se `EXPR1` retornar `nil`, a `EXPR2` é avaliada e o valor de `EXPR2` é retornado.

7.2.99 pick

`(pick COORDSYS GEOMID G V E F P VI EI FI)`

O comando pick é executado internamente em resposta a eventos de seleção (clique duplo no botão direito do mouse).

COORDSYS

= sistema de coordenadas no qual as coordenadas dos seguintes argumentos são especificados. Esse sistema de coordenadas pode ser:

`world` sistema de coordenadas do objeto mundo

`self` sistema de coordenadas do geom (*GEOMID*) selecionado

`primitive`
: sistema de coordenadas do primitivo atual dentro do geom selecionado onde a seleção ocorreu.

GEOMID = id do geom selecionado

G = ponto selecionado (intersecção atual do raio selecionado com o objeto)

V = vértice selecionado, se houver algum

E = aresta selecionada, se houver alguma

F = face selecionada

P = camainho para o primitivo selecionado [0 ou mais]

VI = índice do vértice selecionado no primitivo

EI = lista de índices de extremidades da aresta selecionada, se houver alguma

FI = índice da face selecionada

Módulos externos podem receber informações de eventos de seleção por meio do registro do interesse em chamadas a `pick` por meio do comando `interest`.

No contexto de visualizações de várias dimensões as coordenadas são atualmente pontos de várias dimensões. Eles correspondem a pontos tridimencionais da seleção relativa ao subespaço definido pela janela de visualização da câmera onde a seleção ocorreu. As coordenadas são então preenchidas com zeros e transformadas de volta para o sistema de coordenadas definido por *COORDSYS*.

7.2.100 pick-invisible

`(pick-invisible [yes|no])`
: Escolhe se seleções devem ser sensíveis a objetos cuja aparência faz com que fiquem invisíveis; o padrão é yes. Sem argumentos, retorna a situação/valor atual.

7.2.101 pickable

`(pickable GEOM-ID {yes|no})`
: Informa se GEOM-ID está incluído ou não no conjunto de objetos que podem ser retornados a partir do comando pick.

7.2.102 position

`(position objetoID outroID)`
: Ajusta a transformação de objetoID para aquele de outroID.

7.2.103 position-at

`(position-at objetoID outroID [center | origin])`
: Faz a translação de objetoID para o centro da caixa associada ou para a origem do sistema de coordenadas de outroID (translação paralela). O padrão é center.

7.2.104 position-toward

`(position-toward objetoID outroID [center | origin])`

Rotaciona objetoID de forma que o centro da caixa associada ou a origem do sistema de coordenadas do outroID localize-se sobre a parte positiva do eixo z do primeiro objeto. O padrão é o centro da caixa associada.

7.2.105 process-events

`(process-events)`

Devolve o controle ao laço do evento do Geomview, então continua avaliando o comando do script atual. Se o fluxo atual de entrada tiver sido colocado para cochilar por um dos comandos `(sleep-...)`, então o controle permanece no laço principal até que a "cochilada" do fluxo de entrada atual tenha terminado. Veja seção 7.2.131 [sleep-until], página 152. Veja seção 7.2.130 [sleep-for], página 152.

7.2.106 progn

`(progn DECLARAÇÃO [ ... ])`

evaluates each DECLARAÇÃO in order and retorna o value do last one. Use progn to group a collection of commands together, forcing them to be treated as a single command.

7.2.107 quit

`(quit)` `quit` is a synonym for seção 7.2.54 [exit], página 137.

7.2.108 quote

`(quote EXPR)`

retorna uma expressão simbólica em lisp *EXPR* sem avaliá-la. Note, todavia, que `quote` analisa *EXPR* como se ela pudesse ser avaliada. Veja seção 7.2.49 [eval], página 136.

7.2.109 rawevent

`(rawevent dev val x y t)`

Coloca o evento de linha especificado em uma fila de eventos. Os argumento especificam diretamente os membros da estrutura do evento usada internamente pelo geomview. Esse é o controlador de evento de mais baixo nível e não foi pensado para uso geral.

7.2.110 rawpick

`(rawpick CAMID X Y)`

Process a pick event in camera CAMID at location (X,Y) given in integer pixel coordenadas. This is a low-level procedure not intended for external use.

7.2.111 read

`(read {geometry|camera|aparência|image|ntransform|transform|command} {GEOMETRY ou CAMERA ou ...})`

Lê e interpreta o texto em ... como contendo o tipo de dado fornecido. Útil para definir objetos usando a sintaxe de referência OOGL, e.g.

```
(geometry  thing { INST  transform : T    geom : fred })
(read  geometry  { define fred
   QUAD
   1 0 0  0 1 0  0 0 1  1 0 0
})
(read  transform { define T <myfile})
```

Veja seção 4.1.9 [Referencias], página 68. Veja seção 7.2.59 [hdefine], página 137. Veja seção 7.2.60 [hdelete], página 138.

7.2.112 real-id

`(real-id ID)`

Retorna uma sequência de caracteres canonicamente identificando o ID fornecido, ou `nil` se o objeto não existe. Exemplos: (if (real-id fred) (delete fred)) apaga `fred` se esse arquivo existir mas retorna sem erro se esse arquivo não existir, e (if (= (real-id targetgeom) (real-id World)) () (delete targetgeom)) apaga `targetgeom` se esse arquivo for diferente do World.

7.2.113 redraw

`(redraw CAM-ID)`

Declara que o visual em CAM-ID deve ser redesenhado na próxima passagem através do ciclo principal ou da próxima vez que `draw` for chamada.

7.2.114 regtable

`(regtable)`

shows the registry table

7.2.115 rehash-emodule-path

`(rehash-emodule-path)`

Reconstrói o navegador de (módulo externo) aplicação através da leitura de todos os arquivos .geomview-* em todos os diretórios retornados pelo comando emodule-path. Primariamente pensado para uso interno; quaisquer aplicações definidas por comandos `(emodule-define ...)` fora dos arquivos .geomview-* sobre o comando emodule-path irão ser perdidos. Não ordena as entradas no navegador. Veja seção 7.2.45 [emodule-sort], página 135. Veja seção 7.2.40 [emodule-define], página 134.

7.2.116 replace-geometry

`(replace-geometry GEOM-ID PART-SPECIFICATION GEOMETRY)`

Substitui uma parte do geométrico for GEOM-ID.

7.2.117 rib-display

`(rib-display [frame|tiff] FILEPREFIX)`

Ajusta o display Renderman para janela temporária (janela que surge na tela) ou para um arquivo de disco no formato TIFF. FILEPREFIX é usado para construir nomes da forma `prefixNNNN.suffix`. (i.e. foo0000.rib) O número é incrementado a cada chamada a `rib-snapshot` e retorna a 0000 quando `rib-display` é chamado. Arquivos TIFF são fornecidos com o mesmo prefixo e número que o arquivo RIB correspondente (i.e. foo0004.rib gera foo0004.tiff). O padrão FILEPREFIX é `geom` e o formato padrão é TIFF. (Note que geomview apenas gera um arquivo RIB, o qual deve ser processado.)

7.2.118 rib-snapshot

`(rib-snapshot CAM-ID [nomearquivo])`

Escreve um instantâneo Renderman (no formato RIB) de CAM-ID gavando em <nomearquivo>. Se não for especificado um nomearquivo, veja seção 7.2.117 [rib-display], página 150 para uma explanação do nomearquivo usado.

7.2.119 scale

`(scale GEOM-ID FATOR [FATORY FATORZ])`

Ajusta proporcionalmente GEOM-ID, multiplicando seu tamanho por FATOR. Os fatores devem ser números positivos. Se FATORY e FATORZ estiverem presentes e forem não nulos, o objeto é ajustado proporcionalmente por FATOR na direção x, por FATORY na direção y, e por FATORZ na direção z. Se somente FATOR estiver presente, o objeto é justado proporcionalmente por FACTOR nas três direções x, y, e z. Ajustes proporcionais somente fazem sentido realmente no espaço de Euclides. Ajustes proporcionais usando o mouse nos outros espaço não é permitido; o comando scale pode ser emitido em outros espaços mas deve ser usado com cuidado pelo fato de poder fazer com que dados extendam-se para além dos limites do espaço.

7.2.120 scene

`(scene CAM-ID [GEOMETRY])`

Make CAM-ID look at GEOMETRY instead of at the universe.

7.2.121 set-clock

`(set-clock TEMPO)`

Ajusta o relógio para esse fluxo de comando para ler *TIME* (em segundos) a partir do momento que o comando é recebido. Veja seção 7.2.131 [sleep-until], página 152. Veja seção 7.2.25 [clock], página 132.

