MANGABA, UM SISTEMA DE GEOMETRIA DINÂMICA ESPACIAL

Rodrigo da Silva Moreira

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por:

Prof. Cláudio Esperança, Ph. D.

Prof. Antônio de Oliveira, Ph. D.

Prof. Luiz Carlos Guimarães, Ph. D.

Prof. Rafael Garcia Barbastefano, Ph. D.

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2004

MOREIRA, RODRIGO DA SILVA

Mangaba, um Sistema de Geometria Dinâmica [Rio de Janeiro] 2004

VII, 92 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2004)

Tese-Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas de Geometria Dinâmica

I. COPPE/UFRJ II. Título (Série)

Agradecimentos

Este trabalho não existiria se não fosse pela ajuda generosa de diversas pessoas, incluindo aquelas que são mencionadas aqui.

Em primeiro lugar, agradeço ao meu orientador, Professor Cláudio Esperança, por suas críticas objetivas e seus comentários esclarecedores e principalmente por acreditar na minha capacidade. Também gostaria de agradecer aos meus colegas do grupo PACE do Instituto de Matemática da UFRJ e do Laboratório de Computação Gráfica pelas inúmeras sugestões e críticas, e por terem fornecido parte do material bibliográfico que muito contribuiu para o enriquecimento deste trabalho.

Um agradecimento especial a meu pai e minha mãe, por terem me dado apoio e principalmente a oportunidade de chegar até aqui.

Agradeço a Deus, o ser criador dos seres vivos. A Jesus Cristo e Maria que estão sempre ao nosso lado, até mesmo em momentos mais difíceis quando não achamos saídas, indicando-nos o caminho mais adequado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

MANGABA, UM SISTEMA DE GEOMETRIA DINÂMICA ESPACIAL

Rodrigo da Silva Moreira

Maio/2004

Orientador: Cláudio Esperança

Programa: Engenharia de Sistemas e Computação

Esta dissertação apresenta o estudo da dinâmica e do movimento de sistemas em que objetos estão conectados e têm suas posições restritas. Discutimos os sistemas de geometria dinâmica existentes e como é feito o cálculo da posição de objetos articulados formando um mecanismo de objetos sem massa (cinemática inversa) e com massa (restrições dinâmicas). Em particular, desenvolvemos um sistema de geometria dinâmica baseado em construções para a atualização de objetos geométricos que fazem parte de uma construção e estão dispostos em uma cena tridimensional. O sistema desenvolvido na presente dissertação visa à elaboração de um *software* de ensino de Geometria Espacial, no qual se pode comprovar a veracidade de alguns teoremas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

MANGABA, A DINAMIC SPATIAL GEOMETRY SYSTEM

Rodrigo da Silva Moreira

May/2004

Advisor: Cláudio Esperança

Department: Computing and Systems Engineering

This Thesis introduces the study of the dynamics and movement in systems where linked objects have constrained positions. We discuss about the existing dynamic geometry systems and how we can calculate the position of linked objects in a mechanism of objects without mass (inverse kinematics) and objects with mass (dynamic constraints). In particular, we developed a constructive based dynamic geometry system to calculate the position of geometric objects in a geometric construction distributed in a tridimensional scene. The system developed in this thesis intends to elaborate a Spatial Geometry teaching software, where the veracity of some theorems can be proved.

Sumário

Capítulo 1

Introdução

A Geometria é uma ciência que traz diversas dificuldades de entendimento para o aluno de ensino médio. Isso é fácil ser compreendido, pois, como habitantes de um mundo tridimensional, estamos habituados a visualizar objetos tridimensionais. Ao representar objetos geométricos tridimensionais em quadros negros, o aluno não os visualiza com facilidade, pois somente a projeção do objeto é representada.

Com o advento do computador, tornou-se possível o desenvolvimento de novas ferramentas de ensino de Matemática que facilitam o entendimento das geometrias plana e espacial pelos alunos por meio da visualização das relações geométricas. Com essas ferramentas, o aluno pode interagir dinamicamente com os objetos geométricos mudando suas posições. As relações geométricas entre os objetos se mantêm durante a interação. Este método de ensino surgiu em meados dos anos 80 com a geometria dinâmica, disciplina que estuda a dinâmica e o movimento de objetos geométricos envolvidos em uma construção geométrica. Uma construção geométrica é um conjunto de objetos geométricos no qual a posição de alguns destes depende da posição de outros objetos do conjunto. Por exemplo, podemos definir uma construção onde o eixo de um cilindro deve ser paralelo a uma reta e passar por um ponto. Se a reta ou o ponto tiver sua posição alterada, a posição do cilindro é modificada de forma a manter as relações geométricas.

O presente trabalho está ligado ao projeto PACE do Instituto de Matemática da UFRJ, que tem por objetivo desenvolver ferramentas computacionais adequadas à confecção de materiais de ensino de matemática, para serem usados em disciplinas de primeiro ano de universidade, programas de reciclagem de professores de matemática de nível médio e estudantes do ensino médio. Como fruto resultante do trabalho em equipe do projeto, foram desenvolvidas as ferramentas Tabulae[1], Mangaba[1] e MathChat[1].

O Tabulae é um *software* feito em Java destinado ao ensino de geometria plana. Algumas ferramentas computacionais de ensino de geometria plana já existem: Cinderella [2,3,4,5,6], Cabri [7,8,9,10,11] e o Sketchpad[12,13,14,15]. O Tabulae possui o mesmo status de funcionalidade desses *softwares*. O MathChat é um programa que possibilita cálculos matemáticos e ensino à distância de Matemática por meio de troca de mensagens. O *software* Mangaba foi projetado com base em estudos feitos nas áreas de Redes de Computadores e de Engenharia de *Software* e se destina ao ensino de geometria espacial. Com o Mangaba, podemos visualizar e verificar a veracidade de alguns teoremas e propriedades da geometria espacial. A verificação de um teorema pode ser vista no Apêndice B. Na literatura encontramos poucas ferramentas de ensino de geometria espacial [16,17]. Em geral, as ferramentas de ensino de geometria espacial possuem a elaboração gráfica limitada, representando os objetos geométricos apenas com pontos e retas. Já a geometria dinâmica plana está presente na literatura há bastante tempo. Muitas ferramentas como o Cinderella [2,3,4,5,6], Cabri [7,8,9,10,11], o Sketchpad[12,13,14,15], GéoSpécif [18,19,20], o UniGéom [21,22] e o Juno-2 [23] foram desenvolvidas. Duas teses que foram defendidas em 1999 [5,24] apontavam a geometria dinâmica espacial como área a ser explorada. Hoje, a geometria dinâmica espacial é um tema que está em evidência.

A presente dissertação destina-se ao estudo do desenvolvimento do sistema de geometria dinâmica espacial chamado Mangaba dando ênfase ao sistema de cálculo da posição dos objetos nas construções geométricas. No Mangaba a representação gráfica dos objetos foi feita com OpenGL – uma biblioteca gráfica que possibilita o desenho de um conjunto pequeno de objetos: pontos, retas, polígonos, imagens e *bitmaps* em ambientes bidimensionais e tridimensionais. OpenGL possui vários recursos para o controle do desenho como luminosidade, permitindo uma representação gráfica mais realista e com isto facilitando o entendimento das construções por parte dos alunos.

No capítulo 2, apresentamos um estudo sobre trabalhos relacionados. Nesse capítulo são abordados os sistemas de geometria dinâmica mais conhecidos.

No capítulo 3, expomos um estudo sobre o comportamento de objetos ligados e que têm suas posições restritas. Dois assuntos semelhantes à geometria dinâmica são abordados: cinemática inversa e restrições dinâmicas. Em cada um dos assuntos, apresentamos ou demonstramos como a posição de um objeto pode ser calculada dada a mudança na posição dos objetos de que ele depende.

No capítulo 4 apresentamos um estudo da interface do sistema de geometria dinâmica do Mangaba. Uma interface bem elaborada possibilita melhores visualização e interação com objetos pertencentes às construções geométricas. No capítulo 5 apresentamos um estudo sobre como é feito o cálculo da posição dos objetos geométricos de uma construção.

No apêndice A expomos uma tabela de restrições geométricas. A tabela mostra o nome de cada restrição geométrica, os objetos envolvidos e como é a relação entre os objetos. No apêndice B é apresentada uma discussão sobre dois teoremas Geométricos e como pode ser verificada a veracidade dos mesmos com o uso do Mangaba.

Capítulo 2

Trabalhos Relacionados

Como discutido no capítulo 1, existem muitas áreas em que se estuda o comportamento de objetos ligados por relações de restrição de suas posições ou orientações. Em problemas de geometria dinâmica estuda-se o comportamento de objetos geométricos que fazem parte de uma construção. Em uma construção, alguns objetos geométricos têm suas posições dependentes de outros objetos geométricos.

2.1 Geometria Dinâmica

O computador tem se revelado uma importante ferramenta no estudo da Matemática. Em particular pela utilização de programas adequados conhecidos como sistemas de geometria dinâmica, é possível montar interativamente construções envolvendo objetos geométricos como esferas, planos, pontos e cones. Usando tais sistemas, o usuário pode não só desenhar objetos e movê-los na tela, mas também restringir a posição de alguns objetos em relação a outros com o objetivo de estudar alguns teoremas relacionados às geometrias plana e espacial. As restrições podem envolver paralelismo, perpendicularidade, distâncias e outras. Por exemplo, o usuário pode verificar como a interseção entre um plano P_1 e uma reta r se comporta ao mudar interativamente a posição da reta ou do plano. Com um sistema de geometria dinâmica a posição de todos os objetos de uma construção pode ser calculada e atualizada.

Muitos sistemas de geometria dinâmica [2,9,12,18,19,24,25,26,27,28] foram desenvolvidos nos últimos anos. Alguns usam apenas o conceito de orientação a objeto (seção 2.5.3). Nesses sistemas, os objetos da construção participam ativamente da atualização das suas posições. Em outros (seção 2.5.2), a atualização é feita usando um grafo de dependências de objetos simples. No sistema descrito por Winroth [24] e por Van Labeke N. [16], a atualização da construção é feita com o auxílio de um grafo de dependências de objetos compostos (seção 2.5.1). Existem dois tipos básicos de sistemas

de geometria dinâmica: um é baseado em restrições e o outro, em construções. O GéoSpécif [18,19,20], o UniGéom [21,22] e o Juno-2 [23] são ferramentas que utilizam sistemas baseados em restrições. Já o Cabri [7,8,9,10,11], o GSP [13,14,15] e o Cinderella Café [3,4,5,6] são ferramentas que usam sistemas baseados em construções.

Neste capítulo serão abordados os sistemas de geometria dinâmica, enfatizando como as restrições impostas aos objetos podem ser resolvidas em sistemas baseados em restrições, e em construções, e suas principais diferenças. Esta discussão está detalhada em [24].

2.2 Conceitos de orientação a objeto (OO)

2.2.1 Objetos, métodos, atributos e mensagens

Os objetos no mundo real possuem estados como, por exemplo, cor e tamanho, e comportamentos como, exemplificando, a capacidade de frear um carro. Em orientação a objeto, um objeto mantém seus estados em variáveis e implementa seus comportamentos com métodos. Tudo que um objeto sabe (estado) é expresso por variáveis; e tudo que um objeto pode fazer (comportamento) é expresso por métodos. Os objetos se comunicam por mensagens. Quando um objeto *A* quer que o objeto *B* execute uma ação, o objeto *A* envia uma mensagem ao objeto *B*. Ao recebê-la, um objeto executa um método para alterar ou verificar seu estado.

2.2.2 Classe

Considere uma bicicleta. A sua é apenas uma das existentes no mundo. As bicicletas possuem estado e comportamento em comum (têm duas rodas, um guidom, etc). A classe é um molde para a construção de um objeto. Na terminologia OO, sua bicicleta (objeto) é uma instância da classe *Bicicleta.* Com uma classe podem ser instanciados (criados) vários objetos que possuem os mesmos atributos e métodos (bicicletas de mesma cor e estilo), porém objetos instanciados de uma mesma classe podem possuir estados independentes (bicicletas de cores diferentes). Para cada objeto instanciado é alocado um espaço de memória.

2.2.3 Encapsulamento

O encapsulamento é usado para esconder detalhes de implementação de uma classe. Para executar um método, um objeto não precisa saber como aquele foi implementado, basta saber a interface (nome) do método. Um objeto pode disponibilizar algumas variáveis e métodos para que outros possam utilizar. Já certos métodos e variáveis não são disponibilizados, pois só são usados pelo próprio objeto. As variáveis e métodos disponíveis são ditos públicos. As variáveis e métodos não disponíveis são ditos privados.

2.2.4 Herança

A herança descreve o relacionamento entre classes definidas a partir de outras classes. Toda subclasse herda os atributos e os métodos públicos definidos na superclasse e acrescenta atributos e métodos próprios. Por exemplo, uma *mountain bike* é uma bicicleta com estados e comportamentos próprios. Pode ser criada uma subclasse *MountainBike* que se estende (herda) da superclasse *Bicicleta.*

2.2.5 Classe abstrata

Classe abstrata é aquela na qual pelo menos um de seus métodos está declarado, mas não tem implementação associada. Essa classe não pode ser instanciada, mas pode servir como uma superclasse para outras classes. Cada subclasse deve implementar o método que não é implementado pela classe abstrata. A classe abstrata, portanto, serve para organizar a herança das classes.

2.2.6 Polimorfismo

Polimorfismo é a habilidade de diferentes instâncias, de classes diferentes que se estendem de uma mesma classe, responderem à mesma mensagem de maneiras diversas. Considere três classes:

1. Uma classe *Caixa*, que representa uma caixa que abre;
2. Uma classe *Dado*, que representa um dado;
3. Uma classe *Cubo*, que representa um *Cubo*.

As classes *Dado* e *Caixa* se estendem da classe *Cubo* e implementam seus próprios métodos de desenho (estes métodos devem ter o mesmo nome do método de desenho da classe *Cubo*). Quando uma instância da classe *Cubo* recebe uma mensagem para se desenhar, ela executa o método de desenho da classe *Cubo* que desenha seis faces sem textura. Quando uma instância da classe *Caixa* recebe uma mensagem para se desenhar, executa o método de desenho da classe *Caixa*, que desenha seis faces com textura sendo que uma das faces (a tampa) é ligada a uma outra face apenas por uma aresta. Quando uma instância da classe *Dado* recebe uma mensagem para se desenhar, executa o método de desenho da classe *Dado*, que desenha seis faces sem textura e pontos sobre as faces indicando os números.

2.3 Restrições Geométricas

Uma restrição geométrica é uma restrição (ver tabela de restrições no apêndice A) dada à posição de um objeto geométrico. Considere a construção apresentada na Fig. 2.1a onde há duas restrições e que envolve os seguintes objetos: um plano Π, uma reta r e dois pontos P_1 e P_2. O plano Π e o ponto P_1 são livres. A reta deve ser perpendicular ao plano e passar pelo ponto P_1 (a posição de r está restrita). A posição do ponto P_2 deve ser restrita à interseção entre Π e r.

Toda construção pode ser representada por um grafo onde os nós correspondem aos objetos geométricos, e os arcos às restrições entre eles. O grafo correspondente à construção da Fig. 2.1a é mostrado na Fig. 2.1b. O sistema desenvolvido deve ser capaz de manter a construção da construção quando a posição dos objetos é mudada.

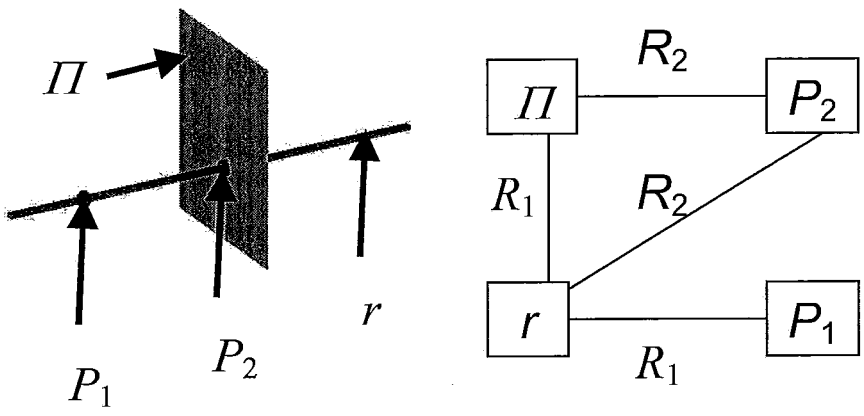


Figura 2.1a: Reta perpendicular a um plano e passando por um ponto livre (*RetaPePlanoPonto*). Um segundo ponto é definido como sendo o ponto de interseção entre o plano e a reta (*IntRetaPlano*).
(b): Grafo relacionado à construção em (a).

Por exemplo, se o ponto P_1 for transladado, o sistema deve ser capaz de atualizar a posição de todos os objetos que compõem a construção de forma a manter as restrições criadas. Nas duas próximas seções é discutido como os problemas de geometria dinâmica podem ser resolvidos usando sistemas baseados em construções e sistemas baseados em restrições.

2.4 Sistemas baseados em restrições

Em sistemas baseados em restrições [18,19,20,21,22,23] toda construção pode ser representada com um grafo não direcionado de restrições onde os nós representam os objetos da construção, e os arcos as restrições de um objeto em relação ao outro. Quando um objeto da construção é movido, todos os nós que com ele se relacionam no grafo são atualizados. Quando P_2 se move, a reta r e o ponto P_1 têm suas posições atualizadas. Quando P_1 se move, tanto r quanto P_2 têm suas posições atualizadas. Quando o plano Π muda de posição, as posições de P_2 e r são atualizadas.

2.4.1 Problemas de Ambigüidade

O grafo não é um grafo direcionado, ou seja, a informação de atualização pode

ser percorrida em qualquer direção. Em construções simples, sistemas baseados em restrições conseguem resolver a construção, mas, em construções mais complexas, a restrição pode permitir duas soluções possíveis para a mesma construção. Por exemplo, considere a construção da Fig. 2.2 consistindo de um plano Π mediano a dois pontos P_1 e P_2.

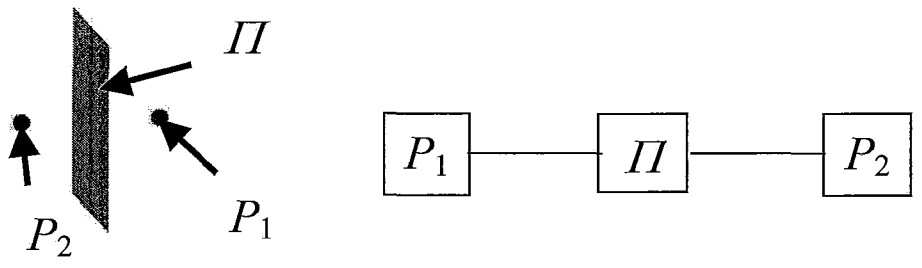


Figura 2.2: (a) Plano mediano a dois pontos (*PlanoPontoPonto*) (ambigüidade). (b) Grafo relacionado à construção em (a).

A Fig. 2.2a mostra uma construção consistindo de um plano mediano a dois pontos (*PlanoPontoPonto*). O grafo correspondente está representado ao lado. Quando o ponto P_1 é transladado, existem três possibilidades em que a construção permanece válida. A primeira é o plano Π mudar de posição e o ponto P_2 ficar parado. A segunda é o ponto P_2 mudar de posição e Π ficar parado. A terceira é P_2 e Π mudarem de posição.

Em sistemas baseados em restrições, deve-se ter um método geral para manter válidas as construções. Observando o grafo, podemos associar os seus arcos às equações do sistema que deve ser resolvido, e as variáveis às posições dos objetos que estão na mesma componente conexa. De uma forma geral, calcular a posição dos objetos de uma construção com sistemas baseados em restrições envolve um sistema de equações polinomiais não linear com várias variáveis. Para resolver tal tipo de problema existem métodos numéricos e métodos algébricos.

2.4.2 Construções com mais de uma solução

Calcular a posição dos objetos de uma construção é complicado principalmente

porque não é possível prever se o usuário fará uma construção que não tenha nenhuma solução ou que tenha várias possíveis. Por exemplo, considere a construção envolvendo os seguintes objetos: um diedro D, dois pontos P_1 e P_2 e uma reta r (veja a Fig. 2.3). O diedro D e a reta r são livres. Os pontos P_1 e P_2 são os pontos de interseção entre o diedro e a reta.

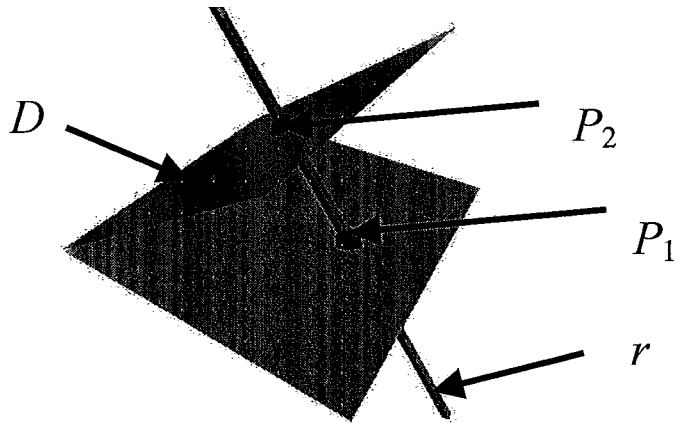


Figura 2.3: Pontos P_1 e P_2 que são a interseção entre o diedro D e a reta r.

Nesse caso, um sistema baseado em restrições monta um sistema de equações e calcula a posição dos pontos. A restrição imposta a cada ponto implica apenas que eles devem ser um ponto de interseção. Não existe uma restrição extra indicando que os pontos não podem possuir a mesma posição no espaço. Sem tal restrição, os pontos P_1 e P_2 podem ficar na mesma posição. Existe uma solução que satisfaz a configuração, que é a que realmente interessa, em que as posições dos pontos P_1 e P_2 são diferentes. Para que esta solução seja considerada, o sistema baseado em restrições acrescenta uma outra restrição ao sistema de equações indicando que as posições dos pontos no espaço devem ser distintas. O exemplo da Fig. 2.3 é bem simples, mas, mesmo em uma construção mais complexa, os sistemas baseados em restrições devem ter a capacidade de verificar se o sistema de equações tem muitas soluções e acrescentar as restrições mais adequadas para que a solução encontrada seja a que o usuário estava realmente esperando visualizar.

2.4.3 Acrescentando restrições temporárias para a interação com objetos

Independentemente da construção montada, quando o usuário muda a posição de um objeto da construção, todos os objetos que estão envolvidos na construção devem ter suas posições atualizadas suavemente. Por exemplo, considere a construção que envolve os seguintes objetos: Uma esfera *S* e um ponto *P*. O ponto *P* deve ficar sobre a esfera.

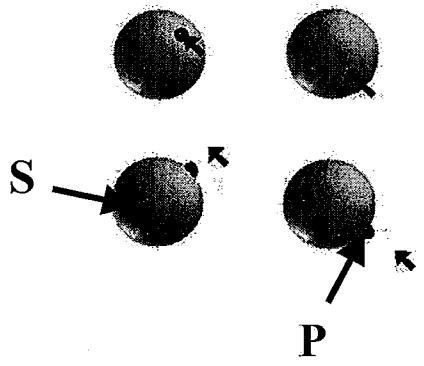


Figura 2.4: Ponto sobre uma esfera.

Se o usuário selecionar e transladar o ponto, independentemente da posição do *mouse*, o ponto deve ficar sobre a esfera. Quando a posição do *mouse* está sobre a esfera na tela, o comportamento do movimento do ponto deve ser diferente de quando o *mouse* não estiver sobre a esfera. Por exemplo, quando o *mouse* está sobre a esfera, a posição do ponto pode ser a projeção ortogonal da posição do *mouse* sobre a esfera na direção da normal que sai da tela. Quando o *mouse* não está sobre a esfera, a projeção nunca cai sobre a esfera. Portanto, em tempo real, enquanto o usuário muda a posição de um objeto em uma construção, um sistema baseado em restrições acrescenta uma restrição temporária relacionada com a posição do *mouse* ao sistema de equações, de forma a adequar a solução do sistema de equações (no caso a solução é a posição do ponto) ao que o usuário esperava visualizar, ou seja, um movimento consistente e suave do ponto.

