Framework para Construção de Pacientes Virtuais: Uma Aplicação em Laparoscopia Virtual

Carla Maria Dal Sasso Freitas[1], Isabel Harb Manssour[2], Luciana Porcher Nedel[1], Julierme Krüger Gavião[1], Thiago Corrêa Paim[1], Anderson Maciel[3]

[1]Instituto de Informática/PPGC – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

[2]Faculdade de Informática – Pontifícia Universidade Católica do R.G. do Sul (PUCRS)
Av. Ipiranga, 6681 – Prédio 30 – 90.619-900 – Porto Alegre – RS – Brasil

[3]Virtual Reality Lab – Swiss Federal Institute of Technology (EPFL)
CH 1015 – Lausanne – VD – Switzerland

{carla,nedel,julierme,tpaim}@inf.ufrgs.br,
manssour@inf.pucrs.br,anderson.maciel@epfl.ch

Abstract. *The paper describes the framework VPAT ("Virtual Patients") for virtual reality and computer graphics applications in medicine. VPAT is an object-oriented framework which has been used as a basis for several graphics applications in our laboratory. Recently, it has been extended to support applications which use a virtual laparoscopy device.*

Resumo. *Este artigo descreve o framework VPAT ("Virtual Patients") que suporta aplicações de realidade virtual e computação gráfica na Medicina. VPAT é orientado a objetos e tem sido usado para uma série de aplicações em nosso laboratório. Recentemente foi estendido para suportar aplicações que fazem uso de um dispositivo de laparoscopia virtual.*

1. Introdução

Na área médica, as modalidades de diagnóstico por imagens se estabeleceram já há algum tempo, valendo-se de uma variedade de equipamentos de aquisição como tomografia computadorizada, ressonância magnética e ultra-som. As imagens revelam desde órgãos completos até estruturas macroscópicas como, por exemplo, tumores e pólipos, ou processos ocorridos ou em desenvolvimento nessas estruturas, tais como inflamações ou fraturas.

Freqüentemente, a análise visual de imagens é insuficiente para a determinação de características das estruturas em estudo e faz-se necessário obter modelos geométricos que as representem e que possibilitem a extração de medidas e a simulação de procedimentos. Assim, a idéia de criar "pacientes virtuais", ou melhor, modelos de representação de humanos virtuais (mesmo que parciais), para uso em aplicações de computação gráfica e realidade virtual na área médica, representa uma linha seguida por diferentes grupos [Grimson et al. 1999, Preim et al. 2002]. Tais pesquisas visam permitir o melhor entendimento da forma humana, suas funções e seu desenvolvimento. Analogamente aos simuladores de vôo, este tipo de tecnologia permitiria a um estudante participar

seguida e repetidamente de situações raras e/ou de emergência. Assim, um modelo de paciente virtual objetiva também a educação em diversos níveis, especialmente para profissionais na área da saúde, os quais, em futuro próximo, poderão ser treinados para desenvolver, ensinar e diagnosticar usando pacientes eletrônicos virtuais. Alguns sistemas comerciais já estão disponíveis (ver *LapSim BasicSkills*, http://surgical-science.com/, e *Kismet - Kinematic Simulation, Monitoring and Off-Line Programming Environment for Telerobotics*, http://iregt1.iai.fzk.de/), mas são complexos e bastante onerosos, não representando uma solução viável para a maioria dos centros médico-hospitalares.

A construção de um modelo humano virtual pode ser dividida em três etapas básicas: obtenção das imagens médicas; reconstrução tridimensional do modelo; e simulação de movimento e deformações.

A primeira etapa consiste na aquisição e processamento de imagens médicas, usadas como base para a reconstrução tridimensional das partes que compõem o corpo humano. Nesta etapa, são empregadas técnicas usuais para melhoria das imagens e seu registro (alinhamento) para aplicações específicas, se for o caso. Na segunda etapa, métodos de segmentação, modelagem geométrica e topológica de elementos básicos (i.e., ossos, músculos, pele, tendões, vasos, gordura, etc.) fornecem os componentes fundamentais para a reconstrução do corpo humano. Finalmente, a terceira etapa do processo consiste na simulação do movimento e deformação dos tecidos, sendo necessários algoritmos para simulação de corpos humanos baseados em conceitos anatômicos, suprindo assim às necessidades impostas ao uso de seres humanos virtuais em aplicações médicas.

Considerando as etapas de modelagem mencionadas, o *framework* VPat ("*Virtual Patients*") foi projetado com o propósito de suportar o desenvolvimento de aplicações de computação gráfica na área médica. O *framework* serviu de base tanto para ferramenta de visualização direta de imagens médicas [Manssour *et al.* 2002] como para a simulação anatomicamente embasada de movimentos articulares [Maciel *et al.* 2002].

Neste artigo, será descrito o *framework* VPat (seção 2) e sua utilização nos trabalhos mencionados, assim como sua extensão para o suporte ao treinamento em laparoscopia através de procedimentos laparoscópicos virtuais (seção 3). A seção 4 apresenta a aplicação experimental desenvolvida, enquanto a seção 5 discute alguns sistemas descritos na literatura. Comentário finais são apresentados na seção 6.