7.2.122 set-conformal-refine

`(set-conformal-refine CMX [N [SHOWEDGES]])`

Ajusta os parâmetros para o algorítmos de refinamento usado no desenho no modelo conformacional. CMX é o cosseno do maior ângulo e arestas podem entortar antes de serem refinadas. Seus valores devem estar entre -1 e 1; o

padrão é 0.95; deminuindo seu valor irá fazer causar menos refinamento. N é o númeo máximo de iterações de refinamento; o padrão é 6. SHOWEDGES, que pode ser `no` ou `yes`, determina se arestas interiores no refinamento são desenhadas.

7.2.123 set-emodule-path

`(set-emodule-path (PATH1 ... PATHN))`

Ajusta o caminho de busca por módulos externos. Os PATHi devem ser nomes de caminho de diretórios contendo, para cada módulo, o arquivo executável do módulo e um arquivo .geomview-<modulename> que contém um comando (emodule-define . . .) para aquele módulo. Esse comando implicitamente chama (rehash-emodule-path) para reconstruir o navegador de aplicações a partir do novo caminho ajustado. O nome especial de diretório `+` é substituído pelo caminho atual, então e.g. (set-emodule-path (meudir +)) anexa meudir no início do caminho.

7.2.124 set-load-path

`(set-load-path (PATH1 ... PATHN))`

Ajusta o caminho de busca para comando, objeto geométrico, arquivos, etc. Os PATHi são sequências de caracteres fornecendo os nomes de caminho de diretórios a serem buscados. O nome especial de diretórios `+` é substituído pelo caminho atual, então e.g. (set-load-path (meudir +)) anexa meudir no início do caminho.

7.2.125 set-motionscale

`(set-motionscale X)`

Ajusta o fator de proporcionalidade do movimento para X (o valor padrão é 0.5). Esses comandos ajustam proporcionalmente seu movimento por uma quantidade que depende da distância do quadro ao centro e do tamanho do quadro. Especificamente, ele ajusta de forma proporcional dist + scaleof(frame) * motionscale onde dist é a ditância a partir do centro ao quadro e motionscale é o fator de ajuste proporcional do movimento ajustado por essa função. Scaleof(frame) mede o tamanho do objeto frame.

7.2.126 setenv

`(setenv nome string)`

ajusta a variável de ambiente `nome` para o valor *string*; o nome é visível para o geomview (como em nomes de caminho contendo `$nome`) e para processos cria-se, e.g. módulos externos.

7.2.127 setq

`(setq SIM EXPR)`

Associa o símbolo *SIM* ao valor de *EXPR*. *SIM* deve ser um símbolo não qualificado, i.e. sem aspas duplas, e literal: `(setq "algumacoisa" outracoisa)` não irá funcionar. Da mesma forma `(setq (outracoisa STUFF) algumacoisa)`

também não irá funcionar, mesmo se `(outracoisa ...)` puder avaliar para um símbolo não qualificado: nomes de variáveis devem ser literais. Note que chamando `(setq SIM ...)` irá alterar o valor de *SIM* dentro do atual namespace: se *SIM*, e.g., estiver associado a uma variável local através de um lambda, uma expressão let ou uma expressão defun, então `(setq SIM ...)` irá mudar o valor da variável local, a associação global irá permanecer inalterada. NÃO é possível desassociar o símbolo. Todavia, posteriores invocações `(setq SIM ...)` irem reassociar *SIM* a outro valor e liberar o objeto do lisp previamente associado a *SIM*. Veja seção 7.2.68 [lambda], página 140. Veja seção 7.2.31 [defun], página 133. Veja seção 7.2.69 [let], página 140.

7.2.128 sgi

`(sgi)` Retorna `t` se estiver executando em uma máquina sgi, `nil` se nào estiver. Uma relíquia dos velhos anos nas estações de trabalho antigas.

7.2.129 shell

`(shell COMANDO-SHEL)`

Executa o solicitadoCOMANDO-SHELL UNIX ussndo /bin/sh. Geomview espera pela execução do comando e ficará em silêncio até que a execução se complete. Um sinônimo é `!`.

7.2.130 sleep-for

`(sleep-for TEMPO)`

Suspende comandos de leitura a partir desse fluxo durante TEMPO segundos. Comandos já lidos irão ser executados apesar disso; `sleep-for` dentro de `progn` não irá retardar a execução do restante do conteúdo do progn.

7.2.131 sleep-until

`(sleep-until TEMPO)`

Suspende comandos de leitura a partir desse fluxo até chegar em TEMPO (em segundos). Comandos já lidos irão ser executados apesar disso; `sleep-until` dentro de `progn` não irá retardar a execução do restante do conteúdo do progn. O tempo é medido conforme o relógio desse fluxo, como ajustado por `set-clock`; se nada for ajustado, o primeiro sleep-until ajusta o relógio do fluxo para 0 (de forma que inicialmente (sleep-until TEMPO) é o mesmo que (sleep-for TEMPO)). Retorna o número de segundos que faltam para chegar até TEMPO.

7.2.132 snapshot

`(snapshot CAM-ID NOMEDOARQUIVO [FORMAT [XSIZE [YSIZE]]])`

Grava um instantâneo de *CAM-ID* em *NOMEDOARQUIVO* (a sequência de caracteres). O argumento *FORMAT* é opcional; pode ser `"ppmscreen"` (o padrão), `"ps"`, `"ppm"`, `"sgi"` (em máquinas SGI), `"ppmosmesa"` (se o geomview for compilado com libOSMesa) ou `"ppmosglx"`. Um instantâneo `"ppmscreen"` é criado lendo a imagem diretamente da janela fornecida; a janela é colocada

em primeiro plano acima de outras janelas e redesenhada/atualizada primeiramente, então seu conteúdo é escrito como uma imagem no formato PPM. Um instantâneo "ppmosmesa" é desenhado pelo software do Mesa redenrizado em um espaço temporário de armazenamento na RAM. Um instantâneo "ppmosglx" é renderizado dentro de um espaço temporário de armazenamento de Pixmap da GLX, que também não aparece na tela mas pode ou não localiza-ser na RAM de vídeo. Renderização pode ou não ser acelerada. O problema com instantâneos em tela é que a janela deve ser mapeada e não obscurecida por outras janelas. De forma que instantâneos em tela não irão funcionar em segundo plano, ou quando uma proteção de tela estiver ativa. Com "ps", cria-se uma imagem Postscript representado a visão daquela janela; remoção de superfície escondida pode ser incorreta. Com "ppm", cria-se uma imagem no formato PPM produzido pelo software rederizador interno do geomview; essa imagem pode ter tamanho arbitrário. Se o argumento *NOMEDOARQUIVO* iniciar-se com "|", é interpretado como um comando `/bin/sh` para o qual os dados PPM ou PS devam ser direcionados a um pipe. Valores opcionais *XSIZE* e *YSIZE* são relevantes somente para formatos "ppm", e renderizados para uma janela daquele tamanho (ou ajustadas proporcionalment para aquele tamanho, com aspecto fixo, se somente *XSIZE* for fornecido)

7.2.133 soft-shader

`(soft-shader CAM-ID {on|off|toggle})`

Seleciona se para usar o sombreamento de software ou de hardware naquela câmera.

7.2.134 space

`(space {euclidean|hyperbolic|spherical})`

Escolhe o espaço associado ao mundo.

7.2.135 stereowin

`(stereowin CAM-ID [no|horizontal|vertical|colored] [gapsize])`

Configura CAM-ID como uma janela stereo. no: janela inteira é um painel simples, stereo desabilitado
horizontal: quebra esquerda/direita: esquerda é olho stereo #0, direita é #1.
vertical: quebra topo/base: base é olho#0, topo é #1.
colored: sobreposição de painéis, vermelho é olho#0 stereo, ciano é #1.

Uma brecha do tamanho de `gapsize` pixels é deixada entre subjanelas; se omitido, subjanelas são vizinhas. Se ambos layout e gapsize forem omitidos, e.g. (stereowin CAM-ID), retorna ajustes atuais como uma lista de comandos `(stereowin ...)`. Esse comando não ajusta projeção stereo; use `merge camera` ou `camera` para ajustar as transformações stereyes, e `merge window` ou `window` para ajustar a razão de aspecto de pixel & posição da janela se necessário.