2.4.4 Definindo a dinâmica de uma construção com restrições temporárias

Nem sempre acrescentar apenas uma restrição temporária ao sistema de equações o deixa com apenas uma solução. Nestes casos, um sistema baseado em restrições deve fixar a posição de um ou mais objetos da construção. Na verdade, um sistema baseado em restrições acrescenta restrições temporárias a determinados objetos da construção enquanto a posição de um objeto está sendo alterada. A escolha das restrições não é arbitrária. Ela deve seguir alguns princípios:

1. Quando a posição de um objeto da construção é mudada, a posição dos outros objetos que fazem parte da construção deve ser intuitiva, contínua e, de certa forma, previsível.
2. As restrições temporárias devem ser tais que permitam alcançar de forma intuitiva qualquer configuração da construção com um número finito de operações efetuadas com o *mouse*.

Considere a construção que envolve os seguintes objetos: um plano Π, uma reta r e um ponto P (Veja Fig. 2.5). O ponto P deve ficar sobre o plano Π. O plano Π deve ser perpendicular à reta r.

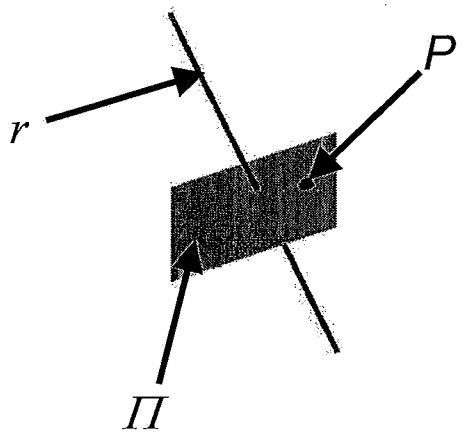


Figura 2.5: Plano perpendicular a uma reta (*PlanoPeReta*) e um ponto sobre o plano (*PontoSoPlano*).

Nessa construção o sistema de equações pode ter várias soluções. Somente as duas restrições da construção (Π perpendicular à reta e ponto sobre plano) não definem apenas uma posição possível para todos os objetos que compõem a construção. Suponha que a reta seja transladada para uma nova posição. O sistema baseado em restrições pode, por exemplo, acrescentar duas restrições:

1. O ponto deve manter as suas posições x e y quando a reta for transladada.
2. O plano deve transladar apenas em relação ao eixo x com o mesmo valor da translação sofrida pela reta. Essas duas restrições temporárias junto com as duas restrições criadas definem apenas uma posição para todos os objetos da construção quando a reta for transladada. Acrescentar restrições temporárias sem levar em conta a como o usuário vê a cena pode tornar a dinâmica da construção difícil de ser entendida. As restrições temporárias acrescentadas devem ser tais que o movimento (dinâmica) dos objetos da construção sejam intuitivas.

2.5 Sistemas baseados em construções

Os sistemas baseados em construções [3,4,5,6,7,8,9,10,11,13,14,15] têm princípios diferentes dos sistemas baseados em restrições. Diferentemente dos sistemas baseados em restrições, nos quais um sistema de equações tem que ser resolvido, em sistemas baseados em construções as restrições são mantidas por meio de um mecanismo de aviso implementado sobre uma estrutura de dados. Existe uma relação hierárquica entre os objetos que fazem parte da construção (objetos *pais* e objetos *filhos*). Assim, digamos, se um ponto P pertence a uma reta r, diz-se que P é *filho* de r e r é *pai* de P. Quando a reta se move, existe um mecanismo de aviso para que a posição do ponto seja modificada e o ponto continue sobre a reta.

Cada restrição geométrica imposta entre um *pai* e um *filho* é dita *restrição*. Por exemplo, considere a construção da Fig. 2.6. Existe uma restrição que envolve os pontos P_1, P_2 e o plano Π_1, uma restrição que envolve os pontos P_2, P_3 e o plano Π_2 e uma restrição que envolve os planos Π_1, Π_2 e a reta r. Diferentemente do que acontece em sistemas baseados em restrições, toda restrição acrescentada à construção tem uma solução, portanto a construção por inteiro só tem uma solução. Ocasionalmente se pode escolher um determinado comportamento para os objetos de uma restrição, definindo,

portanto, a solução desejável para a restrição. Por outro lado perde-se a liberdade proporcionada pelos sistemas baseados em restrições de fazer qualquer tipo de construção.

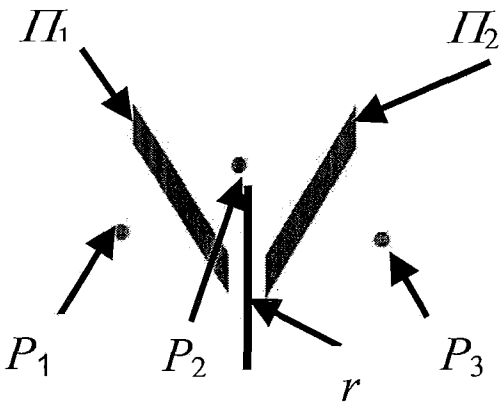


Figura 2.6: Π_1 mediano aos pontos pontos P_1 e P_2 ; Π_2 mediano aos pontos P_2, P_3 e r interseção entre Π_1 e Π_2.

O mecanismo de aviso que permite atualizar a posição dos objetos filhos quando um objeto pai muda de posição pode ser feito de diversas formas, usando vários tipos de estruturas de dados que podem ser vistas como um grafo de dependências, a saber, as mais utilizadas:

1. Grafo de dependência de objetos compostos (cada nó representa um objeto geométrico e a restrição).
2. Grafo de dependência de objetos simples (cada nó representa um objeto geométrico).

2.5.1 Grafo de dependência de objetos compostos

Nos sistemas baseados em restrições, o grafo de dependência não é direcionado. Por exemplo, observe o exemplo da Fig. 2.1. Quando o ponto P_1 é transladado, a posição de P_2 deve mudar. Quando o ponto P_2 é transladado, P_1 deve ter a sua posição atualizada. Já em sistemas baseados em construções, quando um grafo é usado para a

a atualização, ele deve ser direcionado e acíclico. Usando o exemplo da Fig. 2.1, o termo "direcionado" indica que apenas uma das duas possibilidades é possível: quando o ponto P_1 é transladado, a posição de P_2 deve mudar ou quando o ponto P_2 é transladado, P_1 deve ter sua posição atualizada. Isso indica que os sistemas baseados em construções são menos flexíveis que os sistemas baseados em restrições no que toca à atualização das construções.

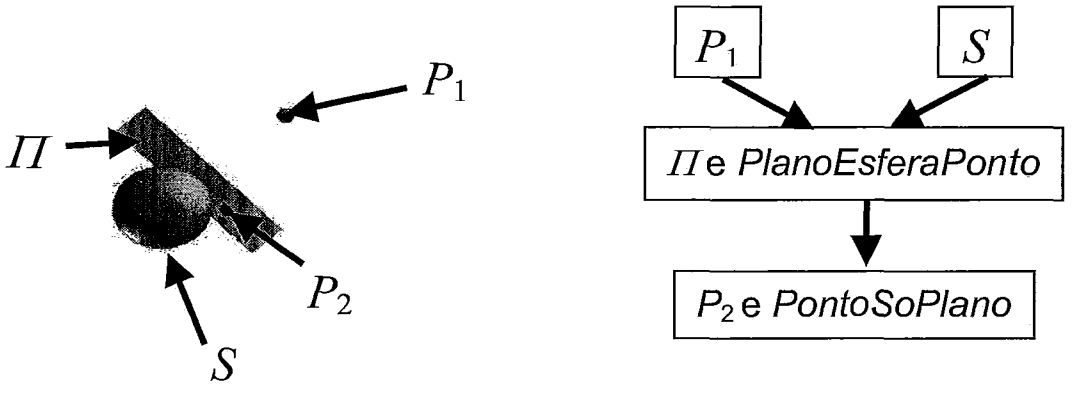


Figura 2.7: Duas restrições: Ponto sobre um plano e Plano tangente a uma esfera e perpendicular a uma reta imaginária que passa por um ponto livre e pelo centro da esfera.

Em sistemas baseados em construções que usam um grafo de dependência composto, os arcos do grafo representam apenas as relações de dependência entre os nós. Um sistema conhecido é o PDB [24]. Em toda a tese, os exemplos dos sistemas de geometria dinâmica plana foram transformados para exemplos no espaço. Nele, cada nó do grafo é uma combinação de um objeto e das restrições impostas a este. Imagine que um ponto livre P_1 e uma esfera livre S sejam criados. Inicialmente só existem no grafo dois nós não ligados.

Depois, um plano Π é criado contendo a seguinte restrição: Π deve ser tangente à esfera e perpendicular a uma reta imaginária que passa pelo centro da esfera e pelo ponto P_1. Neste instante o grafo apresenta três nós, representando uma esfera livre S, um ponto livre P_1 e uma combinação de um plano e uma restrição a S e P_1 (*PlanoEsferaPonto*). Quando a posição de S ou de P_1 são modificadas, o nó S ou P_1 informam ao nó *PlanoEsferaPonto* que a posição de Π deve ser modificada. O nó *PlanoEsferaPonto*, que contém a informação da restrição e da nova posição dos objetos representados pelos nós pais, calcula a nova posição do plano Π. O nó

PlanoEsferaPonto pode permitir que a posição do plano Π seja mudada desde que não afete a posição de S e de P_1, pois a atualização dos objetos do grafo ocorre apenas em um sentido, isto é, na direção dos pais para os filhos. Neste exemplo, as restrições poderiam permitir que o usuário pudesse escorregar Π, mas mantendo Π com a mesma inclinação e tangente à esfera.

Existem construções em que a posição do objeto está totalmente definida pela posição dos objetos pais, como é o caso de um segmento de reta por dois pontos. Como o grafo é direcionado, a posição do segmento de reta não pode ser mudada, pois o segmento está restrito à posição dos dois pontos. Há construções em que a posição do objeto está apenas parcialmente definida pela posição dos objetos pais, como é o caso do plano Π na construção da Fig. 2.7. Π pode ser transladado mantendo sua inclinação e continuando tangente à esfera. Neste caso o sistema pode acrescentar restrições temporárias ao nó *PlanoEsferaPonto*, definir totalmente a posição de Π ou deixar que o usuário acrescente restrições extras para definir por completo a posição de Π.

2.5.2 Grafo de dependência de objetos simples

Nesse caso os objetos devem ser construídos usando o conceito de orientação a objeto. Todos os objetos (cone, esfera, etc) são instâncias de uma classe que se estende de uma superclasse comum. Essa superclasse deve conter as características comuns a todos os objetos como, por exemplo, cor, dilatação, posição espacial, transparência, o tipo do objeto; deve conter os métodos comuns a todos os objetos como, por exemplo, um método que faz retornar sua transparência, posição ou dilatação, um método para desenho, métodos que permitem mudar a dilatação e a posição espacial, e ou também dispor de um método para que o objeto atualize sua própria posição quando um objeto pai muda de posição. Além disso, essa superclasse deve conter uma lista de objetos dependentes, ou seja, de objetos que devem ser atualizados quando o objeto muda de posição no espaço. Essa classe deve ser abstrata, e permitir que as subclasses possam redefinir os métodos caso seja necessário.

O método da classe *Cubo* é diferente do método de desenho da classe *Esfera*. Uma esfera pode ser aproximada e representada por um poliedro com muitas faces. O método da classe *Esfera* desenha apenas um contorno suave das faces do poliedro dando

a impressão de que é uma esfera ideal. Já a classe *Cubo* desenha todas as faces de sua representação interna. O método de manipulação da classe *Ponto* deve redefinir o método de rotação deixando-o vazio, sem nenhuma instrução. A classe *Reta* deve também redefinir o método de rotação, pois visualmente não há diferença entre rodar uma reta em torno do seu eixo ou não mudar seu estado. Considere uma construção com muitos objetos, onde uma reta é a primeira da cadeia de atualização, ou seja, todos os outros objetos dependem, direta ou indiretamente, da posição da reta. Se o método de rotação da classe *Reta* não fosse redefinido, quando a reta fosse rodada em torno de seu eixo os outros objetos da construção receberiam um aviso de que deveriam mudar de posição. Os outros objetos devem ter alguns de seus métodos redefinidos por razões similares.

Uma classe que representa um objeto restrito – como, por exemplo, um plano por três pontos (*Plano3Pontos*) – nesse sistema é definida como uma extensão da classe Plano. Ela herda as características da classe *Plano* só que sua posição depende de três pontos. *Plano3Pontos* tem uma lista contendo a referência para esses três pontos e redefine os métodos de atualização da posição. Então todas as classes devem ser definidas de forma a estender uma classe já previamente definida e acrescentar o que for necessário. Por exemplo, a classe *PontoSoPlano* deve estender a classe *Ponto*. Já a classe *IntRetaPlano* deve estender a classe *PontoSoPlano*.

O sistema de atualização funciona da forma a seguir explicada. Considere uma construção que envolve os seguintes objetos: um plano Π, dois pontos P_1 e P_2 e uma esfera S. P_1 e P_2 são dois pontos que estão sobre o plano Π. S tem o centro em P_1 e tem seu raio definido por P_2. Quando a posição do plano é mudada, Π percorre sua lista de objetos diretamente dependentes e avisa para P_1 e P_2 mudarem suas posições. O mesmo processo acontece para a atualização de S.

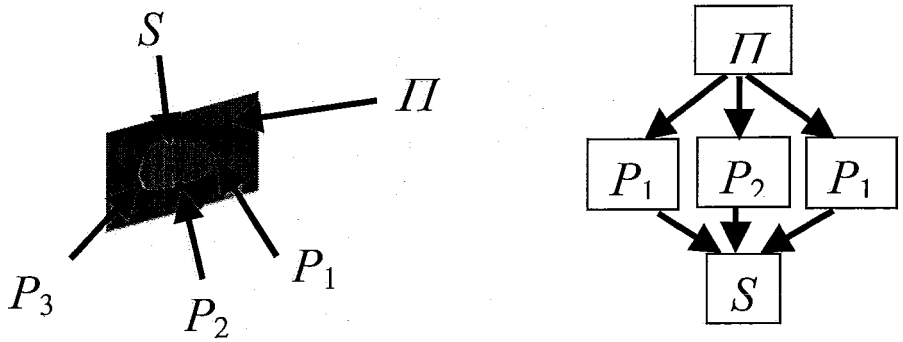


Figura 2.8: Dois pontos P_1 e P_2 sobre um plano e uma esfera com centro em P_1 e raio definido por P_2.

O sistema desenvolvido em [16] é um sistema de geometria dinâmica espacial que é semelhante ao desenvolvido em [24], que é um sistema de geometria dinâmica plana. Cada classe se estende direta ou indiretamente de uma outra que contém características comuns a todos os objetos. Cada classe que representa um objeto geométrico com posição restrita é instância de uma classe que contém uma referência aos objetos pais e um método de cálculo para sua posição quando um objeto pai muda de posição. Por exemplo, a classe *PoSoReta* estende a classe *Ponto* e contém uma referência à reta. A classe *IntRetaPlano* estende a classe *Ponto* e contém uma referência à reta e ao plano. Cada nó do grafo é o próprio objeto, diferentemente de [24], que é uma combinação do objeto e da restrição imposta, que é externa ao objeto. A atualização é feita de forma semelhante, porém o sistema descrito em [24] permite que a restrição imposta a um objeto, por ser externa a este, seja modificada acrescentando restrições temporárias ou permitindo que o usuário acrescente restrições extras.

2.5.3 Uma implementação possível

Neste caso, existem apenas dois níveis de orientação a objeto. Há, como na seção anterior, uma superclasse que engloba características que devem estar presentes em todos os objetos. Essa superclasse não pode ser instanciada. Todas as classes que podem ser instanciadas – como a classe *Esfera*, *Cone*, *Plano* – devem estender essa superclasse, acrescentar suas características, acrescentar alguns métodos e até substituir alguns dos definidos na superclasse, se for necessário.

O sistema deve conter uma lista de objetos que foram criados. Cada vez que o usuário cria um objeto, ele é colocado na lista. A ordem dos objetos da lista não importa.

Cada restrição é representada por uma classe. Cada classe que representa uma restrição deve se estender de uma superclasse *Restrição*. A classe *Restrição* é abstrata e contém uma lista de referências para os objetos pais, uma lista de referências para objetos filhos caso um dos objetos pais tenham mudado de posição. Cada classe que representa uma restrição – por exemplo, *PlanoPaPlano*, *RetaPePlano*, *PlanoTgEsfera* – deve implementar este método abstrato.

O sistema deve conter uma lista de objetos de restrição criados. Um objeto de

restrição é uma instância de uma das subclasses da classe *Restrição*. Cada vez que o usuário faz uma construção, um objeto de restrição é criado e colocado naquela lista.

Cada objeto deve conter uma lista de referências aos objetos de restrição em cujas construções ele está presente. Uma lista encadeada pode ser usada para isso. Considere uma construção em que um ponto P_1 fica sobre um plano Π. Um objeto de restrição C é criado pelo sistema e inserido na lista de objetos de restrição. Uma referência a C deve ser inserida na lista encadeada de Π e de P_1. Considere uma outra construção onde um ponto P_2 deve ficar sobre o plano Π. Um novo objeto de restrição C_2 é criado, colocado na lista de objetos de restrição e uma referência a C_2 é inserida no final da lista encadeada de Π e de P_2. Quando a posição de Π é mudada, Π percorre sua lista encadeada e avisa a todas as restrições (no caso C_1 e C_2) que ele mudou de posição. Como C_1 tem uma referência para Π e P_1, e C_2 tem uma referência para Π e P_2, então C_1 e C_2 executam o método para calcular as novas posições dos pontos P_1 e P_2. Quando o ponto P_1 é transladado, por exemplo, *este* percorre sua lista encadeada e avisa a todas as restrições (no caso C_1) que ele mudou de posição. C_1, então, executa o método para calcular a nova posição de P_1.

É importante notar que toda vez que o usuário cria uma restrição, a referência para o objeto de restrição que representa essa construção é colocada no final da lista encadeada de cada objeto que faz parte da construção. Dessa forma, quando a posição de um dos objetos da construção é mudada, o objeto percorre a lista encadeada que contém as restrições na ordem em que elas foram criadas.

2.6 Comparação entre as abordagens

Sistemas baseados em restrições:	***Sistemas baseados em construções:***
Emprega-se um método de programação para achar a posição dos objetos de toda construção.	O problema é dividido em pedaços menores, em restrições.

Sistemas baseados em restrições:	*Sistemas baseados em construções:*
As construções recaem em um sistema de equações não lineares que pode ser grande dependendo do número de objetos que estão dispostos na construção.	Emprega-se um mecanismo de atualização que é executado com o auxílio de uma estrutura de dados.
Em uma construção qualquer, podemos mudar a posição de qualquer objeto da mesma. A mudança da posição vai refletir em movimento dos outros objetos.	Existe uma relação de dependência entre objetos pais e filhos. Quando mudamos a posição de um dos objetos filhos, o objeto pai não muda de posição.
O grafo de dependências que representa a construção é não direcionado.	O grafo de dependências que representa a construção é direcionado.
Deve ser capaz de resolver problemas de ambigüidade da construção e problemas de construções com mais de uma construção.	Em toda construção cada restrição tem apenas uma solução para toda a construção.
Pode ser feito qualquer tipo de construção, até mesmo construções que resultam em configurações impossíveis.	Não há total liberdade para construir qualquer configuração.

Capítulo 3

Restrições

Neste capítulo serão discutidos dois tipos de problemas semelhantes à geometria dinâmica, mas oriundos de contextos diferentes: problemas de cinemática inversa e questões que envolvem restrições dinâmicas. Em problemas de cinemática inversa, estuda-se o comportamento de objetos sem massa, que estão conectados e que têm suas posições restritas. Em problemas de restrições dinâmicas estuda-se o comportamento de objetos com massa, que estão conectados e têm suas posições restritas. Neste tipo de problema, diferentemente dos da cinemática inversa, como os objetos têm massa, as forças envolvidas devem ser consideradas.

3.1 Cinemática Inversa

A parte da Mecânica que se ocupa do movimento, independentemente das causas, é a cinemática. Um corpo rígido é aquele no qual a distância entre quaisquer dois pontos que fazem parte do objeto não muda. Um corpo rígido ideal não existe, mas existem objetos que têm características físicas que se aproximam bastante de um corpo rígido ideal.

A combinação de corpos rígidos conectados forma um mecanismo ou uma máquina. Em ambos, o movimento relativo dos corpos rígidos conectados é bem definido. Em uma máquina, existe uma relação de força que é transmitida por seus componentes, ou seja, cada corpo rígido exerce trabalho sobre os demais e, assim, todos os corpos rígidos da máquina têm seu movimento definido. Portanto, em máquinas, a massa, as forças, o trabalho, a energia são levados em consideração no cálculo da posição das partes constituintes. Em um mecanismo, os corpos rígidos conectados são considerados sem massa; portanto, para o cálculo da posição de suas partes constituintes, não são levadas em conta relações de força ou energia, mas apenas relações de restrições espaciais.

A diferença entre uma máquina e um mecanismo está ilustrada na Fig. 3.1. Nesta podemos ver uma máquina (à esquerda) na qual a energia do motor é transformada em trabalho pelo cilindro. Um mecanismo análogo está representado à direita onde se observa o esqueleto da máquina. Neste caso, temos somente restrições de espaço para serem resolvidas e não equações que envolvam forças e massas.

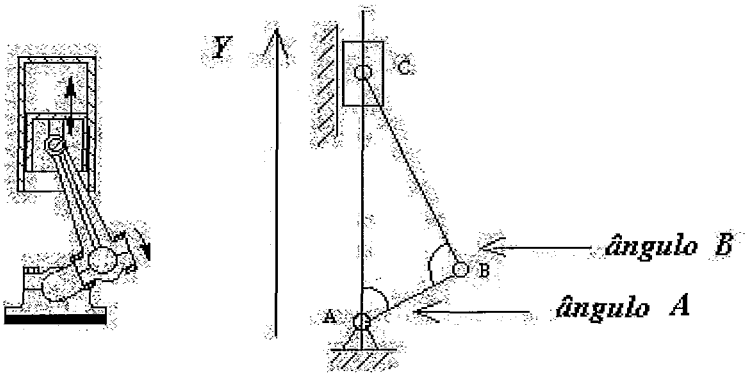


Figura 3.1: Diferença entre uma máquina (à esquerda) e um mecanismo (à direita).

Em computação gráfica, figuras articuladas são usadas para simular seres humanos [29,30,31,32,33,34], animais [35] e outras criaturas com pernas ou patas que aparecem em filmes ou em jogos. A animação dessas figuras pode ser feita com dados coletados da captura do movimento de um modelo. Porém existem ferramentas que utilizam a cinemática inversa para permitir que uma pessoa seja capaz de manipular diretamente a figura e com isso controlar da mesma forma a animação e o resultado da animação. Manipular uma figura complexa é uma tarefa bastante trabalhosa, pois os modelos das figuras podem conter centenas de graus de liberdade.

A cinemática inversa é uma técnica usada para simulação de estruturas articuladas. É bastante utilizada em animação por computador [29,35,36,37], simulação de mecanismos [38] e simulação de objetos articulados [35,39]. Há casos em que os objetos articulados têm um número muito grande de juntas, o que torna o objeto com muitos graus de liberdade [40]. Isso faz com que a pessoa que controla a animação do objeto articulado tenha dificuldade de monitorar a postura do objeto. Com o auxílio de um algoritmo de cinemática inversa podemos simplificar a manipulação dos objetos articulados, fixando a posição de certos pontos do objeto para que depois o algoritmo calcule a posição dos ângulos de certas juntas para satisfazer as restrições impostas [41]. Em robótica, a cinemática inversa geralmente é usada para o cálculo da posição de um

terminal (por exemplo uma mão, um braço) dadas algumas restrições impostas à posição de certas partes do robô [42, 43, 44]. A cinemática inversa também é usada em simulações de tarefas da vida real, nas quais se estuda o ser humano em um meio qualquer. Essa é uma tarefa que abrange o campo da ergonomia em que modelos de seres humanos são usados em simulações com o propósito de prever a reação do corpo humano quando ele se encontra em uma determinada situação. Por exemplo, pode-se estudar qual é a reação do corpo humano durante a batida de um carro.

