

**Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática**

Aline Marques de Moraes

Uma Arquitetura para Programação Visual de Ambientes baseados em Realidade Virtual para Profissionais da Saúde

**João Pessoa
2012**

Aline Marques de Moraes

Uma Arquitetura para Programação Visual de Ambientes baseados em Realidade Virtual para Profissionais da Saúde

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos para a obtenção do título de Mestre.

Orientadores: Prof^a. Dr^a. Liliane dos Santos Machado.
Prof. Dr. Ronei Marcos de Moraes.

**João Pessoa
2012**

Aline Marques de Moraes

Uma Arquitetura para Programação Visual de Ambientes baseados em Realidade Virtual para Profissionais da Saúde

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba como requisito parcial para obtenção do título de Mestre em Informática.

Orientadores: Prof^ª. Dr^ª. Liliane dos Santos Machado.
Prof. Dr. Ronei Marcos de Moraes.

COMISSÃO EXAMINADORA

Prof^ª. Dr^ª. Liliane dos Santos Machado.
Orientadora (CI-UFPB)

Prof. Dr. Ronei Marcos de Moraes.
Orientador (DE-UFPB)

Prof. Dr. Leonardo Vidal Batista
Examinador Interno (CI-UFPB)

Prof. Dr. Christian Azambuja Pagot
Examinador Interno (CI-UFPB)

Prof. Dr^ª. Fátima de Lourdes dos Santos Nunes Marques
Examinadora Externa (USP)

João Pessoa, ____ de _____ de 2012.

Agradecimentos

Agradeço primeiramente a Deus por me direcionar nas escolhas certas ao longo da minha vida. Sou grata também aos meus pais, Maria e Alberto, pelos conselhos e apoio nos momentos mais estressantes do mestrado e da minha vida. Além disso, agradeço a Alana e Sarah, minhas irmãs, pelo companheirismo e parceria de sempre. Ao Eduardo, por toda paciência e compreensão nessa jornada. A meu tio Nicolau (*in memoriam*) pelo exemplo de docência. Enfim, a todos que de alguma forma contribuíram com esse projeto. Por fim, agradeço a Liliane e Ronei, meus orientadores, pela orientação acadêmica e ao projeto INCT-MACC pelo financiamento durante o mestrado.

Resumo

A complexidade envolvida no processo de geração de aplicações médicas baseadas em Realidade Virtual (RV) exige, em sua maioria, um conhecimento específico em programação para manipular estes sistemas computacionais. Um estudo realizado neste trabalho observou que a maioria dos *frameworks* que geram aplicações baseadas em RV não possui interface de programação de aplicações voltada para usuários leigos. Isso se aplica em diversas áreas, inclusive na saúde. Neste contexto, se destaca o CyberMed como *framework* para desenvolvimento de aplicações médicas baseadas em RV. Outro levantamento realizado neste trabalho mostrou que os profissionais da área de saúde possuem vontade de interagir com sistemas em RV de fácil uso. Baseado nesses resultados e em outros estudos executados, esse trabalho apresentará um estudo sobre Programação Visual (PV) em *frameworks* na área de saúde e mostrará como essa técnica pode ajudar no processo de interação e de geração de aplicações médicas, com recursos de estereoscopia, colisão, deformação e sensação háptica, por exemplo, pelos usuários leigos em programação. Como resultado desse estudo, é apresentada a arquitetura de um Sistema de PV chamado de CyberMedVPS baseado em técnicas de PV e que possibilita o desenvolvimento de aplicações com recursos em RV por meio da interação predominantemente gráfica com *frameworks* baseados em RV pelos profissionais de saúde.

Palavras-chave: Realidade Virtual, *Framework* de RV, Programação Visual, Interação, Sistema de Programação Visual.

Abstract

The complexity involved in the generation of medical applications based on Virtual Reality (VR) requires, in most cases, a specific knowledge in programming to manipulate these computer systems. A study in this dissertation noted that most frameworks with VR resources do not have application programming interface targeted for lay users. This fact applies in several areas, including health. In this context, the CyberMed highlights as a framework for developing medical applications based on VR. Another studying shows that health professionals will have to interact with systems in VR easy to use. Based on these results and other studies performed, this dissertation presents a study on Visual Programming (VP) in frameworks in health area and shows how this technique can help during process of interaction and generation of medical applications, with features in stereoscopy, collision, deformation and haptic sensing, for example, by non-programmers . As a result of this study, the architecture of VP system called CyberMedVPS is showed. It is based on VP techniques that enable VR application development by interaction predominantly graphical at VR frameworks to healthcare professionals.

Keywords: *Virtual Reality, Framework based on VR, Visual Programming, Interaction, System Visual Programming.*

Lista de ilustrações

Figura 1. Exemplo de um fluxograma.	10
Figura 2. Diagrama dos componentes da PV.	12
Figura 3. Multidisciplinaridade da RV.	14
Figura 4. Leiaute do ViMeT Wizard.	18
Figura 5. Leiaute do MedicalStudio Wizard.	19
Figura 6. Leiaute do MITK.	20
Figura 7. Leiaute do Framework Julius.	20
Figura 8. Estrutura baseada em fluxograma (a), APV do VisProg (b).	23
Figura 9. Interface visual do NeatVision.	24
Figura 10. Arquitetura do PathSim Visualizer.	25
Figura 11. Estrutura de visualização do PathSim.	26
Figura 12. Protótipo Visual do VR-MED.	27
Figura 13. Interface do MeVisLab.	28
Figura 14. PV aplicada no SCIRun voltado para medicina.	29
Figura 15. Interface visual do LabView.	30
Figura 16. Exemplo de um grafo de cena para uma aplicação com o SOFA.	31
Figura 17. Exemplo de uma aplicação de laparoscopia utilizando o SOFA.	31
Figura 18. Exemplo do uso de PV com InTml.	32
Figura 19. Leiaute do Voreen em um exemplo montagem de fluxogramas.	33
Figura 20. Interesse dos estudantes em relação à RV.	41
Figura 21. Interesse dos professores em relação à RV.	41
Figura 22. Arquitetura do SPV.	45
Figura 23. Diagrama de estados da aplicação gerada no SPV.	46
Figura 24. Diagrama de sequência sobre a criação e execução do SPV.	47
Figura 25. Fluxo dos dados no SPV.	48
Figura 26. Protótipo de interface do CyberMedVPS.	52
Figura 27. Barra de menu do CyberMedVPS.	53
Figura 28. Área lateral da UI e os componente que fazem parte da aplicação visual do CyberMedVPS.	55

Figura 29. Estrutura de um objeto na área de desenvolvimento do CyberMedVPS.	57
Figura 30. Fluxograma montado com componentes do CyberMedVPS.	58
Figura 31. Hierarquia de ligação dos componentes.	58
Figura 32. Área de rodapé da UI do CyberMedVPS.	59
Figura 33. Leiaute completo da UI do CyberMedVPS.	60
Figura 34. Leiaute da classe <i>InitialScreen</i> .	61
Figura 35. Dicionário de dados no CyberMedVPS.	61
Figura 36. Classes que compõem a Camada de Visualização.	62
Figura 37. Especialização entre classes do pacote Model.	63
Figura 38. Classes que compõem a Camada de Comunicação do CyberMedVPS.	65
Figura 39. Diagrama de classes da camada de validação.	65
Figura 40. Dinâmica das linguagens no CyberMedVPS.	68
Figura 41. Diagrama de classes do CyberMedVPS.	70
Figura 42. Selecionando objetos para aplicação visual no CyberMedVPS.	73
Figura 43. Criando ligação no CyberMedVPS.	73
Figura 44. Opções de configuração de parâmetros de objetos do CyberMedVPS.	74
Figura 45. Aplicação final (a) e aplicação final em RV (b).	74
Figura 46. Usuário interagindo com o CyberMedVPS (a) e aplicação final em execução (b).	77
Figura 47. Elementos do CyberMedVPS que foram compreendidos pelo usuário.	79
Figura 48. Quantidade de horas gastas, semanalmente, no computador pelos entrevistados.	79
Figura 49. Componentes de validação do CyberMedVPS.	81
Figura 50. Leiaute da classe FrameworkAsker.	84
Figura 51. Sistemática dos testes de defeitos.	89
Figura 52. Aprovação do Comitê de ética do HUWL.	129

Lista de quadros

Quadro 1. Comparativo entre os frameworks com PV de áreas diversas.	22
Quadro 2. Comparativo entre os frameworks com PV aplicados na área de saúde.	34
Quadro 3. Requisitos funcionais do Sistema de Programação Visual.	44
Quadro 4. Gramática do XML gerada pelo CyberMedVPS.	66
Quadro 5. Objetos visuais que compõem cada categoria de aplicação do CyberMedVPS.	85
Quadro 6. Tabela de ações avaliadas no teste de defeitos do CyberMedVPS.	87

Lista de abreviaturas e siglas

Sigla	Acrônimo
2D	Bidimensional
3D	Tridimensional
API	<i>Application Program Interface</i>
APV	Ambiente de Programação Visual
AV	Ambiente Virtual
CAVE	<i>CAVE Automatic Virtual Environment</i>
CBIS	Congresso Brasileiro de Informática em Saúde
CCS	Ciências em Saúde
CEP	Comitê de Ética em Pesquisa com Seres Humanos
CLV	Classes de Linguagem Visual
CM	Ciências Médicas
EPV	Expressões de Programação Visual
DAO	<i>Data Access Object</i>
GiPSi	<i>General interactive Physical Simulation interface</i>
GHC	<i>Guarded Horn Clauses</i>
GUI	Interface Gráfica – <i>Graphical User Interface</i>
HULW	Hospital Universitário Lauro Wanderley
ID	<i>Identification Number</i>
IHC	Interação Humano-Computador
InTml	<i>Interaction Technique Markup Language</i>
IRVE	<i>Information-Rich Virtual Environments</i>
ITK	<i>Insight Toolkit</i>
iVProg	<i>Internet Visual Programming</i>
JDBC	<i>Java Database Connectivity</i>
JPA	<i>Java Persistence API</i>
LabTEVE	Laboratório de Tecnologias para Ensino Virtual e Estatística
LPV	Linguagens de Programação Visual
MMVR	<i>Medical Meets Virtual Reality</i>
MVC	Modelo/Visão/Controlador
MITK	<i>Medical Imaging Interaction Toolkit</i>
ORM	<i>Object-Relational Mapping</i>
PathSim	<i>Pathogen Simulation</i>
PC	<i>Personal Computer</i>
PPGI	Programa de Pós-Graduação em Informática
PV	Programação Visual

RV	Realidade Virtual
SGBD	Sistema de Gerenciamento de Banco de Dados
SOFA	<i>Simulation Open Framework Architecture</i>
SPV	Sistema de Programação Visual
SSVE	<i>Shared Simple Virtual Environment</i>
SVR	Simpósio de Realidade Virtual e Aumentada
UFPB	Universidade Federal da Paraíba
UI	<i>User Interface</i>
UML	<i>Unified Modeling Language</i>
VAT	<i>Visual AgenTalk</i>
ViMeT	<i>Virtual Medical Training</i>
VI	<i>Virtual Instruments</i>
VRAL	<i>Virtual Reality Abstraction Layer</i>
VTK	<i>Visualization Toolkit</i>
XML	<i>Extensible Markup Language</i>
WIMP	<i>Window, Icon, Menu, Pointing device</i>

Sumário

1 Introdução	1
1.1. Motivação	3
1.2. Relevância	4
1.3. Objetivos	5
1.4. Contribuição	6
1.5. Metodologia	6
1.6. Trabalhos Correlatos na Instituição	7
1.7. Estrutura da Dissertação	8
2 Embasamento Teórico	9
2.1. Programação Visual	10
2.2. Realidade Virtual	13
2.2.1. <i>Frameworks</i> na Realidade Virtual	15
2.3. <i>Frameworks</i> com Programação Visual Aplicados na Área de Saúde	21
2.3.1. VisProg	23
2.3.2. NeatVision	24
2.3.3. PathSim Visualizer	25
2.3.4. VR-MED	26
2.3.5. MeVisLab	27
2.3.6. SCIRun	28
2.3.7. LabView	29
2.3.8. SOFA	30
2.3.9. InTml	31
2.3.10. Voreen	32
2.4. Análise comparativa dos <i>frameworks</i> com PV aplicados na área de saúde	33
2.5. Conclusão	35
3 Arquitetura para PV de Ambientes baseados em RV para Área de Saúde	37
3.1 Sobre a Arquitetura para PV de ambientes em RV para usuários de saúde	38
3.2 Avaliação preliminar com os profissionais de saúde	39
3.2.1. Aspectos éticos do projeto	40
3.2.2. Aplicação do questionário	40
3.2.3. Avaliação dos resultados	42
3.3 Requisitos da arquitetura	42
3.4 Arquitetura do SPV	44
3.5 Visão Lógica do SPV	46
3.6 Visão dos dados do SPV	47
3.7 Conclusão	48
4 Desenvolvimento do CyberMedVPS	49
	X

4.1. Definição da Camada de Visualização	50
4.1.1. Área superior da UI do CyberMedVPS	52
4.1.2. Área lateral da UI do CyberMedVPS	53
a. Componentes de uma aplicação visual baseada em RV	54
4.1.3. Área central da UI do CyberMedVPS	56
4.1.4. Área de rodapé da UI do CyberMedVPS	59
4.1.5. Dicionário de dados	60
4.2. Definição da camada de comunicação	64
4.3. Adição de suporte a frameworks	69
4.4. Estrutura final das classes do CyberMedVPS	69
5 Resultados	71
5.1. Publicações	72
5.2. Exemplo de caso	73
5.3. Avaliação da GUI do CyberMedVPS com o público-alvo	75
5.3.1. Protocolo de avaliação	75
5.3.2. Aplicação da avaliação	77
5.3.3. Resultados da avaliação	78
5.4. Validação do CyberMedVPS	80
5.4.1. Definição do cenário de validação	81
5.4.2. Definição do framework em RV	81
a. Ambiente de Execução para o framework CyberMed	82
5.4.3. Definição das categorias de aplicações em RV	84
5.4.4. Resultados da validação do CyberMedVPS	86
5.5. Teste de defeitos do CyberMedVPS	86
5.5.1. Análise de problemas na interface	86
5.5.2. Análise de problemas na camada de comunicação	89
5.5.3. Conclusões do teste de defeitos	90
6 Conclusão	91
6.1. Contribuições	92
6.2. Trabalhos Futuros	94
6.3. Considerações Finais	94
Referências	96
Apêndices	106
APÊNDICE A – Avaliação das necessidades dos profissionais de saúde	106
APÊNDICE B – Questionário sobre protótipo visual do CyberMedVPS	108
APÊNDICE C – Tutorial de configuração e execução do CyberMed no Windows	110
APÊNDICE D – Resultados obtidos com os cenários de validação do CyberMedVPS	117
Anexo	129
ANEXO A - Aprovação do projeto CyberMedVPS no CEP do HULW	129

1 Introdução

Uma técnica de desenvolvimento de aplicações chamada de Programação Visual (PV) surgiu na década de 60 (GOLDSTINE, 1972), utiliza métodos de visualização de dados e é baseada em fluxogramas. Tal técnica tem como intuito tornar o desenvolvimento de aplicações, em sistemas complexos, uma atividade simples e ágil. Essa complexidade é perceptível no processo de desenvolvimento, que pode apresentar elementos que dificultam a interação do usuário seja pelo número de passos executados até alcançar o objetivo, pela complexidade das instruções ou até mesmo pelo processamento elevado da máquina em execução.

A PV tornou-se uma técnica essencial em sistemas de desenvolvimento rápido (NUNES *et al.*, 2007), em ferramentas que envolvem áreas distintas da computação (JOHNSON e PARKER, 1995) e em sistemas de execuções complexas (AUGUSTINE *et al.*, 2004), como os sistemas com recursos baseados em Realidade Virtual (RV), por exemplo. Outra tecnologia de interface importante é a RV, que segundo Stone *et al.* (2011), é a fusão da computação gráfica interativa tridimensional (3D) com simulações em tempo real e com tecnologias de Interface Humano-Computador (IHC). O seu objetivo é oferecer sistemas que integrem aspectos de imersão e interatividade para simular ambientes realistas que permitam ao usuário interagir e navegar pelo ambiente com objetos e outros elementos do ambiente virtual (GREENLEAF, 2004). Para isso, podem ser utilizados equipamentos específicos que estimulam os sentidos, explorando a sensibilidade visual, tátil, auditiva e olfativa humana.

Os sistemas atuais de RV são considerados complexos porque reúnem conhecimentos de vários segmentos como a eletrônica, a computação, a robótica, a física, a psicologia, dentre outras (ROSEMBLUM *et al.*, 1998). Quanto mais há comunicação entre as áreas divergentes, mais difícil é a concordância entre os objetivos (PREECE *et al.*, 2008). Segundo Bastos *et al.* (2004), toda esta dificuldade atinge negativamente os desenvolvedores de aplicações de RV, afetando diretamente a qualidade da aplicação final, tornando-a mais difícil de manter e utilizar.

Além disso, estruturas de desenvolvimento conhecidas como *frameworks* possibilitam ainda mais que os desenvolvedores sejam capazes de criar as aplicações com recursos em RV. Mas, não são apenas os desenvolvedores que possuem interesse em desenvolver aplicações em RV. É possível detectar que os profissionais de saúde gostariam de criar suas próprias aplicações na área médica (MORAIS e MACHADO, 2011). Assim, a PV vem sendo proposta para sistemas complexos que geram aplicações RV (PAQUET e KROPF, 2000) com o intuito de melhorar a interação dos *frameworks*

para usuários sem conhecimento específico em programação ou para tornar o tempo de desenvolvimento menor a partir desses sistemas.

1.1. Motivação

A RV está inserida em aplicações de diversas áreas, como a educação (QUINN *et al.*, 2003), o entretenimento (JACOBSON e LEWIS, 2005), a aviação (MARION *et al.*, 2007) e a medicina (LAMATA *et al.*, 2006). A criação de aplicações com recursos em RV pode ser realizada por componentes chamados de *frameworks* que, segundo Silva e Lucena (2003), se define como uma estrutura de classes que se relacionam dando origem a uma aplicação que pode gerar um conjunto de aplicações de um determinado domínio.

Os *frameworks* são direcionados para desenvolvedores de aplicações. Por isso, o processo de interação é pouco trivial para usuários inexperientes em programação computacional (MORAIS e MACHADO, 2010). Para usuários leigos em programação, o desenvolvimento de aplicações diretamente por meio de *frameworks* não é atividade recomendada porque, de acordo com Mossman *et al.* (2010), essas estruturas apresentam uma interação complexa por meio de comandos específicos da computação e uma interface visual pouco trivial para usuários de outras áreas durante o processo de desenvolvimento.

Vale ressaltar que a aplicabilidade dos *frameworks* que geram aplicações baseadas em RV é utilizada, por exemplo, em sistemas na área da matemática (ROUSSOU, 2009), da robótica (CAO *et al.*, 1999), de ciências (FICHEMAN *et al.*, 2008) e da astronomia (CAMPOS e SAMPAIO, 2005). Dentre esses diversos segmentos, os *frameworks* na área de saúde merecem destaque e podem permitir o desenvolvimento de aplicações voltadas para o treinamento, aprendizado ou simulação de procedimentos específicos.

O perfil de usuários que pode se beneficiar das aplicações geradas pelos *frameworks* com recursos de RV na saúde é, na sua maioria, os próprios profissionais da área que, muitas vezes, não possuem conhecimentos aprofundados em programação. De acordo com Zajtchuk e Satava (1997), a RV pode ser herdada para melhorar a área médica em quatro atividades principais: a) no aprendizado e treinamento de técnicas; b) no planejamento de desastres e acidentes c) na prototipagem virtual, ou seja, criação ou execução de modelos de ambientes e aplicações com recursos em RV e d) na reabilitação e terapia psiquiátrica.

Uma pesquisa de Merrill *et al.* (1995) afirma que é mais provável que médicos cometam erros em seus primeiros procedimentos cirúrgicos e diagnósticos. Essa afirmação vale também para outros tipos de profissionais da área. Merrill *et al.* (1995) afirmam ainda que o risco operatório pode ser substancialmente reduzido por meio do uso de simuladores que permitem a transferência de habilidades de uma simulação para um contato real com o paciente. Nesse contexto, se inserem as aplicações em RV geradas pelos *frameworks*.

Um estudo realizado por Martikainen *et al.* (2012) revelou que os sistemas computacionais atualmente utilizados para área médica apresentam carência de recursos visuais adequados para apoiar as tarefas típicas desses profissionais, exigem a realização de sequências fixas de passos e tarefas, mal suportam a documentação e recuperação dos dados.

Assim, esse trabalho tem como motivação a ausência de ferramentas computacionais que sejam elaboradas para que profissionais de saúde consigam desenvolver suas próprias aplicações médicas com recursos em RV para treinamento e aprendizado.

1.2. Relevância

Segundo Bastos e Proença (2000), o estudo e o treinamento de estudantes na área de saúde com animais, cadáveres ou peças artificiais provocam um distanciamento em relação aos pacientes reais, principalmente por parte dos médicos. Isso pode ocorrer porque tais elementos, citados anteriormente, não apresentam reações similares as que serão efetivamente enfrentadas nestas profissões. Tal estudo justifica ainda que o profissional de saúde entende inconscientemente o contato com o seu paciente real como uma atividade secundária, pois o mesmo não vivencia essas atividades no período universitário.

De acordo com os resultados de uma pesquisa de von Otter *et al.* (2011), a visualização de determinados dados pode ser determinante no processo de aprendizado na área de saúde. Se esse profissional for um professor, o uso de *frameworks* de RV é adequado para geração de objetos de aprendizagem, desde que o mesmo saiba utilizar corretamente tais *frameworks* para criar suas próprias aplicações em RV e saiba explorar o uso do computador em situações de simulação que permitam ao aluno praticar ou

vivenciar situações abstratas ou reais para as quais ele ainda não esteja preparado ou não tenha visto (MELO e DAMASCENO, 2006).

Segundo Rosen *et al.*(1996), as aplicações médicas com recursos em RV geradas por *frameworks* pode ser de dois tipos: os modelos genéricos, que permite ao usuário interagir com os modelos e visualizar os resultados de suas ações, e os modelos específicos de pacientes, nos quais a aplicação tenta construir o modelo virtual de um indivíduo com todos os seus detalhes específicos. O foco deste trabalho são os modelos específicos, pois se aplicam na área de saúde.

Além disso, os comandos dos *frameworks* que geram aplicações baseadas em RV ficam restritos às linguagens computacionais, caracterizadas por códigos textuais específicos. Dessa forma, o usuário leigo em computação que queira utilizar um *framework* em RV, deverá trabalhar em conjunto com um profissional da computação ou precisará de um treinamento prévio para adquirir conhecimentos em programação (MORAIS e MACHADO, 2010).

1.3. Objetivos

De acordo com Martikainen *et al.*(2012), existem vários estudos que sustentam a ideia de que o desenvolvimento da Tecnologia da Informação (TI) na saúde deve envolver usuários finais, ou seja, os profissionais de saúde como principais usuários desses sistemas. Mas, essa conduta não acontece no desenvolvimento da maioria dos sistemas computacionais, principalmente os mais complexos em termo de interação.

Este trabalho tem como objetivo geral propor uma arquitetura baseada em PV para ambientes em RV a fim de permitir que os profissionais de saúde possam criar aplicações com recursos em RV para o ensino e treinamento, inicialmente com recursos de estereoscopia, interação háptica, deformação e detecção de colisão. Além disso, são objetivos específicos deste trabalho:

- pesquisar sobre o uso de PV em sistemas de RV na área médica;
- determinar as necessidades e as experiências do público-alvo na área de RV;
- agregar estratégias de PV aos *frameworks* que geram aplicações de RV a fim de permitir que os usuários de saúde desenvolvam suas próprias aplicações com recursos de estereoscopia, interação háptica, deformação e detecção de colisão;
- elaborar um protótipo visual de uma arquitetura para um Sistema de PV;

- avaliar esse protótipo visual;
- planejar e desenvolver o Sistema de Programação Visual (SPV), chamado de CyberMedVPS, como resultado da arquitetura proposta;
- desenvolver aplicações com recursos em RV utilizando o SPV desenvolvido;
- verificar a aceitação da UI do CyberMedVPS pelo público-alvo;
- validar cenários de aplicações com recursos de estereoscopia, interação háptica, deformação e detecção de colisão no CyberMedVPS.

1.4. Contribuição

Este trabalho oferece aos usuários da área de saúde a arquitetura de um SPV que adota a PV para prover uma UI (*User Interface*) intuitiva para o processo de desenvolvimento de aplicações em RV com recursos de estereoscopia, sensação háptica, detecção de colisão e deformação. O desenvolvimento gráfico de aplicações com recursos em RV possibilita ao profissional de saúde uma alternativa para estudo e treinamento na área. De acordo com Burnett (1999), isso é possível porque a PV torna o processo de desenvolvimento de aplicações mais acessível, pois pode melhorar a velocidade e a exatidão com que as pessoas executam as tarefas de programação.

As contribuições deste trabalho são: o estudo sobre o uso da PV em *frameworks* da área de saúde e áreas afins, avaliação das necessidades do público-alvo em *frameworks* baseados em RV, a criação da arquitetura de um SPV voltado para *frameworks* que geram aplicações baseadas em RV na área de saúde e a criação de um SPV, denominado de CyberMedVPS, como resultado da arquitetura criada e agregado a diversos *frameworks* em RV da área de saúde. Inicialmente, o SPV proposto será executado com o *framework* CyberMed.

1.5. Metodologia

O processo de desenvolvimento do CyberMedVPS seguiu um modelo híbrido da engenharia de *software* entre o modelo em cascata e o modelo baseado em componentes. Além disso, a metodologia segue a abordagem *top-down* de desenvolvimento e foi descrita de forma incremental. Todo o processo de concepção foi definido com diagramas UML (*Unified Modeling Language*).

Com relação às tecnologias utilizadas para o CyberMedVPS, é possível destacar a linguagem Java na versão 1.6 de sua máquina virtual, a utilização do sistema de gerenciamento de banco de dados MySQL na versão 5.5.8, o MinGW 5.1.4, o MSYS 1.0 e o gdb 5.2.1, para executar o CyberMed na plataforma Windows. Por fim, há a geração de código nas linguagens XML (*Extensible Markup Language*), C++ e Java. Vale ressaltar que a licença de uso do sistema desenvolvido nesse trabalho é de uso aberto.

A metodologia adotada na avaliação do CyberMedVPS foi definida por uma equipe formada por meio da parceria entre pesquisadores na área de IHC da SUNY Oswego e pesquisadores do LabTEVE. Essa metodologia foi composta pelas seguintes etapas: a) definição do protocolo de avaliação, b) aplicação da avaliação com os profissionais de saúde e c) avaliação dos resultados obtidos.

1.6. Trabalhos Correlatos na Instituição

Este trabalho está inserido na linha de pesquisa de Sinais, Sistemas Digitais e Gráficos do Programa de Pós-Graduação em Informática (PPGI) e foi desenvolvido no LabTEVE¹ (Laboratório de Tecnologias para o Ensino Virtual e Estatística) na Universidade Federal da Paraíba (UFPB). Tal laboratório trabalha com temáticas voltadas para o ensino virtual e a distância e com o objetivo prover e apoiar o aprendizado de alunos por sistemas que disponibilizem informações acessíveis de qualquer parte e a qualquer momento, bem como treinar alunos e professores para manipularem tais tecnologias. Uma de suas linhas de pesquisa refere-se a sistemas baseados em RV, sendo um de seus principais produtos o *framework* de código aberto chamado de CyberMed (MACHADO *et al.*, 2009).

Com relação aos estudos sobre PV, foram realizados projetos na área de estatística por meio de uma ferramenta interativa voltada para o ensino à distância chamada de Calculadora Estatística (RODRIGUES *et al.*, 2008). Esse ambiente foi desenvolvido para ajudar os estudantes da estatística na compreensão de alguns erros da área, além de ensiná-los a corrigir as etapas que solucionarão tais problemas. Além disso, a Calculadora Estatística provê a interação por meio da montagem de fluxogramas e sobre a plataforma Web (GOMES *et al.*, 2006).

Na área da RV, houve um trabalho que propôs um sistema de apoio ao ensino (DIAS *et al.*, 2009) capaz de auxiliar o ensino de classificação de imagens. Por meio de

¹ Disponível em <http://www.de.ufpb.br/~labteve>

uma Interface Gráfica (GUI) intuitiva e agregada às técnicas de RV, o usuário é capaz de visualizar e interagir com os elementos para a classificação de uma imagem 3D.

Ainda na área de RV, mais especificamente na área médica, existem vários trabalhos sobre o *framework* CyberMed. Muitas abordagens com este *framework* foram exploradas como, por exemplo: a) o desenvolvimento de uma simulação médica baseada em RV para o ensino e treinamento na área ginecológica (SANTOS *et al.*, 2010) com a integração de um módulo de avaliação baseado na lógica *fuzzy* ao *framework* e b) a criação de um módulo de rede ao CyberMed para permitir atividades colaborativas de tutoria (SALES e MACHADO, 2010).

1.7. Estrutura da Dissertação

Este documento possui sete capítulos subdivididos na seguinte estrutura: Introdução, Embasamento teórico, *Frameworks* de Saúde com Recursos de Programação Visual, CyberMedVPS, Desenvolvimento, Resultados e Conclusão. No Capítulo 1 foi exposto um conhecimento introdutório sobre conceitos voltados para RV e PV, assim como as ideias que motivaram a realização do trabalho proposto. No Capítulo 2 é realizado um levantamento bibliográfico sobre conceitos e técnicas de PV e RV, além de uma análise comparativa entre os *frameworks* em RV e os recursos visuais em sistemas computacionais aplicados na área de saúde. Em seguida, o Capítulo 3 apresenta a arquitetura baseada em PV para Ambientes baseados em RV para área de saúde. No Capítulo 4, é apresentada a etapa de desenvolvimento do SPV proposto. Em seguida, são apresentados os resultados obtidos com o trabalho no Capítulo 5 e suas respectivas considerações finais no Capítulo 6. Por fim, a estrutura desse trabalho é finalizada pelas seções de Referências, Apêndices e Anexos.

2

Embasamento Teórico

Aplicações computacionais baseadas em RV têm se destacado em tarefas de treinamento e ensino na área médica devido à sua capacidade de simular ambientes e cenários interativos, realistas e imersivos. Estas simulações permitem aos usuários treinar habilidades e tomar decisões diante de várias situações. A complexidade envolvida no processo de geração de aplicações médicas baseadas em RV exige a manipulação de ferramentas computacionais. Um fato importante que pode amenizar a complexidade de determinados sistemas é a preocupação com a interação que, segundo Preece *et al.* (2008), consiste em encontrar maneiras de fornecer uma comunicação fácil às pessoas, principalmente se for direcionado para usuários leigos em programação.

Este Capítulo tem como intuito apresentar um estudo sobre as técnicas de PV em sistemas complexos, de acordo com suas vantagens e desvantagens para um usuário leigo em programação. Em seguida será a Realidade Virtual e alguns *frameworks* que utilizam essa interface a fim de mostrar como eles estão caracterizados atualmente.

2.1. Programação Visual

Uma das técnicas que é adotada em problemas que envolvem a dificuldade de programação do usuário em sistemas complexos é conhecida como PV. A PV se iniciou na tentativa de produzir fluxogramas executáveis, particularmente influenciados pela crença de que os fluxogramas seriam bons para ensinar os usuários leigos em programação.

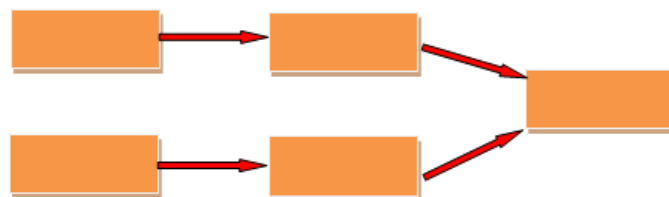


Figura 1. Exemplo de um fluxograma.

Os fluxogramas se caracterizam por ser uma coleção de caixas relacionadas, como pode ser visto na Figura 1. Cada caixa representa uma função com alguns números de entradas e saídas (Lucas *et al.*, 1992). De acordo com Lucas *et al.* (1992), um sistema possui técnicas de PV quando é constituído por fluxogramas. Segundo Nardi (1993), há evidências de que as representações visuais podem otimizar a interação entre o usuário e as ferramentas computacionais porque melhoram habilidades cognitivas humanas.

De acordo com Burnett e Baker (1994), a PV se refere a qualquer sistema que permite ao usuário especificar uma aplicação em forma de duas ou mais dimensões. As dimensões podem ser objetos, relacionamentos espaciais, ou até dimensões de tempo para especificar relações de antes e depois. Cada relacionamento ou objeto multidimensional que é significativo é chamado de símbolo. Embora essa seja uma definição ampla, as linguagens textuais não são consideradas nem bidimensionais, pois o usuário não consegue simultaneamente visualizar estruturas gráficas, que representam modificações ao longo do tempo, e relacionamentos entre seus componentes, por exemplo. Tudo que pode ser visualizado está escrito em uma linguagem de descrição computacionalmente técnica.

A PV é definida como o uso de expressões visuais tais como gráficos, desenhos, animação ou ícones no processo de programação (PAQUET e PLAICE, 1998). Algumas características da PV (DONG, 2002) são: a) As imagens representam um maior nível de abstração, por isso a sintaxe não está envolvida; b) Quando adequadamente projetadas, as imagens podem ser compreendidas pelas pessoas, independentemente da língua que falam e c) As imagens podem transmitir um significado em uma unidade mais concisa do que uma expressão em texto.

De acordo com Green (1995), existem três motivos pelos quais a comunidade computacional estuda PV há tanto tempo, independentemente da técnica. Esses aspectos são importantes até hoje (MORAIS e MACHADOa, 2011). São elas:

- a PV é tecnicamente desafiadora;
- a PV suporta o modelo de fluxos de dados da computação muito melhor do que programação textual convencional;
- embora os programadores sejam usuários, a IHC convencional tem agido pouco em projeto de ambientes de programação.

Outro benefício é que os programas visuais são mais fáceis de adaptar a execução em sistemas de processamento paralelo, pois preserva as informações sobre as dependências em um algoritmo. Esta é uma informação que o computador deve ter disponível para executar em paralelo, por meio de múltiplos processadores, sem acúmulo de erros de dados (SCRIVENER, 1993).

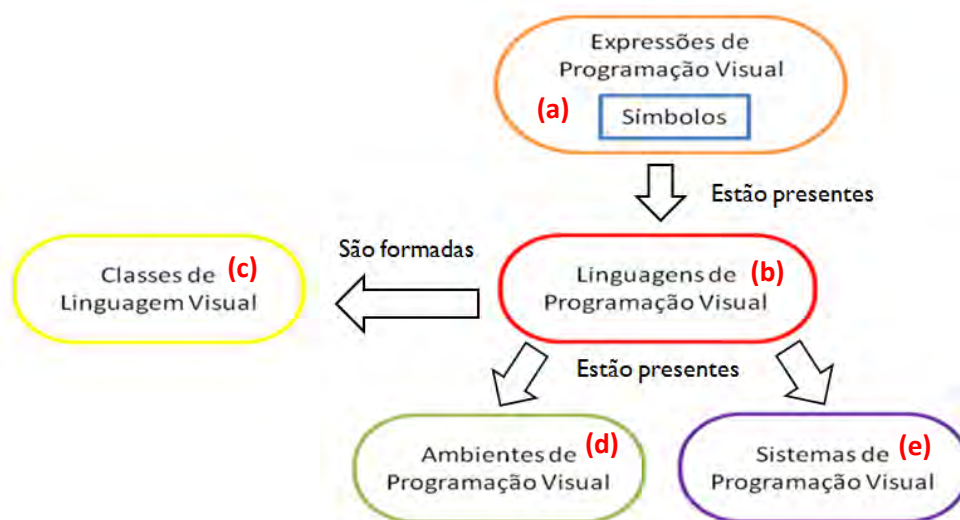


Figura 2. Diagrama dos componentes da PV.

Fonte: Traduzido de Moraes e Machado (2010), pg 2.

A PV engloba os seguintes componentes que podem ser vistos na Figura 2: Expressões de Programação Visual (EPV), Linguagens de Programação Visual (LPV), Classes de Programação Visual (CPV), Sistemas de Programação Visual (SPV) e Ambientes de Programação Visual (APV) (MORAIS e MACHADO, 2010).