2. O *Framework* VPat

VPat é um *framework* orientado a objetos, que contém classes básicas, cujas funcionalidades podem ser compartilhadas ou estendidas, permitindo o desenvolvimento de classes mais especializadas que implementem, por exemplo, algoritmos complexos de visualização ou simulação de movimento. O seu modelo conceitual foi construído de maneira a permitir que sistemas de visualização e exploração de dados médicos possam ser fácil e rapidamente projetados e desenvolvidos.

As classes básicas do *framework* são descritas na seção 2.1, enquanto a seção 2.2 apresenta sua utilização em trabalhos anteriores [Manssour *et al.* 2002, Maciel *et al.* 2002], para visualização e simulação de movimentos de articulações respectivamente.

2.1. Conjunto de classes básicas

A Figura 1 apresenta o conjunto de classes fundamentais do VPat, na forma de um diagrama UML (*Unified Modeling Language*) [Larman 1997]. As classes foram concebidas e implementadas (em C++) de forma independente de plataforma, assumindo o modelo MVC (*Model-View-Controller*) [Buschmann *et al.* 1996] como base para o desenvolvimento de aplicações. Assim, o *framework* conta com classes básicas que representam desde primitivas gráficas tipo "ponto" e classes de mais alto nível, que representam o modelo (objetos gráficos, cenas completas), vistas, etc.

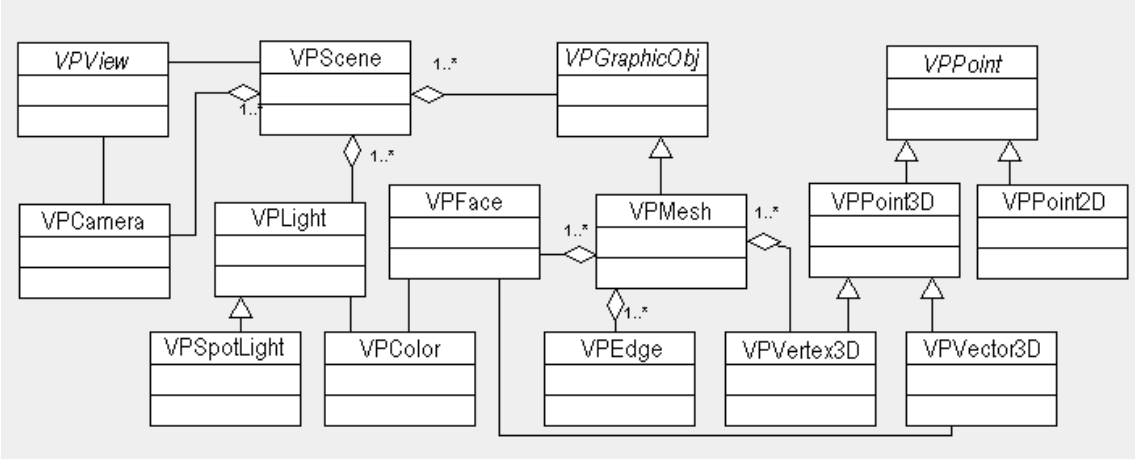


Figura 1. Diagrama de classes básicas do *framework* VPAT

Além de pontos, outras entidades primitivas como vértice, vetor, matriz e cor são implementadas por classes VPat. Pontos são definidos de forma genérica na classe abstrata *VPPoint* e implementados em suas especializações, as classes *VPPoint2D* e *VPPoint3D*. Um vértice, por sua vez, é representado pela classe *VPVertex3D*, que consiste em um ponto associado a um vetor normal com origem nesse ponto. Um vetor é definido pela classe *VPVector3D*, a qual implementa operações com vetores e pontos, tais como adição, produto escalar e produto vetorial. A cor aplicada aos elementos gráficos de uma cena é especificada a partir de instâncias da classe *VPColor* que agrega as componentes básicas RGB e a componente de transparência α.

Outros elementos usualmente presentes em aplicações gráficas são câmeras e luzes. A classe *VPCamera* implementa esse objeto fundamental para o processo de visualização, agregando basicamente a posição e orientação do observador de uma cena, a distância focal, o tipo de projeção e os planos de corte utilizados para eliminar elementos da imagem. Luzes caracterizam-se por atributos como cor, intensidade, posição e atenuação, parâmetros que são definidos na classe *VPLight*. *VPSpotLight* é uma especialização de *VPLight* que define uma direção e um ângulo de dispersão para a luz emitida.

A representação de uma cena virtual é um pré-requisito fundamental para a implementação de um sistema gráfico. Por isso, a classe *VPScene* é definida como a agregação de um conjunto de objetos gráficos, um conjunto de luzes e um conjunto de câmeras virtuais. Seu processo de exibição foi concebido com base no padrão MVC, que consiste em uma tríade de classes frequentemente usadas em sistemas interativos para construção de interfaces com o usuário. A implementação do padrão MVC mantém o núcleo funcional do sistema independente da interface. Assim, as

funcionalidades internas podem permanecer estáveis, mesmo quando a interface necessita ser alterada para se adaptar a novas plataformas e dispositivos de interação.