3.1.1 Grau de liberdade de um corpo rígido

O número de graus de liberdade de um corpo rígido pode ser entendido, a grosso modo, como sendo o número de movimentos de execução possível a ele. No plano xy, o grau de liberdade de um corpo rígido livre é 3, pois ele pode transladar em x, também em y e pode girar. Já um corpo rígido livre no espaço tem grau de liberdade igual a 6, pois a sua posição pode ser obtida por uma sucessão de movimentos simples de seis tipos.

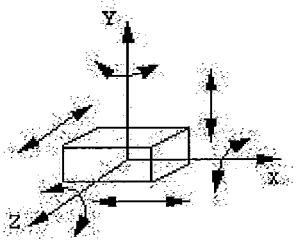


Figura 3.2: Grau de liberdade de um corpo rígido no espaço.

3.1.2 Restrições cinemáticas

O movimento de cada corpo rígido que faz parte de um mecanismo pode ter seu grau de liberdade diminuído devido à imposição de restrições cinemáticas.

Cada tipo de ligação entre as partes do mecanismo define uma restrição diferente. Vamos ver alguns tipos de restrições que podem ser impostas a um mecanismo.

3.1.2.1 Alguns tipos de restrições no plano

A restrição imposta a uma haste (veja a Fig. 3.3) diminui o seu grau de liberdade para 1, pois ela pode rodar somente em torno de um ponto (ponto A), não podendo transladar em x nem em y.

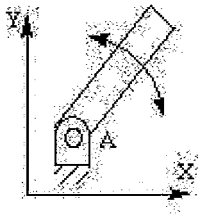


Figura 3.3: Restrição em que a haste só pode rodar em torno do ponto A em um plano.

Quando um corpo rígido é posto para deslizar sobre um plano, o seu grau de liberdade é de 1 porque ele só pode transladar em uma direção como mostra a Fig. 3.4.

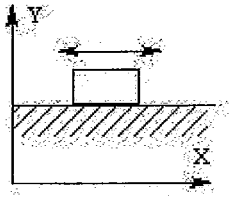


Figura 3.4: Corpo sobre um plano.

3.1.2.2 Alguns tipos de restrições no espaço

A restrição ilustrada na Fig. 3.5 mantém o centro de duas esferas fixas, impedindo qualquer tipo de translação. A haste que está presa à esfera mais interna só pode rodar em torno do seu eixo, enquanto que a haste que está presa à esfera externa pode girar em torno de seu eixo e do eixo da outra haste. Como podemos ver na Fig. 3.5, o mecanismo como um todo tem seu grau de liberdade reduzido a 3.

Figura 3.5: Restrição de duas esferas concêntricas.

Imagine uma haste presa a um paralelepípedo que está sobre uma superfície. O paralelepípedo deve ficar sobre a superfície. Neste caso, a haste pode transladar em duas direções (quando o paralelepípedo se move sobre a superfície). Como o paralelepípedo também pode girar em torno do eixo da haste, então o grau de liberdade do conjunto é 3.

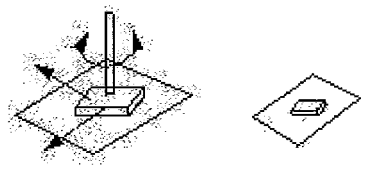


Figura 3.6: Conjunto plano-haste.

Imagine um cilindro deslizando ao longo de uma haste. Neste caso o cilindro pode transladar sobre a haste e girar em torno do eixo dela. Portanto, o grau de liberdade do mecanismo é 2.

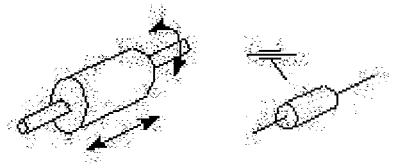


Figura 3.7: Cilindro sobre uma haste.

Imagine uma haste presa a um cilindro. O cilindro está sobre uma haste, não pode correr sobre ela, mas pode girar em torno do eixo. O grau de liberdade do mecanismo é 1, pois o único movimento que o cilindro pode fazer é girar em torno do eixo.

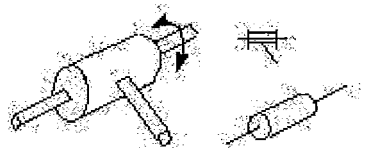


Figura 3.8: Uma haste presa a um cilindro. O cilindro contém uma haste presa.

O estado de um mecanismo é o conjunto de características que determina a posição das partes que o constituem. Este conjunto contém as variáveis independentes, as variáveis dependentes e as restrições cinemáticas impostas ao movimento do mecanismo. Por exemplo, considere o mecanismo ilustrado na Fig. 3.1. As restrições

impostas ao mecanismo são: o ponto A deve ser fixo, as hastes AB e BC estão ligadas de forma que somente um grau de liberdade de movimento seja permitido entre elas (o ângulo B entre elas pode mudar) e o cilindro só pode transladar na direção do eixo *Y*. A variável independente é o ângulo em A, no qual pode variar independentemente de qualquer outro parâmetro. As variáveis dependentes são o ângulo B e a posição Y do cilindro. O mecanismo como um todo tem apenas um grau de liberdade, pois tem somente uma variável independente.

3.1.3 Métodos para resolver problemas de cinemática inversa

Em geral, os métodos podem ser divididos em métodos numéricos e métodos analíticos. Os métodos analíticos calculam sempre todas as soluções possíveis para a configuração e podem ser divididos em duas categorias: os de formas fechadas e os baseados em eliminação algébrica. Em um método de forma fechada, o estado do mecanismo pode ser calculado com um conjunto de equações de forma fechada. Já um método baseado em eliminação algébrica, o estado do mecanismo é calculado com a solução de um sistema de equações polinomiais. Geralmente essas equações têm um grau maior do que 4, o que requer o emprego de subrotinas numéricas.

Existem alguns métodos numéricos para resolver um sistema de equações não lineares, sendo o de Newton-Raphson um dos mais conhecidos. Diferentemente dos métodos analíticos, os métodos numéricos convergem para uma solução interativamente depois de dada uma estimativa inicial para o estado do mecanismo. Nas próximas seções vamos discutir a idéia geral dos métodos numéricos e analíticos.

3.1.3.1 Métodos analíticos

Para configurações simples de manipuladores robóticos com poucos graus de liberdade, a solução da configuração pode ser achada com a resolução de um sistema de equações não lineares [45,46]. Manipuladores robóticos são geralmente projetados de forma a sempre terem uma solução analítica para o problema. É importante projetar o manipulador dessa forma, para permitir que simulações possam ter soluções calculadas em tempo real.

Os métodos analíticos são robustos, o que é vantajoso em animação por computador. Muitos pesquisadores têm estudado o caso de braços e pernas articuladas. Korein [47] desenvolveu uma solução analítica interessante para um braço com 7 graus de liberdade e Tolani et al [48] discutem técnicas similares. Não existe, porém, um método analítico para resolver estruturas articuladas em geral. Em tais casos, costuma-se empregar métodos numéricos.

3.1.3.2 Métodos numéricos

3.1.3.2.1 O Método da razão de movimento (motion rate)

Em seu trabalho pioneiro, Whitney [49] introduziu o método da razão de movimento no qual se baseiam métodos mais complexos. O método desenvolvido por Whitney tem o seguinte esquema:

- Dada uma configuração inicial do mecanismo, as equações não lineares que representam o mecanismo são linearizadas.
- Monta-se uma matriz Jacobiana para representar o sistema de equações diferenciais.
- O sistema de equações é resolvido e uma nova configuração mais próxima do objetivo é encontrada.
- O processo é repetido várias vezes, até que a solução encontrada seja considerada próxima o suficiente do objetivo final.

Este método é inspirado no de Newton-Raphson para a resolução de equações não lineares [50].

Novos métodos foram desenvolvidos de forma a explorar as redundâncias do problema. Whitney [49] usa o método descrito em [51,52] para minimizar o peso da norma da variação da coordenada das juntas. Deo et al. [53] propuseram usar a norma infinita para obter as menores magnitudes possíveis das variações das juntas. Liégeois [54] explorou o espaço nulo da matriz jacobiana.

3.1.3.2.2 O Método do Jacobiano transposto

Este método é discutido por Welman [55] e só difere do método da razão de movimento ao usar a transposta da matriz Jacobiana ao invés da sua inversa. Este método foi introduzido por Wolovich e Elliot [56] e aprimorado por Sciavicco e Siciliano [57] e por Das et al. [58].

Com este método, a interação pode ser feita com rapidez, pois dispensa operações de inversão de matriz. Entretanto, por causa de sua convergência ruim, principalmente perto da solução, o número de passos requeridos pode ser muito maior que no método da razão de movimento.

3.1.3.3 Comparação dos métodos

Métodos:	*Analítico*	*Numérico*
Cálculo da solução para o estado do mecanismo.	É mais rápido.	É mais lento pois requer várias interações do algoritmo.
Quanto às soluções:	Acha todas as soluções para o estado do mecanismo	Acha apenas uma solução depois de convergir
Configurações complexas do mecanismo:	Solução difícil de ser encontrada	É um método preferível em relação ao analítico.

Os métodos analíticos são mais rápidos que os métodos numéricos, porque estes requerem várias interações do algoritmo antes de se achar a solução para o estado do mecanismo. Além disso, os métodos algébricos acham todas as soluções para o estado do mecanismo enquanto que os numéricos apontam apenas uma solução depois de convergir. Em casos de mecanismos que estão mal posicionados, ou melhor, quando a solução é difícil de ser encontrada, os métodos numéricos são preferíveis.

3.2 Restrições Dinâmicas

Simulações físicas por computador têm sido usadas em vários campos como

animação[59, 60, 61], cálculo de forças de contato entre superfícies [62], simulação de objetos com massa articulados [63,64,65], e controle de câmera interativa [66]. O computador pode ser usado para simular eventos físicos em mundos virtuais [67], simular objetos articulados com massa e máquinas onde princípios físicos são diretamente utilizados [68].

Toda simulação física interativa requer bom desempenho. Uma simulação rápida faz com que a manipulação de um modelo pré-definido seja mais natural. A flexibilidade das ferramentas de simulação física é outro fator importante. Por exemplo, deve ser possível modificar dinamicamente os modelos pré-definidos quebrando e criando ligações entre as partes constituintes e adicionando novos modelos. Modelos simples podem ser ligados por meio de pinos e juntas, para formar modelos mais complexos. Outro aspecto importante é o que diz respeito ao realismo físico.

Assim como os sistemas de geometria dinâmica levam em conta restrições geométricas, uma ferramenta de simulação física requer a computação e a atualização de um sistema de restrições dinâmicas [59,69,70].

Para manter o realismo físico, o sistema baseado em restrições dinâmicas usado deve tratar com eficiência as restrições impostas às partes constituintes do modelo. Cada restrição é tratada como um conjunto de forças que atuam nos objetos que fazem parte do modelo. O sistema de restrições dinâmicas acrescenta forças extras para que a força resultante obedeça às restrições espaciais impostas aos objetos. Por exemplo, considere uma esfera sendo empurrada em uma calha inclinada. Se o usuário aplicar uma força que não esteja na direção da calha, o sistema baseado em restrições deve acrescentar uma força para que a força resultante fique na direção da calha e a esfera obedeça à restrição de se movimentar sobre a calha. A adição e a retirada de restrições espaciais dinamicamente em um sistema físico muda o sistema de equações. Com isso, quando alguma restrição espacial é retirada, as forças que mantinham a restrição devem ser mudadas para que a nova configuração seja mantida. Um meio interativo deve ser capaz de responder a essas mudanças automaticamente sem que muito tempo seja gasto para se formar e resolver as novas equações. Alguns métodos [102,103,104,105] foram desenvolvidos para formar e resolver as novas equações de forma eficiente, porém os modelos são restritos a corpos rígidos.. Um outro método de restrições dinâmicas foi descrito em [69] onde, além de corpos rígidos, curvas paramétricas também podem ser manipuladas e conectadas usando restrições. Este sistema também pode ser usado para

problemas não lineares como interpretação de imagens e *model fitting*, e em animação de personagens construídos pela conexão de peças elásticas. Uma extensão do uso de restrições dinâmicas foi descrita em [70] para ser usada com uma mistura de modelos contínuos e discretos, enfatizando os problemas de restrições de *layout* que aparecem em arquitetura, em circuitos elétricos e em configuração de páginas. Nestes problemas, os objetos podem ser postos em diversos lugares e ter suas dimensões alteradas. Mas a execução destas operações está sujeita a restrições que previnem interpenetração, que mantêm as relações de adjacência, e que definem limites de dimensão.

Em todas estas simulações de modelos físicos usando restrições dinâmicas, o usuário pode interagir diretamente com o modelo aplicando forças virtuais.

3.2.1 Posição de um corpo com massa no espaço

Quando uma força age diretamente sobre um corpo rígido, sua velocidade muda. Mas a força em si não é responsável por mudar diretamente a velocidade de um corpo, mas apenas sua aceleração (segunda lei de Newton). Para se saber a posição de um corpo com massa no espaço, deve ser discutido como uma força muda o estado de um objeto. Para isso devemos examinar as relações entre aceleração, força, velocidade e posição [68].

O vetor posição $\vec{r}$, o vetor velocidade $\vec{v}$ e o vetor aceleração $\vec{a}$ estão diretamente relacionados através de derivadas como mostra a Eq. 1.

$$\frac{d^2\vec{r}}{dt^2} = \frac{d\vec{v}}{dt} = \vec{a} \tag{1}$$

A Segunda Lei de Newton nos diz como uma força $\vec{F}$ afeta a aceleração de um corpo com massa m. Ela relaciona a força com a derivada da velocidade multiplicada pela massa. A massa multiplicada pela velocidade é chamada de momento linear ($\vec{p}$).

$$\vec{F} = \frac{d\vec{p}}{dt} = \frac{d(m\vec{v})}{dt} = m\vec{a} \tag{2}$$

Como a massa na grande maioria das simulações é considerada constante, então ela pode ser retirada da derivada e chegamos à famosa Segunda lei de Newton: $\vec{F} = m\ \vec{a}$. Se estivermos lidando com simulações de pontos que têm massa, essa fórmula relaciona a força com a aceleração de forma correta. Porém isso não tem muito valor quando se trata de corpos rígidos com a massa distribuída por uma área ou um volume. Para isso as fórmulas devem ser estendidas um pouco mais.

Um corpo rígido com massa pode ser visto como um conjunto de pontos que têm massa. O momento linear total $\vec{p}_t$ de um corpo rígido é a integral dos momentos de todos os pontos de massa como mostra a Eq. 3.

$$\vec{p}_t = \int \vec{v}(m).dm \tag{3}$$

A análise de um corpo rígido pode ser simplificada com o uso do conceito do centro de massa. O centro de massa é um vetor que é a combinação linear dos vetores posição de todos os pontos do corpo rígido multiplicados pelas suas massas, dividido pela massa total do corpo *M*. Com a definição do centro de massa *cm*, a Eq. 3 pode ser reescrita da seguinte forma:

$$M\vec{v}_{cm} = \frac{dM\vec{r}_{cm}}{dt} = \int \frac{d(\vec{r}(m))}{dt} dm = \int \vec{v}(m) dm = \vec{p}_t \tag{4}$$

A Eq. 4 nos mostra que a massa total do corpo rígido multiplicada pela velocidade do centro de massa ($\vec{v}_{cm}$) é igual ao momento total $\vec{p}_t$. Isso significa que não é preciso fazer o somatório da Eq. 4 para achar o momento se a massa total e a velocidade do centro de massa forem conhecidas.

Similarmente, a força resultante aplicada no corpo rígido é a derivada do momento linear total. O conceito de centro de massa, portanto, pode ser usado outra vez para simplificar a equação da força resultante como na Eq. 5.

$$\vec{F}_t = \frac{d(\vec{p}_t)}{dt} = M\frac{d(\vec{v}_{cm})}{dt} = M\vec{a}_{cm} \tag{5}$$

A Eq. 5 mostra que a força resultante ($\vec{F}_t$) que age sobre o corpo rígido pode ser considerada como uma força que age diretamente sobre o centro de massa. Dividindo a força resultante pela massa total chega-se à aceleração linear do centro de massa $\vec{a}_{cm}$. Se apenas a translação do corpo rígido for considerada, a posição desse corpo pode ser calculada integrando a velocidade, que por sua vez pode ser calculada integrando a aceleração linear.

Para considerar os efeitos de rotação, uma variável escalar deve ser definida. Uma variável escalar μ que representa a orientação de um corpo rígido em 2D pode ser definida. μ é o ângulo que um eixo fixo ao corpo rígido faz em relação ao eixo x contando no sentido anti-horário. A partir de μ, a velocidade angular ω pode ser definida como a derivada de μ em relação ao tempo, e a aceleração angular α como a derivada de ω em relação ao tempo. A relação pode ser vista na Eq. 6. Para se calcular a translação do corpo rígido, a aceleração linear do centro de massa deve ser calculada. A aceleração linear do centro de massa pode ser calculada dadas a força resultante e a massa total. Um raciocínio análogo pode ser feito para o cálculo da orientação do corpo rígido no espaço. A orientação pode ser achada integrando a velocidade angular ω em, relação ao tempo, que por sua vez é calculada integrando α em relação ao tempo.

Considere um ponto fixo o no corpo rígido o qual não sofre o efeito da rotação do corpo rígido. Pelo teorema de Charles, a velocidade de um outro ponto qualquer $\vec{v}_b$ no corpo rígido pode ser calculada da seguinte forma:

$$\vec{v}_b = \vec{v}_o + \vec{\omega r}_{\perp ob} \tag{6}$$

A velocidade de um ponto b qualquer é a velocidade do ponto o ($\vec{v}_o$) somado à velocidade angular ω do corpo rígido multiplicado por um vetor. Este vetor tem o mesmo módulo do vetor que tem origem em o e destino em b e é perpendicular a este mesmo vetor que tem origem em o e destino em b ($\vec{r}_{\perp ob}$). A velocidade de qualquer ponto no objeto pode ser calculada somando a velocidade linear do ponto o com a velocidade gerada pelo efeito da rotação.

A Eq. 7 abaixo diz respeito ao momento angular e é análoga à Eq. 2. O momento angular de um ponto b em relação a um ponto a é definido da seguinte forma:

$$L_{ab} = \vec{r}_{\perp ab} \bullet \vec{p}_b \tag{7}$$

O momento angular de um ponto b em relação a um ponto a (L_{ab}) é definido como o produto escalar do momento linear do ponto b com um vetor ($\vec{r}_{\perp ab}$). Este vetor tem o módulo igual a 1 e é perpendicular ao vetor que tem origem em a e destino em b.

Assim como a derivada do momento linear define uma força, a derivada do momento angular define o torque t_{ab}.

$$t_{ab} = \frac{dL_{ab}}{dt} = \frac{d(\vec{r}_{\perp ab} . \vec{p}_b)}{dt} = \vec{r}_{\perp ab} . m\vec{a}_b = \vec{r}_{\perp ab} . \vec{F}_b \tag{8}$$

O torque quantifica quanto uma força aplicada a um ponto é capaz de mudar a orientação de um objeto no espaço em relação a outro ponto. O objetivo é chegar a uma relação tal que seja possível calcular a aceleração angular dadas as forças aplicadas ao objeto. A orientação (posição) do objeto no espaço é calculada integrando-se a aceleração angular duas vezes.

O momento angular total em torno de um ponto A é definido da seguinte forma:

$$L_{at} = \int_i \vec{r}_{\perp ai} . \vec{p}_i = \int_i \vec{r}_{\perp ai} . m_i . \vec{v}_i \tag{9}$$

O momento angular total em torno do ponto a (L_{at}) é a integral do momento angular $\vec{p}_i$ de todos os pontos calculados em relação ao ponto a. O cálculo pode ser simplificado com o uso do momento de Inércia, da mesma forma que o centro de massa simplifica o cálculo do momento linear. A Eq. 6 mostra como a velocidade de um ponto b pode ser calculada usando um ponto de referência fixo no espaço, um ponto o que não sofre efeitos da rotação do objeto e a velocidade angular do objeto. Se o ponto de referência for mudado para o ponto o então apenas o termo que contém a velocidade angular aparece. A Eq. 9 pode ser reformulada usando a Eq. 6 para se chegar ao momento de Inércia. Se o ponto a da Eq. 9 for tratado como o ponto de referência da Eq. 6 e o índice i da Eq. 9 como o ponto b da Eq. 6, a Eq. 9 pode ser escrita da seguinte

forma:

$$L_{at} = \int_i \vec{r}_{\hat{a}i} . m_i . \varpi . \vec{r}_{\perp ai} = \varpi \int_i m_i \vec{r}_{\perp ai} . \vec{r}_{\perp \hat{a}i} = \varpi \int_i m_i (\vec{r}_{\perp ai})^2 = \varpi I_a \quad (10)$$

O primeiro passo foi escrever o momento angular em termos da velocidade angular $m_i . \varpi . \vec{r}_{\perp ai}$. O segundo passo foi tirar ω da integral, pois é igual para todos os pontos do corpo rígido. Resta então um produto escalar do vetor que liga o ponto *a* ao ponto *i* por ele mesmo $\vec{r}_{\perp ai} . \vec{r}_{\perp \hat{a}i}$. Isto é o quadrado do módulo do vetor. A integral da massa multiplicada pelo módulo deste vetor ao quadrado é o momento de inércia do corpo rígido 2D em relação ao ponto *a* (I_a). O momento de inércia indica o quanto é difícil rodar um corpo em torno de um ponto. O torque é igual ao momento angular total do corpo rígido t_{ar}. O momento angular total do corpo rígido é igual ao momento de inércia multiplicado pela aceleração angular como mostra a Eq. 11.

$$t_{at} = \frac{dL_{at}}{dt} = \frac{d(I_a \varpi)}{dt} = I_a \frac{d(\varpi)}{dt} = I_a \alpha \quad (11)$$

A Eq. 11 relaciona o torque total sobre o corpo e a aceleração angular. Um raciocínio análogo ao cálculo da translação pode ser feito para calcular a orientação do corpo rígido. A posição do corpo é calculada integrando-se a aceleração angular duas vezes em relação ao tempo. Para calcular a aceleração angular basta ter o torque total que age sobre o corpo e o momento de inércia em relação a um ponto.

3.2.2 Restrições Dinâmicas

Na seção anterior foi discutido como forças aplicadas a um corpo com massa mudam sua posição no espaço. Consideremos agora o seguinte exemplo: Um cilindro que está encaixado a uma haste como na Fig. 3.6.

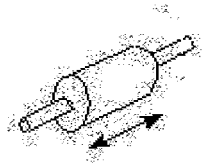


Figura 3.6: Cilindro deslizando ao longo de uma haste.

Neste caso será considerada apenas a translação do cilindro sobre a haste, ou seja, somente a Segunda Lei de Newton será levada em conta. A Segunda Lei de Newton dá a relação entre as forças que agem sobre o cilindro e a sua aceleração. A força e a aceleração têm a mesma direção. Mas a restrição espacial do cilindro sobre a haste implica que o cilindro nunca deve se mover em uma direção que seja diferente da direção da haste qualquer que seja a força aplicada ao cilindro. Isso implica que a aceleração deve ter a mesma direção da haste e, em conseqüência, a força resultante sobre o cilindro deve ter também essa direção independentemente da direção da força virtual $\vec{f}_v$ que o usuário está aplicando ao cilindro. Conseqüentemente, uma força de restrição $\vec{f}_c$ deve ser adicionada pelo sistema de restrições dinâmicas de tal forma que a força resultante seja:

$$\vec{F} = \vec{f}_v + \vec{f}_c = kt \tag{12}$$

onde t é um vetor tangente a haste e k é um escalar. As restrições dinâmicas são um sistema que faz com que o comportamento de um objeto seja consistente (atendendo as restrições de movimento) aplicando forças de restrição $\vec{f}_c$. [69,71,72,73,74,75,76,77] discutem mais a fundo a matemática envolvida nas restrições dinâmicas. Basicamente eles discorrem sobre métodos para a resolução do sistema de equações que governa as restrições espaciais. As equações desses sistemas em geral contêm as forças de restrição, as forças virtuais, a aceleração linear e angular dos objetos e outras constantes como massa e momento de inércia.