A coleção de um ou mais símbolos é conhecida como EPV, ou apenas expressões visuais (Figura 2a). Essas expressões usadas na PV, segundo Burnett (1999), podem ser diagramas, ícones, desenhos à mão-livre ou demonstração de ações realizadas por objetos gráficos. Quando uma linguagem de programação contém expressões visuais (Figura 2b) ela passa a ser chamada de LPV.

Uma LPV, também conhecida como programação diagramática, é qualquer linguagem de programação que permite aos usuários criar programas por meio da manipulação gráfica de elementos do programa ao invés de especificá-los textualmente (JOHNSTON *et al.*, 2004). Cada LPV é formada por CLV (Figura 2c) que são famílias de linguagens visuais em termos de atributos sintáticos de seus objetos gráficos e relações espaciais que podem ser usadas para compor sentenças visuais. Costagliola *et al.* (2002) mostram como exemplo que um objeto gráfico na família de uma linguagem gráfica baseada em fluxogramas é uma caixa com atributos sintáticos identificando pontos específicos na borda da caixa. Além disso, os relacionamentos típicos são as interligações, que são usados para ligar os objetos gráficos por meio de seus pontos de fixação e são visualizados por múltiplas linhas.

Quando expressões visuais são usadas num ambiente de programação como atalho para gerar código de edição que pode ou não ter uma sintaxe diferente daquela usada para editar código, o ambiente é chamado de APV (Figura 2d). Segundo Burnett (1999), tais ambientes de PV proporcionam um meio termo entre as LPVs e as linguagens textuais conhecidas.

Um SPV é um sistema para a criação alto nível de aplicações que foram projetadas com LPV (Figura 2e). Um SPV utiliza uma interface essencialmente visual ao invés de estrutura composta por programação linguística. Geralmente, eles são de natureza pictórica, desde os ícones para representar os dados, até operações. Por sua vez, os APVs são aqueles onde a língua é textual, mas o depurador, o verificador, e outros elementos, podem ter elementos visuais e editores de texto visual que suportam programação textual, fazendo uso de técnicas visuais, como o recuo, fontes e cores (NARDI, 1993).

De acordo com Michel *et al.* (2010), independente da técnica de PV adotada, as ferramentas computacionais que as utilizam tem as seguintes características: a) o conhecimento sobre a linguagem de programação utilizada não é mais obrigatório e b) a sintaxe pode ser assegurada de acordo com a gramática, que pode vir incorporada nos elementos gráficos.

Além do uso de PV, de acordo com Lucanin e Fabek (2011), é perceptível o grau de melhora das GUIs devido ao poder computacional cada vez maior dos PCs (*Personal Computers*) domésticos. Baseado nisso, Kaucic e Asic (2011) afirma que os sistemas computacionais que apresentam PV podem estar enquadrados nas seguintes categorias: a) sistemas para melhorar uma tarefa dada como crítica, b) sistemas de desenvolvimento de jogos, c) sistemas que utilizam produtos para o controle industrial, d) sistemas de visualização de dados científicos e e) sistemas complexos na interação. Os sistemas baseados em RV podem ser enquadrados em diversas categorias, citadas anteriormente, mas neste trabalho eles são caracterizados como sistemas complexos.

2.2. Realidade Virtual

A RV é uma interface avançada do usuário para acessar aplicações executadas no computador, propiciando a visualização, movimentação e interação do usuário, em tempo real e em ambientes tridimensionais gerados por computador. A RV também pode ser definida como o uso de tecnologia computacional para criar o efeito de um mundo

interativo em 3D no qual os objetos têm um senso de presença espacial (KAMATH, 2012). Além disso, é capaz de explorar os diversos sentidos humanos a fim de tornar suas aplicações imersivas para o usuário.

A RV também objetiva o enriquecimento da experiência do usuário por meio de aplicações que utilizam dispositivos multissensoriais (KIRNER e SISCOOTTO, 2007). Uma aplicação com recursos em RV pode contemplar desde equipamentos convencionais de interação com o computador como teclado, *mouse*, até dispositivos para aplicações específicas como o Phantom (MACHADO e MORAES, 2010) e a luva de dados (FREITAS *et al.*, 2003), por exemplo.

Segundo Machado e Moraes (2010), a RV é uma tecnologia de caráter multidisciplinar, ou seja, existem diversas áreas, além da computação, que são utilizadas pela RV para o desenvolvimento de sistemas, como mostra a Figura 3.

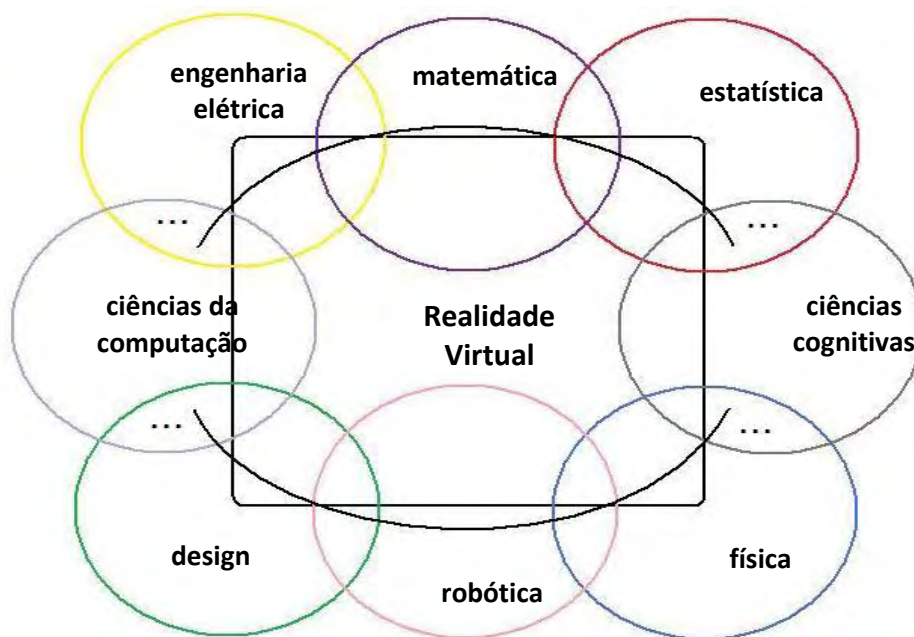


Figura 3. Multidisciplinaridade da RV.

Fonte: Adaptada de Machado e Moraes (2010), pg 86.

O treinamento médico é um exemplo típico, no qual a aplicação com recursos em RV é capaz de simular em objetos virtuais as mesmas sensações de toque, deformação e visualização dos tecidos reais. Para isto, são usados dispositivos hápticos e modelos geométricos deformáveis para se ajustarem às diversas situações de pressão e ruptura. Além desse tipo de aplicação com recurso em RV, existem diversos tipos de aplicações que podem oferecer a imersão e simulação para o usuário. O processo de desenvolvimento de aplicações com recursos em RV não é trivial, mesmo para

desenvolvedores. Mas, o tempo de desenvolvimento pode ser reduzido com a utilização de componentes denominados *frameworks*.

2.2.1. Frameworks na Realidade Virtual

Os *frameworks*, segundo Bastos *et al.* (2004), são implementações abstratas para o desenvolvimento de aplicações e com a vantagem de reutilizar componentes. Eles também consistem em uma técnica de construção de *software* que possibilita a reutilização da análise, do projeto, do código e dos testes (OLIVEIRA e MARQUES, 2009). Dessa forma, os *frameworks* são direcionados para desenvolvedores de aplicações e são responsáveis por oferecer *design* e construtores reutilizáveis para o desenvolvimento de aplicações em RV.

Os *frameworks* que geram aplicações com recursos de RV propiciam um modo diferente de ver e experimentar as informações, uma vez que são dinâmicas e imediatas. De acordo com Mossel *et al.* (2012), um *framework* de RV deve ser barato, de rápida compreensão, bem documentado e flexível para extensões e integrações rápidas para novas soluções relacionadas ao *hardware*. Alguns autores, como Assfalg *et al.* (2000), acreditam que a utilização de recursos visuais em *frameworks* através de técnicas interação visual como o WIMP (*Window, Icon, Menu, Pointing Device*) poderia melhorar ainda mais o tempo de desenvolvimento da aplicação.

De acordo com Shaw *et al.* (1993), para um *framework* de RV ser útil é necessário que: a) gere imagens estereoscópicas (3D) para causar imersão, b) reaja às ações do usuário, b) suporte o multiprocessamento, c) oculte os detalhes da comunicação dos dados e d) monitore o desempenho dos usuários.

Baseado nesse contexto foi encontrado *frameworks* que geram aplicações em RV e que possuem aplicações na área de saúde. Levando em consideração o processo de desenvolvimento de aplicações, tais *frameworks* de RV podem ser divididos em dois grupos: a) *frameworks* sem GUI e b) *frameworks* com o recurso *Wizard*.

Com relação aos *frameworks* sem GUI é possível destacar inicialmente o *General interactive Physical Simulation interface* (GiPSi) que é um *framework open-source* para o desenvolvimento de simulações cirúrgicas. Além disso, o GiPSi fornece uma API (*Application Program Interface*) intuitiva para interface de modelos dinâmicos espaciais e que facilita o desenvolvimento compartilhado de modelos reutilizáveis (CAVUSOGLU *et al.*, 2006). As aplicações geradas pela API do GiPSi podem gerar recursos hápticos, de visualização, de deformação de objetos e com suporte a simulação em paralelo e

distribuída, por exemplo. De acordo com Basdogan *et al.* (2007), com o GiPSi é possível gerar cenários para simulação de diferentes órgãos e com distintos modelos de deformação.

O segundo *framework* é chamado de Spring. Esse *framework* desenvolve aplicações cirúrgicas com recursos em RV e em tempo-real (BRUYNS e MONTGOMERY, 2002). Dentre as características do Spring é possível afirmar que se trata de um simulador multiplataforma que oferece recursos colaborativos para múltiplos usuários e dispositivos, distribuídos ou não. Outro fato importante é que o Spring foi projetado na linguagem C++ e com a biblioteca OpenGL para o suporte gráfico.

Outro *framework* de destaque é o Aladyn-3D que possibilita calcular os movimentos, deformações e interações entre objetos virtuais (MENDOZA e LAUGIER, 2000) e possui uma ampla utilização na robótica (KENNEDY *et al.*, 2002) e na medicina. O ambiente de execução do Aladyn-3D é chamado de Aladyn 3D-L (MENDOZA *et al.*, 2001).

Outro *framework* é chamado basho e é voltado para o desenvolvimento de aplicações com recursos em RV voltados para Ambientes Virtuais (AVs). A relevância do basho é o fato desse *framework* ser compatível com diversos tipos de renderizadores gráficos (HINKENJANN e MANNUSS, 2004), inclusive ligados ao *hardware* quanto simulados por *software*.

O *Shared Simple Virtual Environment* (SSVE) é um *framework* orientado a objeto para o suporte de recursos de colaboração em AVs. De acordo com Linebarger *et al.* (2003), o SSVE gera aplicações com até oito pessoas atuando, remotamente ou não, em um mesmo AV. Esse *framework* funciona integrado a uma biblioteca que suporta o desenvolvimento de aplicações para AVs. Além disso, o SSVE oferece uma estrutura para selecionar tarefas de uma aplicação para AV por meio do reuso de código, do desenvolvimento incremental e da independência de recursos do *hardware* durante o desenvolvimento da aplicação.

O Avocado é um *framework* voltado para a criação de aplicações baseadas em RV em um cenário com recursos colaborativos e interativos utilizados em CAVE (*CAVE Automatic Virtual Environment*) (TRAMBEREND, 2001). Vale ressaltar que esse *framework* é compatível com diversos dispositivos de interface e podem ser extensível com novos tipos de dispositivos de interação (TRAMBEREND, 1999). Além disso, as aplicações geradas com o Avocado podem conter recursos visuais como a visualização

estereoscópica de objetos e a presença de multiprojeção na visualização do AV (TRAMBEREND, 2001), por exemplo.

O *Virtual Patients* (VPat) é um *framework* capaz de gerar modelos anatômicos baseados em RV que compõe aplicações médicas. De acordo com Freitas *et al.* (2003), a geração dessas aplicações obedece as seguintes etapas: a) aquisição das imagens médicas, b) reconstrução 3D dessas imagens e c) adição de deformação no modelo 3D. Na área médica, o desenvolvimento do VPat ajudou na criação de novas abordagens para aplicações em RV, principalmente na área de simulação (MARTINS *et al.*, 2001) e visualização (DIETRICH *et al.*, 2007).

Outro *framework* sem GUI é o *Kinematic Simulation, Monitoring and Off-Line Programming Environment for Telerobotics* (KISMET) e é voltado para simulação em sistemas de treinamento cirúrgico para endoscopia (KUHNAPFEL *et al.*, 2000), por exemplo. Ainda de acordo com KUHNAPFEL *et al.* (2000), o KISMET pode ser utilizado para gerar aplicações baseadas em RV para ensino, planejamento e fiscalização de procedimentos cirúrgicos invasivos.

Por fim, o CyberMed² é um *framework open-source* que tem como principais recursos gerar aplicações baseadas em RV com visualização estereoscópica, com interação por meio de dispositivos convencionais e hápticos, que usa modelos deformáveis, com suporte à detecção de colisão e colaboração (PAIVA *et al.*, 2011). Além disso, o CyberMed possui métodos de avaliação *online* do usuário (MACHADO *et al.*, 2009) e isso destaca esse *framework* de RV em relação aos demais. Além disso, esse *framework* é compatível com diversos dispositivos não-convencionais de interação, como o Phantom e o Shutter *glasses*, que aumentam o grau de imersão na visualização e na interação das aplicações em RV.

Um *Wizard* é um recurso visual que contém várias páginas que reúnem dados do usuário, com o intuito de guiá-lo passo a passo para executar tarefas específicas. Possui um conjunto padrão de botões, como Cancelar, Voltar, Avançar para navegar pelas páginas e Finalizar para concluir a tarefa. Cada *Wizard* tem o seu título, eventualmente um ícone, bem como uma área de informação ou de mensagens de erro. Nesse contexto se destacam *frameworks* de RV que possuem esse recurso. São eles: ViMeT *Wizard*, Medical Studio *Wizard*, Julius e MITK.

² Disponível em: <http://www.de.ufpb.br/~labteve/projetos/cybermed.html>

O ViMeT *Wizard* é um módulo de interação gráfico do *framework Virtual Medical Training* (ViMeT) (NUNES *et al.*, 2007), como pode ser visto na Figura 4, que permite a criação de aplicações, alterações de parâmetros ou até mesmo a chamada de aplicações pré-definidas (OLIVEIRA e MARQUES, 2009). O ViMeT é um *framework* para criação de aplicações baseadas em RV e voltadas para o treinamento médico (OLIVEIRA *et al.*, 2006). Um dos exemplos mais conhecidos do ViMeT na área de treinamento médico, é a aplicação de punção de mama (OLIVEIRA *et al.*, 2006) que oferece recursos de detecção de colisão, estereoscopia, suporte a deformação e utilização de outros dispositivos de interação. Contudo, o ViMeT *Wizard* não oferece todas as funcionalidades que o ViMeT disponibiliza.

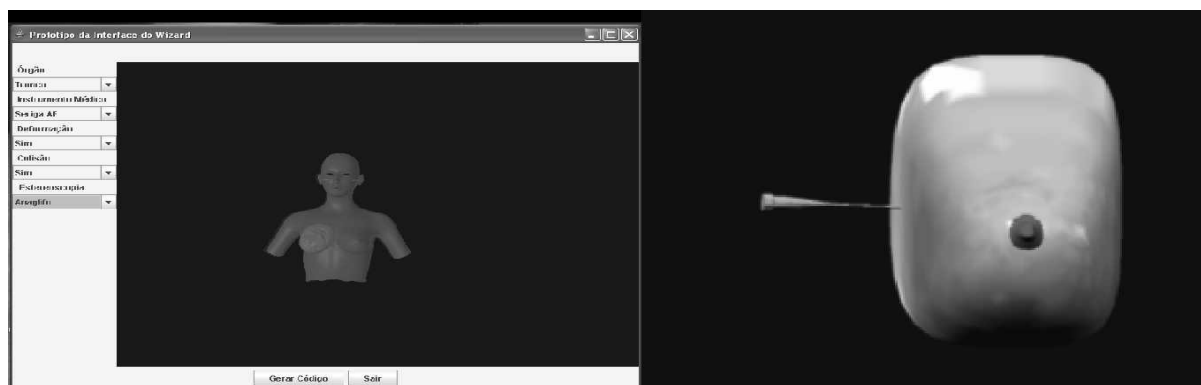


Figura 4. Leiaute do ViMeT Wizard.

Fonte: Oliveira e Marques (2009), pg 2.

O Medical Studio *Wizard* é uma estrutura gráfica do *framework* Medical Studio que auxilia no planejamento, na simulação e nos procedimentos cirúrgicos para garantir um maior grau de realismo do ambiente de trabalho médico por meio do suporte a criação de aplicações médicas em diversos cenários (WIKLUND *et al.*, 2009). É um módulo visual do *framework* de simulação cirúrgica Medical Studio³ com visualização estereoscópica (OLSZEWSKI *et al.*, 2003), como pode ser vista na Figura 5.

³ Disponível em <http://www.medicalstudio.org/>

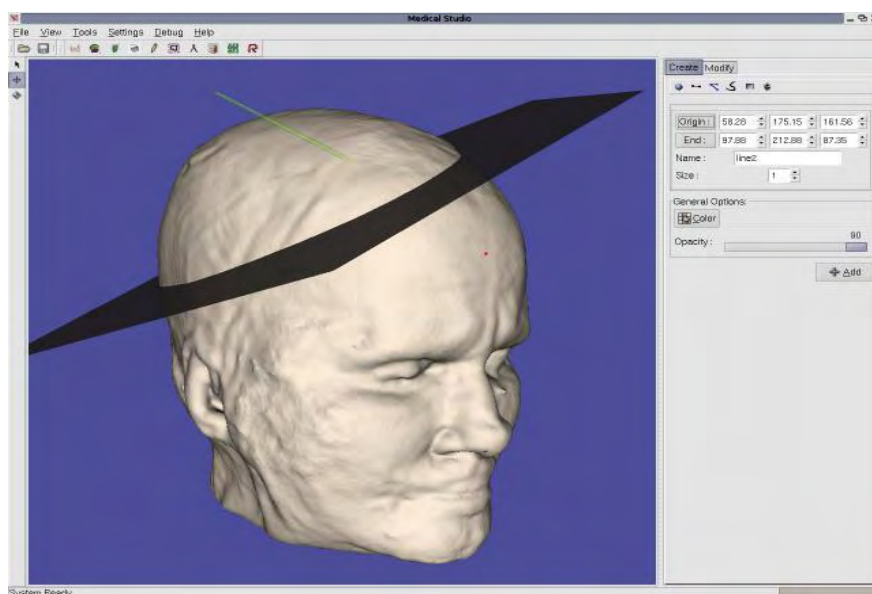


Figura 5. Leiaute do MedicalStudio Wizard.

Fonte: <http://www.medicalstudio.org/>

O *Medical Imaging Interaction Toolkit*⁴ (MITK) é um módulo de visualização criado para facilitar a geração de aplicações médicas baseadas em RV compatíveis com o *Insight Toolkit* (ITK)(IBANEZ et al., 2003) ou com o *Visualization Toolkit* (VTK) (WOLF et al., 2005). A compatibilidade com o VTK e com o ITK é possível porque o MITK faz reuso dos métodos desses dois pacotes de desenvolvimento. As aplicações criadas pelo MITK são utilizadas para a análise de imagens médicas, principalmente em sistemas de planejamentos cirúrgicos. Dentre as características apresentadas pelas aplicações criadas no MITK está na possibilidade de múltiplas formas de visualização (3D) das imagens, como pode ser visto na Figura 6. Além disso, ainda há recursos de segmentação de imagens para visualização.

⁴ Disponível em <http://www.mitk.org/>

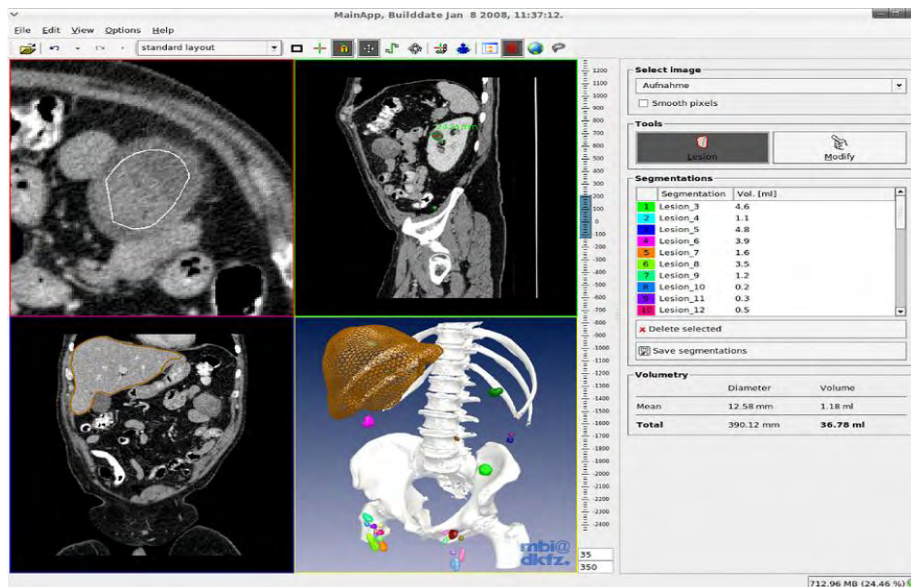


Figura 6. Leiaute do MITK.

Fonte: <http://www.mitk.org/wiki/MacOSX?action=AttachFile&do=get&target=mac.png>

O outro *framework* é denominado de Julius⁵ e é usado para a criação, simulação e navegação de aplicações de visualização médica baseadas em RV (WOLF *et al.*, 2005). De acordo com Viceconti *et al.* (2007), o Julius é permite o desenvolvimento rápido de aplicações médicas de visualização. A UI do Julius, vista na Figura 7, possui elementos visuais que sustentam o conceito de Interface de Documentos Múltiplos (*Multiple Document Interface* - MDI).

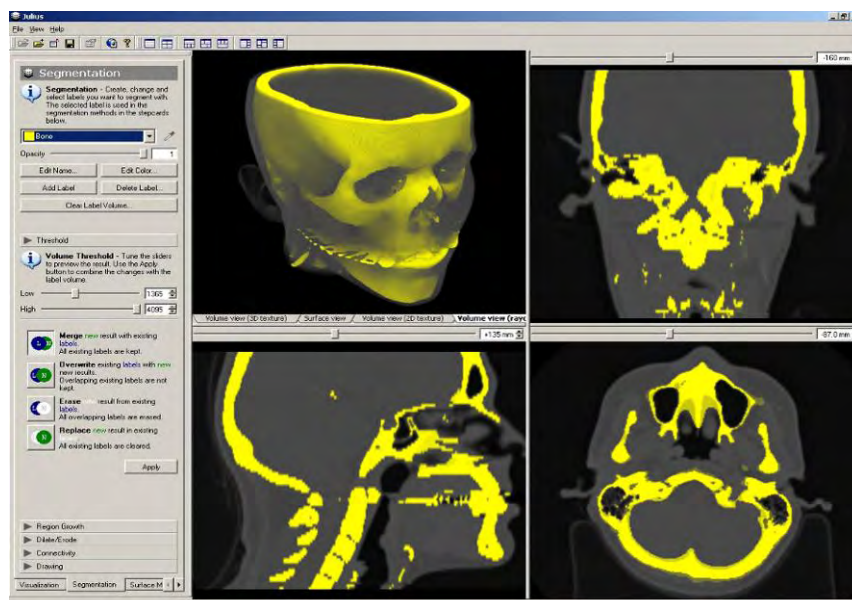


Figura 7. Leiaute do Framework Julius.

Fonte: <http://kde-apps.org/content/show.php/Julius+Framework?content=14201>.

⁵ Disponível em www.julius.caesar.de

Vale ressaltar que todos esses *frameworks* com ou sem *Wizards* são projetados para desenvolvedores de aplicações com recursos em RV, mesmo sendo aplicadas na área de saúde. Assim, o desenvolvimento de aplicações médicas pelos profissionais de saúde não é uma atividade garantida por essas estruturas. Mas, existem técnicas visuais que permite aos usuários, leigos em programação, interagir com sistemas computacionais. O principal exemplo desse recurso é chamado de PV.

2.3. Frameworks com Programação Visual Aplicados na Área de Saúde

Foi esclarecido que *framework* é uma estrutura de suporte, no qual outros programas podem ser organizados e desenvolvidos (RAHMAN, 2007). Além disso, o *framework* pode incluir suporte para programas, código de bibliotecas, linguagem de *script*, ou outro *software* que ajude a desenvolver ou unir diferentes componentes de um projeto de *software*.

É possível observar, na literatura, a presença de diversos *frameworks*, baseados ou não em RV, e de diversas áreas que possuem uma interface visual para o desenvolvimento de suas aplicações. Uma das técnicas que possibilita a existência de uma GUI nos *frameworks*, especificamente no processo de desenvolvimento é a PV. Existem *frameworks* que adotam a PV para permitir o desenvolvimento visual de aplicações em áreas como, por exemplo:

- Geoprocessamento: GeoVista Studio (TAKATSUKA e GAHEGAN, 2002);
- Química: ViPer (SANNER *et al.*, 2002);
- Estatística: Calculadora Estatística (GOMES *et al.*, 2006);
- Jogos: Kodu (MACLAURIN, 2009) e AgentCube (MICHEL *et al.*, 2010);
- Computação: Cantata (YOUNG *et al.*, 1995), ClockWorks (GRAHAM *et al.*, 1996), 3D-PP (OSHIBA e TANAKA, 1999), ALICE (KAMIYA e BRANDÃO, 2009) e iVProg (KAMIYA e BRANDÃO, 2009).

O foco deste trabalho são os *frameworks* aplicados na área de saúde, principalmente. Por isso não serão relatados detalhes estruturais de cada um dos *frameworks* com recursos de PV citados anteriormente. Ao invés disso, foi elaborada uma tabela comparativa, que pode ser vista no Quadro 1, que analisa características como: a) o tipo de PV (visto na seção 2.1.), b) as principais características de cada *framework*, c) as funções de cada um e d) a área de aplicação.

Quadro 1. Comparativo entre os *frameworks* com PV de áreas diversas.

Framework	Tipo de PV	Características	Função	Área de aplicação
<i>Cantata</i>	APV	Projetado dentro do sistema Khoros. Programas visuais são construídos como gráficos direcionados	Facilitar o processamento de imagens para pessoas não especializadas	Computação
<i>GeoVista Studio</i>	APV	Ambiente para o processamento de imagens geoespaciais	Melhorar a análise nas Geociências	Geoprocessamento
<i>Calculadora Científica</i>	APV	Interação baseada em fluxogramas sem utilização de comandos textuais	Auxiliar alunos nos estudos em estatística	Estatística
<i>ViPer</i>	APV	Utilizado na visualização de estruturas moleculares, e na criação de redes	Permitir a construção de redes computacionais descrevendo novos fluxos e visualizações de seus dados, pelo modo gráfico	Computação, Química
<i>Alice</i>	LPV	É desenvolvida em Java e sua licença é aberta. Não foi codificado de maneira internacionalizável	Ajudar nos estudos iniciais em programação	Computação
<i>ClockWorks</i>	APV	Apoiar a criação de protótipos de interfaces gráficas, concentrando-se particularmente sobre o desenvolvimento de interfaces multi-usuário e <i>groupware</i>	Ser uma alternativa gráfica de desenvolvimento para a linguagem Clock	Computação
<i>iVProg</i>	LPV	Baseado no Alice. Possui suporte para a execução <i>web</i>	Ajudar nos estudos iniciais em programação	Computação
<i>3D-PP</i>	SPV	Baseado na linguagem GHC	Oferecer uma possibilidade de interação do GHC totalmente visual	Computação
<i>Kodu</i>	LPV	Linguagem Visual de Desenvolvimento de jogos para Xbox 360 ou para o sistema Windows	Ambiente de desenvolvimento de jogos voltado principalmente para crianças	Computação, Jogos 3D
<i>AgentCube</i>	LPV	Linguagem Visual voltada para alunos do ensino médio	Despertar interesse de estudantes para a computação, especificamente para o desenvolvimento de jogos	Computação, Jogos 3D

Após analisar os parâmetros apresentados na Quadro 1, é possível inferir que, em todos os exemplos, a PV tem como objetivo principal melhorar a forma de comunicação com o sistema sobre o qual estão trabalhando. Outro fato relevante é que o Cantata, ClockWorks, o iVProg e o 3D-PP foram desenvolvidos baseados em *frameworks* já existentes, mas com recursos gráficos de interação ausentes ou insuficientes. Além disso, a técnica mais utilizada nos exemplos apresentados é a APV por ser uma técnica na qual a programação textual pode estar presente de uma maneira mais explícita para o usuário.

O comparativo do Quadro 1, também constatou que não existe um padrão nos recursos de PV utilizados, mesmo que os *frameworks* sejam da mesma área e utilizem um mesmo tipo de técnica. Mas, será que essa ausência de padronização sobre as técnicas de PV também está presente em *frameworks* aplicados na área de saúde? Para compreender essa questão foram realizadas pesquisas nos repositórios da Capes⁶, IEEE⁷, PubMed⁸, ScienceDirect⁹ e Google Acadêmico¹⁰ em trabalhos dos últimos 16 anos (1995-2011) sobre a existência de PV em sistemas computacionais, preferencialmente *frameworks*, aplicados na área de saúde e que podem apresentar recursos baseados em RV nas suas aplicações. Dentre as palavras-chave utilizadas na pesquisa dentro desses repositórios é possível ressaltar que as que tiveram mais resultados foram: a) “*visual programming*” *framework*, b) “*visual programming*” e c) “*visual programming*” “*virtual reality*” *framework*. Os detalhes sobre essa pesquisa serão detalhados a seguir.

2.3.1. VisProg

O VisProg é um APV voltado à análise de dados de genes funcionais. O VisProg utiliza a ferramenta de análise de dados chamada Orange que possibilita aos usuários manipular fluxogramas sem a necessidade prévia em saber programar (CURK *et al.*, 2004).

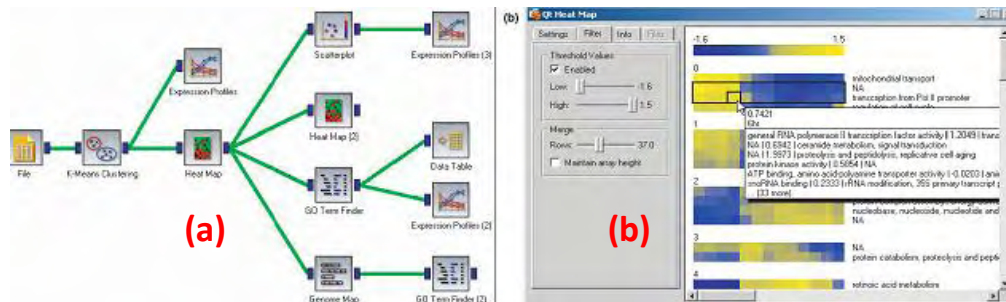


Figura 8. Estrutura baseada em fluxograma (a), APV do VisProg (b).

Fonte: Curk *et al.* (2004), pg 397.

Como pode ser visto na Figura 8 é possível perceber uma aplicação visual, na forma de um fluxograma (Figura 8a). Os quadrados são chamados de *widgets* no VisProg e tem o objetivo de facilitar o uso de componentes no desenvolvimento e manipulação de matrizes microscópicas dos genes pelos pesquisadores e estudantes em genética. Vale

⁶ <http://www.periodicos.capes.gov.br/> >
⁷ <http://ieeexplore.ieee.org/Xplore/dynhome.jsp>
⁸ <http://www.ncbi.nlm.nih.gov/pubmed>
⁹ <http://www.sciencedirect.com/>
¹⁰ <http://scholar.google.com.br/>

ressaltar que o VisProg é multiplataforma, mas seu código não é de livre manuseio, apenas a versão executável.

2.3.2. NeatVision

O NeatVision é voltado para aplicações de diagnóstico médico por meio de processamento e manipulação de imagens radiológicas. Foi desenvolvido com o intuito de facilitar a interação de pesquisadores da área de saúde que não tinham um conhecimento avançado em programação sem PV (WHELAN *et al.*, 2004).

O NeatVision possui 300 imagens em seu acervo para serem manipuladas, porém, quando é necessária uma manipulação mais complexa, a programação textual da ferramenta é acionada novamente. Além disso, como pode ser visto na Figura 9, há a presença de uma área de montagem de uma aplicação visual ou fluxograma. Dessa forma, é possível afirmar que é coerente o NeatVision ser um APV, pois elementos textuais ainda são requeridos.

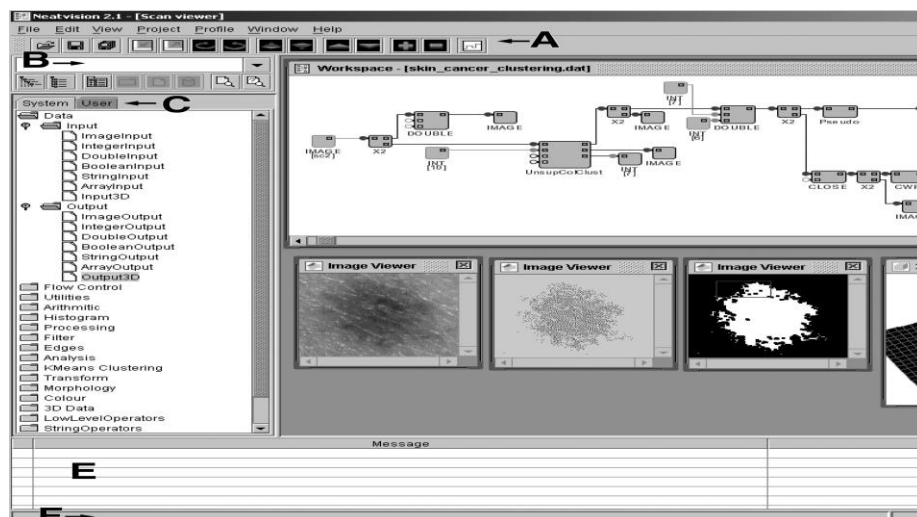


Figura 9. Interface visual do NeatVision.

Fonte: Whelan *et al.* (2004), pg 1782.

A geração das imagens médicas é baseada numa estrutura de fluxogramas que especificam, por exemplo, qual imagem será selecionada, que região da imagem será ampliada para fazer os processamentos de imagem e se há interposição de imagens de origem similar. Vale ressaltar que até para criar novas imagens no NeatVision é preciso uma noção de desenvolvimento de aplicações no modo textual.

2.3.3. PathSim Visualizer

O *Pathogen Simulation* (PathSim) é uma interface visual que permite a geração de ambientes virtuais ricos em informação (IRVEs - *Information-Rich Virtual Environments*) voltados para áreas de bioinformática e de saúde (POLYSa *et al.* , 2004). As aplicações geradas pelo PathSim possuem mais recursos voltados para a realidade aumentada do que para RV, mas seu destaque é importante devido à estrutura de sua arquitetura, que pode visto na Figura 10. A arquitetura do PathSim Visualizer é baseada no princípio que a única camada visível para o cliente será a camada visual.

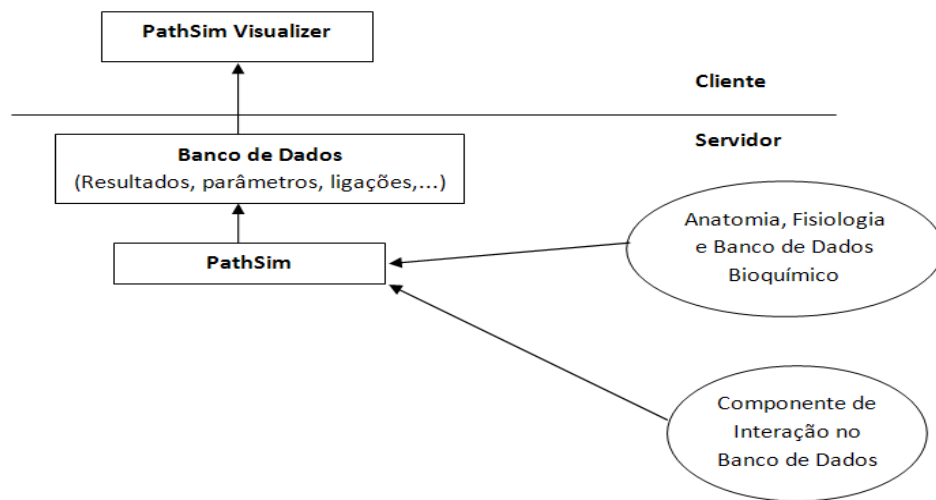


Figura 10. Arquitetura do PathSim Visualizer.