7.2.136 time-interests

`(time-interests deltatempo inicial prefixo [sufixo])`

Indica que todas as mensagens de interesse ou relacionadas, quando separadas por pelo menos `deltatempo` segundos de tempo real, deve ser precedida pela sequência de caracteres `prefixo` e seguida pelo `sufixo`; a primeira mensagem é precedida por `inicial`. Todos os três são sequências de caracteres de formato da printf, cujo argumento é o tempo atual de relógio (em segundos) naquele fluxo. Um `deltatempo` de zero mostra o rótulo de tempo de todas as mensagens. Uso típico:
(time-interests .1 `(set-clock %g) (sleep-until %g)`) ou
(time-interests .1 `(set-clock %g)` "(sleep-until %g) (progn (set-clock %g)" ")") ou
(time-interests .1 "(set-clock %g)" "(if (> 0 (sleep-until %g)) (" "))".

7.2.137 transform

`(transform objetoID centerID frameID [rotate|translate|translate-scaled|scale] x y z [bbox-center|origin] [dt [smooth]])`

Aplica um movimento (rotação, translação, ajuste proporcional) ao objeto `objetoID`; isto é, constrói e concatena uma matriz de transformação com transformações de objetoID. Os 3 IDs envolvidos são o objeto que se move, o centro do movimento, e o quadro em relação ao qual o movimento vai ser aplicado. O centro é facilmente entendido para rotações: se centerID for o mesmo que objetoID então o objeto irá girar em torno de seu próprio eixo; de outra forma o objeto em movimento irá orbitar o centro do objeto. Existe a palavra chave especial `bbox-center` que pode ser usada para `centerID`. Como resultado o movimento irá ser relativamente ao centro da caixa associada de `objetoID`. Normalmente frameID, em cujo sistema de coordenadas os movimentos (de mouse) são interpretados, é `focus`, a câmera atual. Translações podem ser ajustadas proporcionalmente às distâncias entre o alvo e o centro. Suporte ao espaço esférico e ao espaço hiperbólico bem como ao espaço eucidiano é embutido no geomview: use o comando `space` para mudar de um espaço para outro. Como tipo `rotate` x, y, e z são números no formato de ponto flutuante especificando ângulos em RADIANOS. Para tipos `translate` e `translate-scaled` x, y, e z são números no formato de ponto flutuante specificando distâncias no sistema de coordenadas do objeto de centro.

O campo opcional `dt` permite uma forma simples de animação; se estiver presente, o objeto move-se por aquela quantidade durante aproximadamente `dt` segundos, parando em seguinda. Se presente e seguido pela palavra chave `smooth`, o movimento é animado com uma função 3t^2-2t^3, iniciando e parando de modo uniforme. Se omitido, o movimento é aplicado imediatamente.

7.2.138 transform-incr

`(transform-incr objetoID centerID frameID [rotate|translate|translate-scaled|scale] x y z [origin|bbox-center] [dt [smooth]])`

Aplica movimento contínuo: constrói uma matriz de transformação e concatena essa matriz à transformação atual de objetoID a cada atualização de tela (ajusta a transformação incremental de objetoID). A mesma sintaxe da transformação. Se o argumento opcional `dt` estiver presente, o objeto é movido a cada passo de tempo de forma que seu movimento médio iguala-se a uma instância de movimento por `dt` segundos. E.g. (transform-incr World World World rotate 6.28318 0 0 10.0) rotaciona o mundo em torno de seu eixo X em uma volta (2pi radianos) a cada 10 segundos.

7.2.139 transform-set

`(transform-set objetoID centerID frameID [rotate|translate|translate-scaled|scale] x y z [origin|bbox-center])`

Ajusta a transformação de objetoID's para a transformação construída. A mesma sintaxe que transform.

7.2.140 truncate

`(truncate EXPR)`

Arredonda *EXPR* desprezando as casas decimais.

7.2.141 ui-cam-focus

`(ui-cam-focus [focus-change|mouse-cross])`

Ajusta a política de focalização para as janelas de câmera. O padrão é `mouse-cross`: a câmera é tornada ativa (por meio de eventos interativos de mouse) quando o cursor do mouse cruza a janela. Devido a esse comportamento pode tornar-se complicado ativar uma câmera específica (no contexto de múltiplas janelas de câmera) existe também a opção para somente mudar o foco da câmera quando o gerenciador de janelas decidir dar a essa câmera o foco para eventos de entrada. Dessa forma, após especificar `focus-change` essa mudança depende da configuração de mudança de foco de seu gerenciador de janelas quando uma câmera tornar-se a câmera ativa por meio de interação com o mouse. Para mudar esse comportamento permanentemente você poderá vir a, e.g., colocar a seguinte linha no seu arquivo '`${HOME}/.geomview`' (Veja Capítulo 5 [Customizacao], página 103):

```
(progn
  (ui-cam-focus focus-change)
  ... # other stuff
)
```

7.2.142 ui-center

`(ui-center ID)`

Ajusta o centro para movimentos controlados pela interface de usuário (i.e. mouse) para o objeto ID.

7.2.143 ui-center-origin

`(ui-center-origin [origin|bbox-center])`

Ajusta a origem do systema de coordenadas do objeto `CENTER` para movimentos controlados pela interface de usuário (i.e. mouse). A palavra chave `origin` significa a origem do sistema de coordenadas do objeto atualmente selecionado, enquanto `bbox-center` significa usar o centro da caixa associada do objeto atual. A palavra chave `bbox-center` não faz sentido se o objeto geométrico não for do espaço de Euclides. Usando ou `bbox-center` ou `origin` não faz diferença se o objeto do centro não for um *geometry*, e.g. se for uma câmera. O mesmo ocorre se o mundo for o objeto atualmente selecionado.

7.2.144 ui-emotion-program

`(ui-emotion-program PROGRAM)`

É um comando obsoleto. Use seu novo equivalente seção 7.2.40 [emodule-define], página 134 ao invés dele.

7.2.145 ui-emotion-run

`(ui-emotion-run EMODULE)`

É um comando obsoleto. Use seu novo equivalente seção 7.2.46 [emodule-start], página 135 ao invés dele.

7.2.146 ui-freeze

`(ui-freeze {on|off})`

Habilita painés de atualização de usuário. Off por padrão.

7.2.147 ui-html-browser

`(ui-html-browser BROWSER)`

Use *BROWSER* para visualizar a vesão *HTML*n do manual quando o item de menu '`Manual (HTML)`' for selecionado no menu '`Help`'. Se o comando `(ui-html-browser ...)` nunca for executado, então o padrão é usar o navegador armazenado na variável de ambiente `WEBBROWSER`. Se a variável de ambiente estiver vazia então o padrão é dependente da forma como a cópia local do geomview foi compilada.

7.2.148 ui-motion

`(ui-motion {inertia|constrain|own-coordinates} {on|off})`

Habilita ou desabilita certas propriedades de movimento controlado por mouse. O propósito dess comando é acessar aos respectivos comutadores do painél principal (*Main*) no menu de movimento (*Motion*) através de comandos GCL. Veja seção 3.5 [Movimentos do Mouse], página 39.

`inertia` Normalmente, movimentos de objeto possuem inércia: se o mouse estiver em movimento quando o botão for liberado, o objeto selecionado continua a se mover. Quando a opção `inertia` estiver em off, objetos cessam o movimento no mesmo instânte em que o botão do mouse for liberado.

`constrain`
Algumas vezes é conveniente mover objetos em uma direção paralela a um eixo coordenado: exatamente na horizontal ou na vertical. Digitando o comando `(ui-motion constrain on)` muda a interpretação dos movimentos de mouse para permitir isso; arrastos de mouse aproximadamente na horizontal ou aproximadamente na vertical tornam-se exatamente em movimento horizontal ou vertical. Note que o movimento é ainda em torno dos eixos X ou Y da câmera na qual você move o mouse, não necessáriamente no sistema de coordenadas do objeto.

`own-coordinates`
Algumas vezes é conveniente mover objetos em relação ao sistema de coordenadas onde eles foram definidos, ao invés de em relação a alguma visão de câmera. Quando `(ui-motion own-coordinates on)` tiver sido chamado, todos os movimentos são interpretados daquela forma: arrastando o mouse rightward no modo de translação move o objeto na direção de seu próprio eixo `+`*X*, e assim por diante. Pode ser especialmente útil juntamente com o comando `(ui-motion constrain on)`.