3.2.3 Comparando problemas de cinemática inversa, geometria dinâmica e restrições dinâmicas

	Geometria dinâmica	*Cinemática Inversa*	*Restrições dinâmicas*
Contexto	Construções que envolvem apenas objetos geométricos como Cones, Esferas, Cilindros e etc.	Problemas de atuação em vários campos sendo especialmente aplicadas na simulação de figuras articuladas	Problemas em que princípios físicos são diretamente utilizados
Restrições	Envolvem relações apenas geométricas como perpendicularidade, paralelismo, etc.	Restrições de juntas e diminuição do grau de liberdade do objeto em questão	As restrições podem ser de juntas, de interpenetração, limite de potência e etc.
Objetos	Sem massa	Com massa	Sem massa
Cálculo da posição dos objetos constituintes	Os baseados em restrições utilizam métodos numéricos e analíticos. Os baseados em construções dividem o problema em pedaços menores não sendo necessário um método de resolução.	Utilizam métodos numéricos e analíticos.	Utilizam alguns métodos especiais mais detalhados em [100, 106, 107, 108, 109, 110, 111, 112] que utilizam equações diferenciais de primeira e de segunda ordem.

A principal diferença entre um problema de geometria dinâmica e um problema

de cinemática inversa ou de restrições dinâmicas é o contexto. Uma construção de geometria dinâmica lida com problemas que envolvem apenas objetos geométricos como cones, esferas, planos, retas, pontos. As restrições são puramente geométricas e têm um fundamento matemático como, por exemplo, tangências, interseções, paralelismo, perpendicularidade. Já a cinemática inversa lida com problemas em vários campos sendo especialmente aplicada na simulação de figuras articuladas. As restrições neste tipo de problema são restrições de juntas e que fazem com que o objeto articulado em questão tenha uma diminuição do número de graus de liberdade do seu movimento. Tanto em geometria dinâmica quanto em cinemática inversa, os objetos envolvidos são considerados sem massa. Isso faz com que o problema envolva restrições do movimento sem levar em conta forças, trabalhos ou potências. Já as restrições dinâmicas são encontradas em problemas nos quais a massa dos objetos envolvidos é considerada. As forças devem ser consideradas no cálculo da posição dos objetos. As restrições dinâmicas podem ser empregadas em problemas em que princípios físicos são diretamente utilizados. Assim como em cinemática inversa, as restrições neste tipo de problema podem ser de vários tipos: de juntas, de interpenetração, de limite de potência de uma máquina e várias outras.

Quanto aos métodos usados para solucionar esses problemas, podemos destacar os métodos de resolução global de sistemas de equações. No caso dos problemas de cinemática e dos sistemas de geometria dinâmica baseados em restrições, são usados métodos numéricos ou analíticos. No caso dos sistemas de restrições dinâmicas, são usados sistemas de equações que envolvem derivadas de primeira e de segunda ordem. No caso de sistemas de geometria dinâmica baseados em construções, o problema é dividido em pedaços menores, em restrições locais, não sendo necessária a utilização de resolução global de sistemas de equações.

Capítulo 4

Interface Gráfica

A interface com o usuário é um dos pontos mais importantes que devem ser levados em conta no desenvolvimento de um sistema de geometria dinâmica. Uma interface gráfica bem planejada e desenvolvida torna mais fácil e dinâmico o entendimento dos teoremas e o aprendizado de Geometria Espacial.

4.1 Visões

Uma visão é simplesmente uma janela em que parte da cena pode ser vista. Quando o estado de uma construção é mudado em uma das visões, a mudança é refletida nas outras. Com várias visões, é permitido interagir de diversas maneiras com os objetos da construção e visualizar o resultado por diferentes pontos de vista.

A interface gráfica que representa os objetos da cena foi dividida em quatro visões como mostra a Fig. 4.1. Todas as visões possuem grades. O uso de grades [78] faz com que o espaço onde os objetos se situam tenha um referencial, dando uma noção visual de espaço. Quando o usuário tenta mudar a posição de um objeto que faz, ou não, parte de uma construção, a grade ajuda a posicioná-lo dando uma noção de direção, profundidade e sentido. As três grades foram postas nas posições x=-5, y=-5 e z=-5. As linhas das grades possuem um espaçamento igual a 1 entre elas e são desenhadas de modo a formar um plano gradeado que vai da posição –5 até 5. As quatro visões são arranjadas em uma janela, a saber:

- Canto superior esquerdo: visão perspectiva da cena.
- Canto superior direito: visão projetiva (*visãoXY*) em que a cena é vista de frente ao plano XY em vermelho, segundo a direção (0,0,-1) e de forma que o vetor que dá sentido vertical ao desenho é o (0,1,0).

- Canto inferior esquerdo: visão projetiva (*visãoXZ*) em que a cena é vista de frente ao plano XZ em azul, segundo a direção (0,-1,0) e de forma que o vetor que dá sentido vertical para cima ao desenho é (1,0,0).
- Canto inferior direito: visão projetiva (*visãoYZ*) em que a cena é vista de frente ao plano YZ em verde, segundo a direção (1,0,0) e de forma que o vetor que dá sentido vertical ao desenho é (0,1,0).

As três vistas projetivas podem ser ajustadas quanto à aproximação. Nelas, o usuário tem liberdade para definir a matriz de transformação da visão perspectiva e a transformação de projeção [79] da câmera que define a visão perspectiva. O ajuste da visão da câmera se dá por meio do arraste de 10 pontos notáveis: (P1 a P10). Conforme mostrado na Figura 4.1, estes pontos controlam:

- A abertura da lente: P_4 e P_5.
- O centro da câmera: P_1, P_6, P_8,
- O vetor up da câmera: P_2 e P_{10}
- E o olho da câmera: P_3, P_7 e P_9.

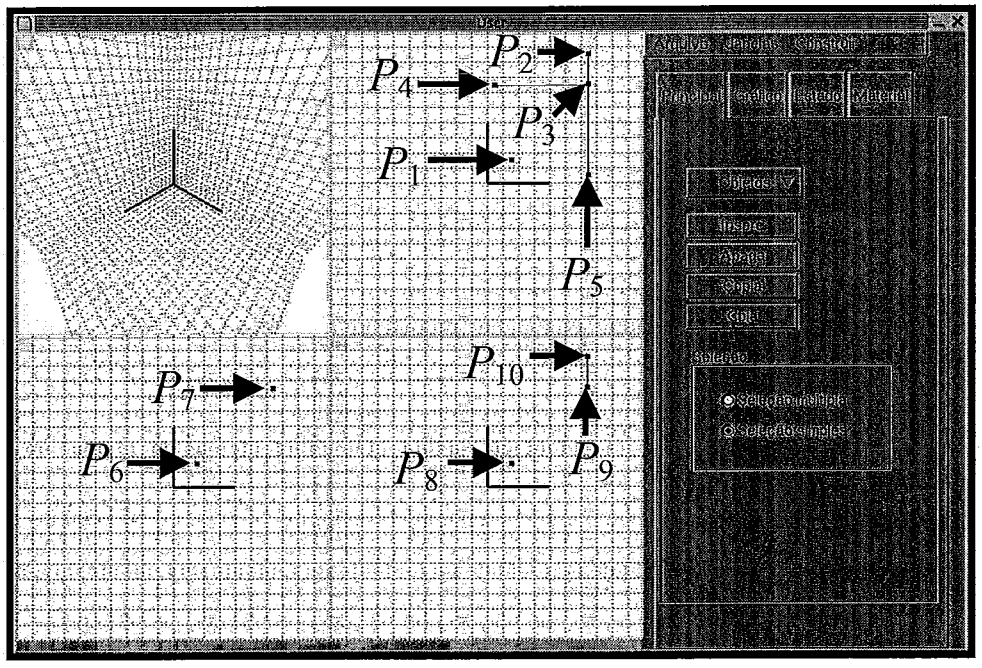
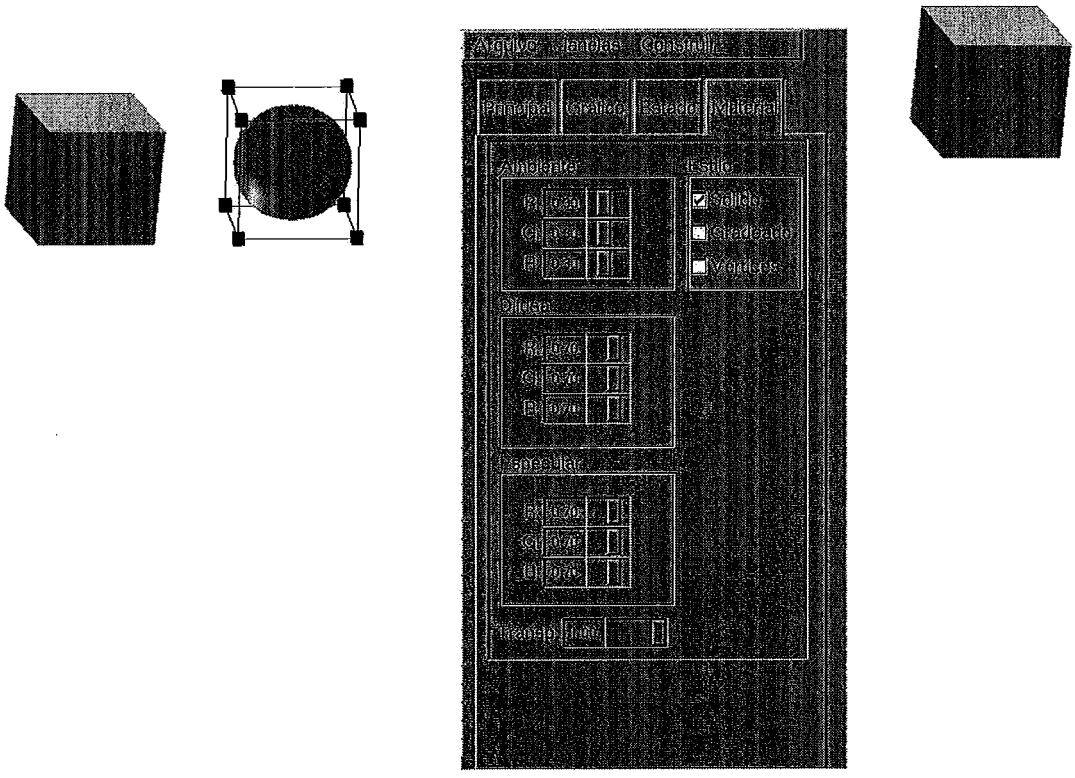


Figura 4.1: Interface Gráfica que representa os objetos da cena.

4.2 Escondendo objetos

Quando uma construção é feita, nem sempre todos os objetos que dela fazem parte são relevantes para a observação da dinâmica da construção. Para facilitar a observação dessa dinâmica, o sistema permite que objetos não relevantes sejam escondidos. Isto é feito ajustando a propriedade de transparência dos objetos em questão, como se vê na Fig. 4.2.



(a) (b)

Figura 4.2: Alguns objetos de uma construção que não são relevantes para a observação da construção podem ser escondidos. 4.2a: Seleção do objeto. 4.2b: Objeto é escondido colocando transparência sobre o mesmo.

4.3 Interação com objetos

Um processo de interação consiste em toda ação executada pelo usuário que é relevante para o sistema. No Mangaba, os principais processos de interação são executados através do *mouse*. Um exemplo de processo de interação é o arraste do

cursor, que pode resultar na translação, rotação ou escala do objeto selecionado. Um outro exemplo é o clique do *mouse* sobre um objeto que resulta na seleção do mesmo.

4.3.1 Seleção dos objetos

Uma técnica que facilita a seleção do objeto é o monitoramento constante do movimento do cursor. Quando o cursor está posicionado de forma que um objeto X pode ser selecionado com um clique do *mouse*, este objeto X é desenhado de forma diferente (ressaltado) para que, de antemão, se saiba que ele vai ser selecionado se o *mouse* for clicado.

No Mangaba, o objeto é ressaltado acrescentando à sua representação o hallo (Fig. 4.3), que consiste em uma coroa envoltória ao objeto que aparece (esfera à esquerda da Fig. 4.3).

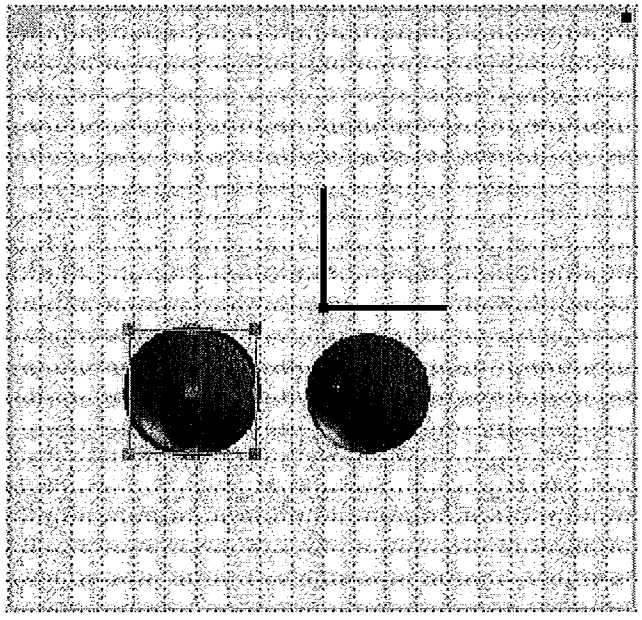


Figura 4.3: Uma esfera com o hallo e outra sem o hallo.

No Mangaba cliques consecutivos do *mouse* com o cursor sobre o mesmo *pixel* alternam os objetos selecionados. Por exemplo, considere o caso mostrado na Fig. 4.4 onde três objetos são interceptados pela reta imaginária.

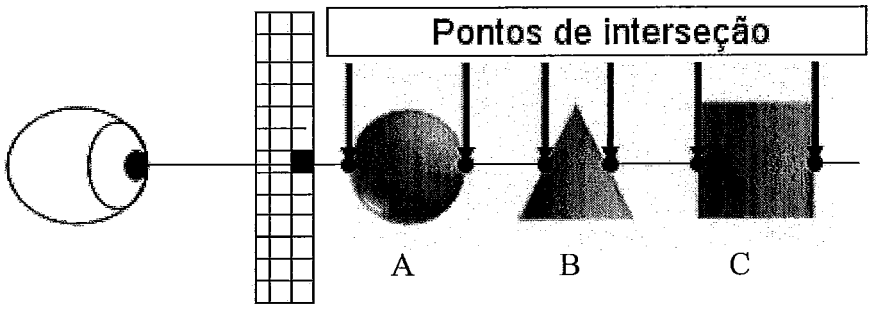


Figura 4.4 Seleção de objetos. Reta imaginária que passa pelo observador e pelo *pixel* onde o cursor estava quando o *mouse* foi clicado. A reta deve ser considerada para a seleção de objetos.

O primeiro clique seleciona A, o segundo B, e o terceiro C. Um quarto clique seleciona A novamente.

Em um sistema de geometria dinâmica espacial, objetos de diferentes dimensões podem ser criados. Quando objetos de menor dimensão se sobrepõem a objetos de maior dimensão, como nas construções *PontoSoPlano*, *PontoSoEsfera*, *PontoSoReta*, *PontoSoCilindro*, *PontoSoCone*, *PontoSoCirculo*, *PontoSoPlano*, *CirculoSoPlano*, *CirculoSoEsfera* é dada prioridade à seleção dos objetos de menor dimensão para posterior manipulação.

4.3.2 Manipulação de objetos

Em ambientes bidimensionais a percepção da manipulação de um objeto é clara, pois quando o *mouse* é arrastado em uma direção, só existe uma posição (coordenada x e y) para onde o objeto selecionado pode ser transladado. Já em um ambiente tridimensional, quando o *mouse* é arrastado, existe uma infinidade de lugares possíveis para onde o objeto selecionado pode ser transladado já que em qualquer visão existe uma direção (perpendicular à visão) para a qual o objeto possa ser transladado e continue sendo desenhado na mesma posição na visão.

No Mangaba, a manipulação de um objeto é feita nas visões projetivas. A manipulação feita desta forma é mais simples de ser desenvolvida e usada, pois apenas duas das três coordenadas são mudadas cada vez que o *mouse* é arrastado.

4.3.3 Translação de objetos

Nas três visões projetivas, duas das três coordenadas da translação do objeto são modificadas por vez. Por exemplo, ao executar o processo sobre *visãoXY*, as coordenadas x e y do centro do objeto vão ser modificadas. Antes de o processo de translação ser feito, o processo de seleção de um objeto é executado. Para transladar o objeto selecionado, o botão esquerdo do *mouse* deve ser mantido pressionado. Ao arrastar o *mouse* deixando o botão esquerdo pressionado, o centro do objeto é transladado para a posição do cursor. O processo de translação acaba quando o botão esquerdo do *mouse* é largado. Os estados do processo de translação são os seguintes:

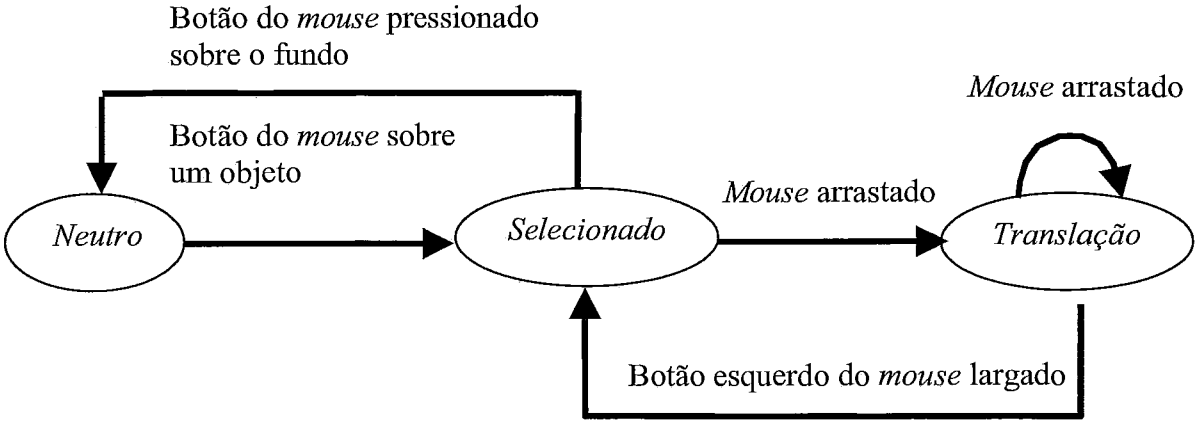


Figura. 4.5 Diagrama de estados do processo de translação.

Translação: O objeto selecionado está sendo transladado.
Selecionado: Um ou mais objetos estão selecionados.
Neutro: Estado em que não há objeto selecionado.

Se o objeto fizer parte de uma construção, a coordenada z é calculada de forma que o objeto obedeça às restrições impostas.

4.3.4 Rotação de objetos

Em aplicações tridimensionais, a orientação dos objetos é freqüentemente [80] ajustada com o auxílio de manipuladores atrelados a um ou mais objetos. Um manipulador é um controle com o qual o usuário pode efetuar interações, tipicamente com o auxílio do *mouse*. Manipuladores de rotação são usualmente implementados segundo um dos paradigmas apresentados a seguir:

4.3.4.1 Ângulos de Euler

A rotação [80] é feita multiplicando todos os vértices dos objetos por três matrizes. Cada matriz representa a rotação em torno de um dos eixos (*x, y* e *z*).

Por exemplo, para desenhar um objeto com essa técnica usando OpenGL, os seguintes comandos podem ser usados para representar a rotação na ordem *x, y* e *z*:

```
glRotatef(angulox, 1.0, 0.0, 0.0);
glRotatef(anguloy, 0.0, 1.0, 0.0);
glRotatef(anguloz, 0.0, 0.0, 1.0);
Desenha_o_Objeto();
```

Neste exemplo as informações de que o sistema precisa são os valores dos ângulos *angulox*, *anguloy* e *anguloz*. glRotatef() é um método que monta uma matriz que representa a rotação em torno do eixo especificado e com um determinado ângulo. O primeiro argumento do método glRotatef() é o ângulo e os três últimos parâmetros indicam o vetor em torno do qual a rotação será executada.

Vantagens:

- O usuário está familiarizado com a rotação em torno dos eixos cartesianos

Desvantagens:

- Uma rotação de valor +10, +30, +45 dada não é desfeita com a aplicação de uma rotação de valores –10, -30, -45.
- Aplicar uma rotação de A graus em torno do eixo cartesiano *z* depois de aplicada uma rotação de 90 graus em torno do eixo cartesiano *x*, equivale a aplicar uma rotação de A graus em torno do eixo cartesiano *y*. O mais

esperado era a aplicação de uma transformação em torno do eixo cartesiano *z* de A graus. Essa rotação é não intuitiva.

4.3.4.2 Vetor e ângulo

Essa técnica [80] foi popularizada pelo OpenGL que usa um ângulo e um vetor como argumentos do método glRotatef(). Antes de o objeto ser desenhado, a posição de todos os seus vértices é multiplicada por uma matriz que representa a rotação de um ângulo (entre 0 e 360) em torno de um eixo unitário.

O vetor e o ângulo podem ser modificados diretamente interagindo com os manipuladores como na Fig. 4.6.

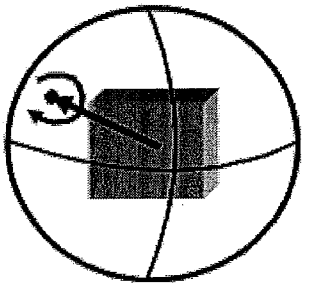


Figura 4.6: Manipuladores usando a técnica vetor e ângulo.

Vantagens:

- Qualquer ponto de visão pode ser alcançado.

Desvantagens:

- Dados dois objetos diferentes em posições diferentes não é intuitivo e fácil escolher dois vetores diferentes para alcançar a mesma posição de visão dos objetos.
- Não é intuitiva.

4.3.4.3 Quaternions / ArcBall

Quaternions [80] permitem descrever a rotação de um objeto com quatro parâmetros. A implementação mais simples de quaternions foi descrita por Ken

Shoemake [81] e é conhecida como ArcBall. Com essa técnica, a posição do objeto é mudada imaginando que o objeto é envolvido por uma esfera e que, ao selecionar e arrastá-la, o objeto é girado acompanhando o movimento da esfera.

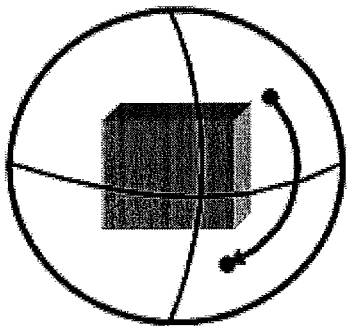


Figura 4.7: ArcBall

Vantagens:

- É uma interface intuitiva e qualquer ponto de visão pode ser alcançado.

Desvantagens:

- O Arcball é um paradigma que requer quatro parâmetros para o cálculo da posição do objeto. Não é intuitivo entrar com quatro parâmetros pela interface e conseguir posicionar o objeto precisamente.

4.3.4.4 Implementação

O paradigma usado para a manipulação dos objetos é o de ângulos de Euler. Apesar de ser o paradigma com o maior número de cálculos, com ele é possível elaborar um manipulador com que o usuário pode indicar a posição exata do objeto no espaço.

Para cada objeto é definido um eixo de referência. Para um cilindro e um cone, o eixo é uma reta perpendicular ao círculo da base e que passa pelo centro deste mesmo círculo. Para um plano, o eixo é uma reta perpendicular ao plano e que passa pelo centro do plano (o plano é definido como um quadrado finito). Para uma reta, o eixo é coincidente com esta reta (veja Fig. 4.8).