Fonte: Polysa *et al.* (2004), pg 9.

Outra característica de PathSim Visualizer é a sua estrutura de visualização que é baseada em fluxogramas e possui alguns elementos de PV como a ligação visual entre componentes para obter determinadas informações no navegador (POLYSb *et al.*, 2004), como pode ser vista na Figura 11.

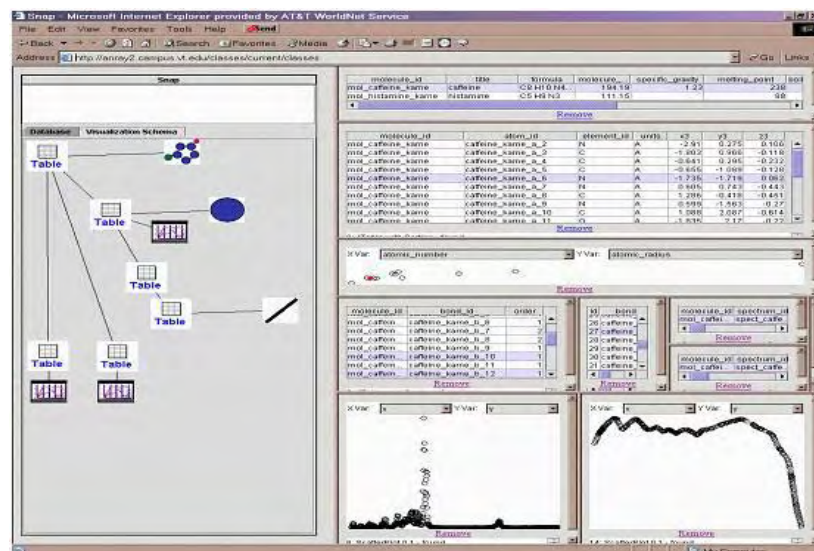


Figura 11. Estrutura de visualização do PathSim.

Fonte: Polysb *et al.* (2004), pg 7.

Todos os outros processos de desenvolvimento como a manipulação com o banco de dados, interação direta com o simulador de interação estão localizados no servidor e atuam de modo transparente para o usuário.

2.3.4. VR-MED

O VR-MED utiliza uma linguagem de domínio específico que utiliza técnicas de programação visual para possibilitar o desenvolvimento de AV aplicados ao ensino na área da Medicina de Família e Comunidade (MOSSMAN *et al.*, 2010).

Na Figura 12, é possível observar uma característica que confirma a presença de PV no VR-MED, como uma área central para a montagem de um fluxograma (Figura 12.3). A montagem do fluxograma pode ser facilitada por um conjunto de classes e notações do VR-MED (Figura 12.1) fruto de um estudo de caso prévio.

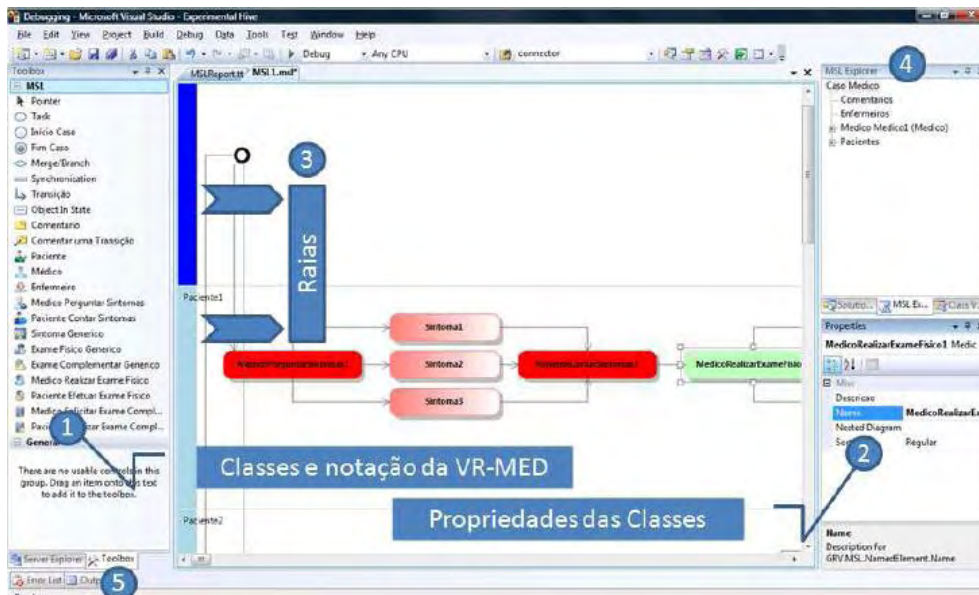


Figura 12. Protótipo Visual do VR-MED.

Fonte: Mossman et al. (2010), pg 17.

Ainda segundo Mossman et al. (2010), essa LPV foi desenvolvida para que programadores e projetistas tivessem uma notação simples e própria nessa área específica da saúde.

2.3.5. MeVisLab

O MeVisLab é um *framework* que fornece uma interface multiplataforma de programação visual modular, com um conjunto de ferramentas para o processamento de imagens e de ferramentas de visualização. Ele tem como objetivo propor uma alternativa para otimizar o tempo de desenvolvimento com o pacote ITK.

Como o ITK não tem uma GUI de desenvolvimento para o usuário, o MeVisLab combina uma série de conceitos de *design* para prototipagem rápida de aplicações voltadas à visualização e análise, fornecendo um conjunto abrangente de processamento geral de imagens e ferramentas de visualização para imagens médicas multidimensionais, bem como métodos avançados de órgãos específicos e as tarefas clínicas (REXILIUS et al., 2004).

Na Figura 13 pode-se observar a plataforma de desenvolvimento MeVisLab, onde cada retângulo (Figura 13b) representa um módulo do MeVisLab que faz chamadas ao ITK. Diferentes cores indicam processamento de imagens, visualização e módulo macro. Pode ser visto, também, vários analisadores como uma interface de pesquisa de módulos (Figura 13a) e os analisadores de ligação (Figura 13c), além do módulo do painel do operador de carga básica (Figura 13d).

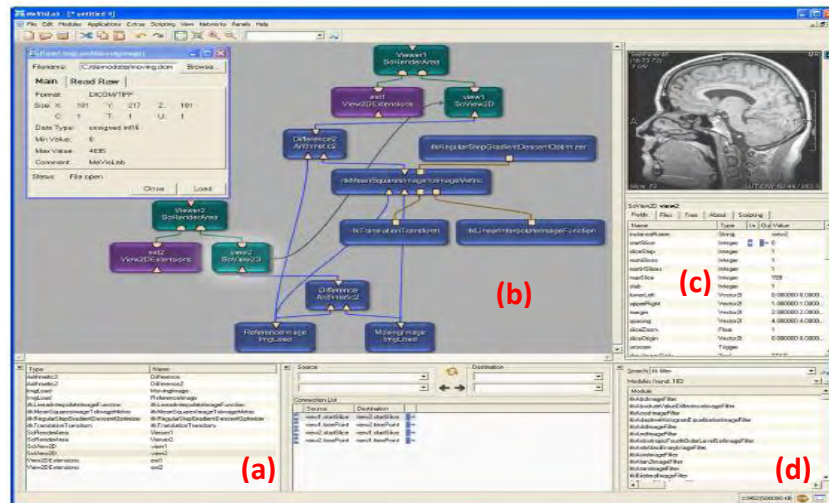


Figura 13. Interface do MeVisLab.

Fonte: Augustine et al. (2004), pg 10.

Pelas características apresentadas de recursos gráficos baseados em módulos, com presença de linguagem textual e expressões visuais, é possível afirmar que o MeVisLab se enquadra num APV.

2.3.6. SCIRun

O SCIRun é um *framework* baseado em RV, que não é voltado exclusivamente para a área médica. Ele possui simulações em grande escala que podem ser compostas, executadas, controladas e ajustadas de forma interativa. O SCIRun foi desenvolvido para reduzir o tempo que o cientista gasta no processo de modelagem e para fornecer uma ferramenta para a exploração de problemas da engenharia (JOHNSON e PARKER, 1995). Além disso, esse *framework* permite que o usuário controle interativamente simulações científicas, enquanto a computação está em andamento.

A criação da simulação é realizada por meio de uma interface de PV, como pode ser vista na Figura 14. Para executar o programa, é necessário especificar os parâmetros de uma GUI do usuário. Dessa forma, é possível afirmar que o SCIRun é um APV devido a composição de elementos gráficos, como fluxogramas, por exemplo.

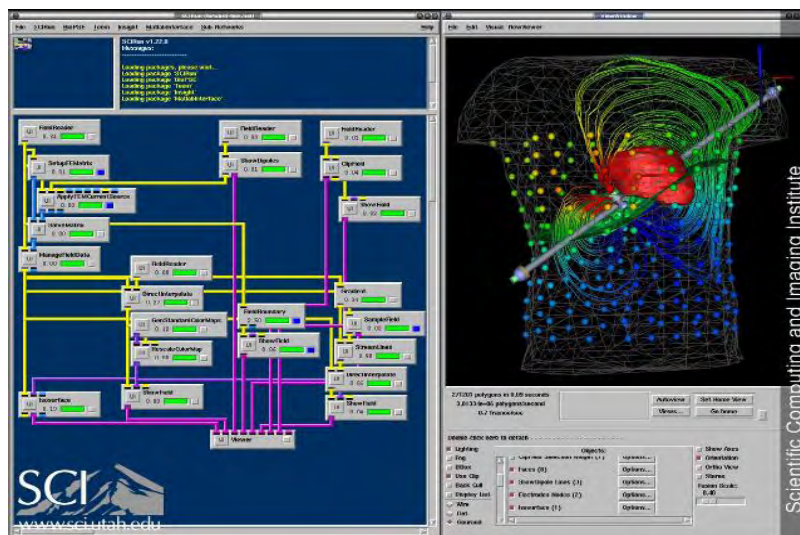


Figura 14. PV aplicada no SCIRun voltado para medicina.

Fonte: http://linux.softpedia.com/screenshots/SCIRun_1.jpg.

A interface de PV do SCIRun permite que os cientistas construam as simulações por meio de componentes visuais de valor computacional. Embora o APV seja o foco central de SCIRun, ele requer um poderoso conjunto de ferramentas computacionais para funcionar da maneira adequada.

2.3.7. LabView

O LabView é um APV que surgiu para oferecer as características requeridas pelos *softwares* de análise de fluxo. Oferece um ambiente que apresenta elementos gráficos chamados de Instrumentos Virtuais (VI – *Virtual Instruments*) que são equivalentes a sub-rotinas de programas em construção (REGAN e GREGORY, 1995). Esse ambiente tem sido utilizado por cientistas que não são da computação ou por programadores que precisam customizar programas de aplicações complexas. É compatível apenas em Mac OS e no *Windows* sua versão compila programas para executar em velocidades comparáveis a programas compilados a partir de seu código em C de 4 a 10 vezes menos tempo na escrita e na depuração do que em linguagens baseadas em texto. É possível notar, na Figura 15, que o LabView também é utilizado na área de saúde.

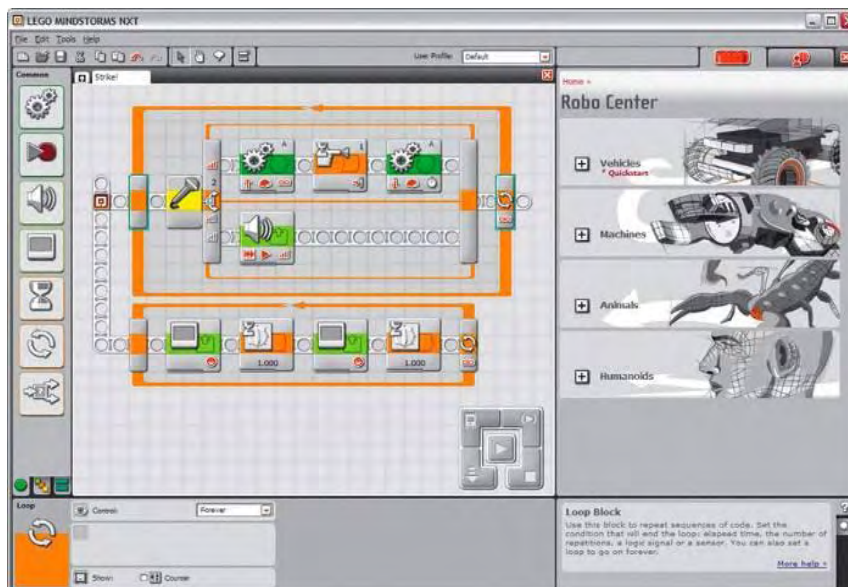


Figura 15. Interface visual do LabView.

Fonte: http://i.livescience.com/images/061012_lego_A_02.jpg.

Sua aplicabilidade pode ser observada, por exemplo, em um sistema para o controle independente de amônia e glutamina em um reator de fibra oca (REGAN e GREGORY, 1995) e na análise de fluxo de caminhos metabólicos (STOLL *et al.*, 1996). Nesses dois casos, o ambiente de PV foi utilizado por ser adequado ao desenvolvimento de programas personalizados para controle de bioprocessos.

2.3.8. SOFA

O *Simulation Open Framework Architecture* (SOFA) é um *framework open-source* voltado para a geração de aplicações baseadas em RV na área médica, principalmente com recursos de deformação e colisão (HALD *et al.*, 2011). A geração da aplicação ocorre por meio de um sistema baseado em componentes no qual a estrutura é montada pelo próprio *framework*.

O SOFA possui uma lógica de desenvolvimento baseada em grafos de cena, visto na Figura 16, para organizar/processar elementos de uma simulação e para separar as tarefas computacionais das tarefas paralelas (PERNOD *et al.*, 2010).

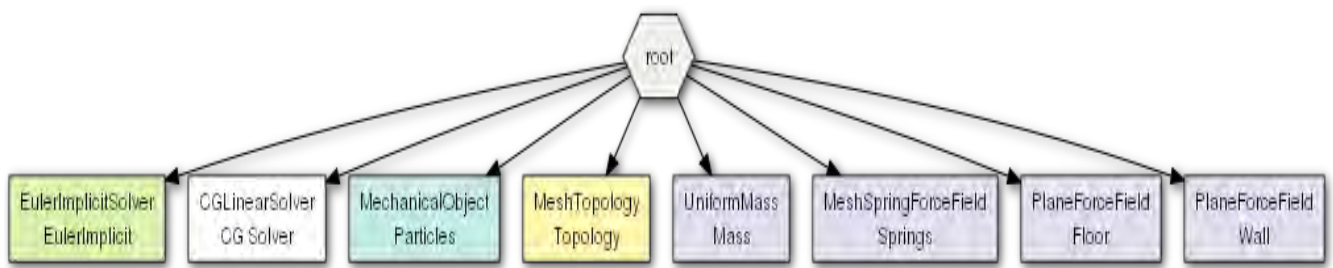


Figura 16. Exemplo de um grafo de cena para uma aplicação com o SOFA.

Fonte: <http://wiki.sofa-framework.org/w/images/graphviz/TutorialBasicCube---ba97c90c8b2fa3ef42677bf5a8a3020fe17bb5a9.png>

Com o SOFA é possível gerar aplicações de simulações médicas como treinamento para laparoscopia (ALLARD et al., 2007), por exemplo, como pode ser visto na Figura 17.

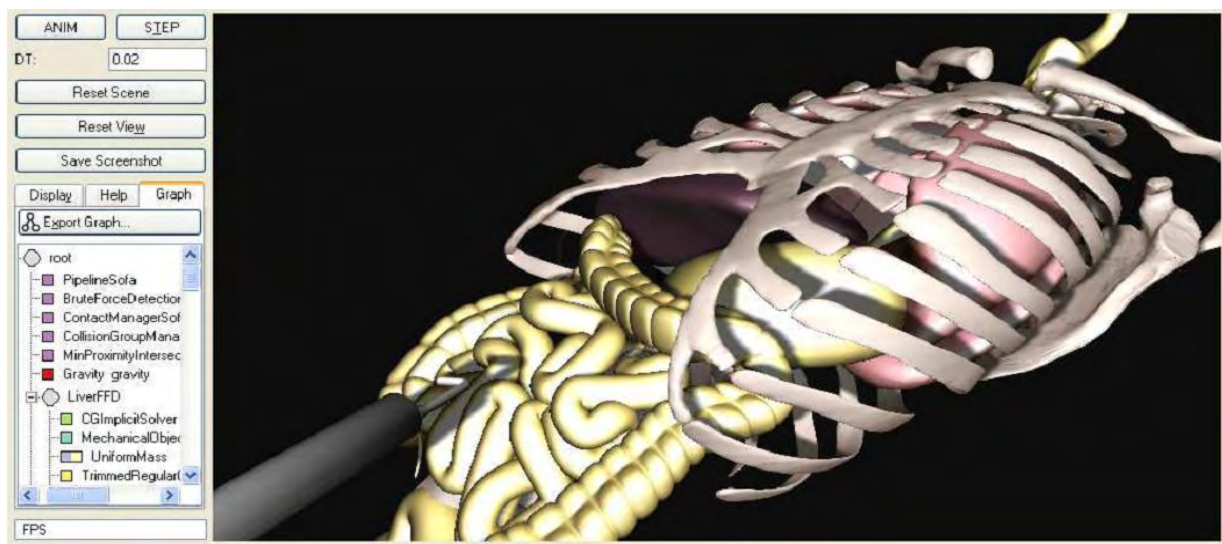


Figura 17. Exemplo de uma aplicação de laparoscopia utilizando o SOFA.

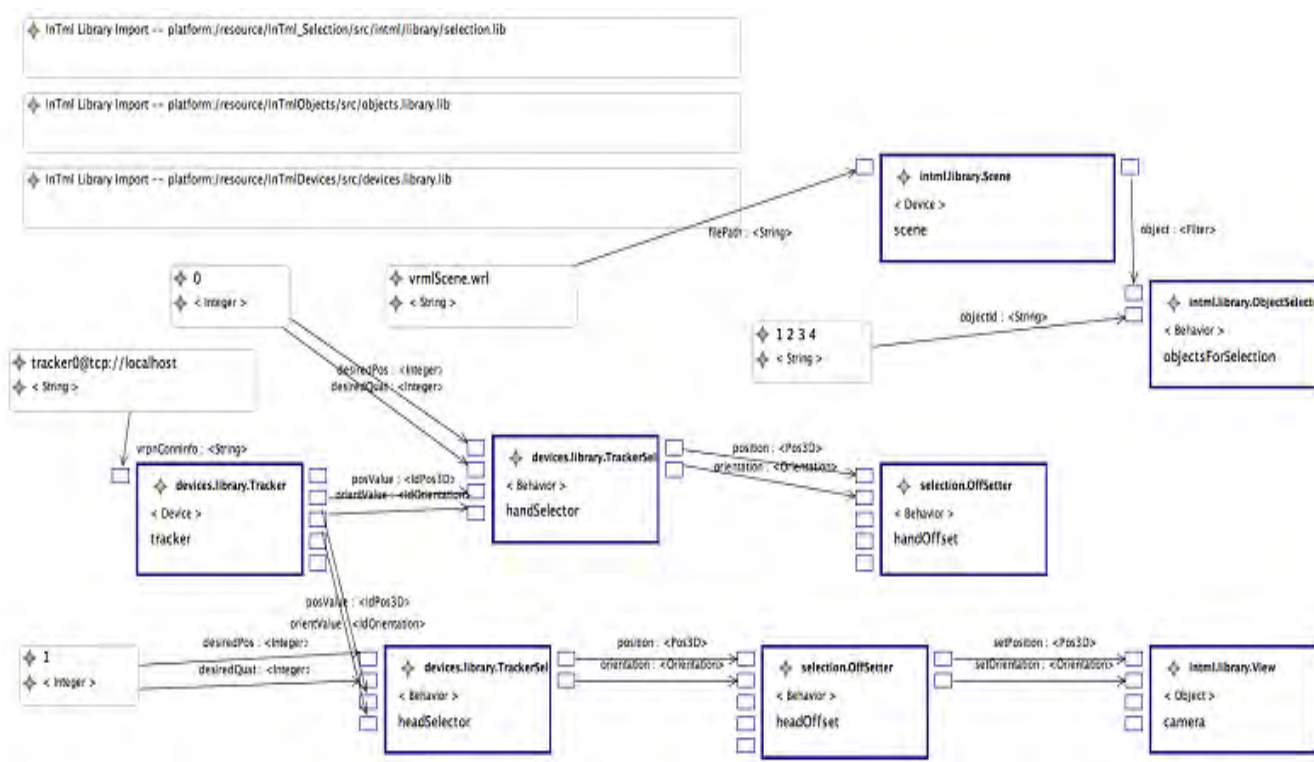
Fonte: Allard et al. (2007), pg 5.

De acordo com Allard et al.(2007), os principais objetivos do *framework* SOFA são: a) proporcionar uma estrutura de *software* comum para a comunidade de simulação médica, b) habilitar o compartilhamento/troca de componentes e reduzir o tempo de desenvolvimento, c) promover a colaboração entre grupos de pesquisa, d) habilitar a validação e comparação de novos algoritmos e e) ajudar na padronização de descrições de conjuntos de dados anatômicos e biomecânicos.

2.3.9. InTml

O InTml trabalha com uma linguagem de marcação para técnicas de interação (*Interaction Technique Markup Language* - InTml) que descreve técnicas de interações

3D, utilizadas em aplicações baseadas em RV (FIGUEROA *et al.*, 2002). Basicamente, o InTml utiliza uma linguagem, em XML, que descreve dispositivos de saída, de entrada, técnicas de interação, aplicações e as características em formato X3D (XML 3D), no qual o InTml é compatível.



Fonte: Figueroa e Castro (2011), pg 26.

2.3.10. Voreen

gerar aplicações médicas com recursos de corte e visualização tridimensional (MEYER-SPRADOW *et al.*, 2010). O desenvolvimento de uma aplicação visual na área médica ocorre por meio da montagem de um fluxograma, como pode ser visto na Figura 19.

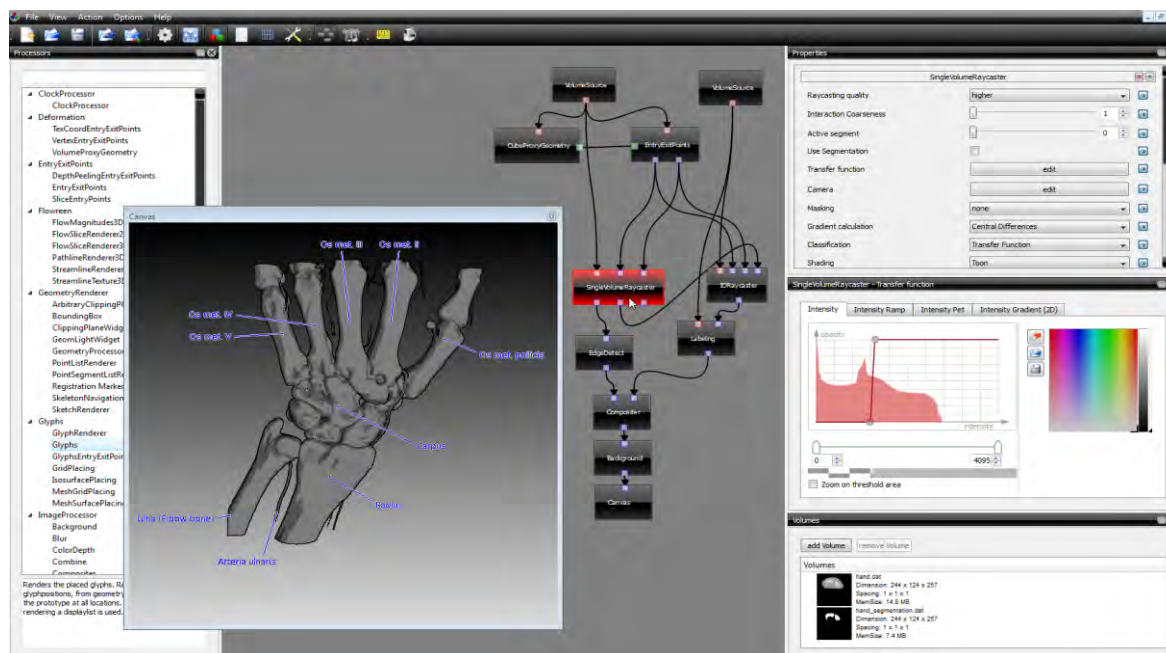


Figura 19. Leiaute do Voreen em um exemplo montagem de fluxogramas.

Fonte: <http://www.voreen.org>

A aplicação visual (em fluxograma) representa toda a lógica de dados da aplicação em RV. Nesse sistema a visualização e a montagem da aplicação possuem a mesma importância no sistema. Dessa forma o Voreen divide o sistema em dois modos: a) Modo de desenvolvimento e b) Modo de visualização (MEYER-SPRADOW *et al.*, 2009).

2.4. Análise comparativa dos *frameworks* com PV aplicados na área de saúde

Neste capítulo, foi possível observar como cada um dos *frameworks* pesquisados, preferencialmente com recursos em RV e aplicados na área de saúde, utiliza as técnicas de PV para resolver problemas de interação ou para tornar o desenvolvimento da aplicação ágil. De acordo com o Quadro 2, uma tabela comparativa foi elaborada para avaliar o comportamento geral dos *frameworks* e como as técnicas de PV foram empregadas. Foram abordados os seguintes parâmetros de avaliação: a) se há a presença de elementos de RV e qual(is) técnicas, b) se o *framework* foi projetado

exclusivamente para atender necessidades da área de saúde, c) qual a técnica de PV está sendo adotada e d) as funções para o qual o sistema foi criado.

Quadro 2. Comparativo entre os *frameworks* com PV aplicados na área de saúde.

Framework	Elementos de RV presentes	Voltado exclusivamente para área de saúde?	Técnica de Programação visual	Funções
VisProg	Não possui	Sim	APV	Possibilita a manipulação a ferramenta <i>Orange</i>
NeatVision	Não possui	Sim	APV	Facilita o diagnóstico a partir de imagens radiológicas
PathSim Visualizer	Visualização 3D	Sim	LPV	Interface visual que permite a geração de Avs
VR-MED	Não possui	Sim	LPV	Possibilita o desenvolvimento de AV aplicados ao ensino na área da Medicina
MeVisLab	Utilização de aparelhos de medição, com retorno de força	Sim	APV	Alternativa para otimizar o tempo de desenvolvimento com o pacote ITK
SCIRun	Retorno háptico, visualização 3D	Não	APV	Reduz o tempo gasto no processo de modelagem
LabView	Não possui	Não	APV	Oferece características requeridas pelos <i>softwares</i> de análise de fluxo
SOFA	Deformação , colisão	Não	LPV	Gera aplicações baseadas em RV na área médica, principalmente com recursos de deformação e colisão de forma livre
InTml	Deformação, colisão, estereoscopia, utilização de dispositivos de interação	Não	LPV	Descreve técnicas de interações 3D, utilizadas em aplicações baseadas em RV
Voreen	Suporte Volumétrico	Não	SPV	Utilizado para aplicações de prototipação rápida, multiplataforma e <i>open-source</i>

Com relação aos *frameworks* aplicados na área de saúde apresentados no Quadro 2, é possível destacar que a PV naqueles que não são baseados em RV é utilizada com o objetivo de: a) tornar o processo de desenvolvimento mais simples, e b) tornar o processo de desenvolvimento mais ágil. Tal constatação foi observada no VisProg, no NeatVision, no VR-MED e no LabView.

Já em relação aos *frameworks* que são baseados em RV, a PV tem o intuito de tornar a criação de aplicações baseadas em RV uma tarefa menos complexa para o desenvolvedor. Essa observação foi confirmada em *frameworks* como o PathSim Visualizer, o MeVisLab, o SCIRun, o SOFA, o InTml e o Voreen. É importante ressaltar que a metade dos *frameworks* pesquisados com PV foi desenvolvida exclusivamente para a área de saúde. São eles: VisProg, NeatVision, PathSim Visualizer VR-MED e MeVisLab. Com essa última informação pode-se sugerir que a PV é tratada como um artifício facilitador no desenvolvimento de aplicações, na área de saúde.

Outro fato relevante é que alguns *frameworks* baseados em RV, que inicialmente não eram aplicados na área de saúde, foram adaptados para suprir necessidades específicas da área. Isso ocorreu com os *frameworks* SCIRun, SOFA e InTml. Porém a adaptação dos *frameworks* de RV, muitas vezes, pode resultar em funcionalidades específicas e restritas para o usuário.

2.5. Conclusão

Ultimamente, é perceptível o aumento da incidência de *frameworks*, baseados ou não em RV, aplicados na área de saúde. Entretanto, a qualidade de interação não está diretamente vinculada ao crescimento quantitativo dessas estruturas. O levantamento realizado neste Capítulo reforça essa afirmação, pois embora existam muitos *frameworks* baseados em RV e aplicados na área de saúde, ainda é restrito o número de *frameworks* que apresentam um escopo de atuação bem definido, uma estrutura visual e de interação voltada para o público-alvo e recursos em RV que sejam efetivamente utilizados pelos usuários.

Retomando o questionamento feito no início do capítulo 2, é possível responder que a ausência de uma padronização visual também está presente em *frameworks* baseados em RV. A pesquisa realizada ainda observou que não há, muitas vezes, um padrão sobre o uso ou não das técnicas de PV entre os *frameworks* aplicados na área de

saúde. Outro fato relevante, que também ajuda a responder o questionamento feito, está presente em uma pesquisa de Santos e Machado (2009) na qual afirma e comprova que, dentre os *frameworks* em RV avaliados, não existe um padrão de *framework* na área de RV para saúde. Essa falta de definição de como seria um *framework* para área de saúde dificulta ainda mais o processo de desenvolvimento de aplicações para o profissional de saúde que deseja utilizar esse tipo de componente e não possui conhecimentos específicos em programação. Baseado nesses pontos, uma arquitetura de PV voltada para o desenvolvimento de aplicações em RV é proposta para ser aplicada na área de saúde. Mais detalhes serão relatados no próximo Capítulo.

3

Arquitetura para PV de Ambientes baseados em RV para Área de Saúde

No capítulo anterior, foi observada a carência de PV para auxiliar usuários leigos em programação durante o desenvolvimento de aplicações em *frameworks* aplicados na área de saúde e que geram aplicações com recursos baseados em RV. Mesmo naqueles que possuem recursos visuais, a interação não ocorre de modo intuitivo como ocorreria em se a PV fosse empregada de forma adequada em *frameworks* voltados para os usuários da área de saúde.

Figueroa e Castro (2011) afirmam que é “difícil desenvolver aplicações com interfaces 3D complexas”, apesar dos progressos visíveis em *frameworks* baseados em RV atualmente. Eles acreditam também que a criação de novas linguagens e *frameworks*, muitas vezes, não facilita o desenvolvimento de componentes reutilizáveis para aplicações interativas em RV.

Baseando nas afirmações de Figueroa e Castro (2011), este capítulo tem como objetivo apresentar uma solução, baseada em PV e compatível com diversos *frameworks*, para a criação de aplicações baseadas em RV pelos próprios usuários da saúde. Tal solução é uma arquitetura para PV de ambientes baseados em RV para área de saúde.

3.1 Sobre a Arquitetura para PV de ambientes em RV para usuários de saúde

A arquitetura de PV se caracterizará como um SPV que apresentará um módulo visual com o objetivo de permitir a programação visual de aplicações em diversos *frameworks* baseado em RV aplicados na área de saúde. Além disso, permitirá a criação e a execução de aplicações baseadas em RV pelos usuários de saúde que são, na sua maioria, leigos em programação. A aplicação gerada por essa arquitetura conterá recursos estereoscópicos, mas a interface de desenvolvimento da mesma não apresentará recursos visuais 3D, devido à complexidade detectada por Figueroa e Castro (2011) e que foi relatada no início deste Capítulo.

Outro fato relevante é que a arquitetura será projetada para ser compatível com diversos *frameworks* baseados de RV, dos quais alguns exemplos podem ser encontrados no Capítulo 2. Por meio desse SPV, os profissionais de saúde poderão criar aplicações baseadas em RV por meio de estruturas de desenvolvimento totalmente gráficas, através da montagem de fluxogramas. Com essa arquitetura, o usuário poderá não depender de um treinamento prévio sobre programação computacional em um *framework* em RV específico para criar suas próprias aplicações. Para gerar a aplicação

com recursos em RV, o usuário de saúde deverá entender apenas quais componentes farão parte da sua aplicação em RV e como eles irão interagir gráfica e logicamente entre si. Linhas de comandos textuais em *frameworks* em RV estarão em segundo plano para criar a aplicação porque essa arquitetura permitirá apenas o desenvolvimento por PV.

O objetivo do SPV é ser efetivamente utilizado por usuários da área de saúde na criação e execução de aplicações em RV. Para que essas tarefas ocorram dentro do esperado, o SPV oferecerá um processo de desenvolvimento de aplicações que obedecerá às técnicas de PV. Assim, professores, alunos ou até mesmos profissionais liberais da área de saúde poderão gerar suas próprias aplicações com recursos em RV para estudo, treinamento ou simulação na área. Tais aplicações poderão apresentar recursos em RV como a estereoscopia, a sensibilidade háptica, o suporte à colisão e a deformação dos objetos virtuais e o suporte a modelos com múltiplas camadas.

Outro fato importante é que o SPV garantirá compatibilidade com vários *frameworks* em RV da saúde. Para isso, será necessária uma série de avaliações seguindo as abordagens da engenharia de *software* a fim de definir quais recursos serão disponibilizados, quais atividades serão realizadas pelo sistema e quais módulos formarão esse sistema, por exemplo.

Antes de definir a arquitetura, será necessário avaliar as necessidades reais do público-alvo e obter informações que delimitarão as funcionalidades e reações do SPV. Então, deve ser realizado um levantamento sobre as necessidades e dificuldades encontradas na área de RV pelos profissionais da área de saúde. Além disso, deve ser avaliado o grau de interesse e conhecimento desses profissionais em utilizar melhor os *frameworks* que geram aplicações em RV.

3.2 Avaliação preliminar com os profissionais de saúde

Foi realizado um levantamento sobre o nível de conhecimento dos profissionais da saúde com *frameworks* baseados em RV. Além disso, foi analisado o grau de interesse desses entrevistados ao utilizar sistemas baseados em RV para o processo de treinamento, aprendizagem e simulação na área de saúde.

Embora a amostra do levantamento não seja estatisticamente significativa, os resultados serviram para definir as necessidades dos usuários da saúde em *frameworks* em RV. Uma amostra não é estatisticamente significativa quer dizer quando a amostra não é aleatória, pois o levantamento foi respondido por voluntários, e o seu tamanho não

representa a quantidade necessária para representar a população de usuários de saúde. Mesmo assim, essa avaliação é importante para elucidar as impressões gerais do público-alvo. Com os resultados da avaliação, também foi possível delimitar quais funcionalidades e técnicas de PV no CyberMedVPS seriam eficazes no processo de desenvolvimento de aplicações em RV pelo usuário da saúde.

3.2.1. Aspectos éticos do projeto

O projeto do CyberMedVPS foi aprovado pelo Comitê de Ética em Pesquisa com Seres Humanos (CEP) do Hospital Universitário Lauro Wanderley (HULW) na UFPB, Protocolo nº 614/10 e Folha de Rosto nº 373810. A coleta de dados, referente a um questionário, foi realizada no Congresso Brasileiro de Informática em Saúde (CBIS 2010) e com profissionais e estudantes vinculados às coordenações de Ciências Médicas (CM) e de Ciências em Saúde (CCS), ambas pertencentes à UFPB. Tal levantamento de dados foi realizado no mês de outubro de 2010 e o questionário empregado pode ser visualizado no Apêndice B.

3.2.2. Aplicação do questionário

O questionário foi composto por 22 questões objetivas separada em duas partes. A primeira parte referiu-se a um levantamento sócio-econômico para determinar o grau de escolaridade, a área de formação, se possui acesso ao computador, entre outros. A segunda parte do questionário constituiu-se de uma avaliação de conhecimento geral para determinar o nível de envolvimento dos profissionais de saúde com sistemas compostos por elementos em RV.

Foram entrevistados 32 profissionais da área da saúde, sendo 50% de estudantes, 21,8% de professores, 6% de profissionais liberais e 21,8% de professores/profissionais liberais; com faixa etária a partir de 18 anos. Nessa amostra, apenas duas pessoas estavam vinculadas a instituições privadas.