Um elemento gráfico da cena é definido de forma abstrata pela classe *VPGraphicObj*, a qual apenas implementa a condição de visibilidade do objeto. Assim, objetos gráficos são instanciados a partir de classes específicas que estendem a classe *VPGraphicObj*. Desse modo, a instância de *VPScene* trata os objetos gráficos de forma genérica. Um exemplo de especialialização de *VPGraphicObj* é a classe *VPMesh*, que implementa o conceito de malhas poligonais, no qual uma superfície é representada por um conjunto de faces. Essas faces são definidas a partir da classe *VPFace* que contém o conjunto de arestas componentes do polígono correspondente à face. As arestas, por sua vez, são definidas como instâncias da classe *VPEdge*, cujos atributos são índices de vértices componentes da malha, instâncias de *VPVertex3D*.

A classe *VPView* tem como objetivo apresentar o modelo, representado aqui pela classe *VPScene*, ao usuário. A visualização da cena depende dos parâmetros de uma das câmeras especificadas para a *VPScene*. Além disso, a *VPView* também é responsável pela interpretação dos eventos gerados pelos dispositivos de interação. Portanto, o modelo definido em um objeto *VPScene* é totalmente independente do modelo de interação especificado nas subclasses da *VPView* que são dependentes de uma plataforma específica. Para completar o modelo MVC, é necessária uma classe *Controller* (não exibida na figura); entretanto, esta é dependente da aplicação e, por isso, não faz parte do conjunto de classes básicas e será abordada na apresentação das aplicações específicas implementadas com o VPat.

Tanto as subclasses de *VPView* como as classes *Controller* são modeladas segundo o padrão de projeto *Observer* [Buschmann *et al.* 1996]. Assim, sempre que o modelo (*VPScene*) muda de estado, ele notifica os observadores, que devem, então, ser atualizados. Portanto, há um mecanismo de atualização automática que mantém um registro dos componentes dependentes do modelo. Alterações no estado do modelo invocam este mecanismo, que é a única ligação entre o modelo, a *VPView* e a *Controller*.

2.2. Classes específicas para visualização volumétrica e modelagem anatômica de articulações

Conforme mencionado, o *framework* VPat foi utilizado em duas aplicações específicas, com características bastante diversas: visualização volumétrica de imagens médicas e simulação do movimento de articulações anatomicamente corretas.

Na Figura 2 é apresentado o modelo UML simplificado utilizado para o desenvolvimento da aplicação envolvendo visualização volumétrica. Novas classes foram incorporadas ao VPat, estando sinalizadas em tonalidade cinza claro, mantendo-se em branco aquelas do *framework* diretamente relacionadas com estas, e em cinza mais escuro, as classes específicas da aplicação. *VPImage* é a primeira delas, uma subclasse de *VPGraphicObj*. Duas subclasses de *VPImage* foram implementadas: *VPImage2D*, para visualização e interação com uma imagem 2D, tal como uma fatia de um exame de tomografia computadorizada, e *VPVolume*, que corresponde a um volume de dados médicos como o obtido a partir de um conjunto de imagens tomográficas.