7.2.149 ui-panel

`(ui-panel NOMEPAINEL {on|off} [JANELA])`
Mostrar ou não mostrar o dado painel de interface de usuário. A caixa alta ou baixa é ignorada em nomes de painél. Atualmente *NOMEPAINEL* pode ser:

geomview	painel principal
tools	controles de movimento
appearance	controles de aparência
cameras	controles de câmera
lighting	controles de iluminação
obscure	controles obscuros (parece não existir mais)
materials	controles de propriedades de materiais
command	caixa de entrada de comandos
credits	créditos do geomview

Por padrão, os painéis `geomview` e `tools` aparecem quando o geomview inicia. Se a janela opcional for fornecida, uma cláusula `position` (e.g. (ui-panel obscure on `{` position xmin xmax ymin ymax `}`) ajusta a posição padrão do painel.

(Somente os valores de xmin e de ymin são atualmente usados.) Uma janela presente mas vazia, e.g. `(ui-panel obscure on {)}` faz com que o posicionamento seja interativo.

7.2.150 ui-pdf-viewer

`(ui-pdf-viewer VIEWER)`

Usa o executável armazenado na variável *VIEWER* para visualizar a versão em *PDF* do manual quando o item de menu '`Manual (PDF)`' estiver selecionado no menu '`Help`'. Se o comando `(ui-pdf-viewer ...)` nunca tiver sido executado, então o padrão é usar o navegador armazenado na variável de ambiente `PDFVIEWER`. se a variável de ambiente estiver vazia então o padrão é determinado pela forma como a cópia local do geomview foi compilada.

7.2.151 ui-target

`(ui-target ID [yes|no])`

Set the target of user actions (the selected line do objeto alvo browser) to ID. The segundo argumento especifica se to make ID the current objeto regardless of its type. If `no`, then ID becomes the current objeto of its type (geom ou camera). The default is `yes`. This command may result in a change of modos de movimento based on target choice.

7.2.152 uninterest

`(uninterest (COMANDO [args]))`

Desfaz o efeito de um comando `interest`. (COMANDO [args]) deve ser idêntico àquele usado no comando seção 7.2.67 [interest], página 139.

7.2.153 update

`(update [timestep_in_seconds])`

Aplica cada movimento incremental uma única vez. Usa timestep se esse tempo de passo estiver presente e não for nulo; de outra forma os movimentos são proporcionais ao intervalo de tempo decorido real.

7.2.154 update-draw

`(update-draw CAM-ID [timestep_in_seconds])`

Aplica cada movimento incremental uma única vez e então desenha CAM-ID. Aplica segundos de `timestep` no valor do movimento, ou usa tempo decorrido real se `timestep` estiver ausente ou for nulo.

7.2.155 while

`(while TESTE CORPO)`

Iteração: *avalia TESTE, se encontrar alguma coisa, avalia CORPO.*

7.2.156 window

`(window CAM-ID JANELA)`

Especifica atributos para a janela de CAM-ID, e.g. seu tamanho ou sua posição inicial, na sintaxe de janela da OOGL. A identificação especial de câmera `default` especifica propiedades de janelas futuras (criadas por seção 7.2.19 [camera], página 131 ou por seção 7.2.91 [new-camera], página 145).

7.2.157 winenter

`(winenter CAM-ID)`

Diz ao geomview que o cursor do mouse está na janela de CAM-ID. Essa função tem propósitos de desenvolvimento e não foi pensada para uso geral.

7.2.158 write

`(write {command|geometry|camera|transform|ntransform|window|bbox} NOMEDOARQUIVO [ID|(ID ...)] [self|world|universe|outroID])`

escreve uma descrição de ID no formato especificado para NOMEDOARQUIVO. O último parâmetro escolhe o sistema de coordenadas para geometry & transform: self: apenas o objeto, nenhuma transformação ou aparência (objeto geométrico somente) world: o objeto como posicionado dentro do Mundo. universe: posição do objeto em coordenadas universais; inclue Worldtransform outroID: o objeto transformedo para o sistema de coordenadas de outroID.

Um nome de arquivo `-` é um caso especial: dados são escritos para o fluxo a partir do qual o comando 'write' foi lido. Para módulos externos, os dados são enviados para a entrada padrão do módulo. Para comandos não lidos a partir de um programa externo, `-` significa a saída padrão do geomview (veja seção 7.2.26 [command], página 132).

O ID pode ou se um simples id ou uma lista entre parêntesis de ids, como `g0` ou `(g2 g1 dodec.off)`.

7.2.159 write-comments

`(write-comments NOMEDOARQUIVO GEOMID PICKPATH)`

escreve objetos COMMENT da OOGL na hierarquia do GEOMID no nível do caminho selecionado para NOMEDOARQUIVO. Especificamente, COMMENTS no nível (a b c . . . f g) irão coincidir caminhos selecionados da forma (a b c . . . f *) onde * inclui qualquer valor de g, e também quaisquer valores de possíveis índices adicionais h,i,j, etc. O caminho selecionado (retornado pelo comando `pick`) é uma lista de contadores inteiros especificando uma subparte de um objeto OOGL hierárquico. Descendo dentro de um objeto complexo (LIST ou INST) adiciona um novo inteiro ao caminho. Traversal de objetos simples incrementa o contador no nível atual. Individual COMMENTS são colocados entre chaves, e A completa sequência de caracteres de zero, um, ou mais COMMENTS (escritos na ordem em que forem encontrados durante a hierarquia transversal) é colocada entre parêntesis.

Note que dados arbitrários podem somente ser informados através das bibliotecas OOGL como objetos OOGL COMMENT completamente fugidios, que

podem serem anexados a outros objetos OOGL através do tipo LIST como descrito acima. Comentários comuns em arquivos OOGL (i.e. tudo após '#' sobre uma linha) são ignorados quando o arquivo é carregado e não pode ser retornado.

7.2.160 write-handle

`(write-handle PREFIX NOMEDOARQUIVO HANDLE)`

Escreve o objeto com o respecitivo identificador para *NOMEDOARQUIVO*. Essa função foi pensada para depuração interna somente.

7.2.161 write-sexpr

`(write-sexpr NOMEDOARQUIVO LISPOBJECT)`

Escreve o dado LISPOBJECT para NOMEDOARQUIVO. Essa função é pensada para depuração inerna somente.

7.2.162 xform

`(xform ID TRANSFORM)`

Concatena TRANSFORM à transoformação atual do objeto (aplica TRANSFORM ao objeto ID).

7.2.163 xform-incr

`(xform-incr ID TRANSFORM)`

Aplica movimento contínuo: concatena TRANSFORM à atual transformação do objeto sempre atualizando a tela (ajusta ID da transformação incremental de objeto para TRANSFORM).

7.2.164 xform-set

`(xform-set ID TRANSFORM)`

Sobrescreve a atual transforma;cão de objeto com TRANSFORM (ajusta a transformação do objeto ID para TRANSFORM).

7.2.165 zoom

`(zoom CAM-ID FATOR)`

Zoom CAM-ID, multiplicando seu campo de visão por FATOR. FATOR deve ser um número positivo.

8 Geometrias Não-Euclidianas

Geomview suporta objetos geométricos hiperbólicos e esféricos bem como objetos geométricos de Euclides. Os três botões parte inferior do painel *Main* são para ajustar o tipo do boeto geométrico geométrico atual.

Em cada uma das três geometrias, três odelos são suportados: *Virtual*, *Projective*, e *Conformal*. Você pode mudar o modelo atual com o navegador *Model* no painel de *Camera*. Cada câmera do Geomview tem seu próprio ajuste de modelo.

O modelo padrão é que todos os três espaços sejam *Virtual*. Isso corresponde à câmera estando no mesmo espaco que, e movendo-se sob o mesmo conjunto de transformações que, o objeto geométrico em si mesmo.

No espaco de Euclides *Virtual* é o modelo mais útil. Os outros modelos foram implementados para espaços hiperbólicos e esféricos e ocorrem apenas para trabalhar no espaço de Euclides também: *Projective* é o mesmo que *Virtual* mas o padrão mostra a esfera unitária, e *Conformal* mostra tudo invertido na esfera unitária.

No espaço hiperbólico, o modelo *Projective* ajusta as informações fornecidas a uma visão do modelo de bola projetiva do espaço tridimensional hiperbólico embutido no espaço de Euclides. A câmera está inicialmente fora da bola unitária. Nesse modelo, a câmera move-se através de movimentos de Euclides e o objeto geométrico move-se através de movimentos hiperbólicos. O modelo *Conformal* é parecido mas mostra o modelo de bola conformal.

No espaço esférico, o modelo *Projective* fornece uma visão de metade da esfera tridimensional embutida no espaço de Euclides tridimensional. Movimentos esféricos fornecem sobem para transformaçes projetivas no modelo *Projective*, e para transformações de Möbius no modelo *Conformal*. Em ambos esses modelos a câmera move-se através de movimentos de Euclides.