Quando perguntados sobre as condições e acesso dos materiais de anatomia, 53,1% da amostra afirmaram que utilizavam materiais sintéticos e naturais nas instituições, dos quais 66,6% relataram que as peças anatômicas estavam sempre disponíveis, mas em mau estado de conservação. Com relação a acesso e as condições dos materiais cirúrgicos, 60% afirmavam que tais elementos eram novos e defasados e 46,6% informaram que raramente utilizavam essas ferramentas na instituição. Além disso,

a totalidade tem conhecimento básico em informática, tem acesso ao computador e à *Internet* corriqueiramente.

Com relação à etapa de conhecimentos em geral, vale ressaltar que a totalidade dos entrevistados utilizou o computador alguma vez como ferramenta de estudo ou trabalho. Porém, 59,3% nunca trabalharam com programa de visualização 3D e 87,5% nunca utilizaram programas de treinamento cirúrgico.

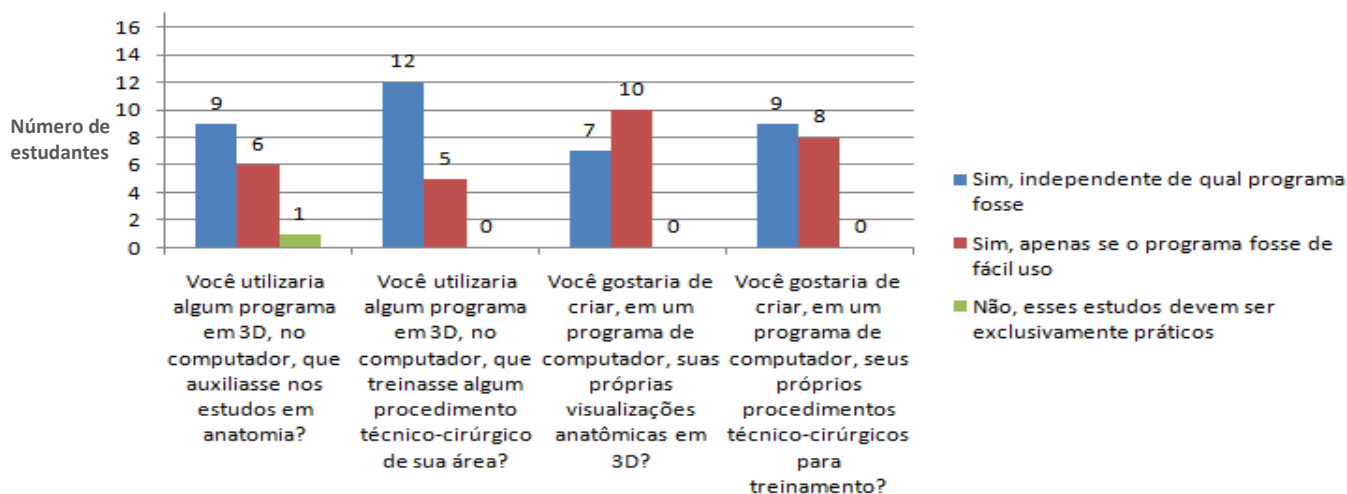


Figura 20. Interesse dos estudantes em relação à RV.

Outras perguntas foram feitas sobre o interesse de estudantes (Figura 20) e professores (Figura 21) sobre a utilização de diferentes abordagens em RV. Foi perguntado aos estudantes e professores se eles gostariam de utilizar programas com visualização 3D que os ajudassem nos estudos de anatomia e na criação de procedimentos técnico-cirúrgicos. A maioria dos estudantes afirmou que utilizaria um programa de visualização 3D independentemente de qual fosse. Enquanto isso, a maioria dos professores relatou que só usaria se o programa fosse de fácil uso.

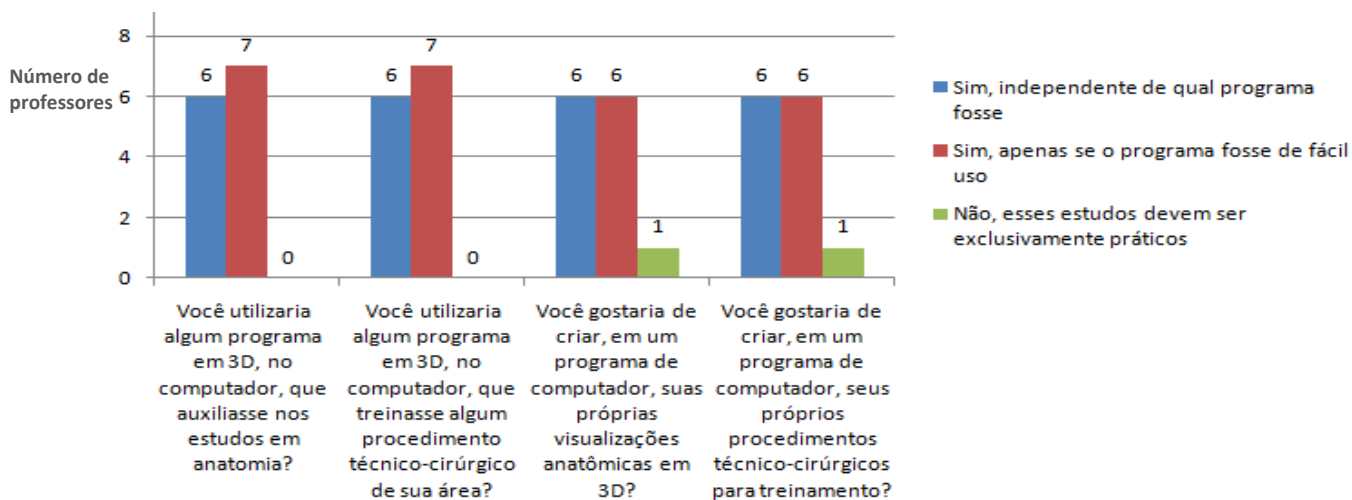


Figura 21. Interesse dos professores em relação à RV.

Ainda baseado nos questionamentos apresentados nas figuras 20 e 21, indagou-se aos estudantes e professores se havia interesse de utilizar um programa no qual fosse possível a criação de suas próprias visualizações e procedimentos técnico-cirúrgicos. A maioria dos estudantes e professores demonstrou interesse em utilizar esses programas.

3.2.3. Avaliação dos resultados

Ao analisar os resultados apresentados na pesquisa, é possível sugerir que apesar das condições muitas vezes inadequadas de peças anatômicas e materiais técnico-cirúrgicos agregado ao amplo acesso do profissional de saúde ao computador como material de estudo, não há o contato efetivo desses profissionais com ferramentas baseadas em RV que poderiam melhorar substancialmente a qualidade de seu aprendizado/ensino. Outro resultado inferido é que a maioria dos entrevistados nunca utilizou programas com recursos de RV para visualização e treinamento de procedimentos, apesar deles demonstrarem claramente o interesse em utilizar esse tipo de programa.

É possível observar que há o interesse, por parte de estudantes e professores, em utilizar uma ferramenta em que seja possível visualizar aplicações em 3D, manipular técnicas cirúrgicas ou até mesmo criar suas próprias aplicações com recursos em RV para visualização ou utilização de uma determinada técnica cirúrgica. Portanto, a criação de um SPV atenderia as expectativas desses segmentos acadêmicos.

Se a amostra fosse estatisticamente significativa seria possível fazer análise de semelhança das amostras, fazer o Teste de chi-quadrado a fim de analisar a distribuição para analisar igualdade entre as categorias de professores e alunos entre outros métodos.

3.3 Requisitos da arquitetura

A avaliação sobre as reais necessidades dos profissionais de saúde em *frameworks* baseados em RV mostrou que a maioria da amostra tem interesse em utilizar sistemas em RV independente de qual seja. Mas, outra parte dos entrevistados afirmou que só utilizaria esse mesmo tipo de sistema se fosse de fácil interação. Portanto, para atender a todos os perfis dos usuários de saúde avaliados, foi definido que o SPV deve:

- a) ser um sistema de criação de aplicações em RV para área de saúde;
- b) ter uma interface visual baseada nos princípios de usabilidade e PV;
- c) ser um sistema de execução de aplicações em RV para saúde;

- d) permitir que o usuário de saúde crie suas próprias aplicações em RV;
- e) permitir que o usuário salve as suas aplicações visuais(ou fluxogramas);
- f) permitir que o usuário abra as aplicações visuais salvas;
- g) permitir que as aplicações sejam montadas apenas pelo modo gráfico;
- h) ser um SPV compatível com diversos *frameworks* em RV para área de saúde;
- i) gerar um código computacional genérico, ou seja, uma linguagem de programação como resultado final que possa ser compatível com *frameworks* de RV.

Além de o SPV prover a criação e a execução de aplicações com recursos em RV, o sistema tem como funcionalidades: abrir e salvar aplicações visuais(fluxogramas), garantir que aplicação final seja válida em termos de parâmetros e relacionamentos, gerar um código-fonte genérico da aplicação visual criada, gerar um arquivo de registro sobre as ações do usuário, ser compatível com diversos *frameworks* da área de saúde, mostrar mensagens de *feedback* e de dicas do sistema para o usuário. Com relação às funções do usuário no SPV, é importante ressaltar que ele deve:

- a) interagir com o sistema baseado em PV;
- b) compreender a função dos componentes que compõem a aplicação visual;
- c) inserir parâmetros de configuração dos componentes pelo modo gráfico;
- d) visualizar registros de suas ações pelo modo gráfico;
- e) salvar a aplicação visual;
- f) abrir uma aplicação visual salva;
- g) entender as mensagens de comunicação com o usuário e de dicas do sistema;
- h) executar a aplicação visual.

Além do levantamento das funcionalidades, é necessário realizar uma definição formal dos requisitos funcionais do CyberMedVPS. Tais requisitos podem ser vistos no Quadro 3, que se baseia na norma NBR:ISO/IEC 9126, que pode ser usada para especificar requisitos funcionais e não-funcionais do cliente e do usuário.

Quadro 3. Requisitos funcionais do Sistema de Programação Visual.

Funcionalidade	O SPV será um sistema de desenvolvimento de aplicações visuais que gera aplicações computacionais baseadas em RV para usuários da área de saúde
Usabilidade	Apresentará recursos baseados nos princípios do <i>design</i> de interface e adotará técnicas de PV
Confiabilidade	O SPV é capaz de validar as aplicações visuais montadas
Eficiência	O sistema deve responder as ações do usuário em tempo de execução
Manutenabilidade	As modificações realizadas pelo usuário serão executadas apenas pelo modo gráfico
Portabilidade	O SPV será compatível com diversos <i>frameworks</i> que geram aplicações em RV e será multiplataforma

Em relação aos requisitos não-funcionais do sistema, há um ponto que pode ser ressaltado: a) o SPV deve ser multiplataforma. Assim, o Sistema Operacional (SO) não é um problema adicional durante o processo de desenvolvimento de aplicações em RV pelo usuário no SPV.

3.4 Arquitetura do SPV

Para o SPV ser compatível com diversos *frameworks* baseados em RV e da área de saúde, foi preciso que sua arquitetura garantisse dois pontos fundamentais. São eles:

- a) a interação com *frameworks* de RV deve ser um processo imperceptível para o usuário que utiliza o SPV, ou seja, a lógica de desenvolvimento da aplicação visual será a mesma para o usuário independentemente de qual *framework* baseado em RV que o SPV esteja utilizando;
- b) todo o processo de desenvolvimento e execução da aplicação visual deve ocorrer apenas pelo modo gráfico, por meio de PV.

Além dos aspectos relatados, foi imprescindível garantir o desacoplamento da estrutura do SPV com os *frameworks* em RV. Portanto, foi definido que a arquitetura de PV é formada por dois módulos principais: a Camada de Visualização e a Camada de Comunicação. A arquitetura dessas camadas pode ser vista na Figura 22 e adotou o modelo arquitetural baseado em camadas.

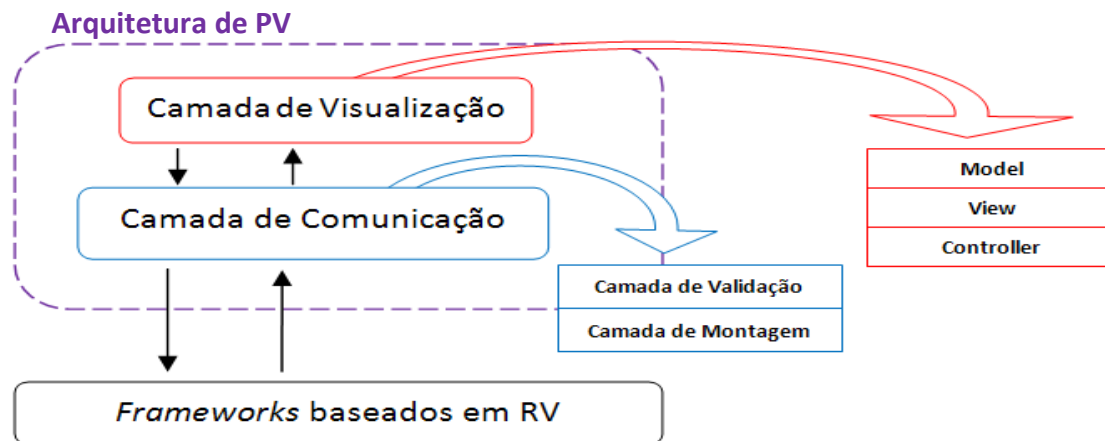


Figura 22. Arquitetura do SPV.

Fonte: Moraes e Machado, 2011, pg 5.

A Camada de Visualização é a única camada na qual o usuário tem contato direto no SPV. Todas as interações de entrada e saída do SPV com o usuário ocorrem nesse módulo. Além disso, é nessa camada que a estrutura de PV está implementada e onde a aplicação visual será montada pelo usuário. Vale ressaltar também que o modelo de padrão de projeto utilizado na camada de Visualização foi o MVC, que é um padrão de projeto de *software* que isola a parte lógica, da UI. Esse padrão foi escolhido devido a características como o isolamento entre a interface visual e o domínio lógico, para garantir a flexibilidade do sistema.

A outra camada é chamada de Camada de Comunicação. Esse módulo é responsável por traduzir a aplicação visual em uma linguagem computacional genérica. Além disso, tem como função habilitar a comunicação desse código com o *framework* de RV. Todo esse processo descrito ocorre de modo transparente para o usuário durante a execução do SPV. Como mostra a Figura 22, a Camada de Comunicação é subdividida em dois pacotes internos:

- a) camada de Validação: essa camada garante que a aplicação visual, gerada por PV, tem seus parâmetros e operações validados e é traduzida em uma linguagem genérica pelo SPV;
- b) camada de Montagem: nessa camada, a linguagem genérica é traduzida e tem seus parâmetros disponibilizados para geradores de código-fonte que obedecem à sintaxe dos *frameworks* em RV.

A camada do *framework* refere-se ao *framework* que geram aplicações em RV e aos componentes responsáveis por gerar o código-fonte de sintaxe compatível com tais

frameworks em RV. Dessa forma, esse código-fonte gerado é utilizado na execução final da aplicação em RV.

Algumas restrições arquiteturais foram previstas no SPV. A primeira delas é se o usuário quiser inserir dados, por meio de linguagem de programação, dentro da aplicação de RV, usando o SPV. Essa ação não é permitida, pois um dos requisitos previstos nesta arquitetura é que todo o processo de desenvolvimento seria utilizando apenas PV. A única solução forma de realizar essa ação seria acessando diretamente o código-fonte da aplicação em RV sem intermédio do SPV.

As outras limitações arquiteturais dizem respeito às restrições dos próprios *frameworks* de RV, como por exemplo, se o *framework* não é multiplataforma ou se ele não possui suporte de algum recurso em RV (sensibilidade háptica, estereoscopia, colisão, deformação, por exemplo). As soluções para essas restrições foram realizar um estudo prévio das limitações do *framework* utilizado e a tentativa de correção, quando for necessário.

3.5 Visão Lógica do SPV

A aplicação em desenvolvimento no SPV passa por quatro estados até que a execução da aplicação aconteça. A Figura 23 mostra que inicialmente não é criada nenhuma aplicação ao iniciar o SPV (Figura 23a). A linguagem só é desenvolvida à medida que a aplicação visual for sendo montada. Nesse momento, a aplicação está no estado de Linguagem Visual (Figura 23b). Ao completar a montagem da aplicação visual, ocorre a validação de seus parâmetros a montagem de uma Linguagem Genérica (Figura 28c) que pode ser convertida para qualquer outra linguagem compatível com *frameworks* em RV, ou seja, é transformada em uma Linguagem Computacional (Figura 23d).



Figura 23. Diagrama de estados da aplicação gerada no SPV.

As principais atividades do usuário, no SPV, estão relacionadas à criação e à execução de aplicações baseadas em RV por meio de PV, como se elucidou nos casos de uso do usuário. A estrutura lógica para essas atividades (Figura 23) utiliza o SPV para

a criação e a validação da aplicação visual. Além disso, o SPV também é responsável pela criação da linguagem genérica. Externo à arquitetura, existe uma classe tradutora de código que montará uma aplicação computacional que será compatível ao *framework* baseado em RV para execução da aplicação gerada.

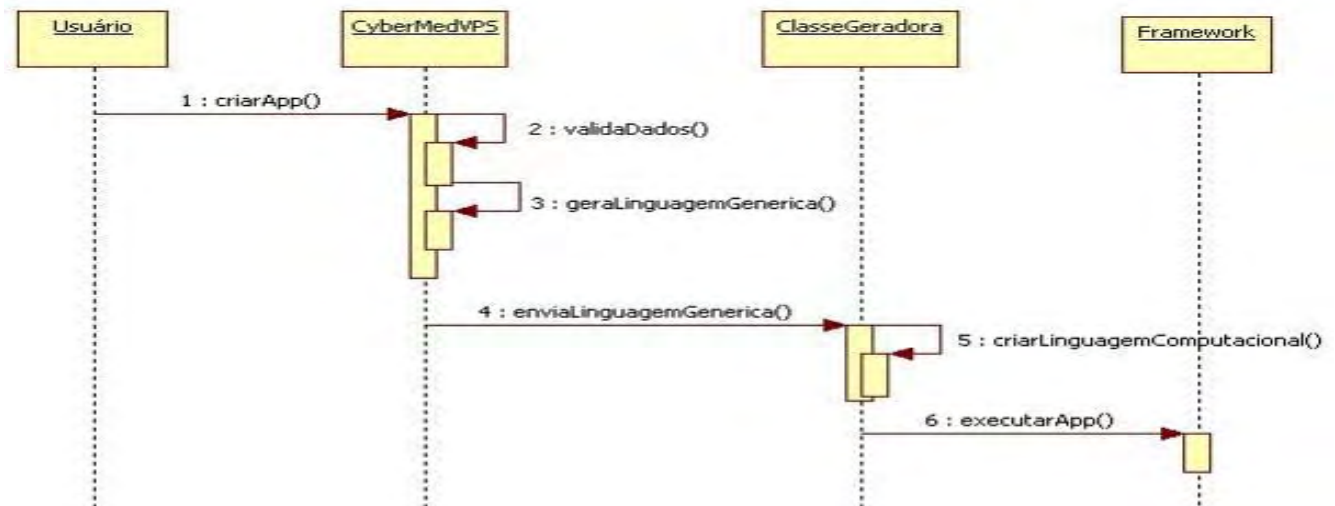


Figura 24. Diagrama de sequência sobre a criação e execução do SPV.

Para definir a visão lógica do SPV é necessário definir estruturas que possibilitem uma visão da dinâmica do sistema a fim de prosseguir com o desenvolvimento do projeto de SPV.

3.6 Visão dos dados do SPV

Após a definição de como os componentes do SPV estão organizados, foi necessário entender como se caracteriza o fluxo de dados da arquitetura. A dinâmica dos dados ocorre como mostra a Figura 25. O usuário se comunica apenas com a estrutura referente à camada de Visualização, por meio da inserção de parâmetros (Figura 25a) na aplicação visual, ou fluxograma, que é montada. O sistema responde as reações do usuário apenas em forma de mensagens no momento da execução do SPV (Figura 25b). À medida que usuário monta sua aplicação visual, a camada de comunicação analisa as ações e os parâmetros inseridos na aplicação durante o desenvolvimento.

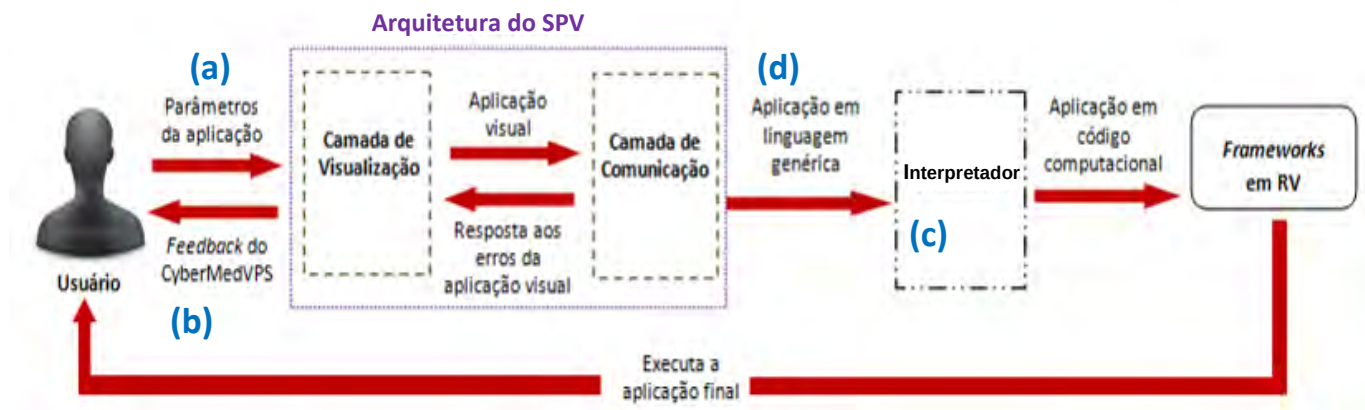


Figura 25. Fluxo dos dados no SPV.

A camada de comunicação faz a transformação da linguagem visual em uma linguagem genérica (Figura 25d). Essa linguagem genérica é codificada por interpretadores (Figura 25c) que estão externos ao SPV e traduzida para um código computacional, ou código-fonte, que atende a sintaxe do *framework* de RV escolhido. Esse código-fonte é executado pelas chamadas dos *frameworks* por solicitação dos tradutores e a resposta é retornada ao usuário.

3.7 Conclusão

Espera-se que por meio da aplicação da arquitetura proposta o usuário possa utilizar a PV para criar aplicativos de maneira. Além disso, processos complexos como a compilação de um código-fonte, por exemplo, poderão ser transparentes para o usuário devido à forma como a arquitetura do SPV foi definida.

Após definir os requisitos de cada camada da arquitetura do SPV é importante avaliar se o que foi proposto é realmente eficiente para o público-alvo. Para isso, serão desenvolvidas as classes que formam as camadas do SPV. Dessa forma, todas as funcionalidades poderão ser aplicadas nessa arquitetura como planejado. O resultado concreto da arquitetura proposta foi chamado de CyberMedVPS. Maiores detalhes sobre o desenvolvimento do CyberMedVPS serão relatados no capítulo 4.

4 Desenvolvimento do CyberMedVPS

O CyberMedVPS tem como intuito ser desenvolvido com base na arquitetura proposta do SPV e relatada no Capítulo anterior. Assim, esse SPV foi implementado com base na hierarquia de camadas para manter a interdependência entre elas. Além disso, a ordem adotada no processo de desenvolvimento das camadas obedeceu à visão do usuário, ou seja, a implementação iniciou-se na GUI e foi se direcionando para camadas mais internas do CyberMedVPS que são responsáveis pelo controle das informações do SPV. Vale salientar que o desenvolvimento desse sistema seguiu três etapas principais: a) a definição da camada de visualização, b) a definição da camada de comunicação e c) a definição da validação.

Apesar do processo de desenvolvimento de *software* adotar modelos tradicionais na maior parte do projeto, ainda há a presença de elementos do modelo de engenharia de usabilidade durante a definição da UI durante o processo de desenvolvimento. Um desses pontos é a presença de um *design* iterativo para o desenvolvimento da camada visual. Mais detalhes sobre o desenvolvimento das camadas do CyberMedVPS serão detalhados a seguir.

4.1. Definição da Camada de Visualização

A Camada de Visualização contém todos os elementos visuais do CyberMedVPS que são oferecidos para o usuário. Além disso, a interação com esses recursos visuais é o único modo de comunicação do usuário com o SPV. Dessa forma, dois aspectos na Camada de Visualização foram definidos. São eles:

- a) a presença da PV no processo de desenvolvimento de aplicações visuais no CyberMedVPS;
- b) a presença de usabilidade na UI do CyberMedVPS para o processo de interação pelos usuários de saúde.

Esses dois pontos têm como objetivo ajudar o usuário de saúde no processo cognitivo de interação com o CyberMedVPS. A cognição está relacionada na maneira como se adquire conhecimentos sobre um sistema em uso (PREECE *et al.*, 1994) (WEBER e VOSGERAU, 2011). Isso inclui aspectos como a compreensão, a lembrança, a razão, a aquisição de habilidades e a criação de ideias, por exemplo.

O primeiro ponto refere-se à estrutura geral do módulo visual do CyberMedVPS, principalmente aspectos relacionados à usabilidade. A usabilidade é uma técnica do

design de interface que permite a melhoria no processo de interação do usuário com o sistema, por meio de elementos visuais como cores, disposição de elementos no leiaute, atalhos visuais e fontes até aspectos abstratos como aprendizagem, memorização, qualidade (DUBEY e RANA, 2011).

Com relação a PV no CyberMedVPS, ela foi implementada de forma que o fluxo de dados fosse visível para o usuário e o desenvolvimento da aplicação ficasse em alto nível de compreensão. Para que as técnicas de usabilidade e PV coexistissem no CyberMedVPS de maneira equilibrada foi necessário que:

- a) a UI apresentasse dados visuais consistentes em termos de formato, cor, tipografia e padronização de componentes, por exemplo;
- b) o SPV apresentasse informações que sejam facilmente assimiladas pelo usuário;
- c) o usuário precise memorizar o mínimo possível do processo de interação;
- d) os dados que foram inseridos no CyberMedVPS fossem consistentes ao longo da execução, ou seja, uma estrutura visual não apresentasse significados diferentes em momentos diferentes do desenvolvimento.

O processo de interação do CyberMedVPS foi realizado prioritariamente por ações realizadas pelo mouse. De acordo com Shneiderman e Plaisant (2010), as chances de erros têm a possibilidade de aumentar se o usuário mover suas mãos entre dispositivos de entrada para realizar etapas seguidas de interação. Mas essa constatação não proíbe que outros dispositivos de entrada de dados também possam estar habilitados. Portanto, quando for necessário, o CyberMedVPS também disponibiliza atalhos de interação pelo teclado.

A ideia inicial para UI do CyberMedVPS foi definir um protótipo visual que ilustrasse a estrutura funcional do sistema. O leiaute do CyberMedVPS tem uma organização de áreas semelhante a maioria dos sistemas *desktops* porque o seu público-alvo é composto por profissionais que não são da computação. Então, quanto menos informações sobre a UI o usuário tiver que aprender no processo cognitivo, mais atenção ele direciona para o processo de desenvolvimento da aplicação visual no CyberMedVPS. Além disso, de acordo com Tidwell (2006), o usuário deve ser capaz de entender a estrutura de informações oferecidas no leiaute para que a compreensão seja eficiente.

O resultado do protótipo da interface do CyberMedVPS pode ser visto na Figura 26 (MORAIS e MACHADOa, 2011). Nesse módulo visual, foi definida uma área lateral no qual os componentes que fazem parte da aplicação visual (Figura 26A) são

disponibilizados. Além disso, foi definida uma aba superior onde serão inseridos os atalhos com as opções do menu (Figura 26B). Outra região é a área central onde é montada a aplicação visual (Figura 26C) por meio de PV. E, por fim, uma região no rodapé que permite a comunicação do SPV com o usuário (Figura 26D).

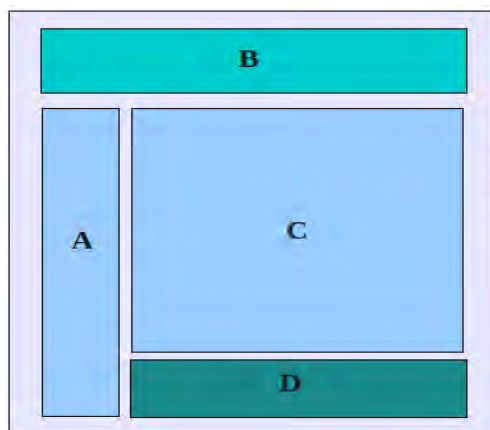


Figura 26. Protótipo de interface do CyberMedVPS.

Fonte: Moraes e Machadoa (2011), pg 2.

A escolha por um leiaute simples baseou-se na ideia de Shneiderman e Plaisant (2010) que afirmam que o desenvolvimento de uma interface pode ser comprometido pela presença de muitas informações. E esse comprometimento pode gerar baixo desempenho e desmotivação do usuário no sistema. Vale ressaltar que a delimitação de áreas de acordo com a função combate esse tipo de problema no protótipo visual do CyberMedVPS. Além disso, garante a consistência e a padronização dos dados apresentados na UI do SPV.

A proposta desta UI é que a informação trafegue pelo sistema por meio de uma série de estágios ordenados. Por isso, a definição do protótipo de interface foi imprescindível para o SPV, pois de acordo com Shneiderman e Plaisant (2010), a definição é necessária para avaliar o contraste, a profundidade de percepção e os impactos das características visuais.

Após delimitar a função de cada uma das subáreas do protótipo visual do CyberMedVPS, foram definidas as características visuais de cada dessas subáreas, obedecendo aos princípios do *design* de interface e a PV.

4.1.1. Área superior da UI do CyberMedVPS

A estrutura superior do CyberMedVPS é referente a barra de menu que fica disposta horizontalmente e no alto da janela principal do sistema. De acordo com Cybis et

al.(2007), essa área é destinada para usuários de saúde que não tem a possibilidade de memorizar opções do sistema por atalhos do teclado.

Outra justificativa é que a criação de um menu de opções (Figura 27) possibilita a interação predominantemente pelo mouse. Dessa forma, o processo de interação com o mouse pode se tornar mais natural para usuários iniciantes ou leigos em sistemas de desenvolvimento de aplicações do que utilizando comandos específicos pelo teclado.



Figura 27. Barra de menu do CyberMedVPS.

A estrutura visual da barra de menu do CyberMedVPS é similar às encontradas na maioria dos aplicativos *desktops*, como nomenclaturas padrões (Arquivo, Editar e Ajuda), por exemplo. Tal similaridade é importante para garantir uma memorização de longo prazo já adquirida em outras estruturas computacionais sobre o processo de interação (SHNEIDERMAN e PLAISANT, 2010). Essa padronização também se aplica para algumas opções como Abrir, Fechar, Sobre, Ajuda, Fazer e Desfazer.

As opções da barra de menu do CyberMedVPS também possuem comandos específicos de um SPV, como as opções referentes a Ligação e Objeto, ambos participantes da aplicação visual. Além disso, houve a preocupação em tornar a nomenclatura das opções específicas diretamente vinculadas às suas funções. Por exemplo, na seção referente à Ligação os comandos tratam da ligação entre os componentes (excluir ou criar) e na seção referente aos Objetos os comandos tratam dos componentes que formam a aplicação visual em desenvolvimento.

Outra funcionalidade que está disponível na barra de menu é atalhos pelo teclado para usuários mais especializados na interação. Segundo Shneiderman e Plaisant (2010), os atalhos são importantes para usuários experientes em um determinado sistema. Isso acontece porque tais atalhos já estão memorizados por esses usuários e isso pode facilitar o processo de interação tornando-o mais eficiente em termos de velocidade de desenvolvimento.

4.1.2. Área lateral da UI do CyberMedVPS

A área lateral da UI do CyberMedVPS é destinada a disponibilizar os componentes visuais da aplicação baseada em RV que é montada. Antes de definir como

a organização desses elementos se caracteriza nessa subárea da UI do CyberMedVPS, é preciso definir quem são os componentes que fazem parte da aplicação visual.

a. Componentes de uma aplicação visual baseada em RV

Uma aplicação baseada em RV e voltada para área médica pode conter recursos como visualização estereoscópica ou monoscópica, sensação háptica no processo de interação, colaboração entre objetos remotos, colisão e deformação dos objetos virtuais utilizados na aplicação. A maneira como essas funcionalidades podem ser geradas e as suas aplicabilidades variam de acordo com o *framework* em RV utilizado.

Então, com base nas funcionalidades genéricas apresentadas e nas terminologias utilizadas pelos profissionais de saúde, foram definidos quais objetos podem formar uma aplicação visual baseada em RV e genérica em relação ao *framework*. Foram especificados os componentes usados para a criação de uma aplicação visual com recursos de visualização, sensibilidade háptica, colisão e deformação:

- **Modelo:** representa o objeto virtual utilizado na aplicação. Ou seja, cada imagem de visualização ou manipulação utilizada na aplicação corresponde a um objeto Modelo como, por exemplo, a estrutura óssea da bacia;
- **Instrumento:** é o objeto virtual o qual exerce uma ação sobre o modelo. Por exemplo, uma aplicação com um objeto virtual no formato de um bisturi pode ser um objeto Instrumento;
- **Toque:** é objeto responsável por representar a ação de colisão e deformação do Instrumento sobre Modelo;
- **Agregador:** componente responsável por agregar um mesmo conjunto de características para vários objetos do tipo Modelo;
- **Ambiente Visual:** é o espaço virtual onde o Modelo é ligado para ser visualizado;
- **Ambiente Tátil:** é o espaço virtual onde o Modelo é ligado para ser sentido com recursos hápticos, ou seja, com sensação de textura, grau de rigidez e retorno de força;
- **Exibição:** é o objeto que define a execução da aplicação finalizada.

A nomenclatura dos objetos da aplicação visual baseou-se na associação entre a linguagem operacional do usuário e a função exercida pelo objeto na aplicação em RV. Além disso, alguns comportamentos foram previamente estipulados no processo de

montagem da aplicação em RV como: o objeto Toque deve estar posicionado entre os objetos Instrumento e Modelo, na aplicação visual. Além disso, para que os objetos fossem visualizados foi necessário que todos os componentes estivessem relacionados ao objeto Ambiente Visual. Se por acaso for uma aplicação háptica é necessário que todos os modelos e instrumentos estejam ligados ao objeto Ambiente Tátil. Tanto o Ambiente Visual quanto o Ambiente Háptico podem ser ligados ao objeto Exibição.

A ideia inicial seria disponibilizar, no menu lateral, *links* textuais que representassem cada um dos objetos. Mas essa proposta ia contra os princípios de um SPV, que prioriza a apresentação na forma gráfica. Em outras palavras, se houver uma alternativa gráfica possível de ser implantada, ela deve ser executada no sistema. Com isso, foi proposto que seria disponibilizado na área lateral da UI do CyberMedVPS a mesma ilustração gráfica dos objetos que geram a aplicação visual. O resultado final da aba lateral, com seus componentes disponíveis, pode ser observado na Figura 28.