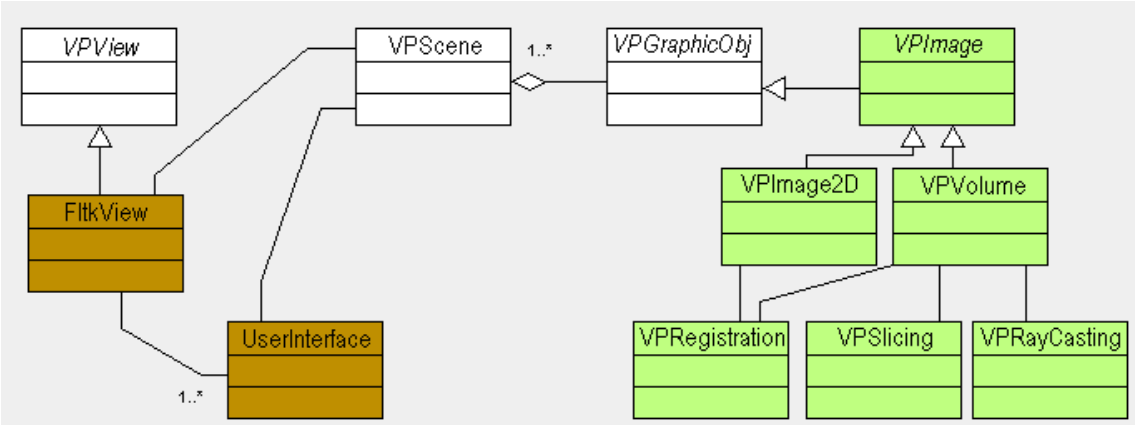


Figura 2. Extensão do framework para visualização volumétrica

A classe *VPRayCasting* contém um conjunto de métodos implementando variações do algoritmo de *ray casting* apresentado por Levoy (1988), tal como MIP, visualização de dados multimodais e de estruturas internas. Para a geração de imagens com *ray casting*, são necessárias tabelas de cor e opacidade, as quais são também implementadas como classes; suas instâncias são atributos da classe *VPVolume*, usadas pela *VPRayCasting*. A classe *VPSlicing* é responsável pela geração de imagens de fatias ortogonais ao volume, isto é, imagens dos planos axial, coronal e sagital. A classe *VPRegistration* está associada com as classes *VPVolume* e *VPImage2D*, pois fornece métodos para realizar o registro (alinhamento) entre imagens 2D ou entre volumes.

A aplicação para visualização volumétrica contêm a classe *UserInterface*, como *Controller* e uma classe estendendo *VPView*. A aplicação foi desenvolvida em plataforma Windows®, sendo essas duas classes implementadas sobre a API *FLTK (Fast Light ToolKit* (http://www.fltk.org).

No caso de modelagem anatômica de objetos articulados, a aplicação foi desenvolvida em ambiente Linux, sendo as classes básicas utilizadas sem modificações. O diagrama simplificado, apresentado na Figura 3, mostra as classes mais relevantes para essa aplicação, seguindo a mesma convenção de cores.

Algumas das classes foram agregadas ao próprio *framework*: por exemplo, a classe *VPJoint*, que utiliza uma instância da classe *VPGraphicObject*, para representação geométrica dos segmentos de corpos que se articulam numa determinada junta, e a classe *VPMatrix*. A subclasse de *VPView* criada nessa aplicação foi implementada utilizando a API *OpenInventor* (http://oss.sgi.com/projects/inventor/), sendo denominada *InventorView*.

O desenvolvimento dessas duas aplicações permitiu observar a viabilidade do *framework* em duas situações diversas do ponto de vista dos modelos de “pacientes virtuais” utilizados. Na modelagem anatômica de articulações foram utilizados apenas modelos geométricos criados com métodos convencionais de modelagem e não há interação entre o usuário e o modelo. A classe *InventorView* valeu-se das facilidades interativas do próprio *OpenInventor*, não sendo necessário o tratamento de interação pelas classes introduzidas. Já na aplicação de visualização volumétrica, os modelos correspondem a dados adquiridos da natureza e a aplicação foi desenvolvida para proporcionar alta taxa de interatividade para um usuário final.