A rotação de objetos geométricos é feita com o auxílio de um manipulador de rotação (Fig. 4.8), que permite ao usuário posicionar o objeto modificando a posição do

seu eixo de referência e girando-o em torno do eixo. Juntamente com a grade, o manipulador dá uma noção visual da orientação do objeto. Considere a construção *RetaPePlano*. O manipulador do plano pode ser posicionado facilmente de forma que o eixo *y* do sistema de coordenadas local ao plano fique na direção (1,1,1), deixando, em conseqüência, a reta nessa direção. Se o botão direito do *mouse* for pressionado quando o cursor estiver sobre o manipulador de rotação, o objeto pode ser rodado de forma a modificar a posição do seu eixo de referência (Fig. 4.9 A e B). Em caso contrário, se o botão direito do *mouse* for pressionado e arrastado quando o cursor estiver sobre o objeto, este é rodado em torno do seu eixo de referência.

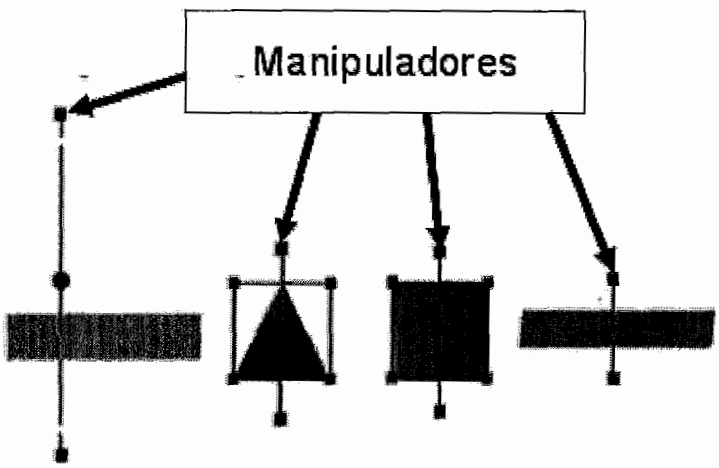


Figura 4.8: Manipuladores de rotação usados no Mangaba.

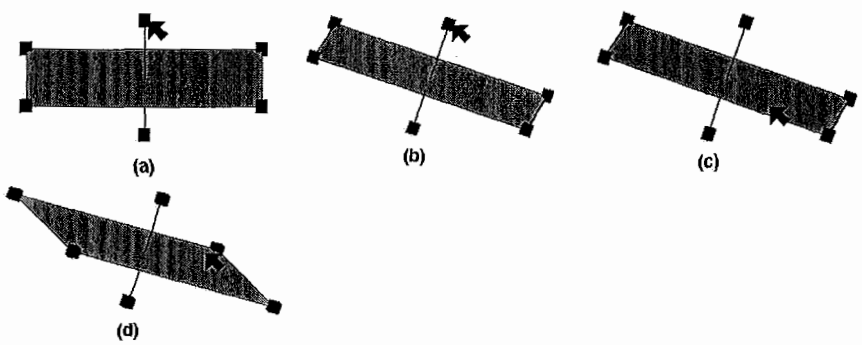
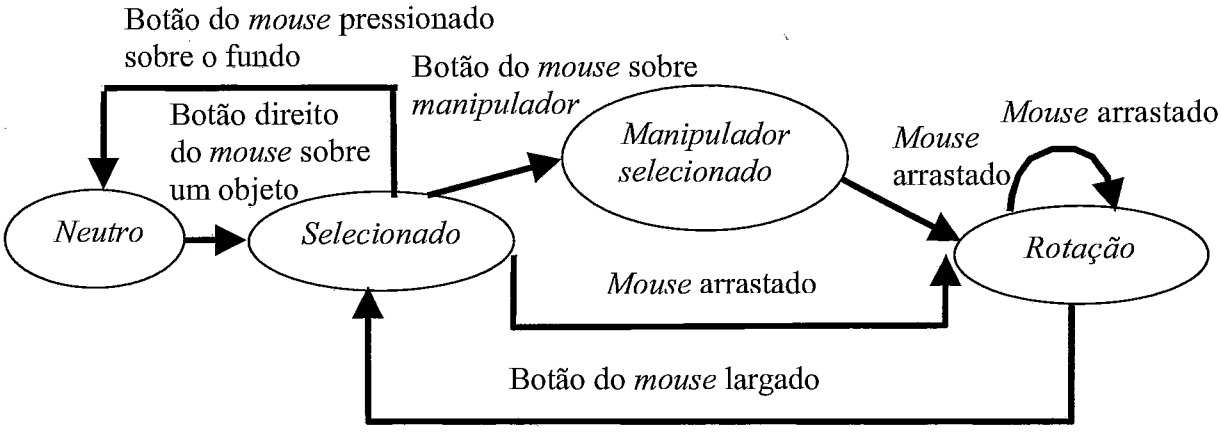


Figura 4.9: Figura que representa a rotação de um objeto por meio de um manipulador.

A Fig. 4.10 mostra o diagrama de estados para a rotação dos objetos no Mangaba:



Neutro: Estado em que não há objeto selecionado.

Rotação: O objeto selecionado está sendo rodado.

Selecionado: Um ou mais objetos estão selecionados.

Manipulador selecionado: O manipulador do objeto selecionado está selecionado.

Figura. 4.10: Diagrama do processo de rotação no Mangaba.

4.3.5 Escala de objetos

A escala geralmente é feita imaginando um sistema de coordenadas fixo (preso) ao objeto. Desta forma, quando o objeto é escalado de dois em relação ao eixo x, está subentendido que é o eixo x do sistema de coordenadas local ao objeto e com isso o objeto não é deformado pela escala.

Para que o objeto possa atingir todas as posições na cena e não ser deformado (não ser contorcido, mas apenas escalado em relação aos três eixos do sistema de coordenadas local ao objeto), as transformações devem ser sempre dadas na seguinte ordem: escala, rotação e translação. No contexto de geometria dinâmica espacial, a deformação de objetos não é um tema de estudo. Por esta razão, as transformações no Mangaba são dadas sempre nesta ordem.

A escala de objetos, assim como a rotação de objetos discutida na seção 4.3.4, é feita com o auxílio de manipuladores.

4.3.5.1 Escala através dos vértices

O Mangaba e algumas ferramentas de CAD possibilitam que a escala de um objeto seja feita igualmente em relação aos três eixos (x, y e z) do sistema de coordenadas locais ao objeto com o uso de um manipulador. Em geral o manipulador usado para este fim são os vértices de um cubo que envolve o objeto. Ao selecionar e arrastar um dos vértices do cubo envolvente, o objeto é escalado igualmente em relação aos eixos x, y e z fixos ao objeto de um valor tal que o vértice que está sendo arrastado continua sempre na posição do cursor como mostra a Fig. 4.10. Este tipo de escala pode ser feito tanto em uma vista perspectiva quanto em uma vista projetiva, pois em ambos os casos só existe um valor de escala que faz com que o vértice, sendo arrastado, fique na posição do cursor.

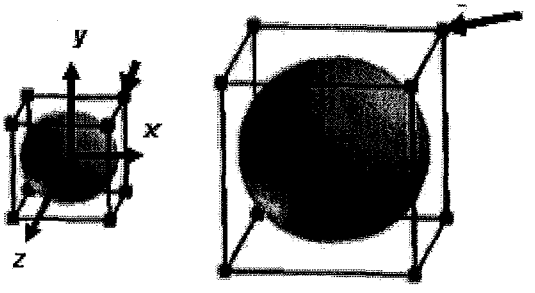


Figura 4.10: Escala a partir de um dos vértices do cubo envolvente ao objeto.

4.3.5.2 Escala através das arestas

Nem sempre se deseja escalar um objeto igualmente nas três direções dos eixos de coordenadas local ao objeto. Um sistema de geometria dinâmica deve possibilitar que a escala do objeto em relação aos eixos do sistema de coordenadas local ao objeto possa ser feita de forma desigual. Veja o exemplo da Fig. 4.11.

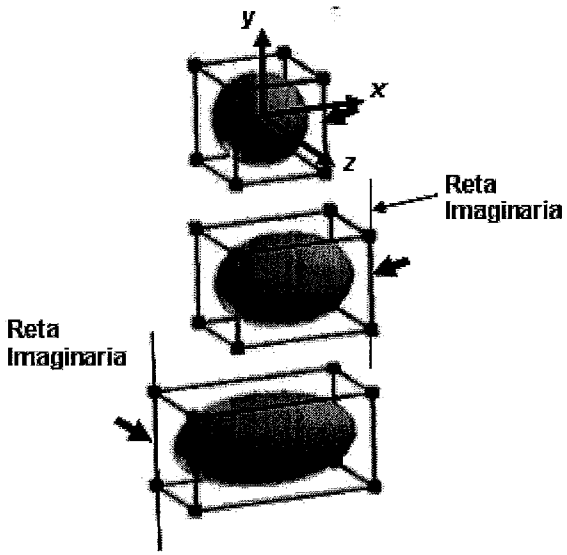


Figura: 4.11 Escala através das arestas de um cubo gradeado envolvente ao objeto.

Ao selecionar e arrastar uma das arestas do cubo envolvente, o objeto é escalado em relação a dois dos três eixos x, y e z do sistema de coordenadas local ao objeto de um valor tal que a reta imaginária que passa pela aresta que está sendo arrastada continue sempre passando pelo cursor. Os dois eixos que são escolhidos para a escala são os perpendiculares à aresta. Esta escala pode ser feita de duas formas como mostra a Fig. 4.11. Na forma 1, o objeto é escalado de forma que apenas a face do cubo envolvente no sentido positivo do eixo x do sistema de coordenadas local ao objeto muda de posição. Na forma 2, o objeto é escalado de forma que as faces do cubo envolvente no sentido positivo e negativo do eixo x do sistema de coordenadas local ao objeto mudam de posição. Na forma 2, o objeto é escalado por um valor maior. Este tipo de escala também pode ser feito tanto em uma vista perspectiva quanto em projetiva, pois em ambos os casos só existe um valor de escala que faz com que a reta imaginária que passa pela aresta arrastada passe pelo cursor.

4.3.5.3 Implementação

O Mangaba, além da escala através dos vértices do cubo envolvente, permite que um objeto seja escalado na direção de um dos eixos por vez, e a escala é feita como descrito na seção anterior. O manipulador usado para este fim são pontos que estão no centro da face do cubo que envolve o objeto. Veja a Fig. 4.12.

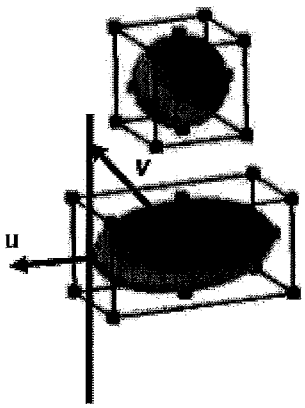
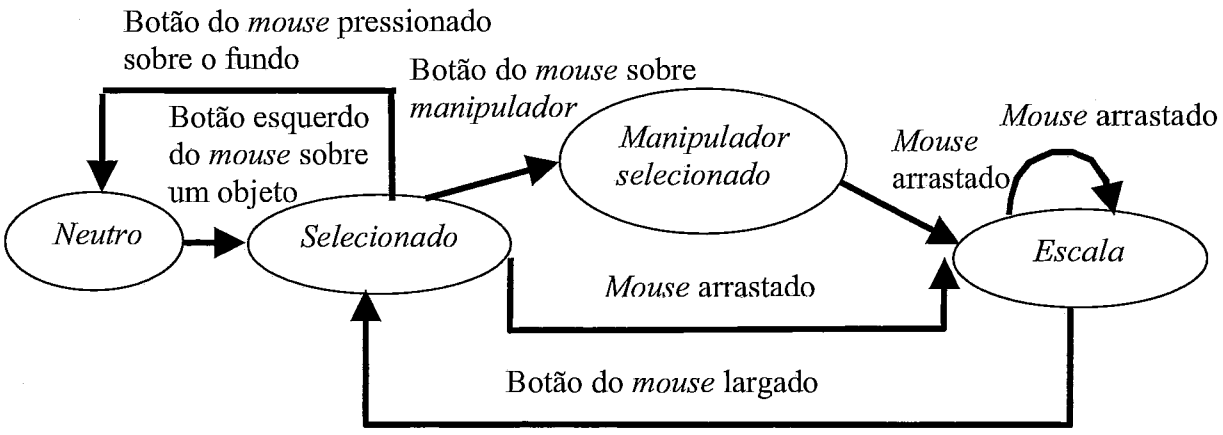


Figura. 4.12: Escala no Mangaba.

Ao selecionar e arrastar um dos pontos manipulador, o objeto é escalado em relação a um dos eixos x, y ou z do sistema de coordenadas local ao objeto. O eixo deve ser o que passa pelo centro do objeto e pelo ponto manipulador que está sendo arrastado. O valor da escala é calculado da seguinte forma: achar um vetor V, que é o resultado da diferença das coordenadas do cursor e do centro do objeto. O valor da escala é o da projeção de V sobre o vetor unitário que tem o sentido do eixo que passa pelo centro do objeto e pelo ponto manipulador que está sendo arrastado. A Fig. 4.13 mostra o diagrama de estados da dilatação de objetos.



Manipulador selecionado: O manipulador do objeto selecionado está selecionado.

Neutro: Estado em que não há objeto selecionado.

Escala: O objeto selecionado está sendo escalado.

Selecionado: Um ou mais objetos estão selecionados.

Figura 4.13: Diagrama de estados do processo de escala de um objeto no Mangaba

Capítulo 5

Sistema de Atualização

Este capítulo descreve o sistema que encapsula a metodologia empregada pelo Mangaba para efetuar a atualização dos objetos dispostos na cena em resposta às interações do usuário.

Como descrito no Capítulo 2, existem dois tipos de sistemas de atualização: os baseados em construções e os baseados em restrições. O sistema de atualização do Mangaba pode ser considerado um sistema baseado em construções, já que não requer a solução de sistemas de equações polinomiais não lineares como os sistemas baseados em restrições. Nestes últimos, construções complexas podem acarretar problemas de ambigüidade que são difíceis de serem solucionados. No Mangaba, a estrutura de dados utilizada para a atualização da posição dos objetos é uma implementação do grafo de dependência de objetos simples (veja Seção 2.5.2).

5.1 Arquitetura

O sistema é dividido em um modulo principal (*Interface*) e três módulos gerenciadores denominados Gerenciador de Processos (*GP*), Gerenciador Gráfico (*GG*) e Gerenciador de *Mouse* (*GM*) respectivamente. A Fig. 5.1 mostra a relação entre *Interface* e os módulos. As setas indicam chamadas de rotinas, isto é, *Interface* pode chamar rotinas de qualquer gerenciador, enquanto que o *GM* só pode chamar rotinas do *GP* e este, rotinas do *GG*.

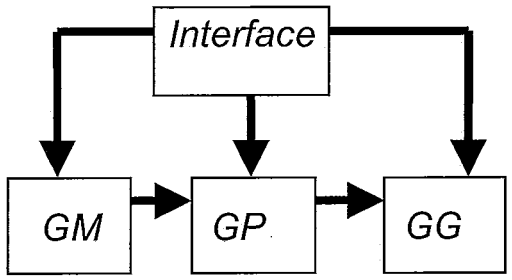


Figura 5.1: Relação entre *Interface* e os módulos.

- Interface: É o responsável pela criação e configuração dos elementos da interface com o usuário, isto é, janelas, botões, listas de opções, menus, etc. Interface também é responsável pelo despacho dos eventos associados a esses elementos.

- Gerenciador de *Mouse* (*GM*): É responsável pelo serviço de eventos associados ao *mouse*, como operações de seleção, movimentação de objetos, manipulação das características dos objetos.

- Gerenciador de Processos (*GP*): É responsável pela manipulação das estruturas de dados globais (seção 5.5) que representam as construções.

- Gerenciador Gráfico (*GG*): É o responsável pelo desenho da cena. Os elementos fixos (grades, câmeras, eixos e background) são desenhados explicitamente. Os objetos pertencentes às construções são responsáveis por seu próprio desenho, em resposta a mensagens enviadas pelo *GG*.

5.1.1 Linguagem de implementação

O Mangaba é um sistema orientado para objetos. Duas linguagens orientadas para objetos são escolhas naturais: Java e C++. Alguns sistemas conhecidos, como o Cabri Java [17] e Cinderella Café [6] foram feitos completamente em Java, enquanto que outros foram feitos em C++ [16]. As duas linguagens possuem vantagens e

desvantagens. Java é uma linguagem que suporta uma vasta gama de facilidades para a construção de interfaces gráficas. Entretanto, por ser uma linguagem tipicamente interpretada, aplicações escritas em Java costumam executar com mais lentidão do que aplicações escritas em linguagens compiladas.

Por ser uma linguagem bastante popular e eficiente, escolheu-se implementar o sistema na linguagem C++ com o uso da API gráfica OpenGL.

5.2 Processamento de eventos

Nesta seção é discutido um caso de uso do sistema para ilustrar o papel das principais unidades do sistema no processamento de interações com o usuário. Em particular, vamos detalhar uma operação de seleção seguida de uma operação de translação, conforme vemos nas Fig. 5.2 e Fig. 5.3.

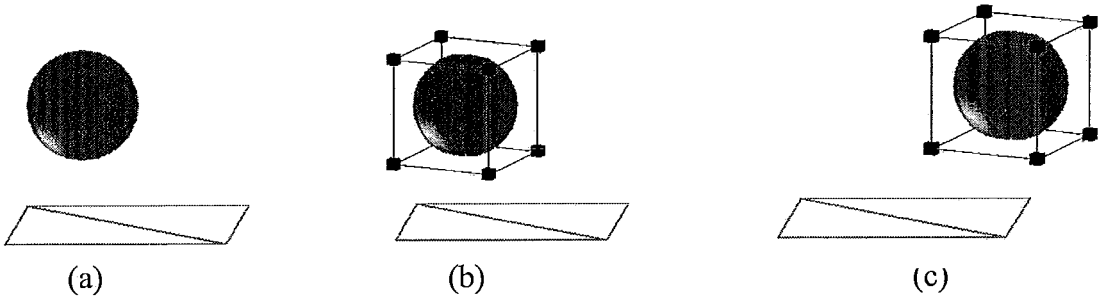


Figura 5.2: Seleção e translação de um objeto usando o botão direito do *mouse*. (a) Cena composta de uma esfera e um plano. (b) Esfera é selecionada. (c) Esfera em uma nova posição.

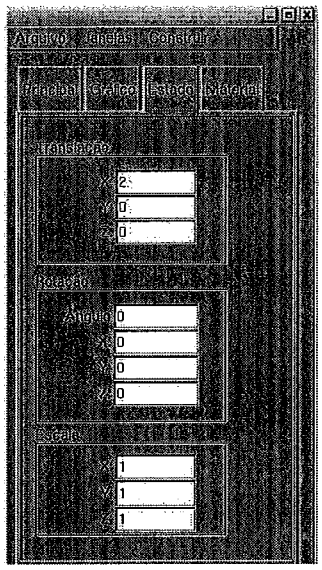


Figura 5.3: A seleção e translação de um objeto pode ser realizada através do evento de mudança do valor da caixa de translação.

A seleção dos objetos se faz clicando diretamente sobre eles, enquanto que a translação pode ser feita de duas formas: arrastando o *mouse* sobre o objeto selecionado (Fig 5.2) ou através de interação com elementos da interface gráfica (Fig. 5.3).

O diagrama de caso de uso da Fig. 5.4 mostra a seqüência de eventos relacionando as principais unidades do sistema. Primeiramente, *Interface* interpreta o clique do *mouse* sobre a tela, passando a posição do cursor para o *GM* (passo 1). O *GM* verifica sobre qual objeto o *mouse* foi pressionado e o indica para o *GP,* que o marca como objeto selecionado (passos a até 3). O usuário então translada o objeto mudando o valor da caixa de texto relacionada à posição (passo 4). *Interface* passa o valor da caixa de texto para o *GP* (passos c e 5), que verifica se o objeto selecionado faz parte de uma construção (passo d). Neste caso, inicia o processo de atualização dos demais objetos da construção (passo d) (Seção 5.6). Depois das atualizações serem concretizadas, *Interface* invoca o *GG* para que este desenhe a cena com os objetos nas posições corretas (passos 6 e e).

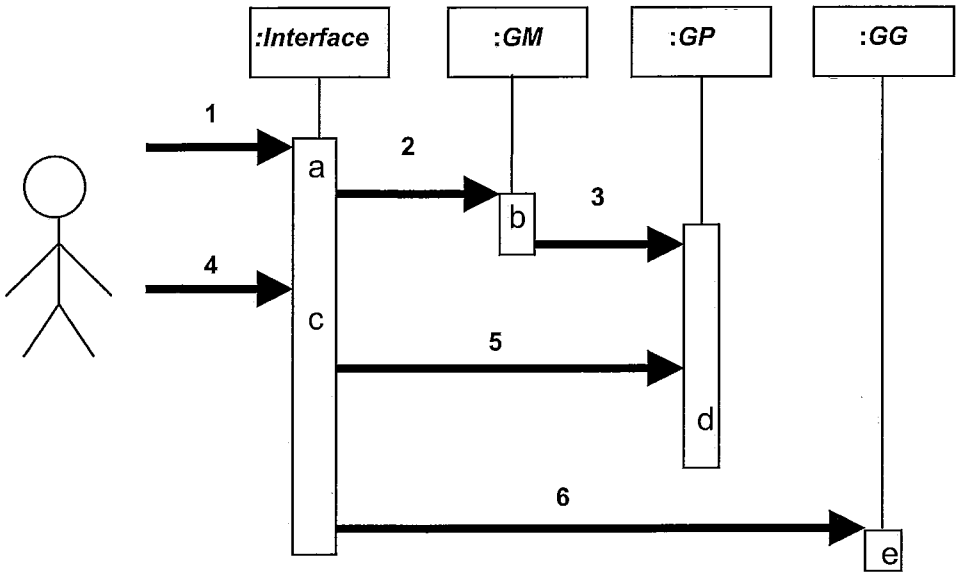


Figura 5.4: Diagrama de seqüência da seleção e translação do objeto selecionado através da caixa de texto de translação.

1 – Ocorre o evento de clique do *mouse* sobre a área de desenho.

a – O evento causa a execução do método *Interface->ManipulaEvento(event* ev). A posição do cursor é armazenada nas variáveis *xPos* e *yPos* : *xPos = Fl::event_x();yPos = Fl::event_y()*;

2 – *Interface* passa o valor da posição do cursor para o *GM* : *GM->mousePos(xPos, yPos)*.

b - *GM* transforma a posição do cursor em coordenadas do mundo (*x, y, z*). *GM* seleciona um dos objetos através do método *GM->seleciona(x, y, z)*. Em *GM->seleionat(x, y, z) GM* pega a lista de objetos armazenada em *GP*, percorre a lista e seleciona um dos objetos.

3 – *GM* passa para *GP* o índice do objeto selecionado na lista de objetos. *GP->modifSelecionado(int i)*.

4 – Ocorre outro evento: o usuário muda o valor do slider de translação em *x* da *Interface.*

c – O evento causa a execução do método *Interface->TransX()* e o novo valor do slider é guardado na variável temporária xTem.

5 – *Interface* passa o novo valor do slider para *GP* através do método *GP->translacaoX(xTemp)*.

d – *GP*, que contém o objeto selecionado, passa as novas coordenadas para o objeto e executa os métodos responsáveis pela atualização dos objetos de uma construção.

6 – *Interface* envia uma mensagem para GG desenhar a cena: *GG->desenhaTela()*.

e – A cena é desenhada.

5.3 Objetos geométricos

Os objetos suportados pelo sistema são: cones, esferas, cilindros, planos, retas e pontos. Todos estes objetos são instâncias de classes que estendem uma superclasse comum chamada *CIObjeto* (Fig. 5.5). Esta modela as características comuns a todos os

objetos como cor, transparência, grau de dilatação, posição, bem como os métodos públicos comuns a todos os objetos, incluindo:

- Métodos de consulta a características do objeto:
  - Rotação: *retornaRotacaoX* (), *retornaRotacaoY* (), *retornaRotacaoZ* ().
  - Translação: *retornaTranslacaoX* (), *retornaTranslacaoY* (), *retornaTranslacaoZ* (), *retornaTranslacao* ().
  - Escala: *retornaEscalaX*(), *retornaEscalaY* (), *retornaEscalaZ* (), *retornaEscala* ().
- Mótodos de modificação das características do objeto:
  - Translação: *translação* (*x, y, z*)
  - Rotação: *rotação* (*angx, angy, angz*)
  - Escala: *escala* (*x, y, z*)
  - Material: *modifAmbiente* (*r,g,b*) , *modifDifusa* (*r, g, b*), *modifEspecular* (*r, g, b*)
  - Desenho: *desenha* ().