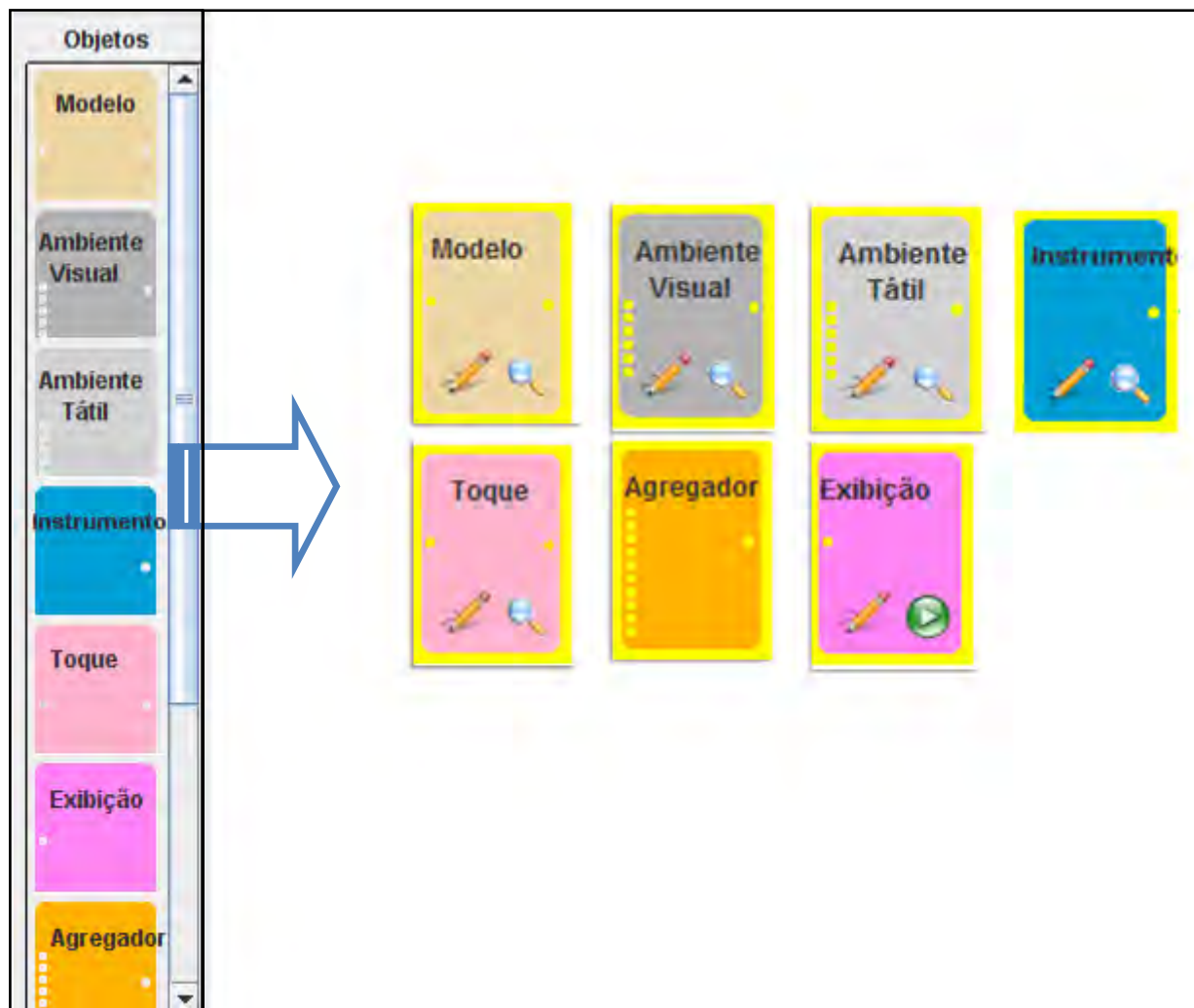


Figura 28. Área lateral da UI e os componente que fazem parte da aplicação visual do CyberMedVPS.

Foi desenvolvida, nessa área lateral, uma lista gráfica que disponibiliza os objetos que podem ser inseridos na aplicação visual. A ordenação desses componentes na área lateral seguiu um dos critérios sugeridos por Shneiderman e Plaisant (2010) que aconselhava o agrupamento de itens relacionados entre si.

Os objetos são visualizados na aba lateral na mesma estrutura gráfica que aparece na aplicação visual. Mas, na área lateral os objetos estão com dimensões menores e com os botões desativados. Isso serve para transparecer ao usuário que um mesmo componente, dependendo da área que esteja localizado no SPV, pode estar em uso ou não.

Vale ressaltar também que as cores dos objetos foram escolhidas a partir de um estudo que visava garantir que sensações equivocadas não fossem repassadas para o usuário pela UI do SPV. Então, os componentes adotaram cores frias próximas a tons pastéis para oferecer neutralidade para o usuário. Além disso, a escolha das cores estava associada à função desses elementos na aplicação visual. Dessa forma, as cores entre os objetos Ambiente Tátil e Ambiente Visual são similares para reforçar a ideia de que eles possuem funções equivalentes dentro da aplicação visual.

O evento responsável por enviar um objeto da área lateral para a área de desenvolvimento é o clique do *mouse*. Automaticamente esse componente é enviado para a área central do CyberMedVPS e algumas funcionalidades de edição e visualização são habilitadas.

4.1.3. Área central da UI do CyberMedVPS

A técnica de PV escolhida para ser implementada no CyberMedVPS foi o SPV, pois como foi visto no capítulo 2, é a forma de PV no qual o fluxo de dados é visível para o usuário, assim como o processo de montagem e a seleção dos parâmetros da aplicação visual. A estrutura que é responsável por garantir que o fluxo de dados fique visível é um fluxograma que será formado pela ligação dos objetos disponíveis na aba lateral.

O leiaute do objeto que faz parte da aplicação visual pode ser observado na Figura 29. É possível observar a identificação de cada objeto no topo do mesmo (Figura 29a), com uma fonte que facilita a leitura rápida. Além disso, também estão presentes ícones cujas ilustrações estão relacionadas às suas funções como o botão de edição (Figura 29b) e o botão de visualização (Figura 29c). No objeto Exibição o botão de edição é substituído pelo botão de execução (Figura 29d).

O botão de edição (Figura 29b) tem como função disponibilizar janelas secundárias que receberão os parâmetros de cada componente. Vale salientar que cada componente tem sua própria janela, com informações específicas.

O botão de visualização (Figura 29c) serve para que o usuário observe os registros das suas ações em cada componente. Assim, é possível verificar quais parâmetros foram escolhidos pelo usuário.

O botão de execução (Figura 29d) é responsável por executar a aplicação criada por PV. Isso ocorre quando a aplicação visual estiver toda validada e o usuário clicar no botão de execução do objeto Exibição. Mais detalhes sobre o processo de validação serão descritos nas seções seguintes.

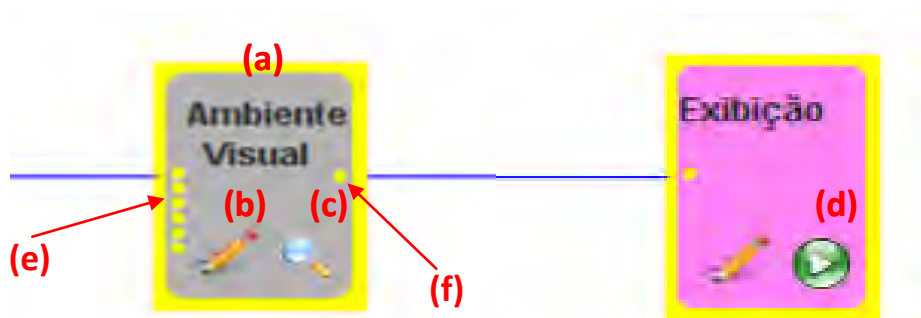


Figura 29. Estrutura de um objeto na área de desenvolvimento do CyberMedVPS.

Na Figura 30 é possível observar um exemplo de uma aplicação visual que utiliza os objetos gráficos definidos anteriormente. Além dos botões, há a presença de ilustrações na forma de quadrados (Figura 29e), para as entradas de dados, e círculos (Figura 29f), para as saídas de dados, dentro dos objetos que demonstram o número possível de entradas (Figura 30a) e saídas (Figura 30b) desses componentes. Existem objetos que só possuem saídas, pois eles são os componentes iniciais da aplicação visual do SPV.

Também existem componentes que só possuem entrada (Figura 30d), pois eles obrigatoriamente devem estar no final da aplicação visual. Os objetos que possuem apenas saídas (Figura 30c) geralmente indicam que estão no início da aplicação visual. E existem ainda objetos que possuem uma entrada e uma saída (Toque, Modelo) e várias entradas e uma saída (Agregador, Ambiente Visual e Ambiente Tátil). Esses dois últimos tipos devem estar localizados no meio do fluxograma. No caso do objeto Modelo (Figura 30e), é possível estar localizado tanto no início quanto no meio da aplicação visual. O que vai determinar esse comportamento é o tipo de aplicação visual criada.

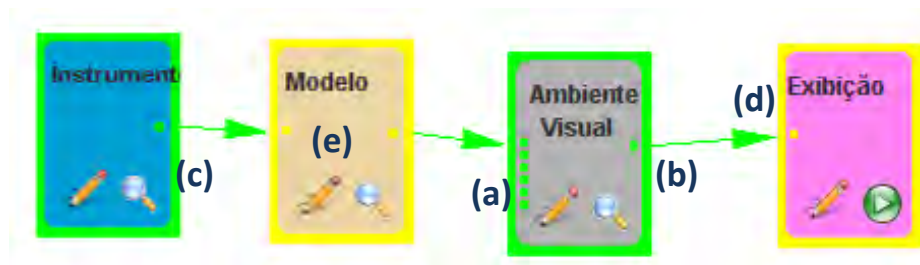


Figura 30. Fluxograma montado com componentes do CyberMedVPS.

Outro fato importante são os aspectos cromáticos das entradas e saídas dos objetos. Quando o objeto é inserido na área de desenvolvimento da aplicação visual, suas bordas, entradas e saídas ficam inicialmente na cor amarela, em sinal de atenção porque nenhum parâmetro foi definido. Após inserir os parâmetros e criar as ligações entre os objetos, suas bordas, entradas e saídas passam a ter a cor verde para enfatizar que os dados estão validados pelo SPV.

Para ligar os componentes entre si é necessário que dois componentes estejam selecionados e obedeça a hierarquia de relacionamento definida na Figura 31. Por exemplo, o objeto Toque (Figura 31A) poderá ter uma ligação com os objetos Modelo, Ambiente Visual, Ambiente Tátil e Instrumento. A ligação do Toque com os outros objetos não é válida e consequentemente não é permitida pelo SPV.

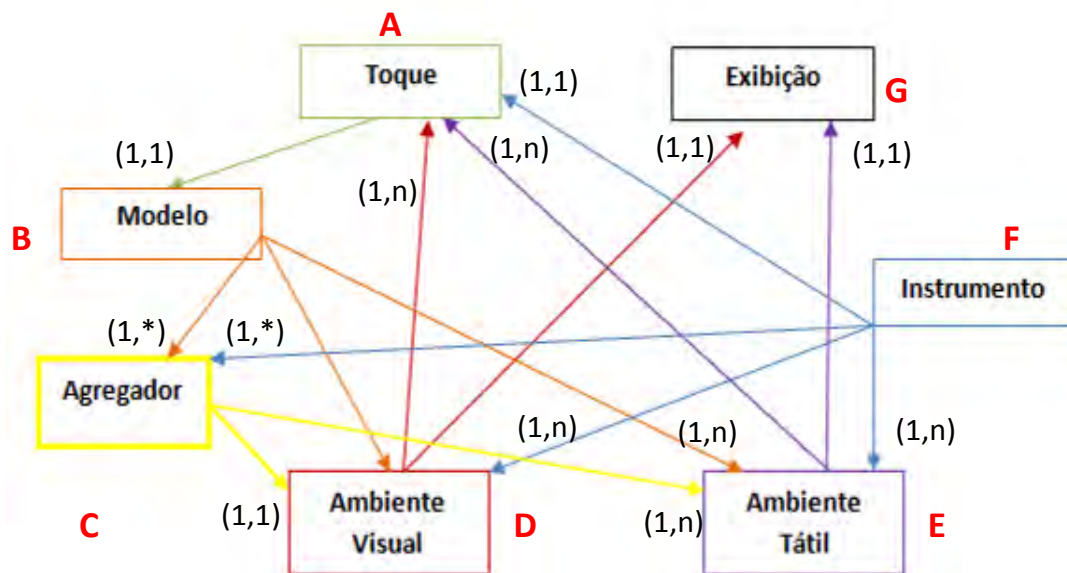


Figura 31. Hierarquia de ligação dos componentes.

A movimentação dos componentes na área central pode ser realizada de duas formas: pelas teclas de direção do teclado ou arrastando os componentes com o mouse pela área central. Assim, os erros provocados por opções limitadas durante interação devem ser minimizados na PV da aplicação.

4.1.4. Área de rodapé da UI do CyberMedVPS

De acordo com Shneiderman e Plaisant (2010), o usuário não é obrigado a lembrar todos os passos para executar uma determinada tarefa. Por isso, é importante que o CyberMedVPS ofereça mensagens de *feedback* para usuários que estejam interagindo com o SPV.

A ideia inicial foi disponibilizar, no rodapé da UI do CyberMedVPS, apenas uma área para transmitir mensagens de sucesso ou erro para o usuário (Figura 26D). Mas, estudos sobre a usabilidade foram realizados e se constatou a necessidade em subdividir tal região em duas seções: de Ações (Figura 32a) e Dicas (Figura 32b). Dessa forma, o usuário pode visualizar mensagens sobre suas ações ou de dicas para a PV de forma direcionada.

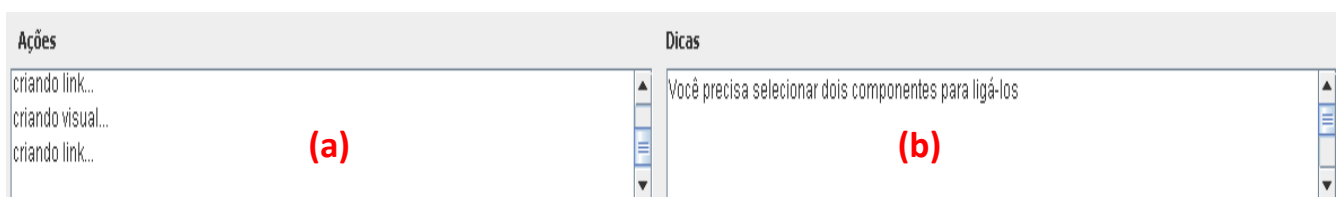


Figura 32. Área de rodapé da UI do CyberMedVPS.

A linguagem do sistema para usuário adota uma linguagem natural de comunicação. Outro fato importante é a presença de ícones com a figura de uma interrogação nos menus internos dos componentes para a consulta do usuário em tempo de execução.

O resultado final do leiaute do CyberMedVPS pode ser visto na Figura 33, que mostra a integração da área de menu e ícones de atalho de Abrir fluxograma, Salvar fluxograma, Desfazer, Refazer, Criar linha e Apagar objeto ou linha (Figura 33a). Além disso, também está presente a área lateral (Figura 33b), a área de desenvolvimento (Figura 33c) e a área de mensagens do SPV (Figura 33d).

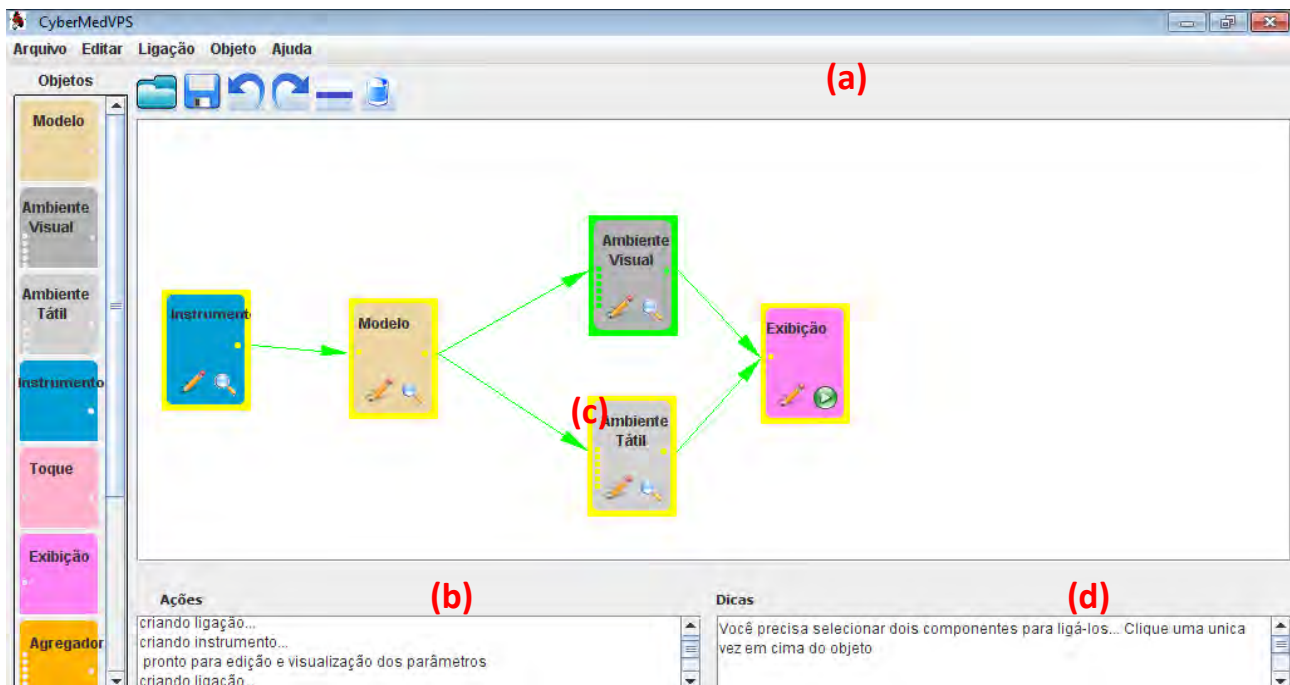


Figura 33. Leiaute completo da UI do CyberMedVPS.

A estrutura prevista pelo protótipo visual do CyberMedVPS não foi alterada, apesar de terem ocorrido modificações em alguns detalhes do leiaute do SPV, especificamente em subáreas. Assim como o protótipo visual, a GUI final do CyberMedVPS manteve a mesma estrutura simples, apesar de ser um sistema complexo, em termos de possibilidades de recursos em RV. Tal simplicidade pode ajudar o usuário no processo de interação devido à ausência de sobrecarga de informação visual para o processo cognitivo no SPV.

4.1.5. Dicionário de dados

Uma funcionalidade prevista durante o processo de implementação da camada de visualização foi a disponibilização de diversos idiomas para o SPV. Além disso, o sistema deveria ser desenvolvido de tal forma que fosse flexível a inserção de novas linguagens. Dessa forma, a comunicação não deve ser um problema do sistema com o usuário.

A solução encontrada foi a criação de um dicionário de dados para se integrado à arquitetura do CyberMedVPS. O dicionário foi definido por uma classe chamada de *DataDictionary* a qual tem os métodos de leitura e inserção de dados no banco. Além disso, tal classe é responsável por definir a estrutura do banco de dados que será composta por apenas uma tabela chamada *datadictionary* que é formada pelos atributos *variável*, *inglês* e *português*. Outro elemento responsável pelo dicionário de dados é o gerenciador de informações representado pela classe *ManagerDB*. Essa classe utiliza a

biblioteca nativa *java.sql.** para realizar as chamadas para o banco. A classe designada para conferir a escolha da linguagem utilizada no CyberMedVPS é a *InitialScreen*, cujo leiaute é mostrado na Figura 34. Tal classe é a tela inicial na execução do SPV.

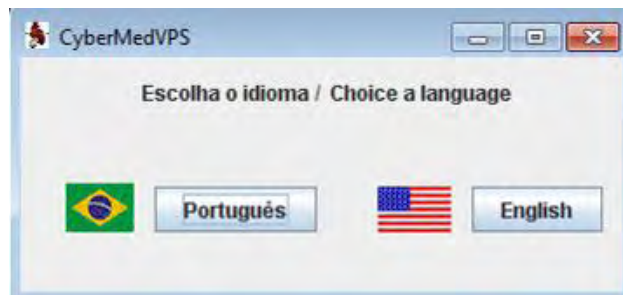


Figura 34. Leiaute da classe *InitialScreen*.

Após o usuário definir qual linguagem deseja utilizar com o CyberMedVPS, a classe *InitialScreen* passa o valor da decisão do usuário por meio de uma variável pública estática chamada de *option* para a classe *ManagerDB* (Figura 35a). Tal classe faz o acesso dos valores delimitados no banco de dados (Figura 35b) e repassa para outras classes do CyberMedVPS, por meio dos métodos *abrirBanco* e *searchLanguage*.

Caso seja a primeira execução do SPV em um determinado SO, é chamada a classe *TesteDB* para criar o banco no sistema e inserir os dados na tabela para o CyberMedVPS funcionar corretamente. A criação do banco de dados ocorre quando o *TesteDB* chama os métodos *criarBanco* e *criarTabelas* do *ManagerDB* (Figura 35c). Em seguida, as informações são armazenadas na tabela e no formato correto, devido à chamada do construtor da classe *DataDictionary* (Figura 35d). Por fim, todas essas informações são armazenadas no banco de dados.

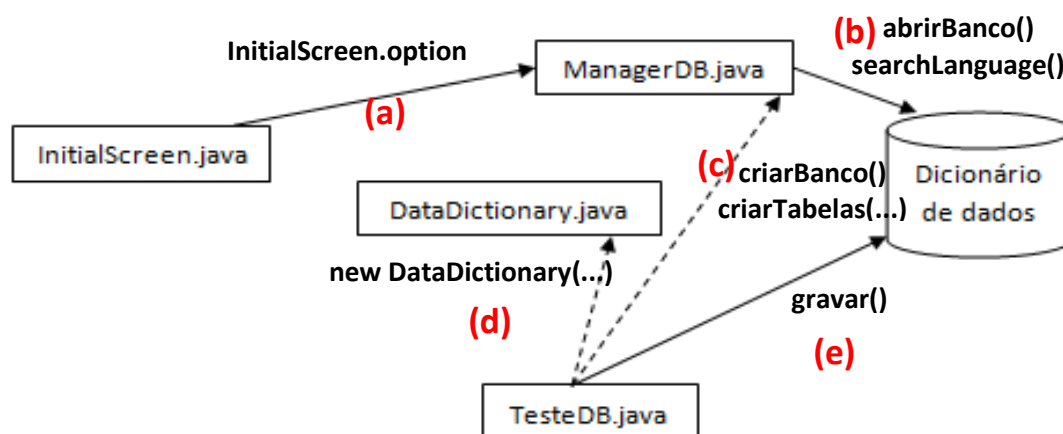


Figura 35. Dicionário de dados no CyberMedVPS.

Para adicionar uma nova linguagem no CyberMedVPS, é preciso criar uma nova coluna na tabela *datadictionary*. Além disso, é necessário adicionar uma nova opção na classe *InitialScreen* para conferir se essa nova linguagem foi a escolhida.

Vale salientar que o SGBD escolhido para gerenciar as informações do dicionário de dados do CyberMedVPS foi o MySQL, devido a fatores como: ser multiplataforma, ter uma biblioteca de acesso nativa na linguagem Java, utilizar a linguagem SQL (*Structured Query Language*) como interface e possuir uma baixa exigência de processamento computacional em comparação aos outros SGBDs (NIXON, 2009).

Definida as funcionalidades da camada de visualização é possível observar a organização final, de acordo com a Figura 36, que mostra como esse padrão subdivide as classes na camada de visualização.

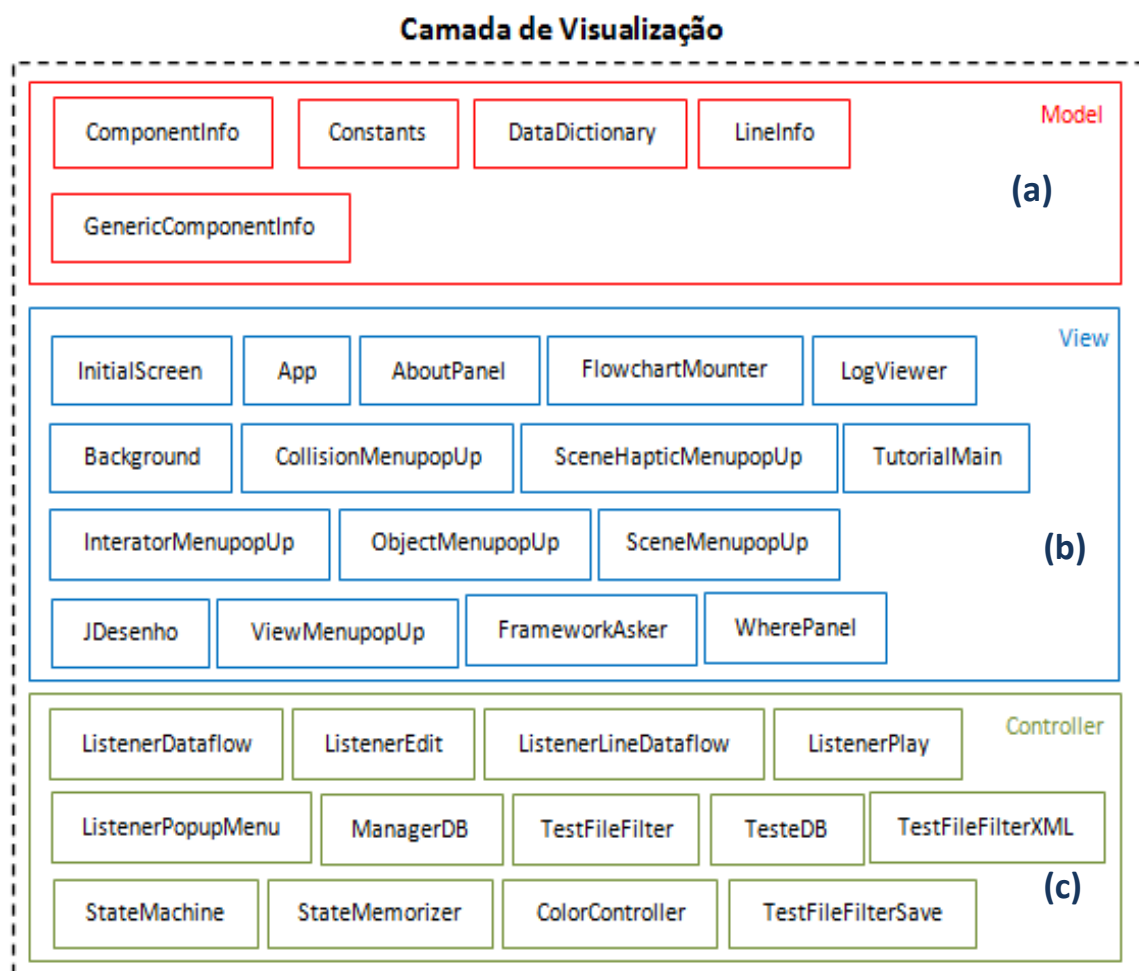


Figura 36. Classes que compõem a Camada de Visualização.

O pacote *Model* (Figura 36a) é formado por classes que definem estruturas genéricas, como a classe *GenericComponentInfo*, por exemplo, que serve para definir as características dos componentes da aplicação visual. Qualquer componente que faz parte

de um fluxograma tem uma instância na classe *GenericComponentInfo*. Essa classe sofre uma especialização (Figura 37) e pode se subdividir em dois tipos distintos: a) se for um componente da aplicação, ela tem comportamento previsto pela classe *ComponentInfo* e b) se for uma linha na aplicação, ela tem características da classe *LineInfo*.

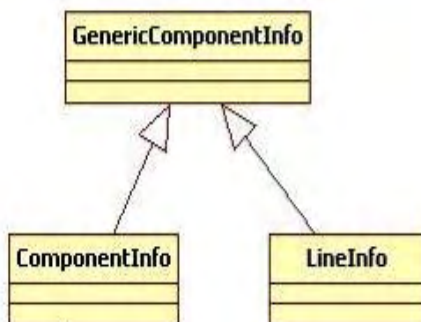


Figura 37. Especialização entre classes do pacote *Model*.

O pacote *View* (Figura 36b) diz respeito às classes responsáveis por toda a estrutura visual do CyberMedVPS. O desenvolvimento da interface visual do CyberMedVPS utilizou a API *Swing* fornecida pela linguagem Java.

A classe principal do pacote *View* é a *App*, pois é quem delimita todas as áreas da GUI na camada de visualização que foram descritas anteriormente. Além disso, essa classe herda métodos e características da superclasse *JFrame*¹¹ da API Java SWING. Os componentes utilizados na aplicação visual são definidos como classes que herdam métodos e características da superclasse *JPanel*¹².

O pacote *Controller* (Figura 36c) diz respeito às classes responsáveis pelo controle dos eventos e dos estados da GUI e da aplicação visual gerada no CyberMedVPS. Nesse pacote há uma chamada de *StateMachine* que é capaz de validar a estrutura da aplicação criada por PV. Além disso, tal classe é responsável por validar as ligações entre os componentes.

Ainda no pacote *Controller*, existe a classe chamada de *StateMemorizer*. Ela é responsável por armazenar os estados de implementação da aplicação visual em desenvolvimento. Dessa forma, o SPV é capaz de disponibilizar as funcionalidades de desfazer (com as teclas *Ctrl* e *Z* ou no botão de desfazer da área de atalho) e refazer (com as teclas *Ctrl* e *Y* ou no botão de refazer da área de atalho) para o usuário.

¹¹ É uma janela da API SWING com um título e uma borda. Disponível em: <http://docs.oracle.com/javase/6/docs/api/javax/swing/JFrame.html>.

¹² É um contêiner genérico na API SWING. Disponível em: <http://download.oracle.com/javase/6/docs/api/javax/swing/JPanel.html>.

Para reagir às ações do usuário na interface visual, foram definidos *listeners* que “escutam” os eventos do usuário durante a execução do SPV, tanto pelo mouse quanto pelo teclado. De acordo com a Figura 36c, o CyberMedVPS possui cinco classes com essa função:

- *ListenerLineDataflow* : responsável por reagir aos eventos da linha na área central do SPV;
- *ListenerPopupMenu* : responsável por reagir aos eventos do menus internos dos componentes do fluxograma. Esses menus são visualizados após o usuário pressionar o botão de Editar de qualquer componente da aplicação visual;
- *ListernerDataflow* : responsável por reagir aos eventos do componente na área central do SPV;
- *ListenerPlay* : responsável por mostrar ou a tela de registros ou executar a aplicação visual quando detectar que o usuário clicou o botão de Execução;
- *ListenerEdit* : inicializa *JDIALOGs* que podem variar de acordo com o componente que iniciou o evento.

Vale ressaltar que após a finalização do desenvolvimento da camada de visualização foi realizada uma avaliação da interface do protótipo visual do CyberMedVPS com o público-alvo desse sistema. Mais detalhes serão relatados no capítulo 5.

4.2. Definição da camada de comunicação

Como foi explicado, na seção anterior, a camada de visualização serve para o usuário montar a aplicação visual baseada em RV apenas por PV. A proposta do CyberMedVPS é que o resultado final do SPV gere algo computacionalmente válido para *frameworks* com recursos baseados em RV. Com base nessa necessidade, foi projetada a camada de comunicação que tem como objetivo solucionar o problema do diálogo entre os *frameworks* em RV e a aplicação visual gerada no SPV. Portanto, a camada de comunicação será a ponte de comunicação entre a camada de visualização e os interpretadores dos *frameworks*.

De acordo com a Figura 38, a camada de visualização é subdividida em duas partes: a camada de validação e a camada de montagem. A camada de validação (Figura 38a) refere-se às classes responsáveis por garantir que as informações geradas na codificação da aplicação visual para aplicação computacional estejam válidas para

execução. A camada de montagem (Figura 38b) é responsável por interpretar a aplicação computacional e disponibilizar os parâmetros definidos pelos usuários para os interpretadores dos *frameworks* em RV.

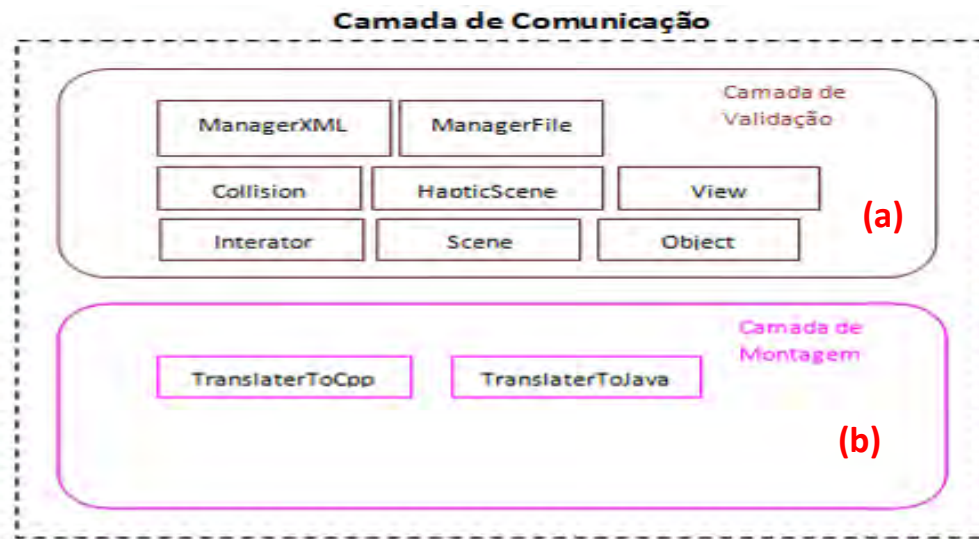


Figura 38. Classes que compõem a Camada de Comunicação do CyberMedVPS.

O processo de codificação da aplicação visual para aplicação computacional ocorre a partir das informações registradas nas classes *Object*, *Scene*, *HapticScene*, *Interator*, *Collision* e *View* que correspondem às características de cada objeto da aplicação visual. Todas essas classes estão associadas à classe *ManagerXML* que solicita os métodos de leitura de cada um dos componentes associados (Figura 39). Esses métodos de leitura (*readObj*, *readSce*, *readHap*, *readInt*, *readCol* e *readVie*) são responsáveis por avaliar a legitimidade do valor atribuído pelo usuário para determinado objeto.

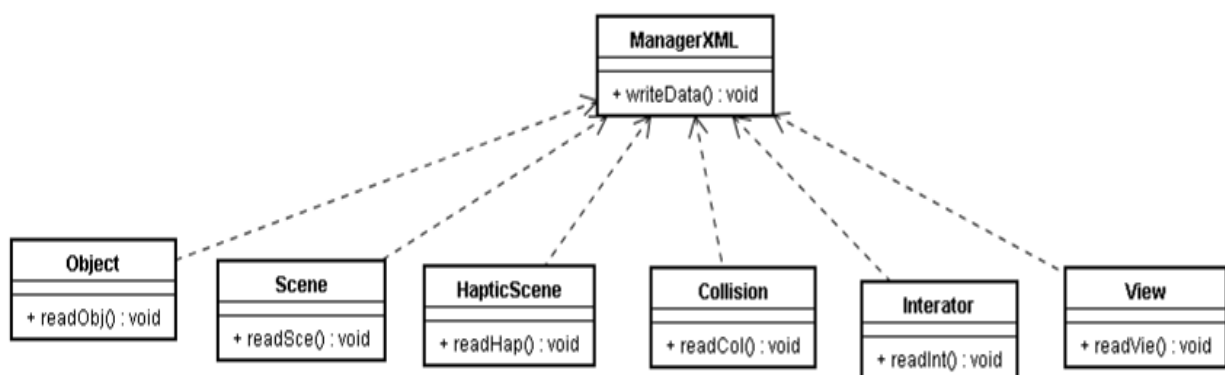


Figura 39. Diagrama de classes da camada de validação.

Se o usuário definir um valor que não seja condizente com o atributo do componente relacionado, a classe *ManagerXML*, por meio do método *writeData* (Figura 39), é responsável por atribuir um valor padrão para tal característica do componente em

análise. Vale salientar que esse processo acontece por meio de uma avaliação individual de cada componente da aplicação visual. Após a validação, o método *writeData* fica responsável por codificar todas as informações em uma linguagem computacional.

A codificação tem como resultado uma linguagem computacionalmente genérica e compatível com interpretadores para *frameworks* baseados em RV. Baseado nisso, a linguagem escolhida para aplicação genérica foi o XML, devido a fatores como: a) ser uma linguagem de marcação com licença de código aberto, b) possuir integração de dados de diferentes fontes, c) ter escalabilidade, d) ser manipulada por diversas APIs na linguagem Java e e) separar o conteúdo da formatação (SALMINEN e TOMPA, 2011) (SCHNEIDER e GIBSON, 2008).

Quadro 4. Gramática do XML gerada pelo CyberMedVPS.