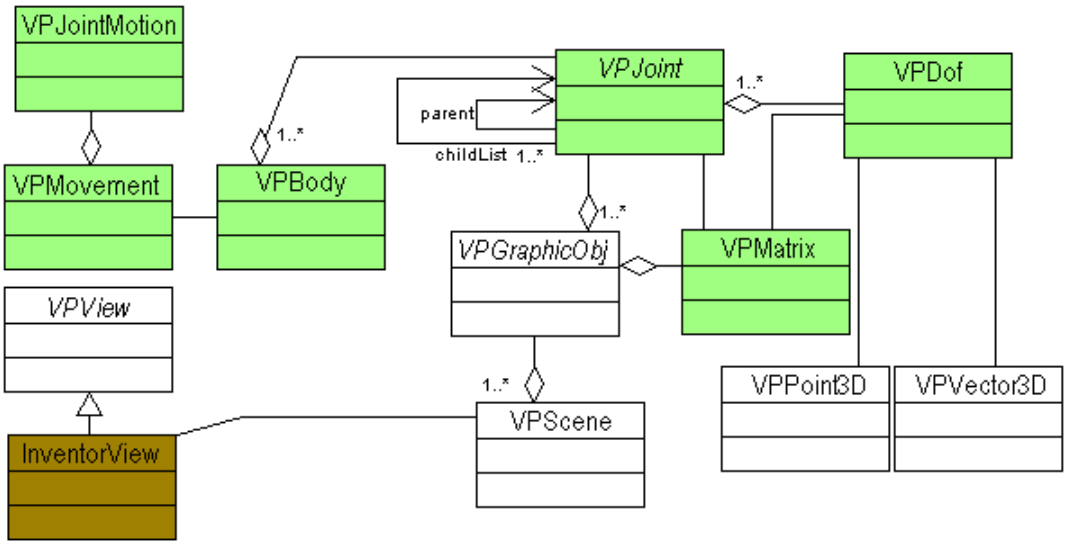


Figura 3. Diagrama de classes para modelagem de corpos articulados

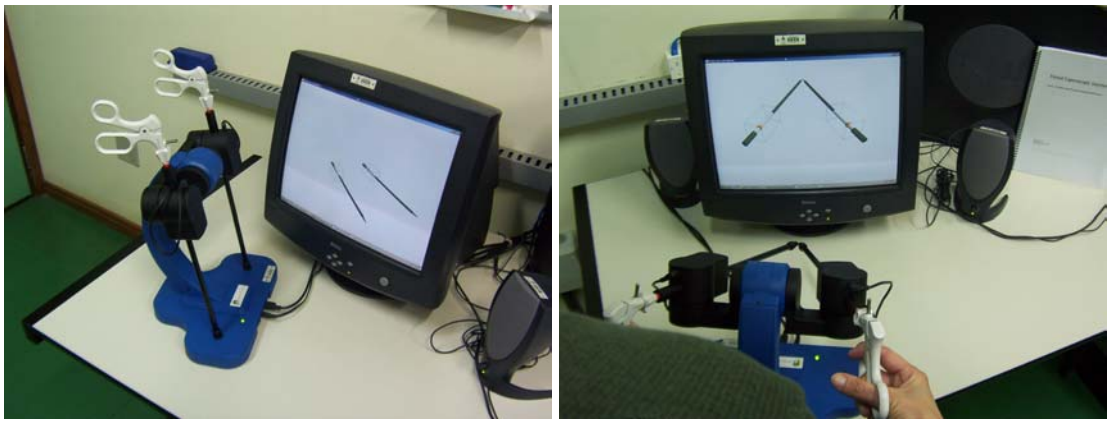
3. Extensão do *framework* VPAT para aplicação de laparoscopia virtual

Nesta seção, é apresentada a extensão do VPat para uma aplicação utilizando dispositivos de realidade virtual, o que permitiu avaliar a facilidade de uso do *framework*, face às necessidades de interação e de representação geométrica desses dispositivos num cenário virtual.

3.1. A interface de laparoscopia virtual

A interface para laparoscopia virtual VLI (*Virtual Laparoscopic Interface*) da *Immersion Corporation* (http://www.immersion.com) é uma interface projetada especificamente para simulações de procedimentos cirúrgicos laparoscópicos em realidade virtual (Figura 4a). O equipamento pode ser visto como um instrumento cirúrgico que se comunica com um computador por meio de uma porta serial e pode ser usado para simular uma grande variedade de ferramentas reais de manipulação cirúrgica.

Cada ferramenta é montada sobre uma junta de modo a permitir a atuação do cirurgião na região da incisão dos instrumentos num paciente. Na Figura 4a, as hastes em preto correspondem aos instrumentos simulados. Na extremidade diretamente usada pelo usuário aparece o manipulador (Figura 4b), de cor branca, semelhante aos manipuladores de tesouras; eles permitem controlar os instrumentos simulados. A ponta da ferramenta simula um instrumento cirúrgico de laparoscopia, a escolha do usuário e de acordo com a aplicação. Para alguns desses instrumentos, devido a sua natureza e utilização como, por exemplo, instrumentos de corte, a abertura/fechamento do manipulador é um grau de liberdade com pouca utilização.



(a) **(b)**

Figura 4. Interface de laparoscopia virtual (Immersion Corporation). (a) Instrumentos em situação de repouso. (b) Manipulação dos instrumentos cirúrgicos simulados.

O *hardware* permite a um computador rastrear precisamente os movimentos do operador utilizando as duas ferramentas cirúrgicas ao mesmo tempo. O dispositivo é um sistema dual de rastreamento com 5 graus de liberdade (Figura 5): três graus de liberdade relativos a rotações, informadas como ângulos de Euler, um grau referente ao movimento de inserção/retração da ferramenta, mais um grau de liberdade para o movimento de abertura/fechamento das pinças ou tesouras. Os ângulos são informados em radianos (α, β, χ), o movimento de retração e inserção (R) em centímetros e o movimento de abertura e fechamento em termos de um intervalo entre os valores mínimo e máximo de abertura.

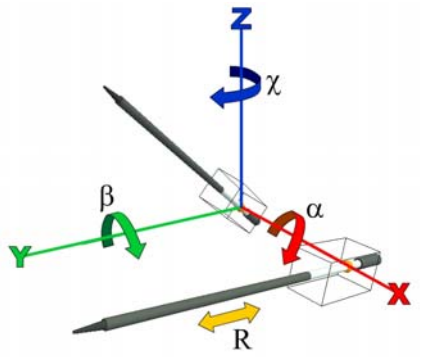


Figura 5. Sistema de referência da Interface de Laparoscopia

3.2. As classes para laparoscopia virtual

Utilizando as classes básicas do *framework* VPat, foram definidas classes específicas para uma aplicação de laparoscopia virtual [Gavião 2003] (Figura 6). Estas classes foram implementadas em ambiente Linux, a exemplo de Maciel *et al.* (2002).