Nos modelos *Projective* e *Conformal*, a esfera unitária é desenhada por parão. Você pode desabilitar e habilitar a exibição da esfera unitária usando o botão *Draw Sphere* no painel de *Camera*. No modelo *Conformal*, polígonos e arestas são sibdivididos se for necessário para fazerem-nos parecerem curvados. Os parâmetros que determinam essa subdivisão podem ser ajustados com o comando GCL `set-conformal-refine`. Veja seção 7.2.122 [set-conformal-refine], página 150.

Existem muitos exemplos de objetos no espaço hiperbólico no subdiretório '`data/geom/hyperbolic`' do diretórios Geomview. Da mesma forma, o subdiretório '`data/geom/spherical`' contém muitos exemplos de objetos no espaço esférico.

9 Gráficos do Mathematica no Geomview ou no RenderMan

Geomview vem com alguns pacotes do Mathematica que permitem a você usar Geomview para mostrar gráficos do Mathematica. Mathematica é um sistema de software matemático comercial disponível a partir de Wolfram Research, Inc.

Existem duas formas de mostrar um gráfico do Mathematica no Geomview.

1. Use Mathematica para escrever um objeto gráfico em um arquivo no formato OOGL ou no formato RIB.
2. Use Geomview como o visualizador padrão para todos as saídas de gráficos 3D no Mathematica.

Você também pode usar esses pacotes para gravar gráficos do Mathematica no formato RenderMan (RIB).

Uma vez que o formato dos objetos gráficos do Mathematica é diferente dos formatos OOGL, ambos esses métodos envolvem traduzir gráficos do Mathematica para o formato OOGL. Geomview é distribuído com um pacote do Mathematica que faz essa tradução. Anes de usar qualquer dos dois métodos acima você deve instalar esse pacote.

9.1 Usando Mathematica para gerar arquivos OOGL

O pacote '`OOGL.m`' permite ao Mathematica escrever objetos gráficos no formato OOGL. Para usá-lo, digite o comando `<< OOGL.m` para Mathematica carregar o pacote. O comando `WriteOOGL[`*arquivo*`,`*nomegrafico*`]` escreve uma descrição OOGL do bojeto gráfico 3D *nomegrafico* para o arquivo chamado *arquivo*.

Esse pacote também fornece o comando `Geomview` que envia um objeto gráfico 3D para o Geomview. A primeira vez que você usa o comando ele carrega uma cópia do Geomview. A partir da segunda chamada em diante ele envia os gráficos para o mesmo Geomview. Existem dois caminhos para usar esse comando `Geomview`.

`Geomview[`*nomegrafico*`]`
: Envia o objeto gráfico 3D *nomegrafico* para o Geomview como um geom chamado `Mathematica`. Subsequentes usos de `Geomview[`*nomegrafico*`]` substitui o objeto `Mathematica` no Geomview com o novo *nomegrafico*.

`Geomview[`*nome*`,`*nomegrafico*`]`
: Envia o objeto gráfico 3D *nomegrafico* para o Geomview como um geom chamado *nome*. Você pode usar múltiplas chamadas dessa forma com diferentes nomes para fazer com que o Geomview mostre muitos objetos Mathematica de uma só vez e permita o controle deles de forma independente.

```
% math
Mathematica 2.0 for SGI Iris
Copyright 1988-91 Wolfram Research, Inc.
 -- GL graphics initialized --

In[1] := <<OOGL.m

In[2] := Plot3D[Sin[x + Sin[y]], {x,-2,2},{y,-2,2}]
```

```
Out[2] := -Graphics3D-
```

O exemplo acima mostra gráfico no caminho usual do Mathematica aqui.

```
In[3] := WriteOOGL["math.oogl", %2]

Out[3] := -Graphics3D-
```

O exemplo acima não mostra nada novo mas escreve o arquivo '`math.oogl`'. Você pode agora carregar o arquivo '`math.oogl`' dentro do Geomview a partir de qualquer computador. Alternativamente, você pode usar o comando `Geomview` para iniciar uma cópia do Geomview a partir de dentro do Mathematica.

```
In[5] := Geomview[%2]

Out[5] := -Graphics3D-
```

9.2 Usando Geomview como Visualizador Padrão 3D do Mathematica

O pacote '`Geomview.m`' faz com que o Geomview seja o programa visualizador padrão para gráficos 3D no Mathematica. Para carregar o pacote '`Geomview.m`', digite o comando `<< Geomview.m` para o Mathematica. De agora em diante, sempre que você mostrar gráficos 3D com o comando `Plot3D` ou `Show`, Mathematica irá enviar o gréfico para o Geomview.

Carregando '`Geomview.m`' implicitamene carrega '`OOGL.m`' também, de forma que você pode usar o `Geomview` e `WriteOOGL` como descrito acima após carregar '`Geomview.m`'. Você não tem que carregar separadamente o pacote '`OOGL.m`'.

```
% math
Mathematica 2.0 for SGI Iris
Copyright 1988-91 Wolfram Research, Inc.
 -- GL graphics initialized --

In[1] := <<Geomview.m

In[2] := Plot3D[x^2 + y^2, {x, -2, 2}, {y, -2, 2}]

Out[2] := -SurfaceGraphics-
```

O comando acima chama o geomivew e carrega o objeto gráfico dentro dele.

```
In[3] := Plot3D[{x*y + 6, RGBColor[0,x,y]}, {x,0,1}, {y,0,1}]

Out[3] := -SurfaceGraphics-
```

O comando imediatamente acima substitui o objeto anterior do Geomview pelo novo objeto.

```
In[4] := Geomview[{%2,%3}]

Out[4] := {-SurfaceGraphics-, -SurfaceGraphics-}
```

Mostra ambos os objetos de uma só vez. Você também pode ter mais de um objeto do Mathematica de uma só vez mostrados no Geomview, e ter controle separado deles, usando o comando do `Geomview` com um nome, Veja seção 9.1 [OOGL.m], página 162.

```
In[5] := Graphics3D[ {RGBColor[1,0,0], Line[{ {2,2,2},{1,1,1} }] }]

Out[5] := -Graphics3D-

In[6] := Geomview["minhalinha", %5]
```

Isso adiciona a `Linha` especificada em `In[5]` ao visualizador Geomview já existente. A `Linha` especificada em `In[5]` pode ser controlada independentemente do objeto do "Mathematica", que atualmente é uma lista de dois gráficos.

```
In[7] := <<GL.m
```

Se você estiver em um SGI, chamando `GL.m` retorna o Mathematica a seu usual visualizador de gráficos 3D. O seguinte plot irá aparecer em uma janela estática normal do Mathematica.

```
In[8] := ParametricPlot3D[{Sin[x],Sin[y],Sin[x]*Cos[y]}, {x,0,Pi},{y,0,Pi}]

Out[8] := -Graphics3D-
```

Podemos retornar ao gráfico do Geomview a qualquer tempo chamando '`Geomview.m`'.

```
In[9] := <<Geomview.m

In[10] := Show[%8]

Out[10] := -Graphics3D-

In[11] := ParametricPlot3D[
   {(2*(Cos[u] + u*Sin[u])*Sin[v])/(1 + u^2*Sin[v]^2),
   (2*(Sin[u] - u*Cos[u])*Sin[v])/(1 + u^2*Sin[v]^2),
   Log[Tan[v/2]] + (2*Cos[v])/(1 + u^2*Sin[v]^2)},
  {u,-4,4},{v,.01,Pi-.01}]

Out[11] := -Graphics3D-
```

O último gráfico é de uma superfície de Kuen, uma superfície de curvatura negativa constante. Parametrização retirada do livro texto de Alfred Gray *Modern Differential Geometry of Curves and Surfaces*.

9.3 Usando Mathematica para gerar arquivos do RenderMan

Adicionalmente aos comandos `WriteOOGL` e `Geomview` descritos acima, o pacote '`OOGL.m`' também define o comando `WriteRIB` que escreve um objeto gráfico 3D para um arquivo RenderMan RIB: `WriteRIB[`*`arquivo`*`,` *`nomegrafico`*`]` escreve *nomegrafico* para o arquivo chamado *aquivo*. RenderMan é um sistema de renderização comercial disponível a partir da Pixar, Inc., que pode produzir imagens de extremamente alta qualidade.

```
In[1] := <<OOGL.m

In[2] := <<Graphics/Polyhedra.m

In[3] := Graphics3D[Cube[]]
```

```
Out[3] := -Graphics3D-

In[4] := WriteRIB["cube.rib", %3]

Out[4] := -Graphics3D-
```

O comando acima gera o arquivo '`math.ri`'b. O arquivo '`math.ri`'b é um arquivo RIB pronto para renderização do objeto geométrico fornecido, usando uma posição de câmera padrão, iluminação, e o sombreador "plastic". Em uma janela de shell, digite `render cube.ri`b paa gerar o arquivo de imagem '`mma.tiff`'. Certamente, você precisa ter o RenderMan instalado para o comando funcionar. Um atalho para renderizar a partir de dentro do Mathematica é `WriteRIB["!render", algumacoisa]`.