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<component>
  <model>
    <modelName></modelName>
    <positionX></positionX>
    <positionY></positionY>
    <positionZ></positionZ>
    <rotationX></rotationX>
    <rotationY></rotationY>
    <rotationZ></rotationZ>
    <translationX></translationX>
    <translationY></translationY>
    <translationZ></translationZ>
    <visibility></visibility>
  </model>
  <visual_scene>
    <number></number>
    <pXi></pXi>
    <pYi></pYi>
    <pXf></pXf>
    <pYf></pYf>
    <viewType></viewType>
  </visual_scene>
  <instrument>
    <typeInt></typeInt>
    <name></name>
  </instrument>
  <collision>
    <typeCol></typeCol>
    <information></information>
    <reaction></reaction>
    <viscosity></viscosity>
    <resistance></resistance>
  </collision>
  <haptic_scene>
    <type></type>
  </haptic_scene>
  <exhibition>
    <name></name>
  </exhibition>
</component>

```

(a)

(b)

(c)

(d)

(e)

(f)

A estrutura adotada no XML do CyberMedVPS é composta por sete *tags* diferentes, como pode ser na gramática do Quadro 4. Essas *tags* obedecem à especificação dos componentes definida na aplicação visual. O principal elemento é a *tag component* que indica as configurações de uma aplicação visual. Dentro dessa *tag*, existem seis tipos de informações no XML do CyberMedVPS.

A primeira *tag* interna refere-se ao *model* e tem como função informar as características do objeto visual Modelo. Além disso, essa *tag* pode aparecer mais de uma vez na aplicação visual, por isso precisa ser identificada por número de identificação (ID). As *subtags* relacionadas à *tag model* (Quadro 4a) refere-se a informações sobre opções de rotação e translação do modelo dentro do ambiente virtual, no eixo x, y e z, e a localização do arquivo que representa o modelo.

A segunda *tag* refere-se ao *visual_scene*, que corresponde ao objeto Ambiente Visual da aplicação visual. As *subtags* relacionadas à *tag model* (Quadro 4b) refere-se a informações sobre a posição do modelo dentro do ambiente virtual, no eixo x, y e z, e o tipo de visualização do modelo.

Em seguida, é descrita a *tag instrument* que possui informações do Instrumento como, por exemplo, o tipo de interação e em qual objeto o Instrumento está vinculado pelo ID. As *subtags* relacionadas ao *instrument* (Quadro 4c) referem-se a informações sobre o tipo de dispositivo de interação e o arquivo do modelo virtual que representa o instrumento.

A quarta *tag*, chamada de *collision* refere-se ao objeto Toque e também é identificado por um número ID para identificar qual instrumento exerce ação sobre qual modelo. Suas *subtags* (Quadro 4d) referem-se ao tipo de deformação e ao tipo de colisão que devem ser oferecidas.

A seguir, surge a *tag haptic_scene* que se refere ao objeto Ambiente Tátil. Suas *subtags* (Quadro 4e) referem-se a um tipo de dispositivo de interação não-convencional. E, por fim, existe a *tag* chamada de *exhibition* que é responsável pelas informações finais da aplicação visual e representa o objeto Exibição. A *subtag* relacionada ao *exhibition* (Quadro 4f) refere-se ao nome da janela da aplicação em RV.

Vale ressaltar que o objeto Agregador não possui uma *tag* que o descreva diretamente na linguagem genérica (XML) porque se trata de um componente lógico, ou seja, ele é responsável por atribuir as mesmas características de toque, de deformação, por exemplo, a diversos *frameworks*. Assim, tal objeto não apresenta parâmetros distintos, mas replica características para quem estiver vinculado ao mesmo.

O conjunto de bibliotecas nativas do pacote *javax.xml.** foi responsável pela montagem da estrutura do XML na classe *ManagerXML* do CyberMedVPS. Para gravação foi necessário apenas instanciar uma classe nativa chamada de *DocumentBuilderFactory* que é responsável por inserir os dados no arquivo de extensão “.xml”. No processo de leitura foi criado o método *readData* que tem como atributo o nome da *tag* e a característica pesquisada. Após concretizar a criação do arquivo XML com as configurações na aplicação visual, é necessário habilitar a camada de montagem classes que entendam a linguagem genérica e consigam codificar essas informações para um interpretador para *framework* em RV.

Portanto, é possível inferir que a camada de comunicação passa por dois processos distintos de transformação de linguagem: o primeiro ocorre na camada de validação e serve para traduzir a aplicação visual em aplicação genérica (XML) (Figura 40a). E o segundo processo ocorre por meio da codificação de uma aplicação genérica em outra linguagem computacional previamente definida (Figura 40b), ou código fonte.

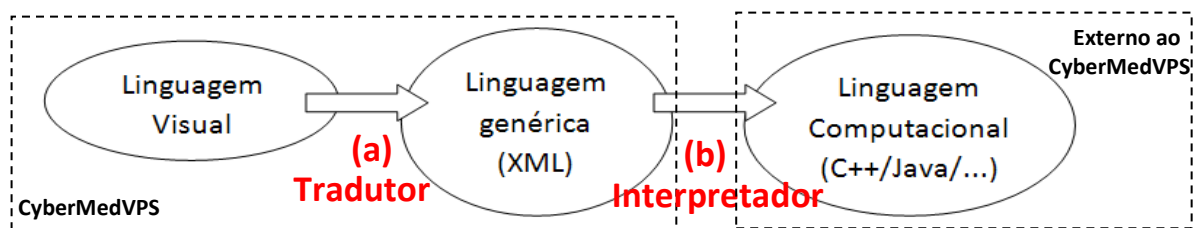


Figura 40. Dinâmica das linguagens no CyberMedVPS.

Na camada de montagem foram criadas duas classes de leitura. A primeira é uma classe tradutora para *frameworks* baseados em RV que são compatíveis com a linguagem C e C++ chamada de *TranslatorCpp*. A outra é uma classe tradutora, chamada de *TranslatorJava*, compatível com *frameworks* implementados na linguagem Java.

O processo de tradução na camada de montagem se inicia a partir do momento em que uma dessas classes de codificação chama o método de leitura *readData* da classe *ManagerXML*. Cada leitura armazena os parâmetros obtidos em variáveis estáticas públicas da classe do tradutor.

Após o processo de tradução no CyberMedVPS, as classes de interpretação, externas ao SPV, definem as bibliotecas que são utilizadas no código final das aplicações compatíveis ao *framework* em RV. Dessa forma, é preciso solicitar variáveis públicas adequadas, que estão dentro dos tradutores, para executar a montagem da aplicação final em linguagem de programação quando classes interpretadoras e externas ao CyberMedVPS precisarem se comunicar com os tradutores da camada de montagem.

4.3. Adição de suporte a *frameworks*

A arquitetura do CyberMedVPS baseada no padrão MVC também garante que o processo de integração de um novo *framework* seja simples. É necessário que exista um conhecimento profundo sobre o *framework* em RV e que as etapas a seguir sejam executadas:

- a) criação de uma classe validadora, externo ao CyberMedVPS e dentro de um pacote chamado *Framework*, compatível com o novo *framework*;
- b) edição da classe *FrameworkAsker*, dentro do pacote *View*, para inserir um novo *framework* como opção.

A classe tradutora, como já foi explicado em seções anteriores, serve para realizar a leitura da linguagem genérica gerada pelo CyberMedVPS. E, a classe *FrameworkAsker* é responsável por ser a parte visual do CyberMedVPS que questiona qual *framework* o usuário deseja utilizar. Por isso, é necessário que os métodos *OpFrameActionPerformed()*, *okbuttonActionPerformed()* e *run()* sejam editados e modificados. É importante que para essa etapa de integração com um novo *framework* no CyberMedVPS exista a colaboração do profissional da área de computação para que erros não esperados sejam atribuídos de forma equivocada no SPV.

4.4. Estrutura final das classes do CyberMedVPS

Após definir e desenvolver as camadas do CyberMedVPS, é possível observar o diagrama das classes que compõem o SPV, na Figura 41. As classes que geram as estruturas visuais que são responsáveis por mostrar os parâmetros de configuração da aplicação visual são denominados de *MenuPopUp* e cada objeto da aplicação visual tem uma estrutura de *MenuPopUp* correspondente (Figura 41a). A comunicação entre classes de diferentes camadas do CyberMedVPS é perceptível na camada de Visualização (Figura 41a) e na camada de Comunicação (Figura 41b). Além disso, é possível ver a presença dos tradutores (Figura 41c) e interpretadores de *frameworks* em RV (Figura 41d).

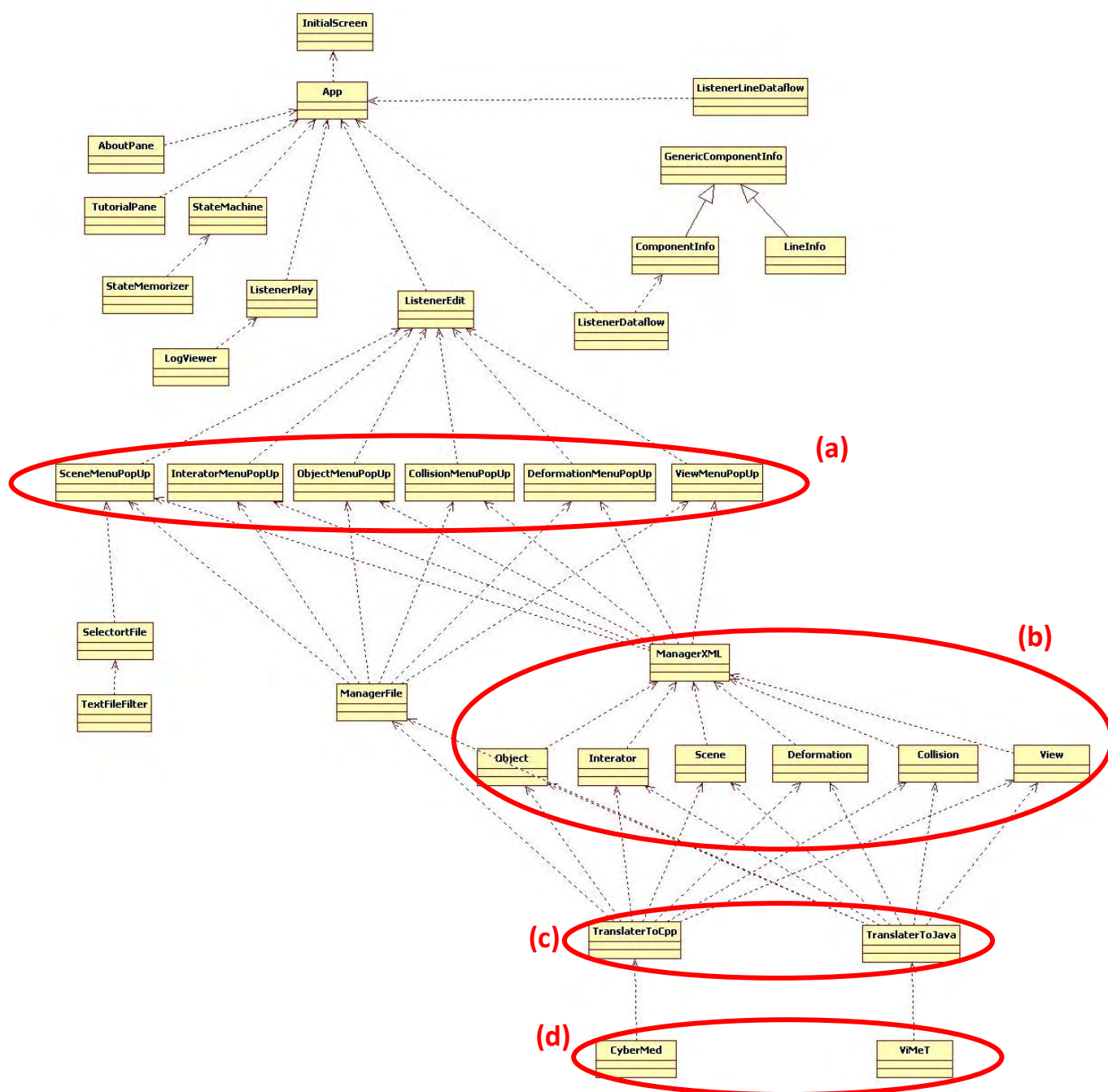


Figura 41. Diagrama de classes do CyberMedVPS.

5 Resultados

Após o detalhamento das etapas de desenvolvimento do CyberMedVPS, serão apresentados os resultados obtidos com a criação desse SPV. Além disso, serão mostrados os trabalhos científicos sobre estudos relacionados ao CyberMedVPS, publicados em eventos científicos.

O processo de desenvolvimento da camada visual do SPV não adotou exclusivamente o processo iterativo. Vale salientar que a avaliação de interface é um passo primordial no processo de *design* em sistemas visuais (ALVES *et al.*, 2011). Com base nessa abordagem, outros resultados foram obtidos com o CyberMedVPS, como: a) avaliação da GUI do SPV antes do desenvolvimento da camada de comunicação, b) a validação das aplicações geradas pelo SPV e c) o teste de defeitos com o CyberMedVPS. Antes de todas essas avaliações será apresentado um exemplo de utilização do CyberMedVPS até a execução da aplicação com recursos em RV.

Por fim, foi realizado o registro de *software* do CyberMedVPS e a criação do instalador multiplataforma para que o SPV possa ser utilizado livremente pela comunidade acadêmica, oferecendo os devidos méritos à equipe.

5.1. Publicações

O trabalho realizado possibilitou a parceria no projeto do INCT-MACC, além de três publicações em eventos acadêmicos. O primeiro foi apresentado em outubro de 2010, no Congresso Brasileiro de Informática em Saúde (CBIS), com o trabalho denominado “Programação Visual para um *Framework* de Desenvolvimento de Aplicações Médicas Baseadas em Realidade Virtual”. O segundo foi aceito no *Medical Meets Virtual Reality* (MMVR 18) com o trabalho “CyberMedVPS: *Visual Programming for Development of Simulators*”. E, por fim, o terceiro foi aceito no XIII Simpósio de Realidade Virtual e Aumentada (SVR 2011) com o trabalho “Análise das ferramentas computacionais com Realidade Virtual voltadas para os profissionais da área médica”, apresentado em maio de 2011. Por fim, foi submetido e está em fase de aguardo, um artigo completo intitulado “CyberMedVPS: *A Visual Programming System based on Virtual Reality for Training and Learning to Users of Medical Area*”, para revista *International Journal of Medical Informatics* que relatará sobre todo o processo de avaliações que foram realizadas, bem como a descrição do processo de elaboração do sistema.

5.2. Exemplo de caso

Para gerar uma aplicação com recursos em RV, é necessário apenas que o usuário crie a aplicação visual corretamente. Para exemplificar como é o funcionamento do CyberMedVPS será criada uma aplicação de visualização de um objeto virtual usando estereoscopia. Uma aplicação com essa funcionalidade só precisa de três componentes para a aplicação visual: a) Modelo: que representa o objeto virtual que será exibido, b) Ambiente Virtual: para visualizar o modelo no cenário e c) Exibição: para mostrar o resultado da aplicação visual gerada.

Primeiramente, será feita a escolha dos objetos relatados por meio da seleção do objeto no menu lateral (Figura 42a) ou selecionando a opção Objeto -> Nome do Objeto, na área das opções de menu localizada na parte superior do SPV (Figura 42b).

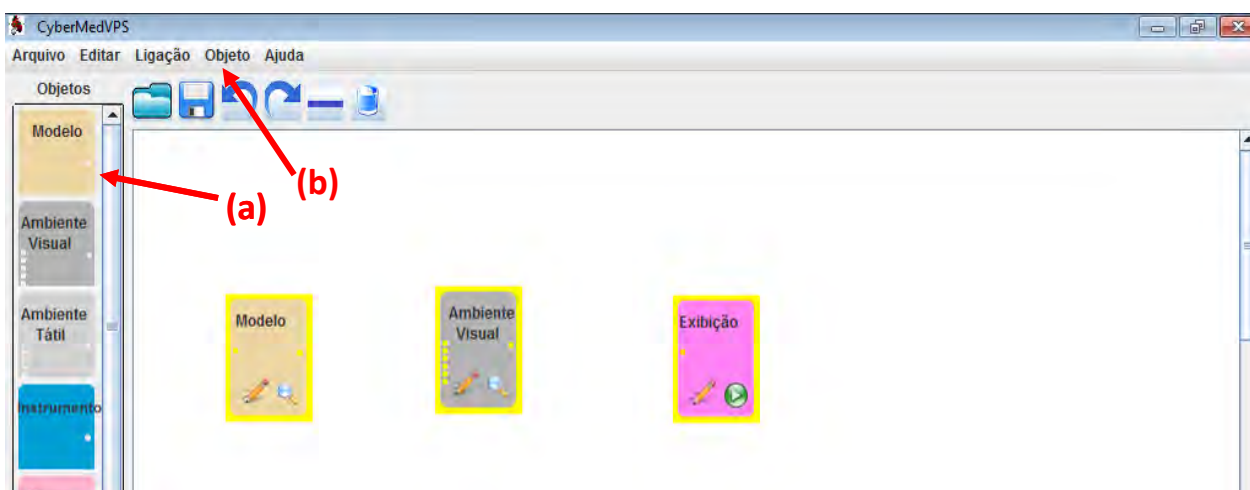


Figura 42. Selecionando objetos para aplicação visual no CyberMedVPS.

Em seguida, serão criadas as ligações entre os objetos da aplicação visual. Basta selecionar dois objetos e clicar na opção criar ligação no menu superior (Figura 43a) ou clicar no atalho abaixo da área de menu (Figura 43b) ou apertar a opção Ctrl+L.



Figura 43. Criando ligação no CyberMedVPS.

O terceiro passo é a configuração das opções de cada componente da aplicação visual. Se o usuário tentasse executar nesse momento da aplicação, o SPV notificaria que não há parâmetros configurados na aplicação visual.

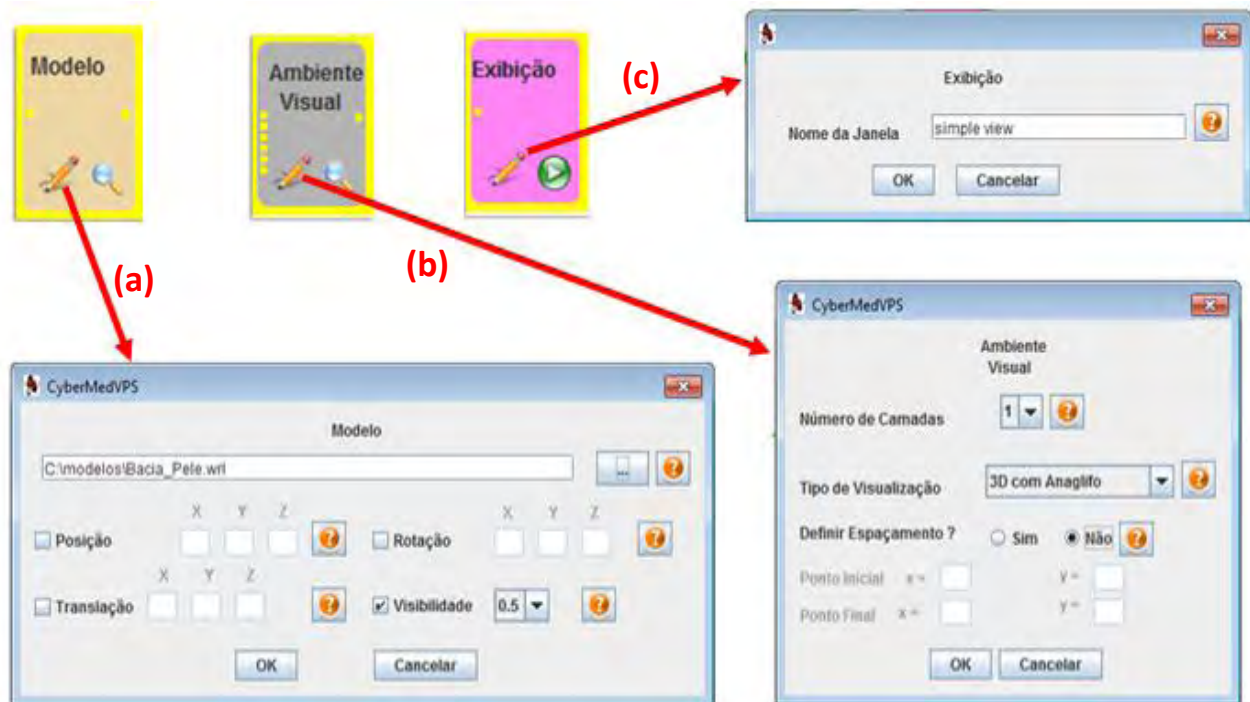


Figura 44. Opções de configuração de parâmetros de objetos do CyberMedVPS.

Na opção do modelo são especificadas opções de referentes às propriedades de rotação, translação e de visibilidade do objeto virtual (Figura 44a). Nas opções referentes ao Ambiente Visual, são configuradas opções referentes ao número de camadas, ao tipo de visualização e posicionamento do modelo no eixo x, y e z (Figura 44b). E na opção da Exibição é definida apenas o nome da janela da aplicação final (Figura 44c).

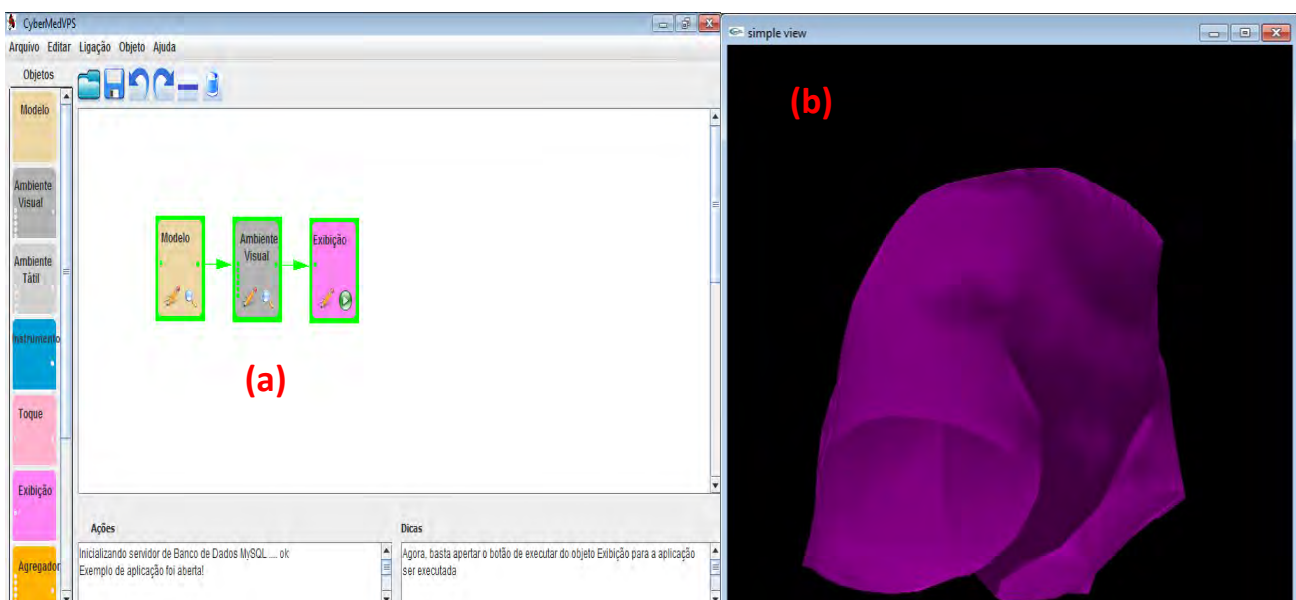


Figura 45. Aplicação final (a) e aplicação final em RV (b).

A linha amarela em volta da caixa torna-se verde para notificar visualmente o usuário quando os parâmetros do objeto são configurados corretamente. Quando toda a aplicação for formada de caixas com bordas verdes, então toda a aplicação visual está validada pelo SPV (Figura 45a). Então é só pressionar o botão de executar no objeto Exibição que a aplicação em RV será executada e mostrada para o usuário (Figura 45b).

5.3. Avaliação da GUI do CyberMedVPS com o público-alvo

A avaliação da GUI do CyberMedVPS teve como intuito mensurar se a camada de visualização foi projetada de forma eficiente para os profissionais da saúde. De acordo com um estudo realizado por Martikainen *et al.*(2012), há a comprovação de que existe um comportamento crítico por parte dos profissionais de saúde , principalmente médicos cirurgiões, em adotar e utilizar sistemas computacionais a sua rotina profissional. Mas, esse estudo foi realizado na Finlândia, por isso houve a necessidade de avaliar se o CyberMedVPS poderia ser aceito pela comunidade de saúde no Brasil.

Assim, após o desenvolvimento da camada de visualização realizou-se uma avaliação sobre a GUI do CyberMedVPS com o seu público-alvo, antes de iniciar o desenvolvimento da camada de comunicação. Foi disponibilizada a execução de uma aplicação em RV já compilada para ser disponibilizada para o usuário caso o mesmo gerasse uma aplicação visual com sucesso. A avaliação foi necessária para analisar se os recursos de PV e de *design* de interface que foram aplicados na camada de visualização estavam sendo compreendidos pelos profissionais da área de saúde.

Na literatura existem diferentes características de técnicas e métodos de avaliação de sistemas, como Teste de Usabilidade (BARNUM, 2011), Avaliação Heurística (TURNER-BOWKER *et al.*, 2011), Critérios e Recomendações Ergonômicas (HUa *et al.*, 2011), entre outros. Aliadas a essas características estão variáveis de tempo, recursos tecnológicos, financeiros e humanos disponíveis. Após o estudo dessas técnicas percebeu-se a necessidade de combinação e adaptação de métodos, envolvendo diferentes áreas, para a avaliação apropriada do CyberMedVPS.

5.3.1. Protocolo de avaliação

Foi elaborado um protocolo de avaliação com o objetivo de coletar as impressões do público-alvo sobre o CyberMedVPS. Esses dados serviram para avaliar o grau de aceitação e eficiência das técnicas visuais aplicadas à GUI do SPV. A equipe responsável

pela avaliação estipulou que o método mais adequado seria a aplicação de uma entrevista anônima, composta por vinte e duas questões. Essas perguntas estão disponíveis no Apêndice B e foram previamente aprovadas pelo Conselho de Ética do HUWL.

Tal entrevista foi dividida em duas partes. A primeira parte referiu-se às informações do usuário como, por exemplo, faixa etária, gênero, área de atuação (professor, profissional liberal, estudante ou professor/profissional liberal), se existe acesso ao computador em casa e no trabalho (para o lazer e para o trabalho), qual o tempo de acesso (para lazer e trabalho), qual sistema operacional utilizado pelo usuário e se o entrevistado já havia programado alguma vez.

A segunda parte da entrevista foi composta por questões referentes às impressões do entrevistado sobre o CyberMedVPS como, por exemplo, se ele já havia utilizado algum sistema como o SPV em avaliação antes, se o usuário se sentiu confortável com o CyberMedVPS, se ele estaria disposto a utilizar esse tipo de sistema no trabalho, se ele conseguiu entender as funções dos componentes, das áreas de *feedback*, os ícones dos componentes e se precisou utilizar algum recurso de ajuda do sistema. Além disso, foi questionado ao usuário se ele conseguiu finalizar aplicação, se conseguiu relacionar o nome dos componentes com a sua respectiva função e se o usuário precisou utilizar as funções de desfazer e refazer.

Outro fato que merece ressalva refere-se à preocupação em tornar a linguagem das questões fáceis de serem compreendidas pelos profissionais de saúde. Para isso, foram criadas perguntas com as seguintes características: a) utilização mínima de termos computacionais e técnicos, b) não eram extensas (com uma explicação inserida na pergunta) e c) respostas que tentavam não gerar dúvida, ou seja, tentou-se abranger a maioria das respostas sem causar ambiguidade. Além disso, também havia a presença de perguntas com respostas abertas para questionamentos de caráter individual.

Após a definição do conteúdo da entrevista, foi decidido que a avaliação da GUI do CyberMedVPS deveria seguir o seguinte roteiro de avaliação:

- Explicação breve sobre o intuito da pesquisa e do CyberMedVPS para o entrevistado, individualmente;
- Ilustração do que era uma aplicação em RV, se o usuário nunca tivesse tido contato;

- Inicialização do CyberMedVPS sem qualquer fluxograma previamente configurado;
- Interação do usuário com o CyberMedVPS (Figura 46a);
- Aplicação da entrevista com o usuário de saúde, ao encerrar a interação.

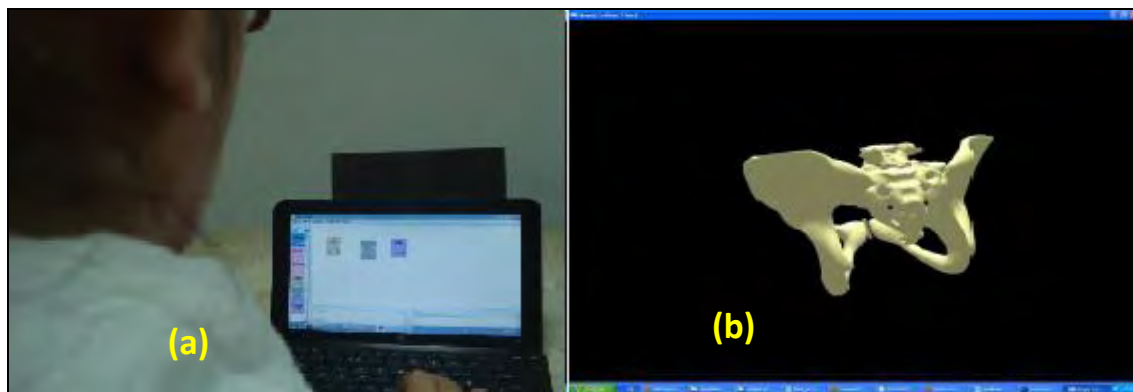


Figura 46. Usuário interagindo com o CyberMedVPS (a) e aplicação final em execução (b).

A explicação breve para o usuário teve como objetivo permitir que o mesmo conseguisse identificar em qual momento o SPV gerou um resultado válido. Na etapa referente à utilização do SPV pelo usuário, houve a solicitação por parte do entrevistador para que o usuário de saúde tentasse criar uma aplicação baseada em RV por meio do CyberMedVPS. A aplicação de RV (Figura 46b) já estava compilada e foi programada para ser apresentada para o usuário quando finalizasse a aplicação visual adequadamente.

5.3.2. Aplicação da avaliação

De acordo com o protocolo de avaliação do CyberMedVPS, presente na seção 5.3.1., foi definido que o tempo médio no processo de avaliação, por cada entrevistado, não deveria exceder 10 minutos. A aplicação era considerada finalizada no momento em que o usuário conseguisse executar a aplicação em RV por meio do SPV. Além disso, não houve interferência dos entrevistadores em nenhuma etapa do processo de interação do entrevistado com o SPV.

A avaliação da GUI com profissionais de saúde ocorreu no mês de julho de 2011 nos Departamentos de Medicina, de Enfermagem, de Odontologia, de Fisioterapia e no HULW, todos localizados na UFPB.

A entrevista foi aplicada a 21 (vinte e um) usuários entre 21 (vinte e um) e 60 (sessenta) anos. Esses entrevistados eram profissionais da saúde nas seguintes especialidades: enfermagem, medicina, fisioterapia e odontologia. Outro fato importante é

que foi possível coletar as impressões gerais dos profissionais de saúde sobre o CyberMedVPS, embora a pesquisa apresentasse uma amostra estatisticamente não-significativa em relação à amostra de profissionais de saúde existentes na UFPB no ano de 2011.

5.3.3. Resultados da avaliação

O estudo dos resultados da entrevista serviu para elucidar aspectos positivos e negativos sobre a eficiência no processo de programação do CyberMedVPS para os usuários da área de saúde.

Os resultados dessa análise revelaram a diferença existente entre o processo de avaliação do CyberMedVPS executado por desenvolvedores/usuários da área de computação e pelo público-alvo. Alguns pontos de interação, dentro do CyberMedVPS, que pareciam ser coerentes para os desenvolvedores poderiam não parecer simples para usuários de outras áreas, especialmente da saúde.

Com relação aos aspectos quantitativos da avaliação, é possível mencionar que a média de tempo de interação com o protótipo pelos usuários não excedeu 5 (cinco) minutos. Essa média de tempo foi melhor que o esperado pela equipe de IHC. Além disso, a maioria dos usuários (98%) afirmaram que gostariam de usar o CyberMedVPS como uma ferramenta para o desenvolvimento para auxiliar em suas atividades profissionais, pois 82% finalizaram a aplicação visual montada e visualizaram a aplicação em RV que já estava compilada.

Embora 76% dos usuários de saúde tenham notado a presença das áreas de rodapé (Figura 47), destinadas às mensagens de comunicação com o público-alvo, menos da metade utilizaram efetivamente as áreas de Ações (38%) e Dicas (9,5%). O mesmo acontece com as áreas destinadas à ajuda, pois apenas dois usuários procuraram tal seção no menu de opções.

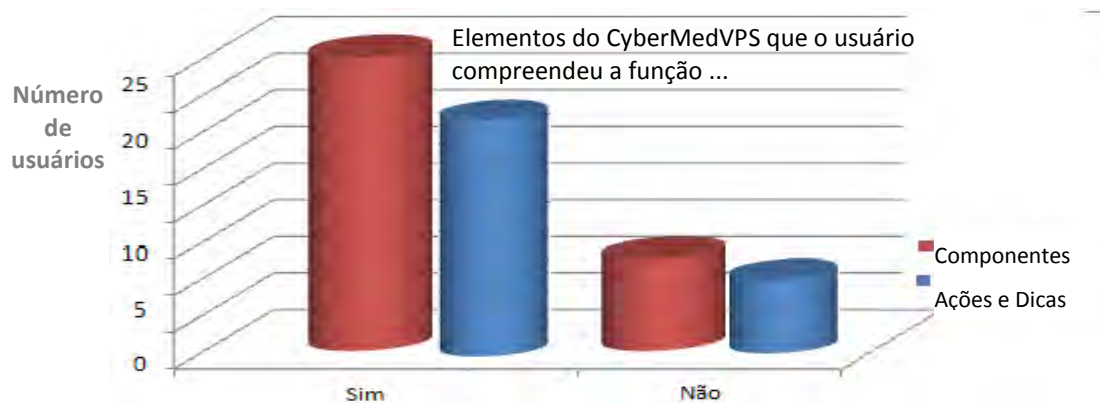


Figura 47. Elementos do CyberMedVPS que foram compreendidos pelo usuário.

A maioria dos profissionais de saúde afirmou que usava o computador predominantemente para o trabalho durante a semana, como mostra a Figura 48. Baseado nisso, foi observado que os entrevistados que declararam passar mais de 15 horas no computador, seja para o trabalho (66,67%) quanto para o lazer (9%), apresentaram uma melhor interação, no CyberMedVPS. Essa afirmação sugere que o CyberMedVPS pode ajudar profissionais da saúde que não estejam tão habituados a utilizar o computador a interagir com novos tipos de aplicações na área e baseadas em RV.



Figura 48. Quantidade de horas gastas, semanalmente, no computador pelos entrevistados.

Quando questionados sobre o uso de diferentes cores entre os componentes, a maioria (66,67%) respondeu que era apenas para distingui-los no SPV. Alguns (9,5%) responderam que a escolha da cor é vinculada à sua função e outros entrevistados (23,08%) consideraram as duas respostas anteriores como corretas.

Foi observada a utilização das funções de desfazer e refazer pelos usuários, assim como a função de seleção de componentes durante a interação do CyberMedVPS (20%). Apenas dois usuários perguntaram aos pesquisadores algo para solucionar dificuldades. Em ambos os casos foi sugerido para eles buscarem as áreas de ajuda da

GUI do SPV. Apenas dois usuários apresentaram muitas dificuldades ao interagir com o CyberMedVPS. Os dois alegaram que precisariam de um pouco mais de tempo para entender e interagir com o SPV.

Durante a interação, foi percebida que outras funcionalidades poderiam ser inseridas no SPV, como: atalhos pelo teclado, habilitação do botão da direita do *mouse* para visualizar opções textuais, um tutorial com um índice lateral e uma documentação e vídeo aulas para desenvolvedores iniciantes.

Durante a pesquisa, alguns aspectos qualitativos foram observados: a) vários usuários (45%) questionaram se era possível usar os próprios modelos de imagens anatômicas no CyberMedVPS, b) questionou-se também se era possível o CyberMedVPS ter disponível alguns modelos anatômicos para serem usados como objetos de aprendizagem (30%), c) a maioria dos usuários que eram professores indagaram se a instalação do CyberMedVPS era simples para que estudantes utilizassem nos computadores domésticos.

Três sugestões feitas pelos usuários podem ser ressaltadas: a) o entrevistado A sugeriu a diferenciação das cores entre os componentes Ambiente Visual e Ambiente Tátil, b) o entrevistado B propôs a criação de um banco de dados com imagens anatômicas que seriam usados como objetos de aprendizagem e c) o entrevistado C sugeriu adicionar um ícone de interrogação nos componentes para ajudas rápidas, caso seja necessário.

É interessante salientar a satisfação expressa pelos entrevistados quando detectavam que a aplicação baseada em RV, que havia sido criada, estava sendo executada com sucesso. Embora os resultados relacionados à avaliação de interface tenham sido preliminares, é possível comprovar a alta receptividade do público-alvo em utilizar ferramentas com PV para a criação de simulações baseadas em RV. Além disso, foram obtidos resultados que sugeriram a importância de interfaces visuais bem projetadas a fim de melhorar o desenvolvimento de aplicações em RV pelos usuários de saúde.

5.4. Validação do CyberMedVPS

A validação do CyberMedVPS foi necessária a fim de analisar o processo de criação e execução de aplicações visuais baseadas em RV no SPV. Tal processo envolvia a análise da aplicação genérica criada pelo CyberMedVPS por meio da

codificação da aplicação visual criada. A análise executada avalia a linguagem computacional criada pelos interpretadores de *frameworks* em RV, externos ao SPV.

Para validar o CyberMedVPS, algumas configurações foram previamente definidas para avaliar se o SPV geraria adequadamente aplicações em RV. São elas: a) definição do cenário de validação; b) definição do *framework* que será utilizado integrado ao CyberMedVPS na validação e c) definição das categorias de aplicações em RV criadas pelo CyberMedVPS.

5.4.1. Definição do cenário de validação

O objetivo principal da validação do CyberMedVPS é garantir que a linguagem genérica em XML possa ser compreendida por interpretadores de *frameworks* baseados em RV. Tais interpretadores são classes ligadas externamente ao CyberMedVPS como, por exemplo, as classes *CyberMed* e *ViMeT* da Figura 49a. Esses componentes são responsáveis por gerar aplicações em linguagem computacional que geram aplicações baseadas em RV.

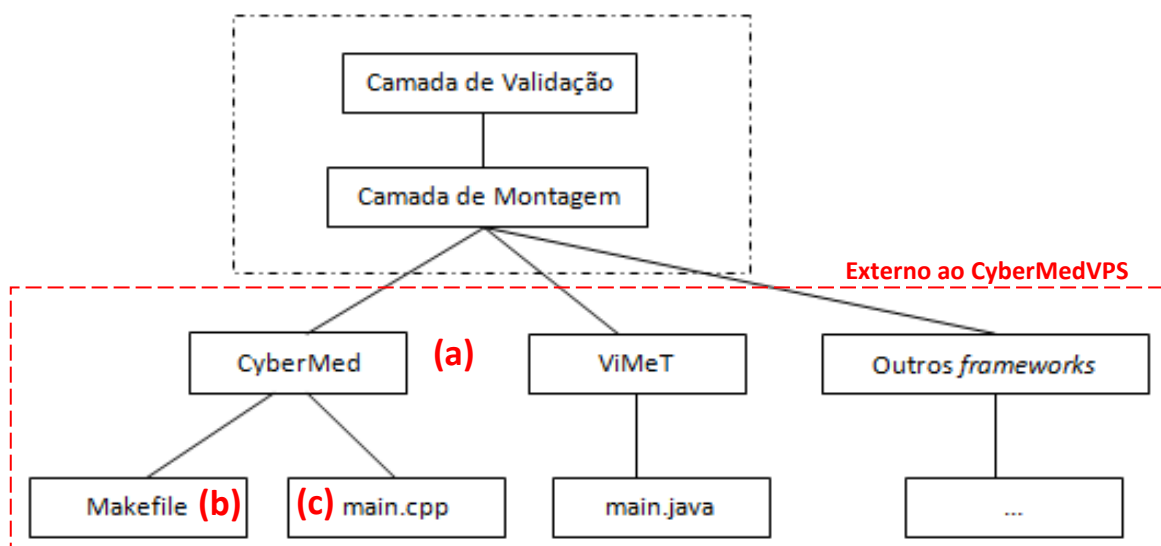


Figura 49. Componentes de validação do CyberMedVPS.

Em alguns casos, como a classe de validação *CyberMed*, foi necessária a criação de mais de um arquivo para a compilação e execução da aplicação baseada em RV e voltada para a área de saúde, como pode ser observado na Figura 49b e 49c.

5.4.2. Definição do *framework* em RV

Além de definir o cenário de execução do CyberMedVPS, é preciso definir qual *framework* baseado em RV será utilizado inicialmente para a validação. Então, é

necessário retomar a ideia de Shaw *et al.* (1993), abordada no capítulo 2, que afirma que para um *framework* em RV ser útil para os usuários é necessário que este apresente, por exemplo, uma geração de imagens estereoscópicas, uma resposta rápida e oculte os detalhes de comunicação com os dados.

Portanto, adotando a ideia de Shaw *et al.* (1993) e analisando os *frameworks* com recursos baseados em RV que foram estudados no Capítulo 2, é possível afirmar que o *framework* mais adequado para validação inicial do CyberMedVPS seria o CyberMed.

Outro fato que determinou a escolha do CyberMed como *framework* para validação foi uma pesquisa realizada por Santos e Machado (2009) que analisou os *frameworks* baseados em RV da área médica. Dentre os *frameworks* estudados neste trabalho, o mais completo em termos de licença, avaliação do usuário, suporte háptico, recursos estereoscópicos, de colisão e de deformação de objetos virtuais foi o *framework* CyberMed.

a. Ambiente de Execução para o *framework* CyberMed

Como foi previsto no processo de concepção da arquitetura (Capítulo 4), algumas alterações poderiam ser realizadas para que *frameworks* baseados em RV possam se integrar de modo consistente, independente do SO, com o CyberMedVPS.

No caso do *framework* CyberMed, a única alteração detectada no estudo seria a necessidade de transformar esse *framework* em um componente multiplataforma, pois até então o CyberMed só era executado na plataforma *Linux*. Baseado nisso, foi definido que para ocorrer a integração do CyberMed com o SPV CyberMedVPS seria interessante executar a migração do *framework* para o *Windows* (MORAIS e MACHADO, 2011). Para tornar o CyberMed compatível com *Windows*, foi necessário avaliar quais bibliotecas e pacotes estavam diretamente dependentes do sistema operacional em uso. Estudos foram realizados e se detectou que:

- A biblioteca OpenGL deveria ser alterada para sua versão correspondente na plataforma *Windows*;
- A biblioteca *Thread*, nativa da plataforma *Linux*, não possui versões exatamente compatíveis ao *Windows*;
- Seria necessário organizar as chamadas do arquivo *Makefile* das aplicações que utilizavam o CyberMed para estruturas de diretórios e bibliotecas condizentes ao *Windows*;

- Era necessário configurar um ambiente que possuísse o compilador g++, que é compilador nativo do *Linux*.