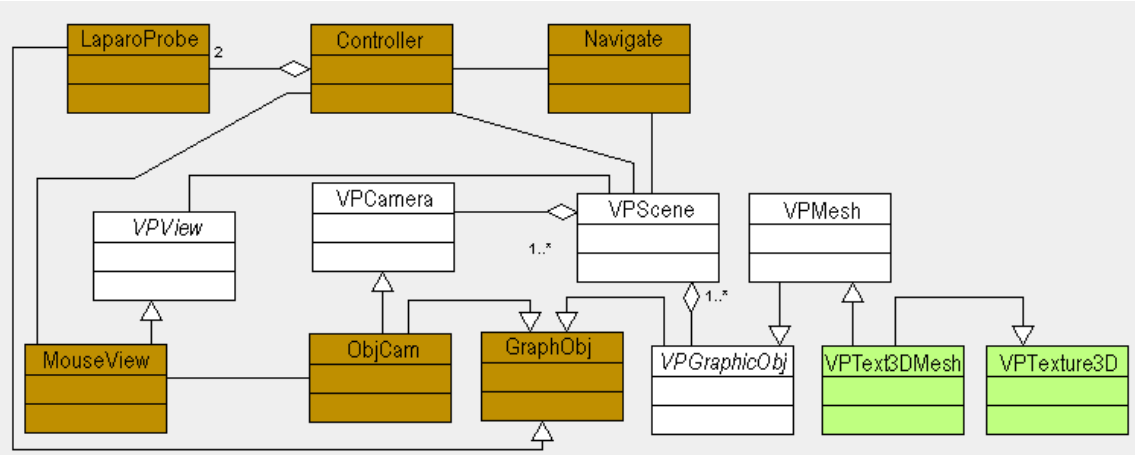


Figura 6. Extensão para a aplicação de laparoscopia virtual

A classe *Controller* gerencia os elementos componentes de uma cena. Todas as vistas (*views*) de uma aplicação de laparoscopia virtual estão acopladas a uma única instância de *Controller*. Por isso, alterações na cena promovidas a partir de uma determinada visão são imediatamente refletidas sobre as outras. Assim sendo, pode-se dizer que esta classe é o elo entre as ações desencadeadas a partir da interface com o usuário e o modelo definido por *VPScene.*

A classe *GraphObj* especializa as definições da classe *VPGraphicObj,* incorporando ao objeto um sistema de referência local e uma posição. Dessa forma é possível realizar computações com vértices em relação ao sistema de referência local. Essas definições de *GraphObj* são importantes para a representação do objeto que simula a sonda laparoscópica, representada no modelo através da classe *LaparoProbe*. A proposta dessa classe é simular os movimentos rotacionais e translacionais obtidos do rastreamento da VLI (ver Figura 5), incluindo a abertura e fechamento dos manipuladores para tocar, segurar ou cortar tecidos.

Quando for desejável ter uma representação visual para uma câmera, deve-se utilizar uma instância de *ObjCam*. Esta classe estende as funcionalidades definidas em *GraphObj* e *VPCamera* para desenhar um *frustum* de visualização. Além disso, ela define uma fonte de luz acoplada logo acima de sua posição e orientação atual. Dessa forma, à medida que a câmera navega através da cena, a fonte de luz a acompanha, simulando o efeito gerado a partir da movimentação das micro-câmeras usadas em cirurgias endoscópicas. Durante a manipulação da câmera ou de qualquer outro objeto da cena, podem ocorrer colisões. O mecanismo de detecção é implementado na classe *Navigate,* que é gerenciada pela classe *Controller*, a qual recebe notificações dos eventos de *hardware.*

Por fim, a classe *MouseView* é definida como a entidade responsável por atender os eventos de *Mouse*. Cada instância dessa classe representa uma visão sobre a cena. Toda interação desse dispositivo, movimentação e clique de botões sobre uma visão da cena são tratados por esta classe. Para simplificar os aspectos de tratamento de interação, a exemplo da aplicação de visualização volumétrica de Manssour *et al.* (2002), para implementação da *MouseView* foi utilizada a API *FLTK*, mas desta vez em ambiente Linux. Assim, a *MouseView* corresponde a uma *FLTKView*.

4. Aplicação experimental sobre o framework VPat

Uma aplicação de treinamento foi desenvolvida sobre o framework VPat, utilizando as classes específicas anteriormente apresentadas.

Nesta aplicação, o usuário pode navegar por uma estrutura geométrica do tipo malha de triângulos (uma instância de *VPMesh* - Figura 7a) representando um cólon, sobre o qual é possível interagir com instrumentos simulados. O objetivo é treinar cortes em tecidos, manipulando os instrumentos com a interface de laparoscopia virtual.

Inicialmente, duas visões da cena são apresentadas ao usuário: a primeira mostra o ponto de vista de um observador localizado na origem do referencial da cena (Figura 7a); a segunda mostra o ponto de vista que teria um observador manipulando a interface como apresentado na Figura 4b. Nesta posição, com uma fonte de luz do tipo *spot*, associada à câmera, simulando o efeito de iluminação de uma *head light*, o observador teria a visão mostrada na Figura 7c.

É possível designar dois tipos de instrumentos nas pontas das ferramentas de simulação de instrumentos laparoscópicos. O primeiro é um cilindro simulando um agarrador ou uma pinça fechada. O segundo imita um instrumento de corte e é usado para efetuar cortes na malha de triângulos que representa o órgão simulado (Figura 7b e 7c).