Como será discutido na seção 5.7, transformações que deformam os objetos não são admitidas pelo sistema.

O estado de rotação de um objeto é representado internamente por três variáveis (*angulox*, *anguloy*, *anguloz*), que representam os três ângulos de rotação em torno dos eixos *x*, *y* e *z*. O estado de dilatação é representado internamente por três variáveis (*escalax*, *escalay*, *escalaz*), que guardam a dilatação na direção dos eixos *x*, *y* e *z* fixos ao objeto. O estado de translação é representado internamente por três variáveis (*transx*, *transy*, *transz*), correspondentes à translação em relação aos eixos *x*, *y* e *z* da cena, respectivamente.

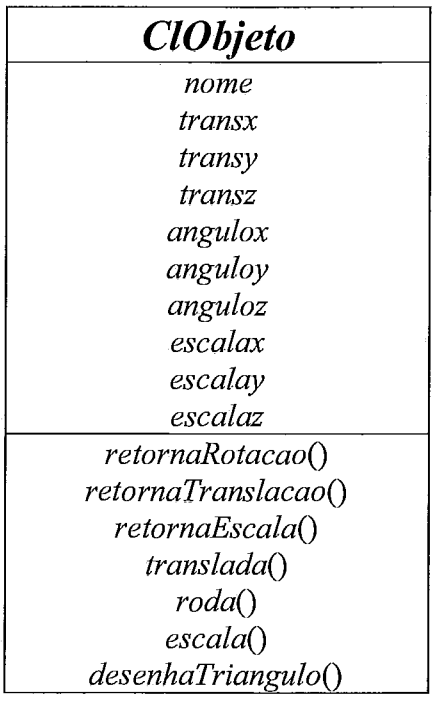


Figura 5.5: Objeto geométrico.

Neste sistema, apenas dois níveis de especialização foram utilizados, ou seja, a superclasse e as classes que representam os objetos geométricos (veja a Fig. 5.6).

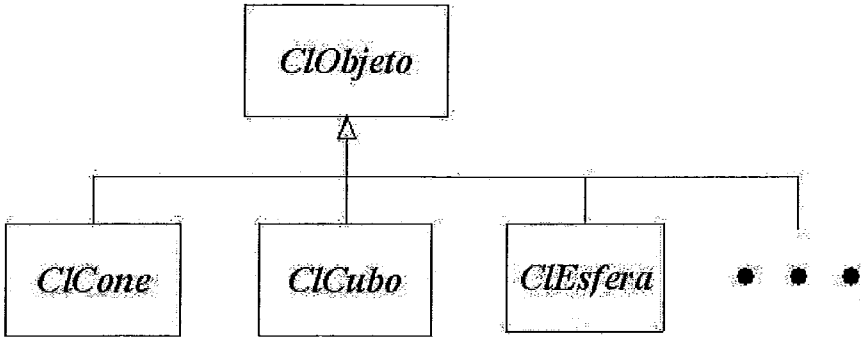


Figura 5.6: Hierarquia de objetos geométricos.

5.4 Objetos de restrição

Toda restrição é representada por um ou mais objetos de uma das classes de restrição – a classe *ClRestrição* (Fig. 5.7) e superclasse de todas as classes de restrição, tais como: *PlanoPorRetaPonto*, *Plano3Pontos*, *PontoPontoEsfera*, *PlanoPontoEsfera*, *IntRetaPlano*, *etc.*. Os objetos de restrição (Veja a Fig. 5.7) encapsulam funcionalidades

relativas ao cálculo do *estado* dos objetos que fazem parte da construção. Chamamos de *estado de um objeto* as características decorrentes das transformações lineares afins que foram aplicadas sobre um objeto primitivo a fim de instanciá-lo dentro da cena, a saber, posição (translação), orientação (rotação) e tamanho (escala). Em se tratando de objetos cujas restrições são inteiramente geometrias, apenas transformações afins que não deformam o objeto são consideradas.

Cada objeto de restrição contém um ou mais objetos geométricos ditos *sujeitos* e um objeto geométrico dito *observador*. Pode-se pensar no objeto observador como dependente dos objetos sujeitos. Por exemplo, numa restrição do tipo *Plano3Pontos* (Ver tabela de restrições no apêndice A), o plano é o observador, enquanto que os pontos são sujeitos. Em termos do grafo de dependências discutido na seção 2.1.3, os sujeitos são os *pais* do observador.

Um objeto de restrição contém uma referência para cada objeto envolvido na restrição. Além disso, ele define seis métodos públicos: *calculaRotacao*(), *calculaTranslacao*(), *calculaEscala*(), *rotacaoObserv*(), *translacaoObserv*(), *escalaObserv*(). Sempre que um dos objetos da restrição muda de estado, estes seis métodos são chamados para a atualização do estado do observador. Os métodos *calculaRotacao*(), *calculaTranslacao*() e *calculaEscala*() são executados sempre que um dos sujeitos da restrição é rodado, transladado ou dilatado, respectivamente. Eles se encarregam de atualizar o estado do objeto observador segundo a restrição. Observe que quando um sujeito muda de posição, tanto a posição do sujeito quanto a do observador devem ser alteradas. Por outro lado, se um observador da restrição é movido, os sujeitos não precisam acompanhá-lo. Nesse caso, os métodos simplificados *rotacaoObserv*(*x,y,z*), *translacaoObserv*(*x,y,z*) e *escalaObserv*(*x,y,z*) são executados quando o *observador* da restrição é rodado, transladado ou dilatado, respectivamente.

ClRestrição
*ListClObjeto * sujeito* *ClObjeto * observador*
calculaRotação() *calculaTranslação*() *calculaEscala*() *rotaçãoObserv*() *translaçãoObserv*() *escalaObserv*()

Figura 5.7: Objeto de Restrição.

Considere o objeto de restrição *PontoSoPlano* (veja Apêndice A). Quando Π – isto é, o sujeito – é rodado, o método *calculaTranslacao*() é executado e calcula a nova posição de P. Já quando o usuário arrasta P, o método *translacaoObserv*(x, y, z) é executado e calcula a nova posição de P.

Considere o objeto de restrição *ReflexãoPlano*. Quando T_1 muda de estado, seja por rotação, translação ou por dilatação, a posição de T_2 é recalculada executando-se um dos métodos *calculaRotacao*(), *calculaTranslacao*() ou *calculaEscala*(). O mesmo acontece quando o outro sujeito da restrição, isto é, Π é alterado.

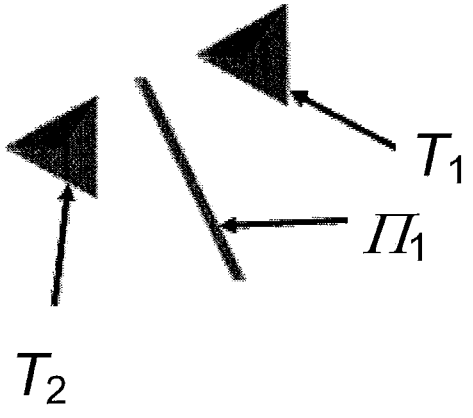


Figura 5.8: T_2 é a reflexão de T_1 em relação ao plano Π.

Observe que cada objeto geométrico pode participar como observador em apenas uma restrição, embora possa desempenhar o papel de sujeito em várias. Ele pode ser restrito a vários objetos pais, mas, depois de definida qual é a restrição que descreve a sua posição espacial, o usuário não pode mais acrescentar uma outra restrição a este mesmo objeto. Cada objeto, portanto, pode ser *observador* de apenas uma construção.

5.5 Estruturas de dados utilizadas para a atualização

O *GP* mantém as seguintes estruturas globais:

- Lista *listaObj*: contém referências para todos os objetos geométricos. A ordem dos objetos nesta lista não importa para o sistema de atualização.
- Lista *listaRestr*: contém referências para todos os objetos de restrição.
- *objManipulado*: referência para o objeto geométrico tendo seu estado modificado.
- *objObservador*: guarda uma referência para o objeto de restrição em que o *objManipulado* é o *observador* (caso exista).

Cada objeto geométrico contém ainda:

- Lista *listaDependentes*: Uma lista de objetos de restrição em que o objeto geométrico é um *sujeito*.

5.6 Esquema de atualização

Digamos que um dado objeto geométrico seja selecionado e, através de uma manipulação comandada pelo usuário, mude de estado. Naturalmente, todos os demais objetos que dele dependem devem mudar de estado de forma consistente. Isto é feito caminhando-se no grafo de dependências a partir do nó que representa o objeto manipulado, efetuando-se as transformações de todos os nós encontrados no caminho até todos os observadores.

Como exemplo, considere uma construção envolvendo os seguintes objetos: dois pontos P_1 e P_2, um plano Π, uma reta R. Este exemplo consiste das seguintes restrições (o último dos objetos é o observador):

- C_1: Ponto sobre plano (Π e P_1).

- C_2: Reta perpendicular a um plano e passa por um ponto (Π, P_1 ,R).
- C_3: Ponto sobre uma reta (R, P_2). Esta construção ilustra três objetos de restrição: *PontoSoPlano*, *PontoSoReta*, *RetaPePlanoPonto*.

A Fig. 5.9 mostra as relações de *sujeito* e *observador* dos objetos envolvidos na construção e os três objetos de restrição.

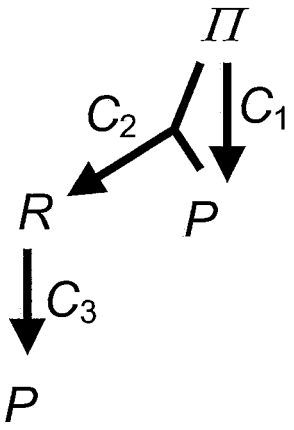


Figura 5.9: Relações de sujeito e observador de uma construção que envolve três objetos de restrição.

Se Π é rodado, então P_1 deve ser transladado, já que ele é dependente (observador) de Π. De forma análoga, R deve ser rodado, o que acarretará a translação de P_2. O algoritmo de atualização é disparado pelo *GP* invocando o método *rotação*() do objeto Π. Este invoca o método *calculaRotacao*() de todos os objetos de restrição em sua variável *listaDependentes*, isto é, C_1 e C_2. O método C_1.*calculaRotacao*(), por sua vez, invocará P_1.*translação*(). O processamento de C_2.*calculaRotacao*() se dá de forma semelhante. O esquema abaixo resume a seqüência de chamadas:

→ Π.*rotação*()
- C_1.*calculaRotacao*()
  - P_1.*translação*()
- C_2.*calculaRotacao*()
  - *R.translação*()
    - C_3.*calculaTranslacao*()
      - P_2.*translação*()
  - *R.rotação*()
    - C_3.*calculaRotacao*()
      - P_2.*translação*()

Repare que, quando o plano roda, a posição do ponto P_1 muda, e portanto R deve rodar e transladar para se manter perpendicular a Π e passar pelo ponto P_1. Para que a posição de todos os objetos de uma construção seja atualizada corretamente, logo após a chamada da função *translação*() do objeto, se ele for sujeito em um objeto de restrição, a função *calculaTranslacao*() do objeto de restrição deve ser executada. Se a posição de P_2 for calculada depois da posição de Π ser totalmente calculada, P_2 irá deslizar para fora da reta. Por esse motivo a função P_2.*translação*() é executada duas vezes quando o plano Π roda.

5.7 Cálculo da posição dos *observadores* de uma restrição

Cada objeto é responsável por se desenhar. Antes de se desenhar, os vértices de cada triângulo são alterados por uma transformação de rotação, uma de translação e uma de dilatação. As transformações são feitas sempre na mesma ordem: dilatação, rotação e translação. Isto permite ao objeto atingir todas as posições na cena, obedecer à restrição imposta pela construção e não ser deformado. Observe-se que, como as transformações são aplicadas sempre na mesma ordem e em relação à posição de inserção do objeto (posição do objeto logo após ser criado), algumas transformações lineares afins não podem ser reproduzidas. Assim, por exemplo, é possível obter um paralelepípedo retangular de quaisquer dimensões a partir de um cubo, mas é impossível obter, digamos, um paralelepípedo geral, isto é, com faces que não sejam retângulos.

O algoritmo de desenho é o seguinte:

$T \leftarrow$ *Matriz_Identidade*;

$T \leftarrow T \times$ *Matriz_Translação*;

$T \leftarrow T \times$ *Matriz_Rotação_em_x;*

$T \leftarrow T \times$ *Matriz_Rotação_em_z;*

$T \leftarrow T \times$ *Matriz_Rotação_em_y;*

$T \leftarrow T \times$ *Matriz_Escala;*

Para todos os triângulos t do objeto fazer

Desenha ($T \times t$);

Para mudar o estado do *observador*, o objeto de restrição calcula e atualiza as variáveis *angulox, anguloy, anguloz, escalax, escalay, escalaz, transx, transy, transz.* Essas variáveis, em última análise, vão gerar os coeficientes das diversas matrizes (*Matriz_Rotação_em_x, Matriz_Rotação_em_y* e *Matriz_Rotação_em_z*) usadas no algoritmo.

À exceção de retas, todos os demais tipos de objetos geométricos com extensão (cilindros, cones, planos), quando criados, são iniciados com o eixo *y* do sistema de coordenadas locais ao objeto alinhado com eixo *y* da cena. Um objeto é rodado através da interface modificando-se quatro parâmetros: três representam uma direção (vetor) e o quarto um ângulo. O eixo *y* do sistema de coordenadas locais ao objeto acompanha o valor do vetor. O quarto parâmetro representa o ângulo de rotação em torno do vetor. A mudança do valor do ângulo acarreta a mudança dos coeficientes da matriz que representa a transformação em torno do eixo *y* (*Matriz_Rotação_em_y*). O objeto é então rodado em torno do eixo *y* do sistema de coordenadas locais. Para que o eixo *y* do sistema de coordenadas locais ao objeto acompanhe o vetor, o valor das variáveis *angulox* e *anguloz* deve ser calculado.

Em todos os objetos de restrição, *angulox* e *anguloz* do objeto observador são calculadas para que o objeto seja rodado de forma a ficar com o eixo *y* do sistema de coordenadas locais na direção de um vetor. Considere o vetor *A* como na Figura 5.10:

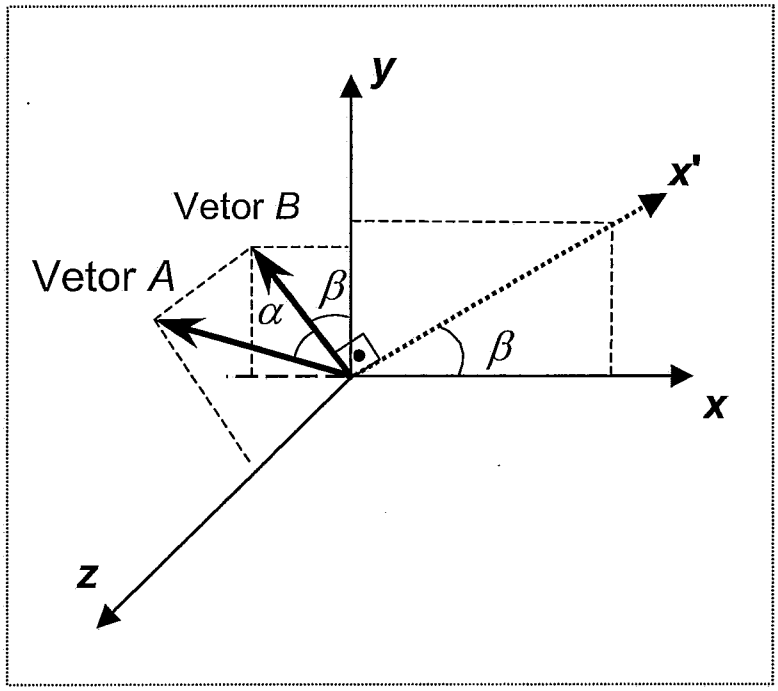


Figura 5.10: Rotação do objeto observador.

Inicialmente, os eixos do sistema de coordenadas locais a um objeto são iniciados na direção dos eixos de coordenadas do mundo. Para posicionar um objeto de forma que o eixo y do sistema de coordenadas locais ao objeto fique alinhado ao vetor A, os ângulos α e β devem ser calculados. A primeira rotação é feita em torno do eixo z (*Matriz_Rotação_em_z*). Ao girar o objeto de um ângulo β em torno do eixo z, o eixo y local ao objeto torna-se paralelo ao vetor B e o eixo x local ao objeto desloca-se para a direção x'. Em seguida, uma segunda rotação de um ângulo α é aplicada em torno do eixo x'. Como resultado, o objeto fica com o eixo y local ao objeto paralelo ao vetor A.

Para ilustrar o processo de rotação do objeto, considere um objeto de restrição *Plano3Pontos* que define um plano Π construído sobre três pontos P_1, P_2 e P_3. Esta restrição implementa o método *calculaTranslacao*() da seguinte forma:

1. *Vetor_X* = P_2->*retornaTranslacao*() – P_1->*retornaTranslacao*().
2. *Vetor_Z* = P_3->*retornaTranslacao*() – P_1->*retornaTranslacao*().
3. *Vetor_A* = *ProdutoVetorial*(*Vetor_X*, *Vetor_Z*);
4. *Vetor_B* = *Projeção*(*Vetor_A*, *PlanoYZ*);
5. β = *Ângulo*(*Vetor_B*, *eixo y*);
6. α = *Ângulo*(*Vetor_A*, *Vetor_B*);
7. Π->*rotação* (β, *Observador*->*retornaRotacaoY* (),α);

Desta forma, quando um dos pontos (sujeitos da construção) é transladado, o método *calculaTranslacao*() é executado, fazendo com que as variáveis *angulox* e *anguloz* do observador sejam modificadas apropriadamente (linha 7 do código acima).

Em algumas restrições, como na restrição *PontoSoPlano,* por exemplo, as variáveis de translação do observador devem ser atualizadas de forma a manter a restrição. Neste tipo de restrição é importante notar que, quando o plano é transladado ou rodado, o ponto deve ser transladado de forma a manter inalterada sua posição em relação às coordenadas locais do plano. Portanto, na restrição *PontoSoPlano*, os métodos *calculaRotacao*(), *calculaTranslacao*() e *translacaoObserv*() devem ser

implementados. O método *rotacaoObserv* (), entretanto, não precisa ser implementado, visto que não faz sentido girar um ponto.

5.8 Comparação com outras abordagens

No Mangaba, os objetos geométricos são representados por transformações que levam o objeto da posição inicial para a posição final, como descrito nas seções 5.3 e 5.7. O mecanismo de atualização dos objetos de uma construção no Mangaba é uma implementação de um grafo de dependências para objetos compostos onde os objetos são tridimensionais. A interface é feita com o uso da biblioteca OpenGL, que é própria para tirar o máximo proveito dos recursos do *hardware* gráfico.

Já os sistemas descritos por [90,91] são implementações de um grafo de dependência de objetos simples. Nestes sistemas de geometria dinâmica, cada objeto geométrico é representado internamente por parâmetros geométricos. Por exemplo, um plano é representado por um vetor normal e um termo independente; já uma reta é representada por um ponto e um vetor e não por transformações. Com isso, a posição dos objetos é calculada de forma mais direta (modificando diretamente o valor do vetor normal e do ponto). A interface gráfica destes sistemas é muito limitada. Os objetos são desenhados basicamente com retas e pontos apenas.

A tabela mostrada abaixo resume as principais características dos diversos sistemas de geometria dinâmica 3D.

	Outras Abordagens	**Mangaba**
Grafo de Dependências	De objetos simples	De objetos compostos.
Representação Interna dos Objetos	Variáveis que descrevem a sua posição (ex. vetor e ponto)	Transformações
Recursos Gráficos	Limitados	Cena desenhada com o uso do OpenGL.

Conclusão

O sistema Mangaba descrito na presente dissertação tem como objetivo ser empregado no ensino de geometria espacial. Este *software* visa facilitar a visualização dos objetos geométricos espaciais pelos alunos, possibilitando o entendimento e a demonstração da veracidade de postulados e teoremas geométricos cuja apresentação em quadro negro é de modo geral inadequada.

Na literatura encontramos poucas ferramentas de ensino de geometria espacial [16,17]. Em geral, as ferramentas de ensino de geometria espacial possuem a elaboração gráfica limitada, representando os objetos geométricos apenas com pontos e retas, dificultando o entendimento dos objetos. O Mangaba, com a utilização de recursos de computação gráfica permite uma renderização interativa e de alta qualidade dos objetos envolvidos. Isto proporciona ao aluno um maior senso de imersão no mundo virtual onde o problema se desenrola.

Para tornar o Mangaba – ou qualquer outro *software* de ensino de geometria espacial interativo – mais rápido, o *software* deve empregar um algoritmo eficiente para a atualização da posição dos objetos. Existem basicamente dois tipos de algoritmos de atualização: os baseados em restrições e os baseados em construções. O Mangaba utiliza um sistema de atualização baseado em construções, o que permite evitar os problemas de ambigüidade de difícil solução inerentes aos sistemas baseados em restrições. Em particular, a implementação do Mangaba emprega um grafo de dependência simples que incorpora características de alguns sistemas conhecidos e que foram adaptadas para a geometria espacial como: objetos sujeitos e observadores e objetos de restrição. A abordagem empregada no Mangaba para construções tridimensionais é nova e pode servir de inspiração para que novos algoritmos de atualização de objetos de uma construção tridimensional sejam criados e implementados.

O *software* Mangaba foi bem sucedido quanto ao seu propósito, possibilitando a construção, demonstração e boa visualização de alguns teoremas geométricos. Entretanto, planejamos acrescentar futuramente algumas características importantes:

1. Muitos teoremas não podem ser visualizados com o pequeno número de objetos e restrições implementados no Mangaba. Em particular, poliedros não podem

participar de construções. Pretendemos implementar portanto novas restrições baseadas em elementos constituintes de poliedros, como por exemplo, reta perpendicular a uma face ou plano perpendicular a uma aresta do poliedro

2. A possibilidade de construir curvas de *locus* é uma ferramenta bastante interessante implementada em sistemas de geometria dinâmica bidimensional. Uma facilidade análoga para construções em 3D é também bastante desejável, mas cuja implementação é consideravelmente mais difícil.

3. A transparência é um recurso que facilita a visualização de objetos em uma construção. A transparência implementada utiliza um expediente relativamente simples (máscara de bits) mas com resultado gráfico apenas razoável, o que limita sua aplicação. Um uso mais intensivo desse recurso requer uma implementação mais complexa e computacionalmente mais dispendiosa.

4. A utilização de sombras melhora o aspecto visual da cena tornando o programa mais interessante e atrativo para o aluno. Nenhum algoritmo de sombra foi implementado.

5. O recurso de detecção de colisão, que foi implementado com o *K-DOPs* [112], pode ser usado para facilitar o usuário fazer construções. Por exemplo, quando uma reta toca um plano, o *software* pode sugerir a restrição de uma reta perpendicular a um plano.

6. Possibilitar a visualização das cenas com óculos 3D.

7. Possibilitar a especificação interativa de objetos através de suas coordenadas.

Apêndice A

Tabela de restrições geométricas

Restrições	Sujeitos	Observador	Funcionalidade
Cilindro3Ponto	Ponto P_1, P_2 e P_3	Cilindro L	L é o cilindro que passa pelo ponto P_3 e tem um eixo que passa por uma reta P_1 e P_2.
CilindroRetaPonto	Reta R e Ponto P	Cilindro L	L é o cilindro que passa pelo ponto P e tem um eixo que passa pela reta R.
Circulo3Pontos	Pontos P_1, P_2 e P_3.	Circulo C	C passa pelos pontos P_1, P_2 e P_3
CirculoSoEsfera	Esfera S	Circulo C	C fica sobre S
CirculoSoPlano	Plano Π	Circulo C	C fica sobre Π
ConeTgEsfera	Ponto P_1 e esfera S	Cone O	O é tangente a esfera S e tem seu vértice passando por P_1.
DiedroReta2Pontos	Ponto P_1, Ponto P_2 e Reta R	Diedro D	D passa pelos pontos P_1 e P_2 e com fronteira uma reta R.
EsferaPontoPonto	Ponto P_1 e Ponto P_2	Esfera S	Esfera com centro em P_1 e raio definido pela distância entre P_1 e P_2.
Esfera3Pontos	Pontos P_1, P_2, P_3	Esfera S	Esfera que passa por P_1, P_2 e P_3.
IntRetaPlano	Reta R e Plano Π_1	Ponto P	P é a interseção entre R e Π.