Em relação à biblioteca OpenGL¹³, a biblioteca a glut para *Windows*, na versão 3.7.6, foi instalada nessa plataforma de licença proprietária para o CyberMed. Tal biblioteca foi inserida para evitar problemas com chamadas relacionadas a recursos de visualização. Com relação à biblioteca *Thread*, foi encontrada uma *interface* chamada de *Pthread*¹⁴ que substituíra as chamadas da biblioteca nativa do *Linux* no *Windows*.

Com o uso de novas bibliotecas no *framework* CyberMed, foi necessário inserir as diretivas *#ifdef* e *#ifndef* em algumas classes do *framework* CyberMed. Tais diretivas foram utilizadas para que o *framework* pudesse chamar as bibliotecas específicas de cada SO. Portanto, as mudanças realizadas não alteraram a sintaxe das aplicações finais que utilizavam os métodos do *framework* CyberMed.

Apesar de não ter ocorrido mudança na estrutura da aplicação em RV que utiliza o *framework* CyberMed, modificações foram necessárias no *Makefile* que complementa a aplicação. No *Windows*, tal arquivo sofreu modificações como, por exemplo, a alteração no diretório das bibliotecas nativas e nas chamadas das dependências.

Outro fato que merece ressalva é a configuração de um ambiente no *Windows* para executar o compilador g++. A solução encontrada foi a instalação do MinGW¹⁵, um compilador que proporciona acesso às funcionalidades da linguagem C e C++. Associado a esse compilador foi instalado os aplicativos *msys* (para executar o *Makefile*) e *gdb* (para realizar o *debug* da aplicação computacional em C/C++). A partir das mudanças realizadas foi elaborada uma documentação de instalação manual do CyberMed no *Windows* que pode ser visto no Apêndice C.

Além da integração com o *framework* CyberMed, o SPV desenvolvido deve estar pronto para trabalhar com outros *frameworks* baseados em RV também. Desse modo o usuário ao solicitar a execução da aplicação visual deve ser capaz de definir qual *framework* ele deseja utilizar na execução. Para suprir essas necessidades, foi criada uma classe chamada de *FrameworkAsker*, na camada de visualização, cujo leiaute pode ser observado na Figura 50. Essa classe é responsável por definir qual interpretador deve ser utilizado para executar a aplicação final.

¹³ Disponível em: <http://www.opengl.org/>

¹⁴ Disponível em: <http://sourceware.org/pthreads-win32/>

¹⁵ Disponível em: <http://www.mingw.org/>

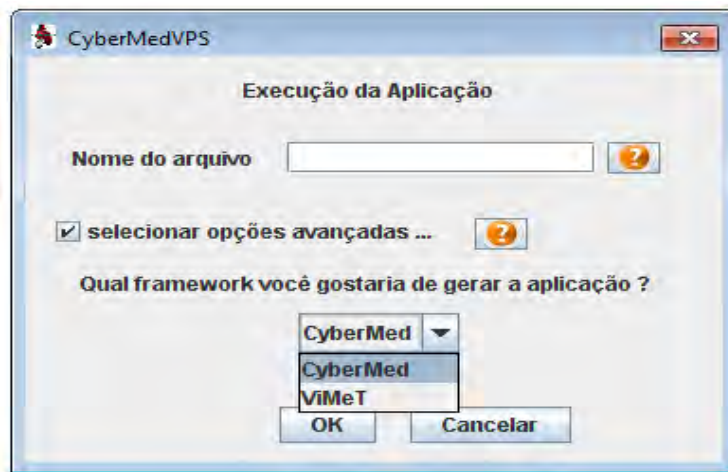


Figura 50. Leiaute da classe *FrameworkAsker*.

Dependendo da escolha do usuário, o CyberMedVPS solicita o validador correspondente à escolha e a execução da aplicação em RV é iniciada. Caso o usuário de saúde não saiba qual *framework* utilizar ele possui uma opção padrão. Nessa opção, é o próprio SPV quem decide qual *framework* deve ser utilizado, de acordo com os parâmetros da aplicação visual criada. Em um primeiro momento, o SPV está configurado para utilizar *framework* CyberMed.

5.4.3. Definição das categorias de aplicações em RV

Após definir qual *framework* baseado em RV será utilizado para validação do SPV, foi necessário definir quais categorias de aplicações, de acordo com a aplicação visual montada, poderiam ser analisadas em um primeiro momento com o CyberMedVPS. Assim, a classe interpretadora *CyberMed* definiu seis categorias de aplicações possíveis, a partir da leitura dos parâmetros do XML gerado pelo SPV. São elas:

- *Simple Visualization*: é uma aplicação em RV apenas com recurso de visualização monoscópica dos modelos;
- *Stereo Visualization*: aplicação em RV com visualização estereoscópica dos modelos;
- *Stereo Shutter Visualization*: aplicação de visualização com estereoscopia e com utilização do dispositivo *shutter glasses*;
- *Simple Interator*: é uma aplicação em RV de visualização (estereoscópica ou monoscópica) que tem a participação de instrumento(s) na manipulação de objetos. Não há detecção de colisão e deformação no toque do instrumento com o modelo;

- *Simple Collision*: é uma aplicação em RV de visualização com detecção de colisão. Mas nesse tipo de categoria a colisão não gera nenhuma deformação no modelo colidido;
- *Simple Deformation*: é uma aplicação de visualização com deformação e detecção de colisão. Nesse tipo de aplicação a colisão gera uma reação no modelo colidido;
- *Only Interator*: é um tipo de aplicação que tem a participação de instrumento(s) hápticos na manipulação de modelos. Além disso, não há a visualização dos modelos, apenas a sensação de toque.

Quadro 5. Objetos visuais que compõem cada categoria de aplicação do CyberMedVPS.

Componente Categorias	Modelo	Agregador	Instrumento	Ambiente Tátil	Toque	Ambiente Visual	Exibição
Simple Visualization	✓	✓ ou ✗	✗	✗	✗	✓	✓
Stereo Visualization	✓	✓ ou ✗	✗	✗	✗	✓	✓
Stereo Shutter Visualization	✓	✓ ou ✗	✗	✗	✗	✓	✓
Simple Interator	✓	✓ ou ✗	✓	✓ ou ✗	✗	✓ ou ✗	✓
Simple Collision	✓	✓ ou ✗	✓	✓ ou ✗	✓	✓ ou ✗	✓
Simple Deformation	✓	✓ ou ✗	✓	✓ ou ✗	✓	✓ ou ✗	✓
Only Interator	✓	✓ ou ✗	✓	✓	✓	✗	✓

No validador *CyberMed*, as categorias das aplicações possíveis servem para prever quais componentes estão presentes na aplicação visual, como pode ser observado no Quadro 5. É possível observar, por exemplo, que as categorias mais simples em relação ao número de componentes utilizados na aplicação são o *Simple Visualization*, *Stereo Visualization* e *Stereo Shutter Visualization*. Em contrapartida, as categorias que podem utilizar todos os componentes de uma aplicação em RV são a *Simple Collision* e *Simple Deformation*.

5.4.4. Resultados da validação do CyberMedVPS

Para a validação do CyberMedVPS, foi proposta a execução de todas as categorias de aplicações no cenário definido anteriormente, utilizando o *framework* CyberMed para gerar as aplicações finais.

Na validação, o resultado é considerado correto a partir da criação de três arquivos distintos: a) um arquivo predominantemente chamado de *application.xml* – resultado da conversão da aplicação visual para linguagem genérica, b) o arquivo *main.cpp* com o código em C++ que utilizará as funções do *framework* CyberMed e c) o *Makefile* para a compilação do *main.cpp*. Esses dois últimos arquivos são resultados da interpretação da linguagem genérica em linguagem computacional.

Quando a aplicação em RV foi executada, houve a comprovação de que os arquivos *main.cpp* e *Makefile* foram criados com sucesso. Mais detalhes sobre outros resultados obtidos com a validação poderão ser vistos e discutidos no Apêndice D. Dessa forma, foi possível garantir que o CyberMedVPS é capaz de gerar aplicações com os recursos essenciais de uma aplicação em RV desde que o interpretador esteja bem configurado.

5.5. Teste de defeitos do CyberMedVPS

Após a correção dos problemas detectados e relatados na etapa de validação, foi elaborado um cenário de teste para observar como seriam as reações do sistema em casos de execução que tenderiam a gerar problemas, seja por erros dos usuários ou mesmo por *bugs* do SPV. Assim, o CyberMed foi avaliado em duas etapas: a) análise da camada de visualização e b) análise da camada de comunicação.

5.5.1. Análise de problemas na interface

Com relação às análises de interface, houve o mapeamento de 15 ações que poderiam ser prejudiciais ao desempenho do CyberMedVPS. Assim, alguns aspectos foram definidos a fim de observar a reação do SPV, como: a) a ocorrência de envio de avisos ao usuário relatando o problema e se o sistema sabe identificar a causa, b) a existência do envio de dicas para o usuário resolver manualmente o problema, se a causa for reversível e c) se não for um erro manual, é verificado se o sistema consegue realizar a análise e reverter o problema sem reiniciar a aplicação visual em andamento. Todos os detalhes desse levantamento podem ser vistos no Quadro 6.

Quadro 6. Tabela de ações avaliadas no teste de defeitos do CyberMedVPS.

Ações	Descrição	Qtde. de testes	Resultado esperado	Resultado obtido?	Houve problema?	Reversão do problema?	Notificação por dica?	Notificação por ações?	Reação do sistema
A	Remoção de objeto que não tem ligação (pelo ícone do SPV)	40	Que o sistema responda e notifique que foi removido com sucesso	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada
B	A + Desfazer Ação (pelo ícone)	40	Que o objeto removido apareça novamente na área de fluxograma	Igual ao esperado	Não	-	-	Sim	Respondeu da maneira esperada
C	A + B + selecionar objeto (com o botão esquerdo do mouse) + duplicar objeto selecionado (pelo ícone)	40	Que apareça na área do fluxograma um objeto igual ao selecionado	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada
D	A + B + C + inserir um novo objeto pelo atalho do teclado	40	Que o sistema insira na área de fluxograma o objeto correto e notifique a ação	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada
E	D + criar ligação entre dois objetos válidos	40	Que o sistema permita criar a ligação válida e notifique a ação	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada
F	Remoção de um objeto com ligações	40	Que o sistema remova da área de trabalho o objeto com suas respectivas ligações da área de fluxograma	A remoção do objeto só atualiza se houver movimentação. Do contrário só apaga a linha antecessora	Se o objeto tiver linhas sucessoras e antecessoras	Sim	Sim	Sim	Depois da correção, funciona dentro do esperado
G	F + Duplicação de um objeto pelo clique do menu lateral	40	Que o sistema duplique o objeto após a opção Duplicar objeto ser clicada pelo mouse	Igual ao esperado		-	Sim	Sim	Respondeu da maneira esperada
H	Edição de parâmetros pelo menu lateral (ao clicar com o botão da direita em cima do objeto)	40	Que um objeto seja inserido na área do fluxograma quando a opção de editar parâmetros do menu lateral for clicada	Igual ao esperado	Não	-	Não	Não	Respondeu da maneira esperada

Caso	Descrição	Qtde. de testes	Resultado esperado	Resultado obtido?	Houve problema?	Reversão do problema?	Notificação por dica?	Notificação por ações?	Reação do sistema
I	Visualização dos registros pelo menu lateral (ao clicar com o botão da direita em cima do objeto)	40	Que o sistema permita a visualização das informações dos objetos ao clicar na opção Ver Informações no menu lateral	Igual ao esperado	Não	-	Não	Não	Respondeu da maneira esperada
J	Salvar fluxograma	40	Que o sistema notifique qual o local que o usuário gostaria de salvar a aplicação visual	Igual ao esperado	Não	-	Não	Sim	Respondeu da maneira esperada
K	Abrir Fluxograma	40	Que o sistema monte a aplicação lida da forma correta	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada
L	Sair do sistema clicando no ícone de sair (X)	40	Que o sistema pergunte antes de fechar se o usuário gostaria de salvar a aplicação visual que estava sendo criada	Igual ao esperado	Não	-	Não	Não	Respondeu da maneira esperada
M	Salvar uma aplicação + Criar novo projeto + pedir para abrir projeto	40	Que o sistema consiga salvar a aplicação corretamente, consiga criar um novo projeto corretamente e abra a aplicação visual da forma correta	Igual ao esperado	Não	-	Sim	Sim	A aplicação visual aberta não fica exatamente na mesma posição da aplicação visual original, pois a aplicação é redesenhada
N	M + Inserção de um novo objeto	40	Que o sistema consiga agregar da forma correta o novo componente	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada
O	N + edição de dados de um objeto no objeto	40	Que o sistema consiga editar as configurações do novo objeto e consiga agregar com as configurações da aplicação visual que foi aberta	Igual ao esperado	Não	-	Sim	Sim	Respondeu da maneira esperada

Era esperado que os procedimentos comuns como edição de parâmetros dos objetos, visualização dos registros dos objetos, duplicação e remoção de objetos e linhas, por exemplo, fossem executados sem maiores complicações. Mesmo assim, esses casos foram mantidos nos testes de execução para garantir que a execução fosse realizada.

Em ações mais complexas foi feita a análise visual das interações entre os objetos do CyberMedVPS. Além desse procedimento, o teste de defeitos executou a depuração das classes do SPV a fim de avaliar com mais detalhes a dinâmica entre as

classes do CyberMedVPS. Na observação executada, as classes da camada de visualização se comportaram da maneira prevista, obedecendo à estrutura definida na arquitetura.

5.5.2. Análise de problemas na camada de comunicação

Para avaliar os possíveis defeitos da camada de comunicação foi necessário o trabalho em conjunto com o interpretador externo do CyberMed que foi implementado para a validação. Assim como o teste de defeitos da interface, a análise da camada de comunicação também foi segmentada em dois passos: a) avaliação da geração do XML (Figura 51a) e b) avaliação do código computacional gerado pelo interpretador (Figura 51b). Para essas duas etapas foi definido que seriam avaliados todos os cenários de aplicação visual que foram anteriormente validados – *SimpleVisualization*, *StereoVisualization*, *SimpleInterator*, *SimpleCollision*, *SimpleDeformation* e *OnlyHaptic*.

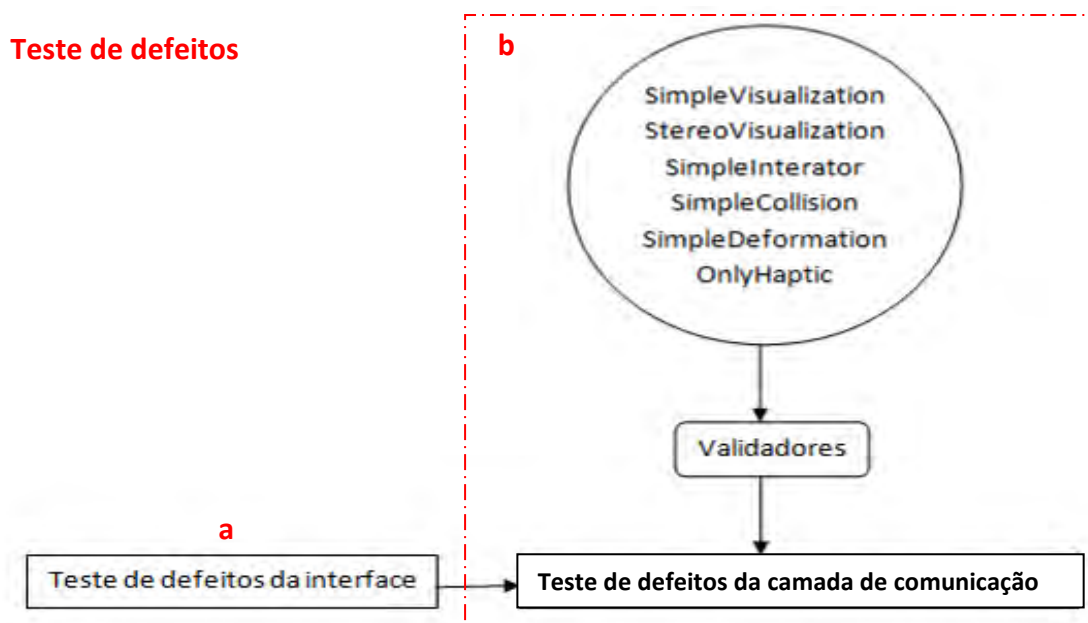


Figura 51. Sistemática dos testes de defeitos.

Na primeira etapa do teste, foi avaliado se a geração do XML aconteceu da forma correta, caso o usuário deixasse de especificar parâmetros na edição dos objetos. Assim, o sistema se comportou da forma esperada, pois o CyberMedVPS em parâmetros importantes para a aplicação baseada em RV conseguiu especificar valores automaticamente.

Quando o usuário especificava parâmetros errados nos objetos, foi percebido que esse problema já era resolvido na camada de visualização, permitindo que o XML que fosse gerado estivesse validado. Se algum valor ainda estivesse errado na aplicação

visual, o XML também estaria corrompido. Nesse último caso a reversão do problema era imprecisa em eficiência, pois o sistema não terá como prever qual foi o valor ou a opção escolhida pelo usuário.

Após garantir que o XML funciona da forma adequada, a etapa de teste se direciona para a geração do código computacional da aplicação em RV que é gerada pelo interpretador. Assim com a etapa de avaliação do XML, esse estágio da avaliação analisa a geração do código computacional para todos os cenários de aplicação em RV esperados pelo sistema.

A leitura do interpretador sobre a linguagem em XML ocorreu da forma esperada para todos os cenários das aplicações em RV. Uma única exceção foi detectada: a criação do código computacional não será bem-sucedida se o XML estiver corrompido, pois a integridade da linguagem genérica é fator determinante para o funcionamento do interpretador. Nesse caso, o erro de execução que deve ser notificado está externo ao CyberMedVPS. O aviso de erro partirá do sistema operacional que executará o SPV como erro de execução.

5.5.3. Conclusões do teste de defeitos

O teste de defeitos do CyberMedVPS serviu para avaliar possíveis falhas em locais do SPV em que o comportamento não estava correto ou não estava em conformidade com a sua especificação. As análises realizadas comprovam que as reações do CyberMedVPS, estão dentro do esperado pela sua especificação.

6

Conclusão

De acordo com Elias *et al.* (2010), a informática se caracteriza como uma área que contribui no processo de capacitação dos profissionais de saúde. No âmbito educacional, a área da computação pode ser definida como uma alternativa capaz de transformar a prática pedagógica em uma estrutura acessível, flexível e dinâmica.

Segundo Dragicevic e Fekete (2004), a necessidade de interfaces computacionais cada vez mais simples é imprescindível para sistemas em RV, pois é observado o crescimento no número de dispositivos de interação de entrada/saída e com diferentes formas de comunicação. Isso demanda uma velocidade de aprendizado mais eficiente para os usuários. Por isso, quanto mais intuitivo for o modo de visualização dessas interfaces, melhor para o aprendizado do usuário.

6.1. Contribuições

No início desse trabalho, foram levantados alguns pontos (GREEN, 1995) que fundamentam as razões pelos quais a PV está presente até os dias atuais. Apesar de o estudo ser relativamente antigo, o presente trabalho justifica que os pontos levantados por este autor não estão defasados. São eles:

- a) PV é tecnicamente desafiadora, pois existem muitos pontos que precisam ser estudados em pesquisa - Isso foi indicado no capítulo 3, por meio de uma pesquisa sobre os *frameworks* da área de saúde com PV e a análise de outros estudos relacionados, que concluiu que não existia um padrão de técnica que deve ser adotada para sistemas que geram aplicações de RV;
- b) A PV suporta o modelo de fluxos de dados da computação muito melhor do que programação textual convencional – com o CyberMedVPS, foi possível observar como a dinâmica dos dados de uma aplicação em RV pode ser melhor compreendida, se comparada a sua aplicação em código fonte;
- c) Embora os programadores sejam usuários, a IHC convencional tem agido pouco em projeto de ambientes de programação – A maioria dos ambientes que adotam alguma das técnicas de PV não foca nos usuários leigos em programação.

Vale ressaltar também que, neste trabalho, os seguintes objetivos específicos foram alcançados: a) foi realizada uma pesquisa sobre o uso de PV em sistemas de RV na área médica, b) foi analisado, por meio de avaliações, as necessidades e as experiências do público-alvo na área de RV, c) foi definida uma arquitetura de PV como

estratégia para possibilitar a interação dos usuários de saúde para que desenvolvam suas próprias aplicações com recursos de estereoscopia, interação háptica, deformação e detecção de colisão, d) foi implementado o CyberMedVPS como resultado concreto da arquitetura proposta, e) foi avaliado a GUI do CyberMedVPS com o público-alvo antes de prosseguir o desenvolvimento da camada de comunicação e f) após todo o desenvolvimento do SPV houve a validação do CyberMedVPS.

A criação da arquitetura baseada em PV para *frameworks* em RV teve como intenção suprir a carência de interfaces visuais que pudessem auxiliar o usuário da saúde no desenvolvimento de aplicações em RV, além de possibilitar que *frameworks* sem interface gráfica pudessem ter um modo de programação gráfico para ser utilizado por usuários leigos em programação. Dessa forma, o distanciamento existente entre o desenvolvedor, juntamente com o *framework* em RV, e o profissional da saúde pode ser reduzido com o uso da PV. Assim, os profissionais de saúde podem ter papéis muito mais ativos no processo de desenvolvimento de aplicações baseadas em RV com o uso do CyberMedVPS.

Desde o momento do planejamento do SPV, observou-se o cuidado em atender às necessidades dos profissionais de saúde e isso foi fundamental para garantir que os resultados das validações (Capítulo 5) fossem satisfatórios. Além disso, é possível afirmar que os requisitos funcionais do CyberMedVPS foram atendidos, como: a) garantir a funcionalidade - ser um sistema de criação de aplicações baseadas em RV para usuários da área de saúde, b) garantir a usabilidade - apresentar recursos baseados nos princípios do *design* de interface e nas técnicas de PV, c) garantir a confiabilidade - o SPV deve validar as aplicações visuais criadas, d) garantir a eficiência – o sistema deve dar as resposta em tempo de execução, e) garantir a manutenibilidade – as modificações das aplicações só serão realizadas pelo modo gráfico e f) garantir a portabilidade - o SPV deve ser compatível com diversos *frameworks* e ser multiplataforma.

Em relação aos SPVs observados na literatura, especificamente aqueles citados no capítulo 2, o CyberMedVPS tem como vantagem ser um sistema de uso aberto, capaz de oferecer recursos em RV como estereoscopia, sensibilidade háptica, suporte à colisão e à deformação, e suporte a modelos com múltiplas camadas. Assim, o CyberMedVPS foi projetado para funcionar integrado a mais de um *framework* em RV devido a sua arquitetura hierárquica.

O CyberMedVPS também tem como funcionalidade de destaque o processo de validação da aplicação visual que ocorre em tempo de execução, ou seja, é um sistema

capaz de realizar os auto-testes durante o desenvolvimento da aplicação visual a fim de validar os dados inseridos pelo usuário no SPV.

Outra característica a favor do CyberMedVPS, em relação aos demais sistemas com recursos de PV, são as avaliações realizadas com o público-alvo no seu processo de implementação. Até o momento foram realizadas duas avaliações: um questionário e uma entrevista. Elas foram primordiais para definir as reais necessidades dos usuários de saúde em sistemas em RV e para avaliar se a GUI do CyberMedVPS estava adequada à realidade computacional dos mesmos e se as técnicas de PV estavam sendo eficientes.

6.2. Trabalhos Futuros

Para as etapas futuras do CyberMedVPS pretende-se avaliar o SPV com uma amostra maior de profissionais da saúde com o intuito de validar o sistema por completo, integrado com outros *frameworks* e validar o sistema com outros cenários de aplicações em RV. Além disso, serão habilitados novos recursos no SPV, como a colaboração e analisar a eficiência dos novos recursos visuais – baseados nos resultados obtidos na entrevista da seção 5.3. Também, será desenvolvida uma documentação de consulta para o usuário, quando o mesmo quiser desenvolver aplicações mais avançadas em RV. Assim, a intenção é que essa documentação seja integrada ao CyberMedVPS, no módulo de ajuda. Ainda, serão analisadas soluções mais simples para o dicionário de dados, ao invés do SGBD. Por fim, futuramente, o CyberMedVPS será validado com outros *frameworks* para que se torne um sistema mais robusto e útil para um número maior de profissionais médicos.

6.3. Considerações Finais

De acordo com uma pesquisa da HealthGrades (2008), 238.337 mil mortes foram causadas por erro médico em hospitais dos Estados Unidos, entre 2004 e 2006. No Brasil, tal panorama não é diferente, pois segundo dados do Supremo Tribunal de Justiça (STJ) o número de processos relacionados a erro médico aumentou 155% nos entre os anos de 2003 e 2009 (JAM, 2009). Uma das possíveis causas desses dados pode estar vinculada à falta de preparo do profissional da área médica. Por isso é importante que ferramentas computacionais, como o CyberMedVPS, sejam capazes de auxiliar no processo de aprendizado e treinamento através de sua utilização como objeto de

aprendizagem ou ferramenta de treinamento para estudantes e profissionais da área de saúde. Isso é possível por diversos motivos como, por exemplo: a) ser multiplataforma, b) abrir e executar exemplos prontos, c) salvar aplicações visuais em desenvolvimento, d) compatível com diversos *frameworks* em RV e e) apresentar módulos de ajuda.

Em todo o estudo realizado, durante o desenvolvimento desse trabalho, não foi encontrado na literatura um sistema de aprendizagem e treinamento com recursos baseados em RV e adepto das técnicas de PV que fosse de uso livre e que fosse elaborada diretamente para usuários da área de saúde. Além disso, a escolha da tecnologia Java foi primordial para garantir que o CyberMedVPS fosse compatível com diversos tipos de sistemas operacionais. Assim, é possível que os usuários de saúde criem suas próprias aplicações visuais e ocupem um papel ativo na utilização de sistemas computacionais que gerem aplicações baseadas em RV para seu treinamento e aprendizado na área.

A maioria dos sistemas que utilizam PV adota essa técnica principalmente para analisar os dados da aplicação visual. Dessa forma, a visualização da aplicação fica com papel secundário (MEYER-SPRADOW *et al.*, 2010). No CyberMedVPS, a visualização do resultado final é tão importante quanto à montagem da aplicação visual devido a funcionalidade de alto-teste e sua compatibilidade com diversos *frameworks* baseados em RV.

Um estudo de Jaspers *et al.* (2004) afirma que quanto mais cedo for o uso de técnicas visuais no processo de concepção de um sistema computacional da área de saúde, maior será a ajuda oferecida para as práticas da área e suportadas pelo sistema. Com os estudos realizados nessa dissertação não foi diferente. Embora algumas funcionalidades ainda pudessem ser mais complexas ou até mesmo compatíveis com outros *frameworks*, o SPV proposto serviu para materializar toda a avaliação realizada com o público de saúde e provar que um usuário de saúde, leigo em computação, é capaz de gerar suas próprias aplicações com recursos em RV sem o intermédio de um profissional da computação. Isso só ocorre desde que exista o auxílio de um sistema com técnicas visuais que o auxilie durante o processo de interação.

Referências

- (ALLARD *et al.*, 2007) ALLARD, J.; COTIN, S.; FAURE, F.; BENSOUSSAN, P.- J.; POYER, F.; DURIEZ, C.; DELINGETTE, H.; GRISONI, L. (2007) *SOFA - an Open Source Framework for Medical Simulation. **Proceedings of Medicine Meets Virtual Reality (MMVR'15)***. Estados Unidos, Fevereiro 2007. P. 2-5. IOSPress.
- (ALVES *et al.*, 2011) ALVES, F. J.; DAMASCENO, E. V.; BEVENUTO, L. P. (2011) **Avaliação De Interfaces De Ferramentas Case Para Elaboração De Diagramas Da UML. Revista Científica 2011** – Atenas Faculdades.
- (ASSFALG *et al.*, 2000) ASSFALG, J.; DEL BIMBO, A.; PALA, P.(2000) *Virtual Reality for Image Retrieval. **Journal of Visual Languages & Computing***, vol. 11, n. 2, Abril - 2000, P. 105-124.
- (AUGUSTINE *et al.*, 2004) AUGUSTINE, K.E.; HOLMES III, D.R.; ROBB, R.A. (2004) *ITK and ANALYZE: A Synergistic Integration. **Proceedings of The International Society for Optical Engineering (SPIE)***, 5367:6-15, Medical Image.
- (BARNUM, 2011) BARNUM, C.M. (2011) **Usability Testing Essentials: Ready, Set Test! Book Review"** *Professional Communication, IEEE Transactions*, N.3, 54:336-337.
- (BASDOGAN *et al.*, 2007) BASDOGAN, C.; SEDEF, M.; HARDERS, M.; WESARG, S.(2007) *VR-Based Simulators For Training In Minimally Invasive Surgery. **IEEE Computer Graphics and Applications** 27*, 2:54–66.
- (BASTOS e PROENÇA, 2000) BASTOS L.A.M.; PROENÇA M.A. (2000) A Prática Anatômica e a Formação Médica. **Revista Panamericana de Salud Publica**, N.6, 7:395-402.
- (BASTOS *et al.*, 2004) BASTOS, T.A., SILVA, R.J.M., RAPOSO, A.B.; GATTASS, M. (2004) *ViRAL: Um Framework para o Desenvolvimento de Aplicações de Realidade Virtual. **Proceedings of VII Symposium on Virtual Reality (SVR)***, P. 51–62.
- (BRUYNS e MONTGOMERY, 2002) BRUYNS, C. D.; MONTGOMERY, K. (2002) *Generalized Interactions Using Virtual Tools Within The Spring Framework: Cutting. **Proceedings of Medicine Meets Virtual Reality (MMVR)***. IOSPress
- (BURNETT e BAKER, 1994) BURNETT, M.M.; BAKER, M.J.(1994) **A Classification System For Visual Programming Languages**. Oregon State University, Corvallis.
- (BURNETT, 1999) BURNETT, M.M. (1999) *Visual Programming. **Encyclopedia of Electrical and Electronics Engineering*** (John G. Webster, ed.), John Wiley & Sons Inc., New York.
- (CAMPOS e SAMPAIO, 2005) CAMPOS, A.A.S.; SAMPAIO, F.F. (2005) Uma Aplicação de Realidade Virtual não Imersiva no Ensino de Astronomia. **Anais do XVI Simpósio Brasileiro de Informática na Educação (SBIE)**, Juiz de Fora – MG, 2005. P. 103-113.
- (CAO *et al.*, 1999) CAO, B; DODDS, G.I.; IRWIN, G.W.(1999) *An Event Driven Virtual Reality System for Planning and Control of Multiple Robots. **Proceedings of the International Conference on Intelligent Robots and Systems (IROS '99)***, 2:1161–1166, Oct. 1999.

- (CAVUSOGLU *et al.*, 2006) CAVUSOGLU, M.C.; GOKTEKIN, T.G.; TENDICK, F.(2006) *GiPSi: A Framework for Open Source/Open Architecture Software Development for Organ-Level Surgical Simulation*. **IEEE Trans. Information Technology in Biomedicine**, N. 2, 2006, 10:312-321.
- (COSTAGLIOLA *et al.*, 2002) COSTAGLIOLA, G.; DE LUCIA, A.; OREFICE, S.; POLESE, G.(2002) *A Classification Framework to Support The Design Of Visual Languages*. **Journal of Visual Languages of Computing** Vol. 13, Ed. 6 (Dec.), P. 573-600, 2002.
- (CURK *et al.*, 2004) CURK, T.; DEMSAR, J.; XU, Q.; LEBAN, G.; PETROVIC, U.; BRATKO, I.; SHAULSKY, G.; ZUPAN, B. (2004) *Microarray Data Mining With Visual Programming*. **Bioinformatics**, n. 3, 21:396-398. Oxford University Press.