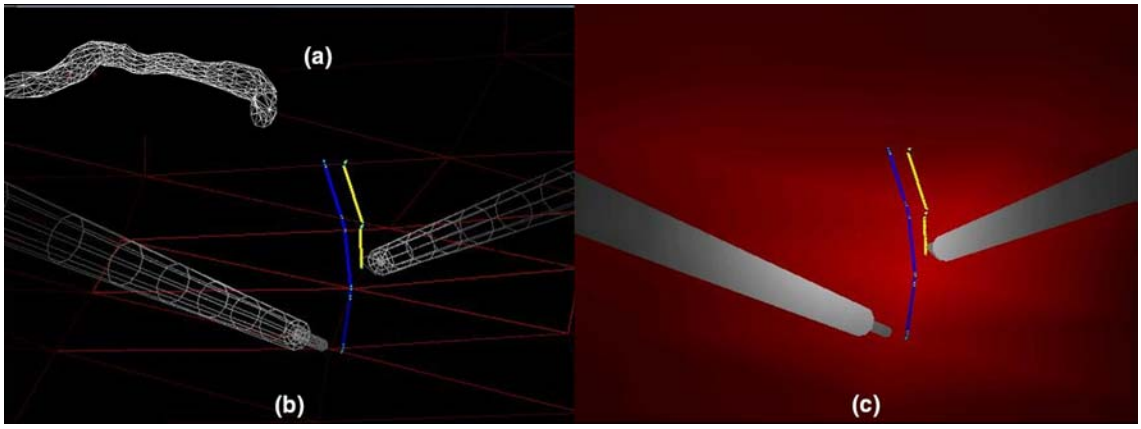


Figura 7. (a) Visão exterior da malha do órgão simulado com uma instância de *VPMesh* (topo a esquerda). (b) Visão interna da simulação de corte com duas instâncias da classe *LaparoProbe* representada em *wireframe* e (c) com iluminação.

Com a pinça é definido um trajeto através do qual o usuário deve executar um corte. O usuário, então, aplica o corte junto ao trajeto com a ponteira de corte. Ao final desse processo, é apresentada uma avaliação do desempenho do usuário, baseada na fidelidade do corte realizado em relação ao trajeto definido.

5. Discussão

Dos ambientes genéricos, orientados a objetos, para construção de aplicações em computação gráfica, VTK – *Visualization Toolkit* [Schroeder *et al.* 2003] é um dos mais conhecidos, sendo constituído por um conjunto de classes para processamento de imagens e visualização científica, que podem ser usadas para aplicações de computação gráfica na Medicina. VROOM (*Volume Rendering by Object-*

Oriented Methods) [Zuiderveld *et al.* 1996] é também uma arquitetura orientada a objetos destinada à visualização de volumes de dados médicos, mas não inclui ferramentas que permitem a simulação de cirurgias. *AnatomyBrowser* [Golland *et al.* 1998] é um *framework* que integra imagens e informações textuais em aplicações médicas, mas também não dá suporte para modelagem de procedimentos cirúrgicos. Um *framework* mais genérico, *Spring* [Montgomery *et al.* 2001], foi desenvolvido para um sistema colaborativo, multiplataforma e que suporta simulação de cirurgia em tempo real. Este sistema inclui a modelagem tanto de tecidos moles como de suturas.

Observa-se, assim, que algumas das ferramentas analisadas permitem a simulação de procedimentos laparoscópicos, enquanto outras possuem apenas funcionalidades para visualização volumétrica. Algumas aplicações correspondem a sistemas e não *frameworks* com possibilidade de extensão para incluir novos procedimentos e modelos para a construção de pacientes virtuais. Muitos são voltados apenas para um tipo de aplicação, por exemplo, procedimentos cirúrgicos virtuais ou visualização. O VPat, apesar de ter sido projetado especificamente para aplicações médicas, inclui suporte para uma variedade maior de situações que muitos ambientes similares.

Creynenbreugel *et al.* (1996) desenvolveram um ambiente orientado a objetos dedicado ao planejamento virtual de cirurgias. Este sistema, que pode ser estendido para simular vários tipos de cirurgia, permite a introdução de objetos, tais como próteses, mas não oferece tratamento de colisão entre objetos. Portanto, ao contrário do VPat, não é adequado para procedimentos de laparoscopia virtual. Já Vilanova (2002) desenvolveu um protótipo para endoscopia virtual considerando um *framework* genérico, elaborado de acordo com os métodos existentes para simular uma endoscopia. Um algoritmo que encontra o caminho central do órgão é usado para mover a câmera, mas, não inclui funcionalidades que permitam diferentes tipos de visualização do mesmo modelo virtual.

Como exemplo de sistemas específicos para simulação de procedimentos de endoscopia pode-se citar Nain (2002), que apresenta uma ferramenta para explorar modelos de qualquer topologia em simulação de endoscopia. Neste caso, o usuário fornece uma trajetória em um ambiente formado tanto pela superfície em análise quanto pelos dados originais que, sincronizados, fornecem uma visão realçada da anatomia visualizada. Uma abordagem diferente, que produz uma animação em VRML da navegação através do órgão sob análise foi apresentada por Loncaric *et al.* (2000). Neste caso, qualquer *browser* pode reproduzir a navegação obtida a partir de um *rendering* volumétrico.