`WriteRI`b trabalha primeiramente convertendo o objeto gráfico do Mathematicao para o formato OOGL usando `WriteOOGL` e então chamando um o programa externo '`oogl2ri`'b para converter o formato OOGL para o formato RIB. O programa oogl2rib recebe muitas opções que você pode especificar em uma sequência de caracteres como um opcional terceiro argumento a `WriteRI`b. A sequência de caracteres padrão de opção é `" -n mma.tiff "`, que indica que o arquivo RIB deve gerar um arquivo renderizado do tipo TIFF chamado '`mma.tiff`'. Uma opção particularmente útil é `-g`, que informa ao oogl2rib para converter somente o objeto geométrico em um fragmento RIB. Você pode nserir aquele fragmento dentro de um arquivo RIB completo de sua própria criação com posições de câmera e sombreamento de sua escolha, para aproveitar o poder completo do RenderMan.

O uso completo do oogl2rib é:

`oogl2rib [-n` *nome*`] [-B` *r*`,`*g*`,b] [-w` *width*`] [-h` *height*`] [-fgb] [`*arquivoentrada*`] [`*arquivosaida*`]`

Por padrão lê de stdin e escreve em stdout. Ou *arquivoentrada* ou *arquivosaida* pode ser '`-`', que significa usar stdin/stdout. As opções são:

`-n` *nome* — Use *nome* para o nome do arquivo TIFF renderizado (o padrão é "geom.tiff") ou para a janela temporária (padrão "geom.rib").

`-B` *r*`,`*g*`,b` — Use cor de fundo (*r*,*g*,b). Cada intervalo de componente vai de 0 a 1. Padrão: nenhum.

`-w` *largura* `-h` *altura* — O quadro Renderizado irá ter *largura* por *altura* pixels.

`-f` — O arquivo RIB renderiza para uma janela temporária na tela ao invés de para o arquivo TIFF.

`-g` — Saída somente do objeto geométrico no formato RIB.

`-b` — Saída somente de um objeto clip Quick Renderman. Ignora -nBwhf.

9.4 Usando Geomview e Mathematica em Computadores Separados

É possível usar o geomview para mostrar gráficos gerados por um Mathematica executado em um computador diferente. Se você deseja usar Mathematica em um computador que

não na mesma rede que o seu computador Geomview, você pode escrever saídas em arquivos *chunk* no Mathematica que você transfere para o computdor do Geomview e então traduz para o formato OOGL para mostrar no Geomview.

9.4.1 Usando um host Geomview na mesma rede

O comando `Geomview` procura elas variáveis de ambiente `DISPLAY` ou `REMOTEHOST` para tentar determinar se você está logado a partir de outro computador. Se qualquer uma dessas variáveis de ambiente indicar que você está, `Geomview` irá tentar executar Geomview no computador onde você se encontra. Com o objeto de fazer funcionar, sua rede deve ser configurada de forma que o computador com o Mathematica possa fazer um `rsh` com sucesso para o computador Geomview sem fornecer uma senha.

Você pode também explicitament ajustar a opção `DisplayHost` para o comando `Geomview` para uma sequência de caracteres que é o hostname desejado, por exemplo:

```
In[1] := << OOGL.m

In[2] := Plot3D[Sin[x + Sin[y]], {x,-2,2},{y,-2,2}]

Out[2] := -Graphics3D-

In[3] := Geomview[%3, DisplayHost->"riemann"]
```

O comando acima mostra o gráfico `%3` no computador remoto chamado `riemann`.

`Geomview` reconhece a sequência de caracteres `"local"` como um valor para `$DisplayHost`; faz com que os gráficos sejam mostrados na máquina local.

Além do nome da máquina que você pode deseja executar Geomview , `Geomview` precisa saber o tipo daquela máquina (o ajuste da variável CPU que corresponde à máquina. Por padrão, `Geomview` assume que é do mesmo tipo de computador que você está executando o Mathematica. A opção `MachType` permite a você explicitamente especificar o tipo do computador `DisplayHost`; deve ser um entre as sequências de caracteres `"sgi"` ou `"next"` ou `"x11"`.

Você pode usar `SetOptions` para mudar o `DisplayHost` padrão e a `MachType` padrão. Por exemplo,

```
In[4] := SetOptions[Geomview, DisplayHost->"riemann", MachType->"sgi"]
```

faz com que `Geomview` execute o Geomview em uma estação de trabalho SGI chamada `riemann`.

9.4.2 Transportando arquivos do Mathematica para o Geomview Manualmente

A função auxiliar `WriteChunk` é para aqueles que somente usar o Mathematica em um computador que não tem Geomview instalado. `WriteChunk[`*`nomearquivo`*`, `*`nomegrafico`*`]` gera um arquivo chamado *nomearquivo* que contém o objeto gráfico *nomegrafico* no formato aceito pelo '`math2oogl`'.

Você pode transferir aquele arquivo para um computador que tem Geomview instalado nele e então usar os programas '`math2oogl`', '`oogl2ri`'b, e '`geomview`' diretamente a partir do shell. Esses programas são distribuídos no subdiretório '`bin/<CPU>`' do diretório do

Geomview, e podem ter sido instalados de forma que eles estejam no seu caminho de busca `path`.

```
In[1]:= <<OOGL.m

In[2]:= Plot3D[ Sin[x + Sin[y]], {x,-2,2}, {y,-2,2} ]

Out[2]= -SurfaceGraphics-

In[3]:= WriteChunk["minhaparte",%2]
```

Escreve o arquivo '`minhaparte`' que contém uma descrição do objeto gráfico. Você pode então transferir esse arquivo para um sistema com Geomview e digitar

```
math2oogl < minhaparte > mma.oogl
```

para converter o minhaparte no arquivo OOGL '`mma.oogl`' que você pode então visualizar usando o Geomview. É equivalente ao comando `WriteOOGL`.

Para um resultado equivalente aos comando `Geomview` ou `Show` do Mathemática, digite

```
math2oogl -togeomview Mathematica geomview < minhaparte
```

O comando `WriteRIb` pode ser emulado a partir do shell como

```
math2oogl < minhaparte | oogl2rib -n mma.tiff
```

9.5 Detalhes do Pacote Mathematica->Geomview

O pacote '`OOGL.m`' usa o programa externo '`math2oogl`' para converter objetos `Graphics3D` para o formato OOGL, pelo fato de um programa extero compilado etar apto a fazer essa conversão muitas vezes mais rápido que o Mathematica.

O conversor irá muitas vezes controlar objetos SurfaceGraphics coloridos corretamente os quais o Mathematica não controla corretamente, o que significa que Geomview[objeto] algumas vezes trabalha onde Show[objeto] mostra erros.

O conversor suporta as primitivas gráficas `Polygon`, `Line`, e `Point`, as diretivas `RGBColor` `Graphics3D`, e objetos `SurfaceGraphics` com ou sem diretivas `RGBColor`, e grande quantidade de quaisquer combinações destes. Silenciosamente igonora todas as outras diretivas.

A conversão do Mathematica para o RenderMan é atualmente um processo em dois passos: Mathematica->OOGL (math2oogl), e OOGL->RenderMan (oogl2rib).

Nos comandos `WriteOOGL` e `WriteRIb`, nome de arquivo pode ou ser uma sequência de caracteres contendo um nome de arquivo, um objeto `OutputStream`, ou uma sequência de caracteres iniciando com uma `!` para enviar a saída para um comando. Objeto pode ser um objeto `Graphics3D`, um objeto `SurfaceGraphics`, ou uma lista desse.

O pacote trabalha melhor com o Mathematica 2.0 ou mais recente. Com a versão 1.2, o visualizador Geomview funciona somente na máquina local.

Recebendo Suporte Técnico para o Geomview

Existem muitas formar de receber suporte para o Geomview.