Restrições	Sujeitos	Observador	Funcionalidade
Plano3Pontos	Ponto P_1, P_2, P_3	Plano Π	Π passa por P_1, P_2 e P_3.
PlanoEsferaPonto	Esfera S e ponto P	Plano Π	Π é tangente a S, e é perpendicular a uma reta imaginária que passa por P e pelo centro de S.
PlanoPaPlano	Plano Π_1	Plano Π_2	Π_2 é paralelo à Π_1
PlanoPaPlanoPonto	Plano Π_1 e Ponto P	Plano Π_2	Π_2 é paralelo a Π_1 e passa pelo ponto P.
PlanoPeReta	Reta R	Plano Π	Π é paralelo a reta R.
PlanoPeRetaPonto	Ponto P e Reta R_1	Reta R_2	R_1 é perpendicular a reta R_2 e passa pelo ponto P
PlanoPlanoInt	Plano Π_1 e Plano Π_2	Reta R	R é a interseção entre P_1 e P_2.
PlanoPlanoReta	Plano Π_1 e Plano Π_2	Reta R	R é a interseção entre Π_1 e Π_2
PlanoPontoPonto	Ponto P_1 e Ponto P_2	Plano Π	Π é o plano mediador a P_1 e P_2.
PlanoRetaPonto	Reta R e Ponto P	Plano Π	Π é o plano que passa por P e por R.
PontoM2Pontos	Pontos P_1 e P_2	Ponto P_3	P_3 é o ponto médio entre P_1 e P_2.
PontoSoCilindro	Cilindro L	Ponto P	P fica sobre L
PontoSoCirculo	Circulo C	Ponto P	P fica sobre C
PontoSoCone	Cone O	Ponto P	P fica sobre O
PontoSoEsfera	Esfera S	Ponto P	P fica sobre S
PontoSoPlano	Plano Π	Ponto P	P fica sobre Π
PontoSoReta	Reta R	Ponto P	P fica sobre R

Restrições	Sujeitos	Observador	Funcionalidade
ReflexãoPlano	Um objeto qualquer A_1 e um plano Π	A_2 do mesmo tipo que o objeto A_1	A_2 deve ser a reflexão de A_1 em relação a Π
RetaPaReta	Reta R_1	Reta R_2	R_2 é paralela a R_1.
RetaPePlano	Plano Π	Reta R	R é perpendicular a Π.
RetaPeRetaPonto	Ponto P e Reta R_1	Reta R_2	R_1 é perpendicular a reta R_2 e passa pelo ponto P
RetaPlanoPonto	Plano Π e Ponto P	Reta R	R é perpendicular a Π e passa pelo ponto P.
RetaPontoPonto	Ponto P_1 e Ponto P_2	Reta R	R passa pelos pontos P_1 e P_2.
SegmentoPontoPonto	Ponto P_1 e Ponto P_2	Segmento M	M é o segmento definido por P_1 e P_2.
SemiretaPontoPonto	Ponto P_1 e Ponto P_2	Semireta U	U é a semireta definida por P_1 e P_2.
SPlanoRetaPonto	Reta R e Ponto P	Semiplano Π	Π tem um lado definido por R e passa por um ponto P.
SPlano3Pontos	Pontos P_1, P_2 e P_3	Semiplano Π	Π passa por P_3 e tem um lado definido por uma reta imaginária que passa por P_1 e P_2.
Triangulo3Pontos	Pontos P_1, P_2, P_3	Triângulo E	E é definido pelos pontos P_1, P_2 e P_3

Apêndice B

O debate sobre postulados e teoremas está detalhado em [107]. A Geometria Espacial examina as propriedades de objetos que são construídos a partir de certos elementos básicos como reta, plano e ponto. O ponto, a reta e o plano podem ser caracterizados por meio de certas propriedades fundamentais, os postulados, que servem como ponto de partida para a teoria a ser desenvolvida.

Postulado 1. Por dois pontos do espaço passa uma e somente uma reta.

Postulado 2. Dada uma reta do espaço, existem pontos que pertencem à reta e pontos que não pertencem a reta.

Postulado 3. Por três pontos do espaço não situados na mesma reta passa um e somente um plano

Postulado 4. Dado um plano no espaço, existem pontos que pertencem ao plano e pontos que não pertencem ao plano.

Teorema1. Se uma reta tem dois de seus pontos em um plano, então ela está contida neste plano.

Decorre dos postulados 3 e 4 que por dois pontos passa sempre um plano. Dada esta afirmativa, o postulado 1 completa a afirmação do teorema.

Com a ferramenta Mangaba, pode ser feita a verificação da veracidade do teorema 1 realizando a seguinte construção:

1. Construir um plano:

- Selecionar um plano no menu;
- Pressionar o botão Inserir;

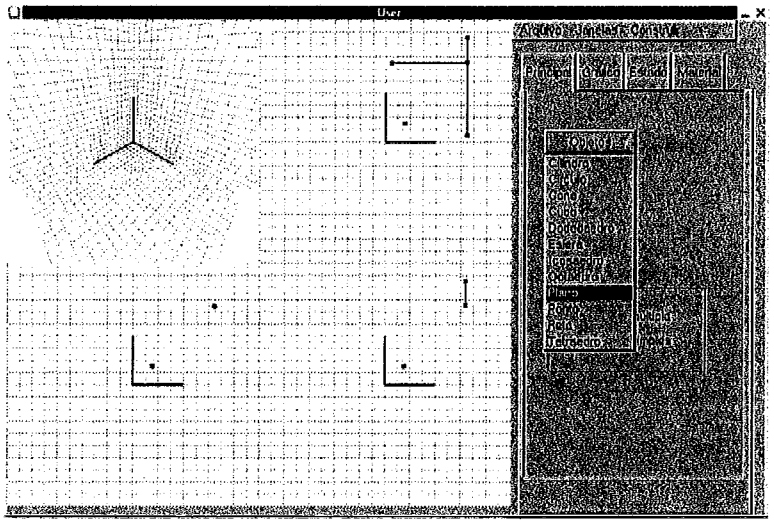


Fig A.1: Escolha do menu plano

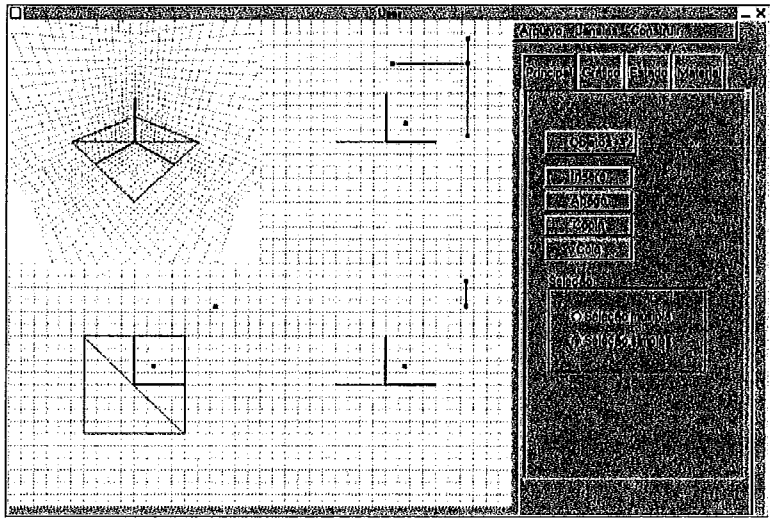


Fig A.2: Plano.

2. Construir dois pontos sobre um plano:

- Selecionar o plano;
- Selecionar no menu Construir a opção ponto sobre um plano duas vezes;

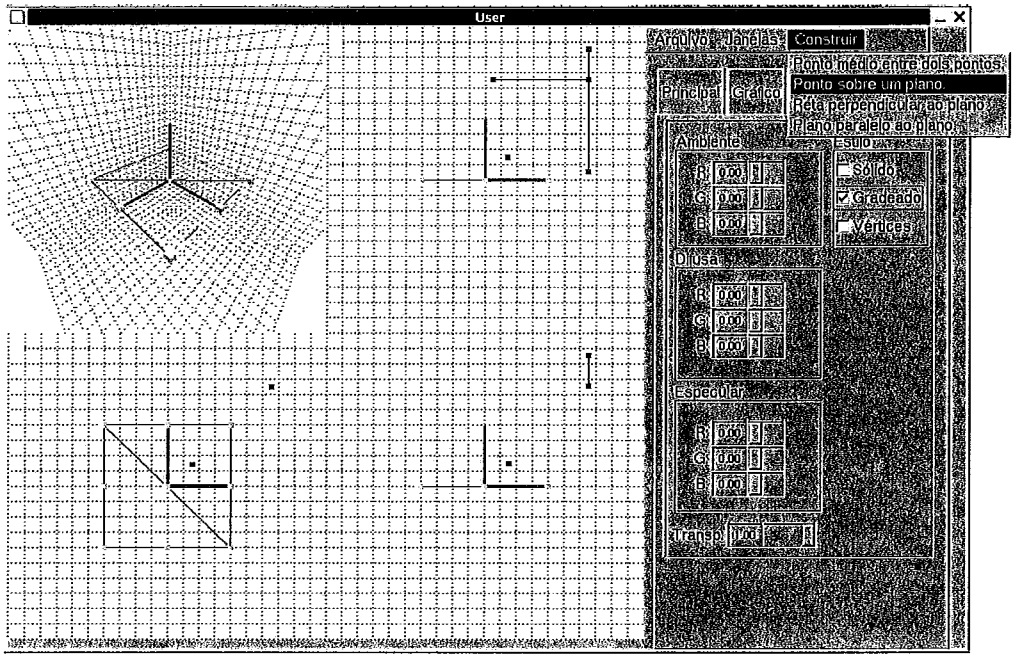


Fig. A.3: Seleção do menu ponto sobre um plano.

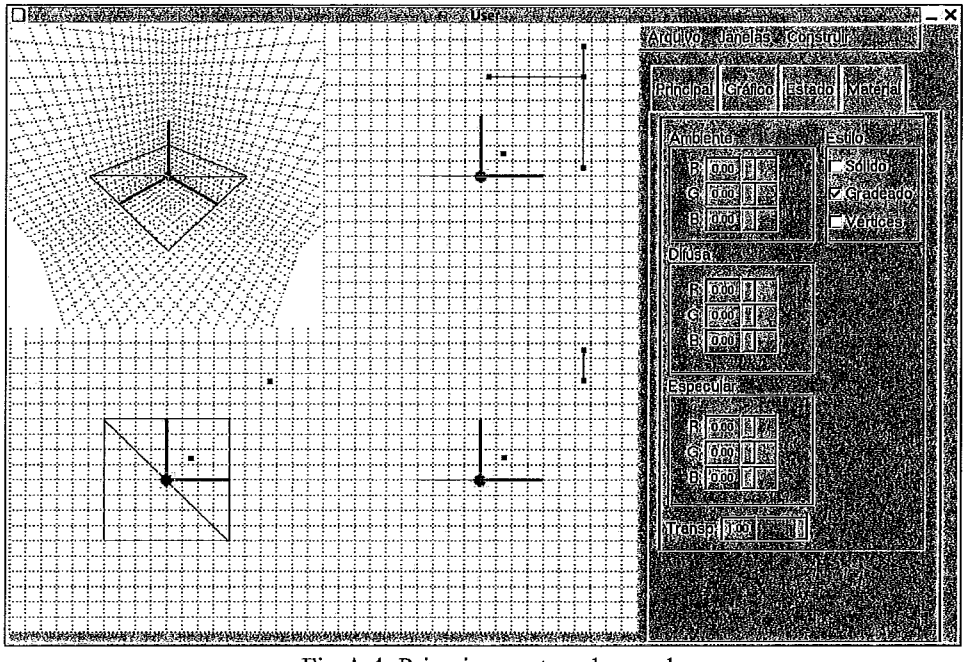


Fig A.4: Primeiro ponto sobre o plano

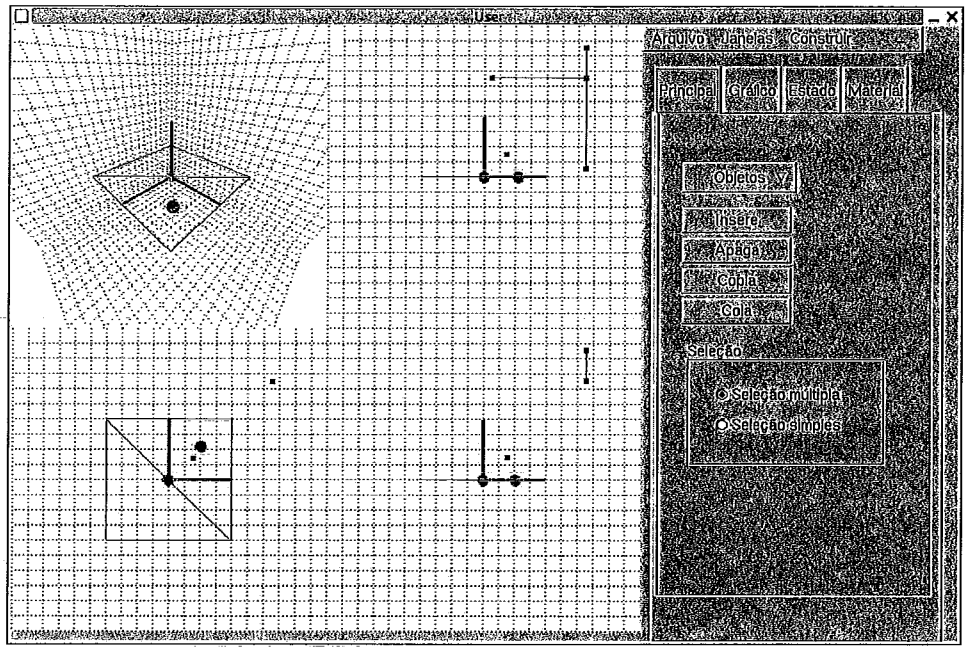


Fig. A.5: Dois pontos sobre o plano.

3. Construir uma reta sobre dois pontos.

- Selecionar os dois pontos;
- Selecionar no menu Construir a opção reta sobre dois pontos;

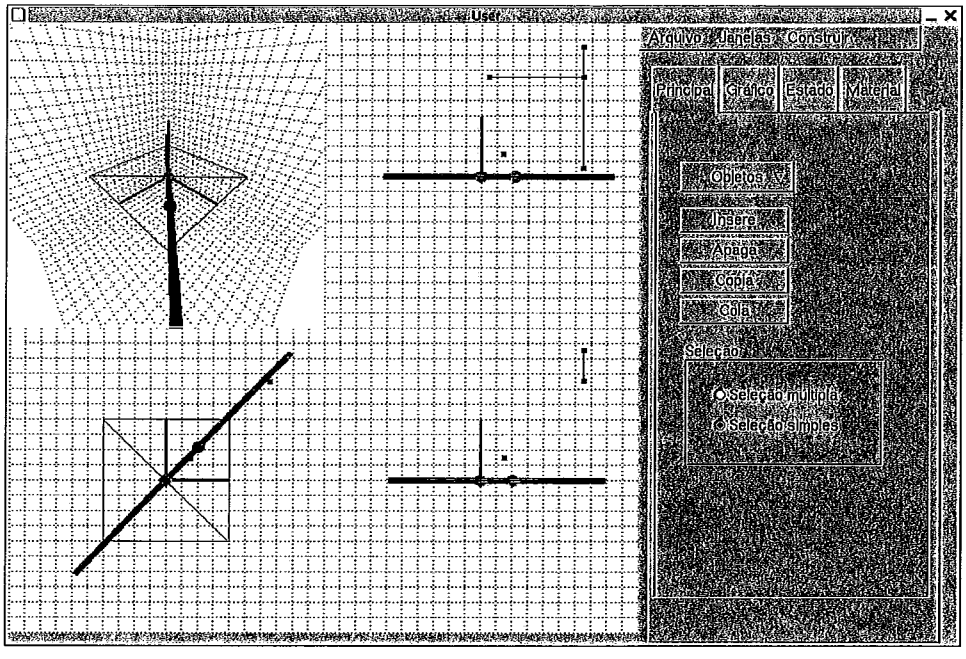


Fig. A.6: Uma reta sobre dois pontos

Modificando a posição dos pontos e do plano interativamente, a reta permanece contida ao plano.

Propriedade: Considere uma pirâmide quadrangular de base P_1, P_2, P_3, P_4 e vértice P_5. Os pontos P_1, P_3 *e* P_5 determinam um plano Π_1 e os pontos P_2, P_4 e P_5 determinam um segundo plano Π_2. A intercessão entre Π_1 e Π_2 é uma reta que passa sempre por P_5 e um ponto de intercessão entre uma reta que passa por P_2 e P_4 e outra reta que passa por P_1 e P_3. P_2 e P_4.

Com a ferramenta Mangaba, pode ser feita a verificação da veracidade desta propriedade seguindo os seguintes passos:

1. Construir os 5 pontos de forma que quatro deles formem a base e o último o vértice.

- Selecionar o menu ponto;
- Pressionar o botão inserir 5 vezes;

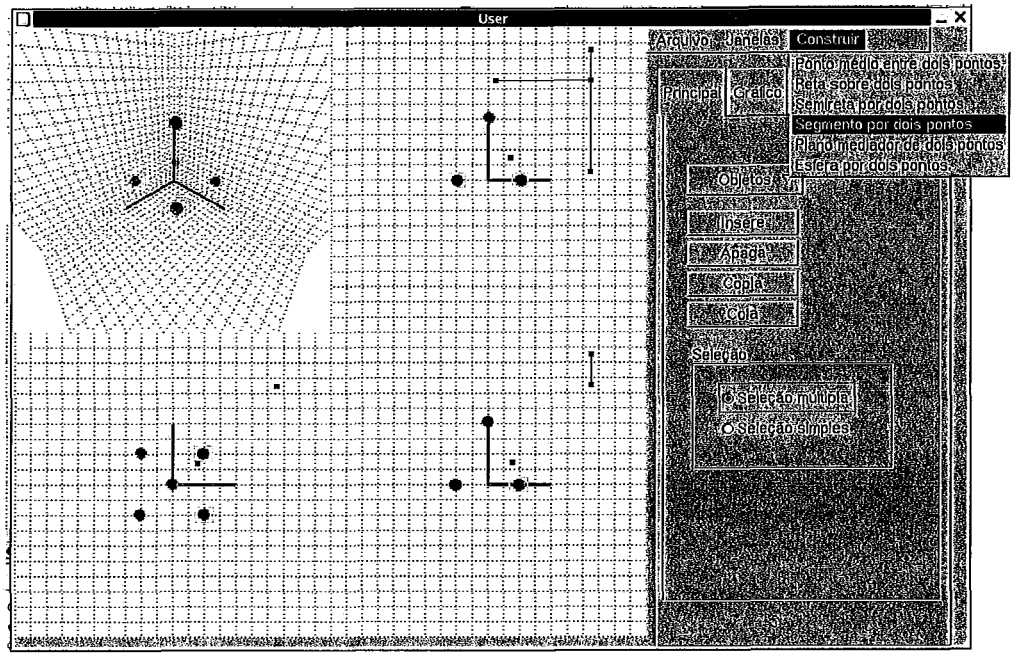


Fig. A.7: 5 pontos criados. Dois deles selecionados e a opção do menu segmento por dois pontos selecionada.

2. Construir as arestas da pirâmide com segmentos de reta.

- Selecionar dois pontos;
- Selecionar no menu Construir a opção segmento por dois pontos; Repetir esses processos até que todas as arestas estejam criadas.

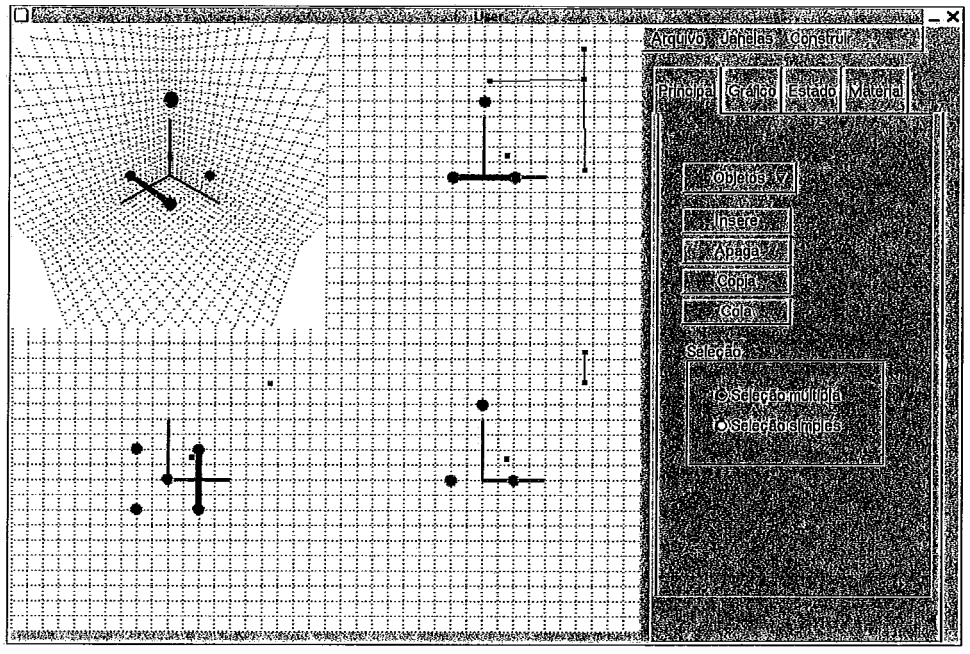


Fig. A.8: Construção de uma aresta da pirâmide

3. Construir os dois planos: Π_1 sobre dois pontos da diagonal do quadrado da base e o vértice da pirâmide e Π_2 restrito a dois pontos da outra diagonal da base e o vértice da pirâmide e a interseção entre eles:

- Selecionar dois pontos da diagonal da base e o vértice da pirâmide;
- Selecionar no menu Construir a opção plano sobre 3 pontos;
- Selecionar os outros dois pontos da diagonal da base e o vértice da pirâmide;
- Selecionar no menu Construir a opção plano sobre 3 pontos;
- Selecionar os dois planos;
- Selecionar no menu Construir a opção Interseção entre dois planos;

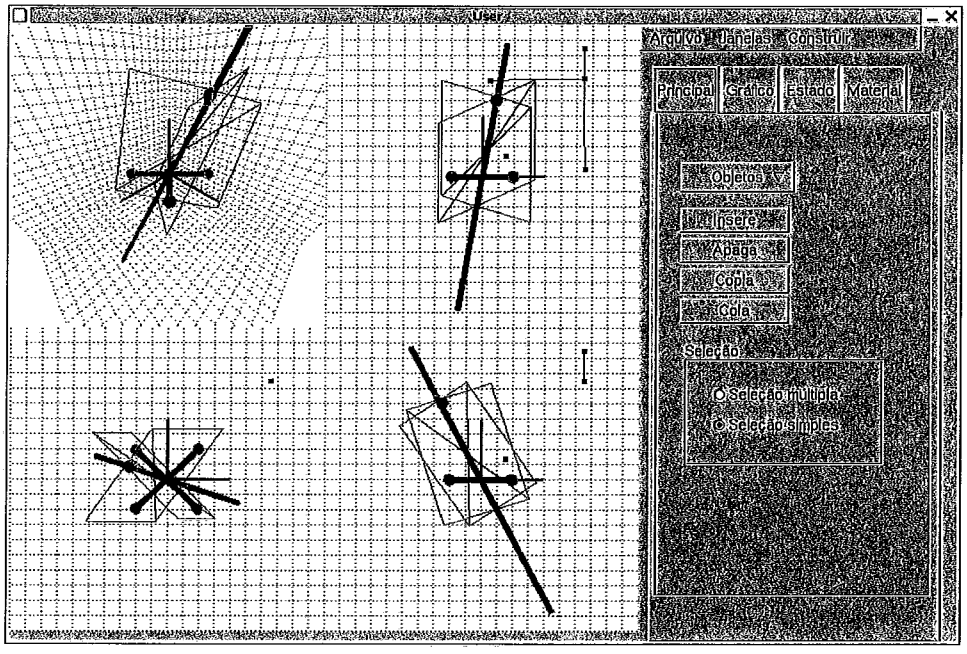


Fig. A.9: Π_1 sobre dois pontos da diagonal do quadrado da base e o vértice da pirâmide e Π_2 restrito a dois pontos da outra diagonal da base e o vértice da pirâmide e a interseção entre os planos.