6. Comentários finais

O VPat consiste em um conjunto de classes que implementa tanto algoritmos de visualização volumétrica, como funções para modelagem e manipulação de objetos geométricos articulados, além de suportar aplicações de realidade virtual. Sua utilização em três situações distintas mostrou a amplitude de aplicação do conjunto das classes escolhidas como básicas, e a facilidade de extensão, em parte inerente à abordagem orientada a objetos. Na seqüência de trabalho, utilizando as características mencionadas acima, deverá ser desenvolvido um sistema utilizando o VPAT para o planejamento de transplantes de fígado. Quanto à aplicação da aplicação experimental para treinamento em laparoscopia, as perspectivas de continuidade do trabalho incluem melhorias nos aspectos visuais do órgão simulado e realização de experimentos com usuários.

Adicionalmente, demonstrando a generalidade do *framework*, ele vem sendo utilizado para o desenvolvimento de uma aplicação cujo escopo é completamente diferente das apresentadas neste artigo e envolve a movimentação de agentes autônomos em ambientes virtuais [Torres *et al.* 2002].

Nesta aplicação estão sendo implementadas novas classes e métodos, responsáveis pela "customização" de movimentos complexos compostos pela ação simultânea de diferentes articulações.

Agradecimentos

Este trabalho vem sendo realizado com financiamento CNPq, FAPERGS e CAPES.

Referências

Buschmann, F. et al. (1996) "Pattern-Oriented Software Architecture: A System of Patterns". Chichester, UK: John Wiley & Sons.

Creynenbreugel , J.V., Verstreken, K., Marchal, G. and Suetens, P. (1996) "A Flexible Environment for Image Guided Virtual Surgery Planning". Lecture Notes in Computer Science 1131, Visualization in Biomedical Computing. Proceedings of 4th International Conference (VBC'96), Hamburg, Germany. p. 501-510.

Gavião, J.K. (2003) "Estudo sobre Aplicação de Laparoscopia Virtual no Contexto do Projeto VPat". Porto Alegre, PPGC da UFRGS (dissertação de mestrado).

Golland, P., Kikinis, R. Umans, C. et al. (1998) "AnatomyBrowser: A Framework for Integration of Medical Information". Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI). Cambridge, MA. p. 720-731.

Grimson, W.E.L., Kikinis, R., Jolesz, F.A. and Black, P.M. (1999) "Image-Guided Surgery". Scientific American, v.280, n.6, p. 54-61.

Larman, C. (1997). "Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design". Prentice-Hall, Upper Saddle River, NJ.

Levoy, M. (1988) "Volume rendering - display of surfaces from volume data". IEEE Computer Graphics and Applications, Los Alamitos, v.8, n.3, p. 29-37.

Loncaric, S., Markovinovic, T. (2000) "Web-Based Virtual Endoscopy". Medical Informatics Europe, Hannover, Germany.

Maciel, A., Nedel, L. and Freitas, C. (2002) "Anatomy based joint models for virtual humans skeletons". Proceedings of IEEE Computer Animation, Geneva, Switzerland. p.110-116.

Manssour, I.H., Furuie, S.S., Olabarriaga, S.D. and Freitas, C.M.D.S. (2002) "Visualizing Inner Structures in Multimodal Volume Data". Proceedings of XV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI). IEEE Computer Society. p. 51-58.

Montgomery, K., Bruyns, C., Brown, J. et al. (2001) "Spring: A General Framework for Collaborative, Real-Time Surgical Simulation". Proceedings of Medicine Meets Virtual Reality (MMVR02), Newport Beach, CA.

Nain, D. (2002) "An Interactive Virtual Endoscopy Tool with Automatic Path Generation". Master Thesis. MAT AI Lab.

Preim, B., Tietjen, C., Spindler, W. and Peitgen, H. (2002) “Integration of Measurement Tools in Medical 3D Visualizations”. Proceedings of IEEE Visualization Conference, Oct./Nov. 2002. p. 21-28.

Vilanova, A. (2002) “Visualization Techniques for Virtual Endoscopy”. PhD Thesis, Technische Universität Wien, Fakultät für Naturwissenschaften und Informatik.

Zuiderveld, K., Koning, A.H.J., Stokkinget, R. et al. (1996) “Multimodality visualization of medical volume data”. Computer & Graphics, Oxford, v.20, n.6, p.775-791.

Schroeder, W., Martin, K. and Lorensen, B. (2003) “The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics”. 3rd ed. Kitware, Inc.

Torres, J.A., Maciel, A. e Nedel, L.P. (2002) “Uma Arquitetura para Animação de Humanos Virtuais com Raciocínio”, In: SVR 2002 - SBC SYMPOSIUM ON VIRTUAL REALITY, 2002, Fortaleza - CE , p. 341-352.