1. Visite o sítio web do Geomview, http://www.geomview.org. Esse sítio conté a documentação mais recente, notícias sobre o desenvolvimento, e a FAQ (Frequently Asked Questions).
2. Envie um email para a lista de mensagens geomview-users@lists.sourceforge.net. Essa é uma lista de mensagens para discutir quaisquer quetões relacionadas ao uso do Geomview. Para juntar-se à lista, envie um email vazio com a palavra 'subscribe' na linha de assunto para geomview-users-request@lists.sourceforge.net, ou visite a página da lista em http://lists.sourceforge.net/mailman/listinfo/geomview-users.
3. Contrato com Geometry Technologies para suporte. Geometry Technologies é um suporte através de contrato e companhia de programação que emergiu do Geometry Center, onde o Geomview foi escrito. Paa mais informação visite o sítio web da Geometry Technologies em http://www.geomtech.com.

Contribuindo para o Desenvolvimento do Geomview

Se você está interessado em contribuir para o desenvolvimento do Geomview, existe muitas coisas que você pode fazer:

1. **Trabalho voluntário de programação**.

   Se você é um programador e fez um melhoramento para o Geomview, contacte os outros autores do geomview enviando uma mensagem para a lista de mesagens '`geomview-users`' (veja `http://lists.sourceforge.net/mailman/listinfo/geomview-users`).

2. **Contrato com a Geometry Technologies**.

   Geometry Technologies, Inc. é uma firma de consultoria que fornece contrato de suporte técnico e serviços de personalização de programas na área de gráficos em 3D. Inclui um amplo leque de serviços relacionados a gráficos em 3D, incluindo mas não limitado a aplicações envolvendo Geomview. Para extender os recursos, Geometry Technologies suporta o desenvolvimento do Geomview; em particular hospeda o sítio web `http://www.geomview.org`, e seu staff faz constantes melhoramentos no Geomview em si mesmo. Se você está em uma posição de pagar por suporte técnico ou trabalho personalizado de programação, contratando com a Geometry Technologies indiretamente suporte para o Geomview. Você pode também contratar com a Geometry Technologies para ter recursos particulares que você deseja adicionar ao Geomview, ou portar o Geomview para uma nova plataforma. Para mais informação veja o sítio web da Geometry Technologies em `http://www.geomtech.com`.

Obrigado.

Índice das Funções

!

!, shell 129, 152

*

* 129

+

+ 130

-

- 130

/

/ 130

<

< 129

=

= 129

>

> 129

?

? 130
?? 130

|

|, emodule-run 130, 135

A

all 130
all camera 130
all emodule defined 130
all emodule running 130
all geometry 130
and 130
ap-override 131

B

backcolor 131
background-image 131
bbox-color 131
bbox-draw 131

C

camera 131
camera-draw 131
camera-prop 131
camera-reset 132
car 132
cdr 132
clock 132
command 132
cons 132
copy 132
cursor-still 132
cursor-twitch 133

D

defun 133
delete 133
dice 133
dimension 133
dither 133
draw 134
dump-handles 134

E

echo 134
emodule-clear 134
emodule-define 134
emodule-defined 134
emodule-isrunning 134
emodule-path 135
emodule-run, | 130, 135
emodule-sort 135
emodule-start 135
emodule-transmit 135
escale 136
eval 136
event-keys 136
event-mode 136
event-pick 137
evert 137
exit 137
ezoom 137

F

freeze 137

G

geometry ... 137
geomview-version ... 137

H

hdefine ... 137
hdelete ... 138
help ... 138
hmodel ... 138
hsphere-draw ... 138

I

if ... 139
inhibit-warning ... 139
input-translator ... 139
interest ... 139

L

lambda ... 140
let ... 140
lines-closer ... 140
load ... 141
load-path ... 141
look ... 141
look-encompass ... 141
look-encompass-size ... 142
look-recenter ... 142
look-toward ... 142

M

merge ... 142
merge-ap ... 142
merge-base-ap ... 142
merge-baseap ... 143
mod ... 143
morehelp ... 143

N

name-object ... 143
ND-axes ... 143
ND-color ... 144
ND-xform ... 144
ND-xform-get ... 144
ND-xform-set ... 144
new-alien ... 145
new-camera ... 145
new-center ... 145
new-geometry ... 145
new-reset ... 145
NeXT ... 145
normalization ... 146
not ... 146

O

or ... 146

P

pick ... 146
pick-invisible ... 147
pickable ... 147
position ... 147
position-at ... 147
position-toward ... 148
process-events ... 148
progn ... 148

Q

quit ... 148
quote ... 148

R

rawevent ... 148
rawpick ... 148
read ... 149
real-id ... 149
redraw ... 149
regtable ... 149
rehash-emodule-path ... 149
replace-geometry ... 149
rib-display ... 150
rib-snapshot ... 150

S

scale ... 150
scene ... 150
set-clock ... 150
set-conformal-refine ... 150
set-emodule-path ... 151
set-load-path ... 151
set-motionscale ... 151
setenv ... 151
setq ... 151
sgi ... 152
shell, ! ... 129, 152
sleep-for ... 152
sleep-until ... 152
snapshot ... 152
soft-shader ... 153
space ... 153
stereowin ... 153

T

time-interests ... 154
transform ... 154
transform-incr ... 155

transform-set 155
truncate 155

U

ui-cam-focus 155
ui-center 156
ui-center-origin 156
ui-emotion-program 156
ui-emotion-run 156
ui-freeze 156
ui-html-browser 156
ui-motion 156
ui-panel 157
ui-pdf-viewer 158
ui-target 158
uninterest 158
update 158
update-draw 158

W

while 158
window 159
winenter 159
write 159
write-comments 159
write-handle 160
write-sexpr 160

X

xform 160
xform-incr 160
xform-set 160

Z

zoom 160

Índice

Introdução ao Geomview **2**

Distribuição **3**

Copiando **4**

LICENÇA PÚBLICA GERAL DO GNU **5**

Introdução 5
TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO 7
Como Aplicar Estes Termos para Suas Novas Bibliotecas 14

História do Desenvolvimento do Geomview **15**

Autores 15

Plantaformas Suportadas **17**

Como Pronunciar "Geomview" **18**

1 Visão Geral **19**

2 Tutorial **20**

3 Interação **31**

3.1 Iniciando o Geomview 31
3.2 Opções de Linha de Comando 31
3.3 Interacao Basica: O Painel Principal 33
3.4 Disponibilizando Objetos dentro do Geomview 36
3.5 Usando o Mouse para Controlar Objetos 39
3.5.1 Selecionando um Ponto de Interesse 44
3.6 Modificando a Forma de Ver as Coisas 45
3.6.1 The Appearance Panel 45
3.6.2 O Painel de Materiais 49
3.6.3 O Painel de Luzes 51
3.7 Cameras 52
3.8 Gravando Seu Trabalho 56
3.9 O Painel de Comandos 60
3.10 Atalhos de Teclado 60

4 Formatos dos Arquivos da OOGL 65
4.1 Convenções 65
4.1.1 Sintaxe Comum a Todos os Formatos de Arquivo da OOGL 65
4.1.2 Nomes de Arquivo 65
4.1.3 Vértices 65
4.1.4 Vértices *N*-dimensionais 66
4.1.5 Direções de superfícies normais 67
4.1.6 Matrizes de transformação 67
4.1.7 Matrizes de transformação ND 67
4.1.8 Formato binário 67
4.1.9 Referências a Objetos Embutidos e a Objetos Externos ... 68
4.1.10 Aparências 69
4.1.11 Mapeamento de Textura 74
4.2 Formatos de Arquivo de Objeto 76
4.2.1 QUAD: coleção de quadriláteros 76
4.2.2 MESH: Malha retangularmente conectada 77
4.2.3 BBOX: Caixas associada simples 78
4.2.4 Superfícies de Bezier 79
4.2.5 Arquivos do Tipo OFF 80
4.2.6 Arquivos do Tipo VECT 82
4.2.7 Arquivos do Tipo SKEL 83
4.2.8 SPHERE Files 84
4.2.9 Arquivos do Tipo INST 85
4.2.9.1 Exemplos INST 88
4.2.10 Arquivos do Tipo LIST 89
4.2.11 Arquivos do Tipo TLIST 89
4.2.12 Arquivos do Tipo GROUP 90
4.2.13 Arquivos do Tipo DISCGRP 90
4.2.14 Objetos do Tipo COMMENT 90
4.3 Objetos nao-geometricos 91
4.3.1 Objetos de aparencia 91
4.3.2 Objetos de imagem 91
4.3.3 Objetos de Transformação 96
4.3.4 Objetos ND-Transform 97
4.3.5 câmera 99
4.3.6 window 101