Modificando a posição do ponto P_5, verifica-se que a reta de intercessão entre os planos sempre passa pela intercessão entre a reta que passa por P_2 e P_4 e a reta que passa por P_1 e P_3.

Referências Bibliográficas

[1] GUIMARÃES, L. C., BARBASTEFANO R., BELFORT E. "Tools for synchronous distance teaching in geometry".

[2] KORTENKAMP, U. *Foundations of Dynamic Geometry*, Ph. D. dissertation, Federal Institute of Technology Zurich, Swiss, 1999.

[3] KORTENKAMP, RICHTER-GEBERT, J. "Making the move: The next version of Cinderella". In: *Proceedings of the First International Congress of Mathematical Software*, World Scientific, pp. 208ff, 2002.

[4] Cinderella, interactive geometry software: http://cinderella-geometry.com/en/home/index.html, Fevereiro, 2004.

[5] KORTENKAMP, U., RICHTER-GEBERT, J. "Cinderella". In: *Erfahrungen mit Java*, ch. 16, pp. 381-401, 1999.

[6] KORTENKAMP, U., RICHTER-GEBERT, J. "Euklidische und Nicht-Euklidische Geometrie in Cinderella". In: *Thomas Weth, Nürnberger Kolloquium zur Didaktik der Mathematik*, Nürnberger, April 1999.

[7] JONES, K. "The Mediation of Learning withing a Dynamic Geometry Enviroment". In: *A Oliver (Ed), Proceedings of the 22nd Conference of the International Group for the Psychology of Mathematics Education.* University of Stellenbosch, South Africa, v. 3, pp. 96-103, 1998.

[8] JONES, K. "Children Learning to Specify Geometrical Relationships Using a Dynamic Geometry Package". In: *Pehkonen E (ed) Proceedings of the 21st Conference of the International Group for the Psychology of Mathematics Education*, v. 3, pp. 121-128, 1997.

[9] REINHARD H. *Im Zugmodus der Cabri Geometrie.* Ph. D. dissertation, University of Augsburg, Germany, November 1994.

[10] LABORDE, JEAN-MARIE, "Some issues raised by the development of implemented dynamic geometry as with Cabri Géomètre". In: *Proceedings of the 15th European Workshop on Computational Geometry*, pp. 7-19, Antibes – Juan-les-Pins, March 1999.

[11] Cabri Project, academic site: http://www.cabri.imag.fr/cabri/index-e.html, Fevereiro, 2004.

[12] FINZER, W. F., BENNET, D. B. "From Drawing to Construction with The Geometer's Sketchpad: Understanding the difference between a drawing and a construction". *Mathematics Teacher*, pp. 428-431, 1995.

[13] JACKIW, *The Geometer's Sketchpad version 3.0.* 1 ed, Key Curriculum Press, CA, 1995.

[14] BENNET, D. *Exploring Geometry with the Geometer's Sketchpad.* 3 ed., Key Curriculum, Emeryville, CA, 1999.

[15] The Geometer's Sketchpad: http://www.keypress.com/sketchpad. Fevereiro, 2004.

[16] VAN, L. N. *Prise en compte de l'usager enseignant dans la conception des EIAO. Illustration dans Calques 3D.* Ph. D. dissertation. University Henri Poincaré / Nancy I., França, 1999.

[17] QASEN, S. *Conception et réalisation d'une interface 3D pour Cabri Géomètre.* Ph. D dissertation, Directeur de thèse Jean-Marie Laborde. Laboratoire Leibniz – IMAG, France, December 1997.

[18] RICHTER-GEBERT, KORTENCAMP, U. *The Interactive Geometry Software Cinderella* (CD-rom with booklet). Springer-Verlag, 1999.

[19] WILSON, D. "An exploration into the Teaching and Learning of Geometry and the Possible Effects of Dynamic Geometry Software". Internet: http://s13a.math.aca.mmu.ac.uk/Daves_Articles/PI/Contents.html, January 1999.

[20] NELSON, G., HEYDON, A. *Juno-2 Language definition,* SRC Technical Note 1997-009, June, 1997.

[21] BOUHINEAU, D., CHANNAC, S. "La programmation logique par constraintes pour l'aide à l'enseignant." In: *Proceedings of 3rh International Conference on Intelligent Touring Systems*, Montréal, Canada, 1996.

[22] CHANNAC, S. "Techniques d'Intelligence Artificielle pour l'Exécution de Programmes Logiques Géométriques". *Journées Infographie Intéractive et Intelligence Artificielle Limoges,* 1996.

[23] HEYDON, A., NELSON, G. *The Juno-2 Constrained-Based Drawing Editor.* In: SRC Research Report 131a, Digital Equipment, 1994.

[24] WINROTH, H. *Dynamic projective geometry*, Ph. D. dissertation, Department of Numerical Analysis and Computer Science, Stockholm, 1999.

[25] TRILLING, L. "On Dynamic Geometry and Logic Geometric Programming". In: *Proceedings of the International Workshop on Constraints for Graphics and Visualization CGV'95 in association with First International Conference on Constraints Programming CP'95,* pp. 143-144, Monash University, Australie, September 18-22, 1995.

[26] ALLEN, R., TRILLING, L. "Dynamic Geometry and Declarative Geometric Programming", *Mathematical Association of America and American Mathematical*

Society Joint Meeting MAA-AMS 95, Contributed Paper (session on Dynamic Geometry, pub. In Geometry turned on!, MAA Notes 41), San Francisco, January 1995.

[27] ALLEN, R., IDT, J., TRILLING, L. "Constrained Based Automatic Construction and Manipulation of Geometric Figures", *IJCAI-93- 13th International Conference on Artificial Inteligence, Morgan Kaufman,* pp. 453-460, Chambery, August 29 – September 3, 1993.

[28] CUCKER, F., ROJAS, J. M. "Foundations of Computational Mathematics". In: *Proceedings of Smalefest 2000*, Homg Cong, pp. 13-17, July 2000.

[29] NORMAN, I. B. "Human Body Models and Animation". In: *IEEE Computer Graphics and Applications,* v. 2, n. 9, pp. 6-7, November 1982.

[30] NORMAN, I. B., O'ROURKE, J., KAUFMAN, B. "Special Problems in Human Movement Simulation". In: *Proceedings of ACM: Computer Graphics*, v. 14, n. 3, pp. 189-197, 1980.

[31] CALVERT, T. W., CHARPMAN, J., PATLA, A. "Aspects of the Kinematic Simulation of Human Movement". In: *Proceedings of IEEE: Computer Graphics and Applications*, v.2, n.9, p.41-50, 1982.

[32] MAUREL, W., THALMANN, D., HOFFMEYER, P. *et al.*, "A Biomechanical Musculoskeletal Model of Human Upper Limb for Dynamic Simulation". In: *Proceedings of 7yh Eurographics International Workshop on Computer Animation and Simulation*, pp. 121-136, Poitiers, France, 1996.

[33] MAUREL, W., *3D Modeling of the Human Upper Limb Including the Biomechanics of Joints, Muscles and Soft Tissues.* Ph. D. dissertation, Laboratoire d'Infographie – Ecole Polytechnique Federale de Lausanne, Paris, 1999.

[34] MAUREL, W., THALMANN, D. "Human Shoulder Modeling including Scapulo Thoracic Constraint and Joint Sinus Cones". In: *Computer Graphics Lab Swiss Federal*

Institute of Technology. DI-LIG, EPFL, CH-1015 Lausanne, Switzerland, v. 24, n. 2, pp. 203-218, 2000.

[35] LAPIERRE, J., WILHELMS, J. *Matching Anatomy to Model for Articulated Body Animation*, M. Sc. Thesis, Computer Science Dept., UCSC, Canada, 2000.

[36] WATT, A., WATT, M. *Advanced Animation and Rendering Techniques – Theory and Practice*. 1 ed., Addison-Wesley, 1992.

[37] SHOEMAKE, K. "Animating Rotation with Quaternion Curves". In: *Proceedings of SIGGRAPH 85*, v. 19, n. 3, pp. 245-254, 1985.

[38] KLEIN, C. A., HUANG, C. H. "Review of Pseudoinverse control for Use with Kinematically Redundant Manipulators". In: *Proceedings of IEEE Trans. Systems, Man, Cybernetics*, v. SMC-13, pp. 245-250, 1983.

[39] KOREIN, J. U., NORMAN, I. B. "Techniques for Generating the Goal-Directed Motion of Articulated Structures". In: *Proceedings of IEEE: Computer Graphics and Applications*, v. 2, n. 9, pp. 71-81, 1982.

[40] ZHAO, J., NORMAN I. B. "Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures". In: *Proceedings of ACM Transactions: Graphics*, v. 13, n. 4, pp. 313-316, 1994.

[41] NORMAN I. B., KAMRAN H. M., GRAHAM W. "Articulated Figure Positioning by Multiple Constraints". In: *Praceedings of IEEE: Computer Graphics and Application*, v. 7, n. 6, pp. 28-38, June 1987.

[42] FEATHERSTONE, R. "Positioning and Velocity Transformations between Robot End-Effector Coordinates and Joint Angles". *International Journal of Robot Research*, v. 3, n. 2, pp.35-45, 1983.

[43] ANGELES, J. *Fundamentals of Robotic Mechanical Systems. Theory, Methods, and Algorithms,* 2 ed., Springer-Verlag, New York, 2002.

[44] GUEAIEB, W., KARRAY, F., AL-SHARHAN S. "Computational Intelligence Based Approach for the Joint Trajectory Generation of Cooperative Robotic Systems". In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Nashville, TN, USA, pp. 3687-3692, October 2000.

[45] HELMY A. *Parallel processing and the dynamics and kinematics of a three degree of freedom planar manipulator.* M. Sc. Dissertation, McGill University, Montreal, Canada, 1994.

[46] BULCA F., ANGELES J., ZSOMBOR-MURRAY P. J. "On the kinematics of mechanisms with redundant loops". In: *Proceedings of 12th Symposium on Engineering Applications of Mechanics (Interaction of Fluids, Structures and Mechanisms)*, pp. 63-72, Montreal, June, 1994.

[47] KOREIN, J. U. *A Geometric Investigation Of Reach.* Ph. D. dissertation, University of Pennsylvania, USA, 1985.

[48] TOLANI, D., BADLER, N. "Real-Time Inverse Kinematics of the Human Arm". *Presence*, MIT Press, v. 5, n. 4, pp. 393-401, 1996.

[49] WHITNEY, D.E. "Resolved Motion Rate Control of Manipulators and Human Prostheses". In: *Proceedings of IEEE Trans. On Man-Machine Systems*, v. 10, n. 2, pp. 47-53, 1969.

[50] ORTHEGA, J. M., RHEINBOLDT, W. C. "Interactive solution of nonlinear equations in several variables". Academic Press, New York, 1970.

[51] BOULLION T.,ODELL, P. L. "Generalized Inverse Matrices". John Wiley and Sons, New York, 1971.

[52] BEN-ISRAEL, A., GREVILLE, T. N. E. "Generalized inverses: theory and applications", John Wiley and Sons, 1974.

[53] DEO, A. S., WALKER, I. D. "Minimum Effort Inverse Kinematics for Redundant Manipulators". In: *Proceedings of IEEE Transactions on Robotics and Automation*, v. 13, n. 5, pp. 767 – 775, 1997.

[54] LIÉGEOIS, A."Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms". In: *Proceedings of IEEE Trans. On Systems, Man, and Cybernetics*, v. SMC-7, n. 12, pp. 868-871, 1977.

[55] WELMAN, C. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*, M. Sc dissertation, Simon Fraser University, Vancouver, Canada, 1993.

[56] WOLOVICH, W. A., ELLIOT, H. "A computational technique for inverse kinematics". In: *Proceedings of 32nd Conference on Decision and Control*, pp. 1359-1362, Dec. 1984.

[57] SCIAVICCO, L., SICILIANO, B. "A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulators". In: *Proceedings of IEEE Journal of Robotics and Automation*, v. 4, pp. 403-410, 1988.

[58] DAS, H., SLOTINE, J-J., SHERIDAN, T. B. "Inverse kinematic algoritms for redundant systems". In: *Proceedings of IEEE ICRA (International Conference on Robotics and Automation)*, pp. 43-48, 1988.

[59] BARAFF, D. "Non-penetrating rigid bodysimulation". State of the Art Reports, Eurographics 93, Ch. 2, Barcelona, 1993.

[60] GRZESZCZUK, R., TERZOPOULOS, D., HINTON, G. "Neuroanimator: fast neural network emulation and control of physics-based models". In: *Proceedings of the*

25th annual conference on Computer graphics and interactive techniques, pp. 9-20. ACM Press, 1998.

[61] Hodgins J. K., Wooten W. L., Brogan D. C. *et al*, "Animating Human Athletics". *Proceedings of Siggraph 95: Computer Graphics*. pp 71-78, 1995.

[62] BARAFF, D. "Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation". In: *Proceedings of Siggraph 90: Computer Graphics*, v.24, n. 4, Dallas, Aug. 1990.

[63] BOULIC R., MAS R., THALMANN, D. *Position Control of the Center of Mass for Articulated Figures in Multiple Support.* In: 6th Eurographics Int. Workshop on Computer Animation and Simulation, Maastricht, D. Terzopoulos and D. Thalmann eds., ISBN 3-211-82738-2, Springer-Verlag Wien, September 1995.

[64] OSHITA, M., MAKINOUCHI, A. "A dynamic Motion Control Technique for Human-like Articulated Figures". In: *Proceedings of Eurographics 2001.* v. 20, n. 3, 2001.

[65] AUSLANDER J., FUKUNAGA A., PARTOVI H., *et al.,* "Further Experience with Controller-Based Automatic Motion Synthesis for Articulated Figures". In: *Proceedings of ACM Transactions on Graphics*, v. 14, n. 4, pp. 311-336, 1995.

[66] NORMAN I. B., CARY B. P., BONNIE L. *Webber Simulating Humans: Computer Graphics Animation and Control.* 1 ed., Oxford University Press, Hardcover, 1993.

[67] PRATT, D. R., BARHAM, P. T., *et al.*, "Insertion of an Articulated Human into a Networked Virtual Environment". In: *Proceedings of the AI, Simulation, and Planning in High Autonomy Systems Conference*, University of Florida, Gainesville, December, 1994.

[68] Game developer. On the front line of Game Innovation: http://www.gdmag.com. Fevereiro, 2004.

[69] WITKIN, A., GLEICHER, M., WELCH, W. "Interactive dynamics". In: *Proceedings of 1990 Symposium on 3-D Interactive Graphics: Computer Graphics*. v. 24, n. 2, pp. 11-21, March 1990.

[70] HARADA, M., WITKIN, A., BARAFF, D. "Interactive physically-based manipulation of discrete/continuous models". In: *Proceedings of Siggraph 95: Computer Graphics*, pp. 199-208, Los Angeles, August, 1995.

[71] BARZEL, R., BARR, A. "Topics in Physically Based Modeling, Course Notes". In: *Proceedings of Siggraph 87*. ACM SIGGRAPH, 1987.

[72] BARZEL R, BARR ALAN H. "A Modeling System Based on Dynamic Constraints". In: *Proceedings of Siggraph 88: Computer Graphics*, v. 22, n. 4, pp. 179-188, 1988.

[73] ISSACS, P., COHEN, M. "Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics". In: *Proceedings of Siggraph 87: Computer Graphics*, v. 21, n. 4, pp. 215-224, July 1987.

[74] PLATT, J. *Constraint Methods for Neural Networks and Computer Graphics*. Ph. D. dissertation, California Institute of Technology, California, USA, 1989.

[75] PLATT, J. C., BARR, A. H., "Constraint methods for flexible models". In: *Proceedings of Siggraph: Computer Graphics*, v. 22, n. 4, pp. 279-288, 1988.

[76] WITKIN, A., FLEISCHER, K., BARR, A. "Energy constraints on parameterized models". In: *Proceedings of Siggraph 87: Computer Graphics*, v.21, n. 4, pp. 225-232, July 1987.

[77] WITKIN, A., KASS, M. "Spacetime Constraints". In: *Proceedings of SIGGRAPH 88*: Computer Graphics, Annual Conference Series, pp. 159-168, ACM SIGGRAPH August 1988.

[78] FOLEY, J. D., DAM, A. V., FEINER S. K., *et al.*, *Computer Graphics, Principles and Practice*, 2nd ed., Addison-Wesley, 1995.

[79] MASON WOO, JACKIE NEIDER, TOM DAVIS *et al.*, *OpenGL Programming Guide,* 3 ed. New York, Addison-Wesley, 1999.

[80] 3D Controls: http://www.psychology.nottingham.ac.uk/staff/cr1/3d.html. Fevereiro, 2004.

[81] SHOEMAKE, K. "ARKBALL: a user interface for specifying tree-dimensional orientation using a mouse". In: *Proceedings of the conference on Graphics interface 92*, pp. 151-156, 1992.

[82] ALLEN, R., TRILLING, L. *Dynamic Geometry and Declarative Geometric Programming. Geometry Turned On.* Mathematical Association of America, 1997.

[83] BAULAC, Y. BELLEMAIN, F., LABORDE, J. M. *Cabri, the Interative Geometry Notebook.* Brooks/Cole Publishing Company, 1992.

[84] ALLEN, R., IDT, J., TRILLING, L. "Constrained Based Automatic Construction and Manipulation of Geometric Figures". In: *Proceedings of 13th IJCAI (Int. Joint Conference on Artificial Intelligence)*, Chamberg, France, 1993.

[85] JONES, K. "Learning Geometrical Concepts using Dynamic Geometry Software". In: *Kay Irwin (Ed), Mathematics Education Research: A catalyst for change.* University of Auckland, Auckland, 2001.

[86] WILSON, D. "An Exploration into the Teaching and Learning of Geometry and the Possible Effects of Dynamic Geometry Software", January, 1999. Available at http://s13a.math.aca.mmu.ac.uk/Daves_Articles/PI/Contents.html.

[87] LABORE, J. M. "Some issues raised by the development of implemented dynamic geometry as with Cabri Géomètre". In: *Proceedings of the 15th European Workshop on Computational Geometry*, pp. 7-19, Antibes, Juan-les-Pins, March 1999.

[88] RICHTER-GEBERT, J., KORTENKAMP, U. *OO – Graphikprogrammierung in Java*. In: Technical report, ETH Zürich 1997-1999.

[89] RICHTER-GEBERT, J., KORTENKAMP, U. "Complexity Issues in Dynamic Geometry". In: *Foundations of Computational Mathematics (Proceedings of the Smale Fest 2000)*, World Scientific, Freie Universität Berlin, 2002.

[90] MathsNet: http://www.mathsnet.net/dynamic, Fevereiro, 2004.

[91] Cinderella: http://www.cinderella.de/Demo/Standalone/cindy.html. Julho, 2003.

[92] LABORDE, J. M. *Exploring non-euclidian geometry in a dynamic geometry enviroment like Cabri Géomètre*. In: James King and Doris Schattschneider, editors, Geometry Turned On, v. 41 of MAA Notes, pp. 185-192. MAA, 1997.

[93] ARMSTRONG, W., GREEN, W., MARK W. "The Dynamics of Articulated Rigid Bodies for Purposes of Animation". In: *Proceedings of Graphics Interfafe 85*, pp. 407-415, 1985.

[94] GIRARD, M., MACIEJEWSKI. A. A."Computational Modeling for the Computer Animation of Legged Figures". In: *Proceedings of Siggraph 85*, pp. 263-270, 1985.

[95] SCHRÖDER, P., ZELTZER, D. "The virtual erector set: dynamic simulation with linear recursive constraint propagation". In: *Proceedings of the 1990 symposium on Interactive 3D graphics*, p.23-31, February 1990.

[96] WILHELMS, J. P., BARSKY, B. A. "Using dynamic analysis to animate articulated bodies such as humans and robots". In: *Proceedings of Graphics Interface 85 on Computer-generated images: the state of the art*, p.209-229, January 1986.

[97] ALLEN, R."Current Perspectives on the Teaching and Learning of Geometry in a Dynamic Environment," *Research Presession, Annual Meeting of NCTM*, San Francisco, pp. 19-24, 1999.

[98] DONALD, B., XAVIER, P., CANNY J., *et al.,* "Kinodinamic Motion Planning". *Journal of the ACM*, v. 40 n.5, pp. 1048-1066, November 1993.

[99] COHEN, M. F. "Interactive Spacetime Control for Animation". In: *Proceedings of Siggraph 92: Computer Graphics*, Annual Conference Series, pp. 293-302. ACM *Siggraph* 92, July 1992.

[100] THOMAS J. N., MARKS, J. "Spacetime Constraints Revised". In: *Proceedings of Siggraph 93: Computer Graphics*, Annual Conference Series, pp. 343-350. ACM *Siggraph*, August 1993.

[101] TERZOPOULOS, D., WITKIN, A., KASS, M. "Energy constraints on deformable models: recovering shape and non-rigid motion". In: *Proceedings of AAAI-87*, Seattle, 1987.

[102] JEFFREY A., THINGVOLD, COHEN, E. "Phisical Modeling with B-Spline surfaces for interactive design and animation". In: *Proceedings of Siggraph 90: Computer Graphics*, v. 24, n.2, pp. 129-138, March 1990.

[103] GLEICHER, M., WITKIN, A. "Drawing with constraints". *The Visual Computer*, 1994.

[104] GLEICHER, M., WITKIN, A. "Through-the-lens camera control". In: *Proceedings of Siggraph 92: Computer Graphics*, 1992.

[105] JOHNSON, J. *Gui Bloopers*, 1 ed., Morgan Kaufmann Publishers, San Francisco, 2000.

[106] POPOVIC, Z., WITKIN, A. "Physically Based Motion Transformation". In: *Proceedings of Siggraph 99: Computer Graphics*, Annual Conference Series, pp. 11-20. ACM *Siggraph*, August 1999.

[107] CARVALHO, P. C. P. *Introdução à Geometria Espacial*, 1ª ed., Gráfica Wagner Ltda, Rio de Janeiro, 1980.

[108] KLINGENER, F. *Inverse Kinematics On The Java 3D Scene Graph.* Brock Engineering, 2000.

[109] GRZESZCZUK, R., TERZOPOULOS, D., HINTON, G. "Neuro Animator: Fast Neural Network Emulation and Control of Physics-Based Models". In: *Proceedings of Siggraph 98: Computer Graphics*, Annual Conference Series, pp. 9-20. ACM *Siggraph*, July 1998.

[110] WITKIN, A., GLEICHER, M., WELCH, W. "Interactive dynamics". In: *Proceedings of 1990 Symposium on 3-D Interactive Graphics: Computer Graphics*, v. 24 n. 2, pp 11-21, March, 1990.

[111] BARZEL R., BARR, A. H. "A Modeling System Based On Dynamic Constraints". In: *Proceedings of Siggraph 87: Computer Graphics*, Annual Conference Series, pp. 179-188. ACM *Siggraph*, August 1988.

[112] K. JAMES, H. MARTIN, M. JOSEPH, *et al.* "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs". In: *IEEE Transactions on Visualization and Computer Graphics*. v. 24, pp. 21-36. January, 1999.