5 Customização: arquivos '`.geomview`' 103

6 Módulos Externos ... 104

6.1 Como Módulos Externos Interagem com o Geomview ... 104
6.2 Exemplo 1: Módulo Externo Simples ... 104
6.3 Exemplo 2: Módulo Externo Simples Usando o Painel de Controle FORMS ... 108
6.4 A Biblioteca XForms ... 112
6.5 Exemplo 3: Módulo Externo com Comunicação Bi-Direcional ... 113
6.6 Example 4: Simple Tcl/Tk Module Demonstrating Picking ... 122
6.7 Module Installation ... 125
6.7.1 Private Module Installation ... 126
6.7.2 System Module Installation ... 126

7 GCL: a Linguagem de Comandos do Geomview ... 127

7.1 Conventions Used In Describing Argument Types ... 128
7.2 GCL Reference Guide ... 129
7.2.1 ! ... 129
7.2.2 < ... 129
7.2.3 = ... 129
7.2.4 > ... 129
7.2.5 * ... 129
7.2.6 / ... 130
7.2.7 + ... 130
7.2.8 - ... 130
7.2.9 ? ... 130
7.2.10 ?? ... 130
7.2.11 | ... 130
7.2.12 all ... 130
7.2.13 and ... 130
7.2.14 ap-override ... 131
7.2.15 backcolor ... 131
7.2.16 background-image ... 131
7.2.17 bbox-color ... 131
7.2.18 bbox-draw ... 131
7.2.19 camera ... 131
7.2.20 camera-draw ... 131
7.2.21 camera-prop ... 131
7.2.22 camera-reset ... 132
7.2.23 car ... 132
7.2.24 cdr ... 132
7.2.25 clock ... 132
7.2.26 command ... 132
7.2.27 cons ... 132
7.2.28 copy ... 132
7.2.29 cursor-still ... 132
7.2.30 cursor-twitch ... 133
7.2.31 defun ... 133

7.2.32 delete . . . 133
7.2.33 dice . . . 133
7.2.34 dimension . . . 133
7.2.35 dither . . . 133
7.2.36 draw . . . 134
7.2.37 dump-handles . . . 134
7.2.38 echo . . . 134
7.2.39 emodule-clear . . . 134
7.2.40 emodule-define . . . 134
7.2.41 emodule-defined . . . 134
7.2.42 emodule-isrunning . . . 134
7.2.43 emodule-path . . . 135
7.2.44 emodule-run . . . 135
7.2.45 emodule-sort . . . 135
7.2.46 emodule-start . . . 135
7.2.47 emodule-transmit . . . 135
7.2.48 escale . . . 136
7.2.49 eval . . . 136
7.2.50 event-keys . . . 136
7.2.51 event-mode . . . 136
7.2.52 event-pick . . . 137
7.2.53 evert . . . 137
7.2.54 exit . . . 137
7.2.55 ezoom . . . 137
7.2.56 freeze . . . 137
7.2.57 geometry . . . 137
7.2.58 geomview-version . . . 137
7.2.59 hdefine . . . 137
7.2.60 hdelete . . . 138
7.2.61 help . . . 138
7.2.62 hmodel . . . 138
7.2.63 hsphere-draw . . . 138
7.2.64 if . . . 139
7.2.65 inhibit-warning . . . 139
7.2.66 input-translator . . . 139
7.2.67 interest . . . 139
7.2.68 lambda . . . 140
7.2.69 let . . . 140
7.2.70 lines-closer . . . 140
7.2.71 load . . . 141
7.2.72 load-path . . . 141
7.2.73 look . . . 141
7.2.74 look-encompass . . . 141
7.2.75 look-encompass-size . . . 142
7.2.76 look-recenter . . . 142
7.2.77 look-toward . . . 142
7.2.78 merge . . . 142
7.2.79 merge-ap . . . 142

7.2.80 merge-base-ap 142
7.2.81 merge-baseap 143
7.2.82 mod 143
7.2.83 morehelp 143
7.2.84 name-object 143
7.2.85 ND-axes 143
7.2.86 ND-color 144
7.2.87 ND-xform 144
7.2.88 ND-xform-get 144
7.2.89 ND-xform-set 144
7.2.90 new-alien 145
7.2.91 new-camera 145
7.2.92 new-center 145
7.2.93 new-geometry 145
7.2.94 new-reset 145
7.2.95 NeXT 145
7.2.96 normalization 146
7.2.97 not 146
7.2.98 or 146
7.2.99 pick 146
7.2.100 pick-invisible 147
7.2.101 pickable 147
7.2.102 position 147
7.2.103 position-at 147
7.2.104 position-toward 148
7.2.105 process-events 148
7.2.106 progn 148
7.2.107 quit 148
7.2.108 quote 148
7.2.109 rawevent 148
7.2.110 rawpick 148
7.2.111 read 149
7.2.112 real-id 149
7.2.113 redraw 149
7.2.114 regtable 149
7.2.115 rehash-emodule-path 149
7.2.116 replace-geometry 149
7.2.117 rib-display 150
7.2.118 rib-snapshot 150
7.2.119 scale 150
7.2.120 scene 150
7.2.121 set-clock 150
7.2.122 set-conformal-refine 150
7.2.123 set-emodule-path 151
7.2.124 set-load-path 151
7.2.125 set-motionscale 151
7.2.126 setenv 151
7.2.127 setq 151

7.2.128 sgi . . . 152
7.2.129 shell . . . 152
7.2.130 sleep-for . . . 152
7.2.131 sleep-until . . . 152
7.2.132 snapshot . . . 152
7.2.133 soft-shader . . . 153
7.2.134 space . . . 153
7.2.135 stereowin . . . 153
7.2.136 time-interests . . . 154
7.2.137 transform . . . 154
7.2.138 transform-incr . . . 155
7.2.139 transform-set . . . 155
7.2.140 truncate . . . 155
7.2.141 ui-cam-focus . . . 155
7.2.142 ui-center . . . 156
7.2.143 ui-center-origin . . . 156
7.2.144 ui-emotion-program . . . 156
7.2.145 ui-emotion-run . . . 156
7.2.146 ui-freeze . . . 156
7.2.147 ui-html-browser . . . 156
7.2.148 ui-motion . . . 156
7.2.149 ui-panel . . . 157
7.2.150 ui-pdf-viewer . . . 158
7.2.151 ui-target . . . 158
7.2.152 uninterest . . . 158
7.2.153 update . . . 158
7.2.154 update-draw . . . 158
7.2.155 while . . . 158
7.2.156 window . . . 159
7.2.157 winenter . . . 159
7.2.158 write . . . 159
7.2.159 write-comments . . . 159
7.2.160 write-handle . . . 160
7.2.161 write-sexpr . . . 160
7.2.162 xform . . . 160
7.2.163 xform-incr . . . 160
7.2.164 xform-set . . . 160
7.2.165 zoom . . . 160

8 Geometrias Não-Euclidianas . . . 161

9 Gráficos do Mathematica no Geomview ou no RenderMan 162

9.1 Usando Mathematica para gerar arquivos OOGL 162
9.2 Usando Geomview como Visualizador Padrão 3D do Mathematica 163
9.3 Usando Mathematica para gerar arquivos do RenderMan 164
9.4 Usando Geomview e Mathematica em Computadores Separados 165
9.4.1 Usando um host Geomview na mesma rede 166
9.4.2 Transportando arquivos do Mathematica para o Geomview Manualmente 166
9.5 Detalhes do Pacote Mathematica->Geomview 167

Recebendo Suporte Técnico para o Geomview 168

Contribuindo para o Desenvolvimento do Geomview 169

Índice das Funções 170

Lista de Figuras

Figura 2.1: Tela Inicial do Geomview. 21
Figura 2.2: Olhando para o Objeto Mundo. 24
Figura 2.3: A Painel Aparência. 25
Figura 2.4: Painel de Escolha de Cores. 26
Figura 2.5: O Painel de Arquivos. 28
Figura 2.6: Trevo e Dodecaedro. 30
Figura 3.1: O Painel Principal. 34
Figura 3.2: O Painel de Arquivos. 37
Figura 3.3: O Painel de Chamar Arquivos. 39
Figura 3.4: O Painel de Ferramentas. 40
Figura 3.5: O Painel de Aparência. 46
Figura 3.6: Painel de Escolha de Cor. 47
Figura 3.7: O Painel de Materiais. 50
Figura 3.8: O Painel de Câmera. 53
Figura 3.9: O Painel Gravar. 57
Figura 3.10: O Painel de Comandos. 60