- (CYBIS *et al.*, 2007) CYBIS, W.; BETIOL, A.H.; FAUST, R. (2007) **Ergonomia e Usabilidade – Conhecimentos, Métodos e Aplicações**. Ed. Novatec., P. 73.
- (DIAS *et al.*, 2009) DIAS, G.P.; MACHADO, L.S.; MORAES, R.M. (2009) Um Sistema Baseado em Construtivismo e Realidade Virtual para Apoio ao Ensino de Classificação de Imagens. **Proceedings of the International Conference on Engineering and Computer Education (ICECE2009)**. Buenos Aires, Argentina.
- (DIETRICH *et al.*, 2007) DIETRICH, O.; RAYA, J.G.; REEDER, S.B.; REISER, M.F.; SCHOENBERG, S.O. (2007) *Measurement of signal-to-noise ratios in MR images: influence of multichannel coils, parallel imaging, and reconstruction filters*. In **Journal of Magnetic Resonance Imaging 2007**; 26:375–385.
- (DONG, 2002) DONG, L.(2002) *Transforming Visual Programs into Java and Vice Versa*, **Master's Thesis**, Dalhousie University, Halifax, Nova Scotia, Canadá.
- (DRAGICEVIC e FEKETE, 2004) DRAGICEVIC, P.; FEKETE, J-D. (2004) *Support for input adaptability in the ICON toolkit*. **Proceedings of the 6th international conference on Multimodal interfaces**.
- (DUBEY e RANA, 2011) DUBEY, S.K.; RANA, A.(2011) *Usability Estimation of Software System by using Object-Oriented Metrics*. **ACM SIGSOFT Software Engineering Notes**, N. 2, 36:1-6.
- (ELIAS *et al.*, 2010) ELIAS, S.; PIRES, S. R.; PATROCINIO, A. C.; MEDEIROS, R. B. (2011) *Uso de Software Como Ferramenta Pedagógica no Processo de Ensino-Aprendizagem da Mamografia Digital*. **Radiol Bras**, n.2, 42:115-120.
- (FICHEMAN *et al.*, 2008) FICHEMAN, I.K.; CORRÊA, A.G.D.; SAUL, J; BIAZON, L.C.; CAMARGO, M.A.; MANTOVANI, M.S.M.; ZUFFO, M.K.; LOPES, R.D.(2008) *Uma Aventura no Espaço com a Nave Mário Schenberg: Ambiente Colaborativo em Realidade Virtual para Aprendizagem de Ciências*. **Anais do Simpósio Brasileiro de Informática na Educação (SBIE)**. P. 746-755.
- (FIGUEROA e CASTRO, 2011) FIGUEROA, S.; CARMEN, M.; CASTRO, R.G. AND TERRAZAS, B. V.; PÉREZ, A. G. (2011) *Essentials in Ontology Engineering: Methodologies, Languages, and Tools*. **Proceedings of CIB Conference W078-W012 - 2nd Workshop on eBuildings Data Models**, P. 26-28.
- (FIGUEROA *et al.*, 2002) FIGUEROA, P.; GREEN, M.; HOOVER, H.J. (2002) *Intml: A Description Language for VR Applications*. **Proceedings of the Seventh International Conference on 3D Web Technology**, P.53-58.

- (FREITAS *et al.*, 2003) FREITAS, C.M.D.S.; NEDEL, L.P.; MANSSOUR, I.H.; GAVIÃO, J. K.; MACIEL, A.; PAIM, T.C.(2003) *Framework Para Construção De Pacientes Virtuais: Uma Aplicação em Laparoscopia Virtual. **Proceedings of SBC Symposium on Virtual Reality (SVR)***, P. 283-296. CD-ROM.
- (GOLDSTINE, 1972) GOLDSTINE, H.H. (1972) *The Computer From Pascal To Von Neumann*. Editora Princeton University Press, ISBN: 0-691-08104-2, 1972.
- (GOMES *et al.*, 2006) GOMES, R.G.S.; MACHADO, L.S.; MORAES, R.M.(2006) *The Calculadora Estatística for Present and Distance Learning. **Proceedings of the International Conference on Teaching Statistics (ICOTS-7)***. Salvador. P. 1-7.
- (GRAHAM *et al.*, 1996) GRAHAM, T.C.N., MORTON, C.A.; URNES, T.(1996) *Clockworks: Visual Programming Of Component-Based Software Architectures. **Journal of Visual Languages and Computing***, Academic Press.
- (GREEN, 1995) GREEN, T. R. G. (1995) *Noddy's Guide To ... Visual Programming. Interfaces, **The British Computer Society, Human-Computer Interaction Group***. 1995.
- (GREENLEAF, 1995) GREENLEAF, W.J.(1995) *Virtual Reality Applications in Medicine. **Proceedings of the WESCON'95 Conference Record***. Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies, P. 691.
- (GREENLEAF, 2004) Greenleaf, W.(2004) *Medical Applications of Virtual Reality. Disponível em: <http://www.greenleafmed.com/publications/VR%20Med%20overview.pdf>*. Acessado em Dezembro de 2011.
- (HALD *et al.*, 2011) HALD, N.; SARKER, S.K.; ZIPRIN, P.; VILLARD, P.; BELLO, F. (2011) *Open Surgery Simulation of Inguinal Hernia Repair. **Proceedings of Medicine Meets Virtual Reality(MMVR)***, N. 163, P. 202-208.
- (HEALTHGRADES, 2008) HEALTHGRADES (2008) *The Fifth Annual HealthGrades Patient Safety in American Hospitals Study*, news story, Apr. 2008.McCloy, R.; Stone, R. (2001), *Virtual Reality in Surgery. BMJ*. 2001 October 20; 323(7318): 912–915.
- (HINKENJANN e MANNUSS, 2004) HINKENJANN A., MANNUSS F. (2004) *basho - A Virtual Environment Framework. **Proceedings of 7th Symposium on Virtual Reality(SVR)***. São Paulo, Brazil.CD-Rom.
- (HUa *et al.*, 2011) HU, B., MA, L.; ZHANG, W.; SALVENDY, G.; CHABLAT, D.; BENNIS, F. (2011) *Can Virtual Reality Predict Body Part Discomfort And Performance Of People In Realistic World For Assembling Tasks? **International Journal of Industrial Ergonomics***, 41: 64-71.
- (IBANEZ *et al.*, 2007) IBANEZ, L., SCHROEDER, W.; NG, L.; CATES, J. (2003) *The ITK,Software Guide*. Citeseer.
- (JACOBSON e LEWIS, 2005) JACOBSON, J.; LEWIS, M.(2005) *Game Engine Virtual Reality with CaveUT. **IEEE Computer***, 38: 79-82, n.4.
- (JAM, 2009) *Jornal da Associação Médica (JAM)* (2009), “Número de processos por erro médico no STJ cresce 155% em seis anos”. *Defesa Profissional*. p. 6 . Dezembro 2008/Janeiro 2009.

- (JASPER *et al.*, 2004) JASPERS, M. W.M.; STEEN, T.; BOSB, C.; GEENENB, M. (2004) *The think aloud method: a guide to user interface design*. **International Journal of Medical Informatics**, 73:781-795.
- (JOHNSTON *et al.*, 2004) JOHNSTON, W.M.; HANNA, J.R.P.; MILLAR R.J. (2004) *Advances In Dataflow Programming Languages*. **ACM Computing Surveys**, N. 1, 36:1-34.
- (JOHNSON e PARKER, 1995) JOHNSON, C.R.; PARKER, S.G. (1995) *Applications in Computational Medicine Using SCIRun: A Computational Steering Programming Environment*. H.W. Meuer, editor, **Supercomputer '95**, P. 2–19. Saur-Verlag, 1995.
- (KAMATH, 2012) KAMATH, R. S. (2012) Development of an improved algorithm for visualization of components. University of Shivaji University. Cap 1. On-line.
- (KAMIYA e BRANDÃO, 2009) KAMIYA, R.R.; BRANDÃO, L.O. (2009) iVProg - Um Sistema Para Introdução À Programação Através de Um Modelo Visual Na *Internet*. **Anais do XX Simpósio Brasileiro de Informática na Educação (SBIE)**. Florianópolis, SC.
- (KAUCIC e ASIC, 2011) KAUCIC, B.; ASIC, T.(2011) *Improving Introductory Programming with Scratch?* **Proceedings of the 34th International Convention, MIPRO**, P. 1095 – 1100.
- (KENNEDY *et al.*, 2002) KENNEDY, C.W.; HU, T.; DESAI, J.P.; WECHSLER, A.S.; KRESH J.Y.(2002). *A Novel Approach to Robotic Cardiac Surgery Using Haptics and Vision*. **In Cardiovascular Engineering: An International Journal**, N. 1, 2:15-22(8).
- (KIRNER e SISCOOTTO, 2007) KIRNER, C.; SISCOUTO, R. (Org.), *Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações*, **Sociedade Brasileira de Computação (SBC)**, Porto Alegre, 2007.
- (KUHNAPFEL *et al.*, 2000) KUHNAPFEL, U.; CAKMAK, H.K.; MAA, H. (2000) *Endoscopic Surgery Training Using Virtual Reality And Deformable Tissue Simulation*. **Computers & Graphics**, N. 5, 24:671-682.
- (LAMATA *et al.*, 2006) LAMATA, P.; GÓMEZ, E.J.; BELLO, F.; KNEEBONE, R.L.; AGGARWAL, R. (2006) *Conceptual Framework for Laparoscopic VR Simulators*. **IEEE Computer Graphics and Applications**, P. 69-79, 2006.
- (LINEBARGER *et al.*, 2003) LINEBARGER, J.M.; JANNECK, C.D.; KESSLER, D.G. (2003) *Shared Simple Virtual Environment: An Object-Oriented Framework for Highly Interactive Group Collaboration*. **Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '03)**, Washington, DC, USA, P. 170-180.
- (LUCANIN e FABEK, 2011) LUCANIN, D.; FABEK, I.(2011) *A Visual Programming Language for Drawing and Executing Flowcharts*. **Proceedings of the 34th International Convention**. MIPRO, 2011, P. 1679 – 1684.
- (LUCAS *et al.*, 1992) LUCAS, B.; ABRAM, G.D.; COLLINS, N.S.; EPSTEIN, D.A.; GRESH, D.L.; MCAULIFFE, K.P. (1992). *An Architecture for a Scientific Visualization System*. **Proceedings of the 3rd Conference on Visualization '92**.
- (MACHADO e MORAES, 2010) MACHADO, L.S.; MORAES, R.M.(2010) *Intelligent Decision Making in Training Based on Virtual Reality*. **Book Chapter**. Da Ruan. (Org.) **Computational Intelligence in Complex Decision Systems**, Cap 4. Paris: Atlantis Press.

- (MACHADO *et al.*, 2009) MACHADO, L.S.; MORAES, R.M.; SOUZA, D.F.L; SOUZA, L.C.; CUNHA, I.L.L. (2009) *A Framework for Development of Virtual Reality-Based Training Simulators*. **Medicine Meets Virtual Reality 17**, 142:174-176, 2009. IOSPress.
- (MACHADO, 2010) MACHADO, L.S. (2010) Dispositivos Não-Convencionais para Interação e Imersão em Realidade Virtual e Aumentada. Book Chapter. Brega, J.R.F.; Kelner, J. (Org.) **Interação em Realidade Virtual e Aumentada**, Cap 2, P. 23-33.
- (MACLAURIN, 2009) MACLAURIN, M. (2009) Kodu: *End-User Programming And Design For Games*. **Proceeding of the 4th International Conference on Foundations of Digital Games(FDG'09)**. New York, 2009.
- (MARION *et al.*, 2007) MARION, N.; SEPTSEAULT, C.; BOUDINOT, A.; QUERREC, R. (2007) *GASPAR: Aviation Management on an Aircraft Carrier using Virtual Reality*. **Proceedings in the International Conference on Cyberworlds (CW '07)**, P. 15–22.
- (MARTIKAINEN *et al*, 2012) Martikainen, S.; Viitanen, J.; Korpela, M.; Lääveri, T. (2012) Physicians' Experiences of Participation in Healthcare IT Development in Finland: Willing but Not Able. **International Journal of Medical Informatics**. N. 2, 81:98–113.
- (MARTINS *et al.*, 2003) MARTINS, R.B.; MACIEL, A.; NEDEL, L.P.(2001) SkelMod-3D: Um Sistema Livre e Multiplataforma para Modelagem e Visualização de Corpos Articulados. **Anais do II Workshop Sobre Software Livre (WSL2001)**, N. 2, P. 21-24.
- (MELO e DAMASCENO, 2006) MELO F.N.P.; DAMASCENO M.M.C. (2006) A Construção de um Software Educativo Sobre Ausculta dos Sons Respiratórios. **Revista da Escola de Enfermagem da USP**, N. 4, 40:563-569.
- (MENDOZA e LAUGIER, 2000) MENDOZA, C.A.; LAUGIER, C.(2000) *A Solution for the Difference Rate Sampling Between Haptic Devices and Deformable Virtual Objects*. **Proceedings of International Symposium in Robotics and Automotion (ISRA)**, November 2000.
- (MENDOZA *et al.*, 2001) MENDOZA, C.A.; LAUGIER C.(2001) *Realistic Haptic Rendering for Highly Deformable Virtual Objects*. **Proceedings of the IEEE Conference on Virtual Reality**, Yokohama, Japan, 13–17 March 2001, P. 264–269.
- (MERRIL *et al.*, 1995) MERRIL, G.; RAJ, R.; MERRIL, J. (1995) *Changing the Focus of Surgical Training*. **VR World**, March/April 1995, P. 56-61.
- (MEYER-SPRADOW *et al.*, 2009) MEYER-SPRADOW, J.; ROPINSKI, T.; MENSMANN, J.; HINRICHS, K.H.(2009). *Voreen: A Rapid-Prototyping Environment for Ray-Casting-Based Volume Visualizations*. **IEEE Computer Graphics and Applications(Applications Department)**, N. 6, 29:6–13.
- (MEYER-SPRADOW *et al.*, 2010) MEYER-SPRADOW, J.; ROPINSKI, T.; MENSMANN, J.; HINRINCHS, K. (2010). *Interactive Design and Debugging of GPU-Based Volume Visualizations*. **Computer Graphics Theory and Applications**, P. 239-245.
- (MICHEL *et al.*, 2010) MICHEL, F.; FERBER, J.; LAUR, P.-A.; ALEMAN, F. (2010) *Situational Programming: Agent Behavior Visual Programming for MABS Novices*. **Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (MABS)**. Toronto-CA. P. 5-18.

- (MOSSEL *et al.*, 2012) MOSSEL, A.; SCHÖNAUER, C.; GERSTWEILER, G.; KAUFMANN, H. (2012) . ARTiFICe - Augmented Reality Framework for Distributed Collaboration . Technical Report. TR-188-2-2012-02.
- (MORAIS e MACHADO, 2010) MORAIS, A.M.; MACHADO, L.S. (2010) Programação Visual para um *Framework* de Desenvolvimento de Aplicações Médicas Baseadas em Realidade Virtual. **Anais do XII Congresso Brasileiro de Informática em Saúde (CBIS)**, Porto de Galinhas, 2010. CD-ROM.
- (MORAIS e MACHADOa, 2011) MORAIS, A.M.; MACHADO, L.S.(2011) *CyberMedVPS: Visual Programming for Development of Simulators*. **Studies in Health Technology and Informatics**, vol.163. Amsterdam: IOSPress.
- (MORAIS e MACHADOb, 2011) MORAIS, A.M.; MACHADO, L.S.; MORAES, R.M. (2011) Ferramentas de Realidade Virtual para Profissionais da Área Médica. **Proceedings of Symposium on Virtual and Augmented Reality 2011 (SVR 2011)**, Uberlândia/MG. CD-ROM.
- (MOSSMAN *et al.*, 2010) MOSSMAN, J.B.; MARONI, V.; PINHO, M.; FLORES, C.D.; DAHMER, A. (2010) VR-MED: Linguagem de Domínio Específico para Ambientes Virtuais Aplicados à Educação Médica. **Anais do XIII Congresso Brasileiro em Informática na Saúde**. Porto de Galinhas - Brasil. CD-ROM.
- (NARDI, 1993) NARDI, B.A. (1993) **A Small Matter Of Programming: Perspectives On End User Computing**, edited by TechBooks, ISBN: 0-262-14053-5, 1993.
- (NIXON, 2009) NIXON, R. (2009) **Learning PHP, MySQL, and JavaScript**. Publisher: O'Reilly Media. July 2009. P. 405.
- (NUNES *et al.*, 2007) NUNES, F.L.S.; OLIVEIRA, A.C.M.; ROSSATTO, D.J. (2007) ViMeTWizard: Uma ferramenta para instanciação de um *framework* de Realidade Virtual para treinamento médico. **Proceedings of XXXIII Conferência Latinoamericana de Informática**, 1:1-8. San José.
- (OLIVEIRA e MARQUES, 2009) OLIVEIRA, A.C.G.; MARQUES, F.L.N. (2009) ViMeT and ViMeTWizard: *Process of Building a Framework and an Instantiation Tool for Programming Application in Medical Training Domain Using Free Technology*. **Proceedings of the Society for Imaging Informatics in Medicine (SIIM)**, Jun, 2009.
- (OLIVEIRA *et al.*, 2006) OLIVEIRA, A.C.M.; NUNES, F.L.S.; PAVARINI, L.; BOTEAGA, L.C. (2006) Desenvolvimento e Implementação de *Framework* de Realidade Virtual para Treinamento Médico. **Anais do VI Workshop de Informática Médica (WIM 2006)**, Maio, Brazil. CD-Rom.
- (OLSZEWSKI *et al.*, 2003) OLSZEWSKI, R.; NICOLAS, V.; MACQ, B.; REYBHLER, H. (2003) Acro 4D: *Universal Analysis for Four-Dimensional Diagnosis, 3d Planning And Simulation In Orthognathic Surgery*. **Proceedings of Computer Assisted Radiology and Surgery (CARS)**, London, June.
- (OSHIBA e TANAKA, 1999) OSHIBA, T.; TANAKA, J. (1999) 3D-PP: *Three-Dimensional Visual Programming System*. **Proceedings of the IEEE Symposium on Visual Languages (VL'99)**, P. 189–199, Tokyo, Japan, September 1999.
- (PAIVA *et al.*, 2011) PAIVA, P.V.F.; MACHADO, L.S; OLIVEIRA, J.C.(2011) *Simulation And Performance Evaluation of a Distributed Haptic Virtual Environment Supported*

By The Cybermed Framework. **Proceedings of XIV Symposium on Virtual Reality(SVR)**, P.188-196, Uberlândia, MG, Brasil. CD-Rom.

- (PAQUET e KROPF, 2000) PAQUET, J.; KROPF, P. (2000) *The GIPSY Architecture. Proceedings of the Distributed Computing on the Web*. Quebec City, Canada, 2000 , 1830:144–153.
- (PAQUET e PLAICE, 1998) PAQUET, J.; PLAICE, J. (1998) *Dimensions and Functions as Values. Proceedings of the Eleventh International Symposium on Lucid and Intentional Programming*, Sun Microsystems, Palo Alto, California, EUA, 1998.
- (PERNOD et al., 2010) PERNOD, E.; SERMESANT, M.; RELAN, J.; DELINGETTE, H. (2010) *Interactive Real Time Simulation of Cardiac Radio-Frequency Ablation. Eurographics Workshop on Visual Computing for Biology and Medicine*. P. 91-98.
- (POLYSa et al., 2004) POLYS, N.F.; BOWMAN, D.A.; NORTH, C.; LAUBENBACHER, R.C.; DUCA, K.(2004) *PathSim Visualizer: An Information-Rich Virtual Environment Framework for Systems Biology. Proceeding of the Ninth International Conference on 3D Web Technology*, Web3D 2004, Monterey, California, USA, 5–8 April 2004. New York: ACM Press, 2004; P. 7–14.
- (POLYSb et al., 2004) POLYS, N.F.; BOWMAN, D.A.; NORTH, C.(2004) *Information-Rich Virtual Environments: Challenges And Outlook. Paper presented at the NASA Virtual Iron Bird Workshop*, NASA Ames.
- (PREECE et al., 1994) PREECE, J.; ROGERS, Y.; SHARP, H.; BENYON, D.; HOLLAND, S.; CAREY, T. (1994). *Human-Computer Interaction*. Ed Addison-Wesley.
- (PREECE et al., 2008) PREECE, J.; ROGERS, Y.; SHARP, H. (2008) **Design de Interação – Além da interação homem-computador**. Ed. Bookman, P. 29-31. 2008.
- (QUINN et al., 2003) QUINN, F.; KEOGH, P.; MCDONALD A.; HUSSEY, D.(2003) *A Study Comparing the Effectiveness of Conventional Training and Virtual Reality Simulation. European Journal of Dental Education*, P. 164-169. ISSN: 1396-5883. 2003.
- (RAHMAN, 2007) RAHMAN, M.M. (2007) *An Application Of Prototyping Models For Web Application Development Using Php Framework Cakephp*, Independent University, Bangladesh, 2007.
- (REGAN e GREGORY, 1995) REGAN, L.; GREGORY, M. (1995) *Flux Analysis Of Microbial Metabolic Pathways Using A Visual Programming Environment. Journal of Biotechnology 42*, P. 151-161.
- (REXILIUS et al., 2004) REXILIUS, J.; SPINDLER, W.; JOMIER, J.; KÖNIG, M.; HAHN, H. K.; LINK, F.; PEITGEN, H.(2004) *A Framework for Algorithm Evaluation and Clinical Application Prototyping using ITK. MICCAI Workshop 2005*.
- (RODRIGUES et al., 2008) RODRIGES, H.F.; DIAS, G.P.; GOMES, R.G.S.; MORAES, R.M.; MACHADO, L.S. (2008) *Uma Interface Visual para o Ensino de Estatística Por meio de da Calculadora Estatística. Proceedings of the International Conference on Engineering and Technology Education (Intertech 2008)*. Santos, Brasil. P. 554-558.
- (ROSEMBLUM et al., 1998) ROSEMBLUM, L., BURDEA, G.; TACHI, A.(1998) *VR Reborn. IEEE Computer Graphics and Applications Nov/Dez*, P. 21-23, 1998.

- (ROSEN *et al.*, 1996) ROSEN, J. M.; SOLTANIAN, H.; REDETT, R. J.; LAUB, D. R. (1996) *Evolution of Virtual Reality from Planning to Performing Surgery*. **IEEE Engineering in Medicine and Biology Magazine**, P. 16-22, 1996.
- (ROUSSOU, 2009) ROUSSOU, M. (2009) *A VR Playground for Learning Abstract Mathematics Concepts*. **IEEE Computer Graphics & Applications**, N. 1, 2009, 29:82-85.
- (SALES e MACHADO, 2010) SALES, B.R.A.; Machado, L.S. (2010) *Interactive Collaboration for Virtual Reality Systems related to Medical Education and Training. Proceedings of the 6th International Conference on Technology and Medical Sciences (TMSI)*. Porto. 2010.
- (SALES *et al.*, 2010) SALES, B.R.A.; MACHADO, L.S.; MORAES, R.M. (2010) Colaboração Interativa para Sistemas de Realidade Virtual voltados ao Ensino e Treinamento Médico. **Anais do XII Congresso Brasileiro de Informática em Saúde (CBIS 2010)**, Porto de Galinhas, Brasil.CD-ROM.
- (SALMINEN e TOMPA, 2011) SALMINEN, A.; TOMPA, F. (2011) **Communicating with XML**. Editora Springer, 2011, P. 11-27.
- (SANNER *et al.*, 2002) SANNER, M.F.; STOFFLER, D.; OLSON, A.J. (2002) ViPER, a *Visual Programming Environment for Python*. **Proceedings of the 10th International Python Conference**. P. 103–115, 2002.
- (SANTOS e MACHADO, 2009) SANTOS, A.D.; MACHADO, L.S. (2009) Realidade Virtual Aplicada ao Ensino de Medicina: Taxonomia, Desafios e Resultados. **Anais do Workshop de Realidade Virtual e Aumentada**, Santos, Brasil.
- (SANTOS *et al.*, 2006) SANTOS, I.H.F.; RAPOSO, A.B.; GATTASS, M. (2006) *A Service-Oriented Architecture for a Collaborative Engineering Environment in Petroleum Engineering*. **Proceedings of Virtual Concept 2006**. Playa Del Carmen, Mexico, November 26th – December 1st, 2006.
- (SANTOS *et al.*, 2010) SANTOS, A.D.; MACHADO, L.S.; MORAES, R.M.; GOMES, R.G.S. (2010) Avaliação Baseada em Lógica Fuzzy para um Framework Voltado a Construção de Simuladores Baseados em RV. **Proceedings of the XII Symposium on Virtual and Augmented Reality (SVR)**. Natal, Brasil, P. 194-202.CD-Rom.
- (SCHNEIDER e GIBSON, 2008) SCHNEIDER, R.D.; GIBSON, D. (2008) **Microsoft SQL Server 2008 All-in-One Desk Reference for Dummies**. Wiley Publishing Inc., P. 5-45.
- (SCRIVENER, 1993) SCRIVENER, A.B.(1993) *The Impact of Visual Programming in Medical Research*. **Proceedings of Medicine Meets Virtual Reality II (MMVR) -- Interactive Technology and Healthcare: Visionary Applications for Simulation, Visualization and Robotics**, Califórnia. EUA. IOPress.
- (SHAW *et al.*, 1993) SHAW, C.; GREEN, M.; LIANG, J.; SUN, Y. (1993) *Decoupled Simulation In Virtual Reality With The MR Toolkit*. **ACM Transactions on Information Systems (TOIS)**, N. 3, 11:287-317, Julho 1993.
- (SHNEIDERMAN e PLAISANT, 2010) SHNEIDERMAN, B.; PLAISANT, C (2010) **Designing the User Interface - Strategies for Effective Human-Computer Interaction**. 5th Edition. Ed Pearson. P. 231.
- (SILVA e LUCENA, 2003) SILVA, V.; LUCENA, C. (2003) *From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language*, **Technical**

Report CS2003-10, School of Computer Science, University of Waterloo, Canada, 2003.

- (STOLL *et al.*, 1996) STOLL, T.S.; RUFFLEUX, P.A.; VON STOCKAR, U.; MARISON, I.W. (1996) *Development Of An On-Line Control System For The Cultivation Of Animal Cells In A Hollow-Fiber Reactor Using Flow Injection Analysis And A Visual Programming Language*. **Journal of Biotechnology**, 51:37-48.
- (STONE *et al.*, 2011) STONE, R.J.; PANFILOV, P.; SHUKSHUNOV, V.(2011) *Evolution of Aerospace Simulation: From Immersive Virtual Reality to Serious Games*. **Proceedings of 5th International Conference on Recent Advances in Space Technologies**. P. 655-662. Istanbul.
- (TAKATSUKA e GAHEGAN, 2002) TAKATSUKA, M.; GAHEGAN, M.(2002) *Geovista Studio: a Codeless Visual Programming Environment For Geoscientific Data Analysis And Visualization*. **Computers & Geosciences**, P. 1133, N. 28. Pensilvânia, EUA.
- (TIDWELL, 2006) TIDWELL, J.(2006) **Designing Interfaces – Patterns for Effective Interaction Design**. Ed. O'Reilly. P. 89.
- (TRAMBEREND, 2001) TRAMBEREND, H.(2001) *A Display Device Abstraction for Virtual Reality Applications*. **Proceedings of AFRIGRAPH '01**, Capetown, South Africa, November 2001.
- (TRAMBEREND *et al.*, 1999) TRAMBEREND, H. (1999) *Avocado: A Distributed Virtual Reality Framework*. **Proceedings of the IEEE Virtual Reality**, P.14, March 13-17, 1999.
- (TURNER-BOWKER *et al.*, 2011) TURNER-BOWKER , D.M.; SARIS-BAGLAMA, R.N.; SMITH, K. J. ; DEROSA, M. A.; PAULSEN, C.A.; HOGUE, S.J. (2011) **Telemedicine and E-Health**. January/February 2011, 17:40-45.
- (VICECONTI *et al.*, 2007) VICECONTI, M.; ZANNON, C.; TESTI, D.; PETRONE, M.; PERTICONI, S.; QUADRANI, P.; TADDEI, F.; IMBODEN, S.; CLAPWORTHY, G. (2007) *The Multimod Application Framework: A Rapid Application Development Tool For Computer Aided Medicine*. **Proceedings of Computer Methods and Programs In Biomedicine**. 85:138–151.
- (VON OTTER *et al.*, 2011) A.M. VON OTTER, M. KARLSTEEN, N. BOKISCH, A. FORSBERG. (2011) *3D Visualization as a Possibility for Interactive Learning within the Sports Realm*. **Proceedings on 3rd International Conference on Education and New Learning Technologies (EDULEARN11)**. P. 5393-5397. ISBN: 978-84-615-0441-1.
- (WEBER e VOSGERAU, 2011) WEBER, A.M.; VOSGERAU, G. (2011) *Is Cognition Grounded in Action? Conceptual Considerations and Empirical Evidence*. **Proceedings of European Perspectives on Cognitive Science**.
- (WHELAN *et al.*, 2004) WHELAN, P.F.; SADLEIR, R.J.T.; GHITA, O.(2004) *NeatVision: Visual Programming for Computer-aided Diagnostic Applications*. **Informatics in Radiology (INFORAD)**, Published online 10.1148. P.1779-1789.
- (WIKLUND *et al.*, 2009) WIKLUND, J.; NICOLAS, V.; RONDAO, P.; ANDERSSON, M.; KNUTSSON, H.(2009) *T-flash: Tensor Visualization in Medical Studio*. **Book Chapter: Tensors in Image Processing and Computer Vision**. P. 455-466.

- (WOLF *et al.*, 2005) WOLF, I.; VETTER, M.; WEGNER, I.; BOTTGER, T.; NOLDEN, M.; SCHOBINGER, M.; HASTENTEUFEL, M.; KUNERT, T.; MEINZER, H.P. (2005) *The Medical Imaging Interaction Toolkit. Proceedings of I Medical image analysis 2005*, 9:594-604
- (YOUNG *et al.*, 1995) YOUNG, M.; ARGIRO, D.; KUBICA, S. (1995) *Cantata: Visual Programming Environment for the Khoros System. IEEE Computer Graphics and Applications*. Edição de maio de 1995.
- (ZAJTCHUK e SATAVA, 1997) ZAJTCHUK, R.; SATAVA, R.M.(1997) *Medical Applications of Virtual Reality. Communications of the ACM*, New York, N. 9, 40:63-64. 1997.

Apêndices

APÊNDICE A – Avaliação das necessidades dos profissionais de saúde

Universidade Federal da Paraíba



UFPB

Programa de Pós-Graduação em Informática

Laboratório de Tecnologias para Ensino Virtual e Estatística

Questionário de Avaliação



1ª parte - Sócio-Econômico:

01. Qual sua faixa etária?

- (A) Menor que 17 anos
- (B) Entre 17 e 27 anos
- (C) Entre 28 e 37 anos
- (D) Acima de 38 anos

02. Qual sua formação acadêmica?

- (A) Técnico
- (B) Graduado
- (C) Mestre
- (D) Especialista
- (E) Doutor(a)
- (F) Outro: _____

03. Ano de formação: _____

04. Qual sua área de atuação?

- (A) Biomedicina
- (B) Enfermagem
- (C) Fisioterapia
- (D) Fonoaudiologia
- (E) Medicina
- (F) Nutrição
- (G) Odontologia
- (H) Outro: _____

05. Atividade atual:

- (A) Estudante
- (B) Professor
- (C) Profissional Liberal

06. Instituição da sua atividade:

- (A) Pública
- (B) Privada

07. Descreva os materiais utilizados nas disciplinas de anatomia da sua instituição?

- (A) Eram somente materiais sintéticos

- (B) Eram somente materiais naturais
- (C) Eram materiais sintéticos e naturais

08. Como era (é) o acesso aos materiais da aula de anatomia na sua instituição?

- (A) Somente nas aulas
- (B) Raramente disponíveis para o estudo fora da aula
- (C) Sempre disponível quando precisava, mas em mau estado de manutenção
- (D) Sempre disponível quando precisava e em bom estado de manutenção
- (E) Levava até para estudar em casa

09. Como eram os materiais utilizados nas disciplinas de práticas cirúrgicas de sua instituição?

- (A) Somente materiais defasados
- (B) Materiais novos e defasados
- (C) Somente materiais novos

10. Como era o acesso aos materiais técnico-cirúrgicos na sua instituição?

- (A) Somente nas aulas
- (B) Raramente disponíveis para o estudo fora da aula
- (C) Sempre disponível quando precisava, mas em mau estado de manutenção
- (D) Sempre disponível quando precisava e em bom estado de manutenção
- (E) Levava até para estudar em casa

11. Você tem conhecimento em informática?

- (A) Sim
- (B) Não

12. Possui acesso ao computador?

- (A) Não possui acesso ao computador
- (B) Possui computador em casa

- (C) Acesso da escola/faculdade
- (D) Acesso do trabalho
- (E) Acesso em *lan houses* / *cybercafés*
- (F) Outros: _____

13. Possui acesso à Internet?

- (A) Não possuo acesso ao Internet
- (B) Posso computador em casa
- (C) Acesso da escola/faculdade
- (D) Acesso do trabalho
- (E) Acesso em *lan houses* / *cybercafés*
- (F) Outros: _____

2ª parte – Conhecimentos em Geral:

14. Você já utilizou o computador como ferramenta do seu trabalho/ estudo?

- (A) Sim
- (B) Não

15. Você já trabalhou com algum tipo de programa com visualização 3D (aparelho de ultrassonografia, ressonância, ou algo do tipo)

- (A) Sim
- (B) Não

---- Se SIM passe para a questão 16 ----

---- Se NÃO passe para a questão 17 ----

16. Qual?

- (A) Aparelho de ultrassonografia
- (B) Aparelho de ressonância
- (C) Aparelho de ecocardiografo
- (D) Outros: _____

17. Você já utilizou algum programa de computador para treinamento cirúrgico?

- (A) Sim
- (B) Não

18. Você utilizaria algum programa em 3D, no computador, que auxiliasse nos estudos em anatomia?

- (A) Sim, independente de qual programa fosse
- (B) Sim, apenas se a programa fosse de fácil uso
- (C) Não
- (D) Não, os estudos em anatomia devem ser exclusivamente práticos

19. Você utilizaria algum programa em 3D, no computador, que treinasse algum procedimento técnico-cirúrgico de sua área?

- (A) Sim, independente de qual programa fosse
- (B) Sim, apenas se a programa fosse de fácil uso
- (C) Não, procedimentos técnico-cirúrgicos devem ser exclusivamente práticos
- (D) Não. Por quê? _____

20. Você gostaria de criar, num programa de computador, suas próprias visualizações anatômicas em 3D?

- (A) Sim, independente de qual programa fosse
- (B) Sim, apenas se a programa fosse de fácil uso
- (C) Não, procedimentos técnico-cirúrgicos devem ser exclusivamente práticos
- (D) Não. Por quê? _____

21. Você gostaria de criar, num programa de computador, seus próprios procedimentos técnico-cirúrgicos para treiná-los?

- (A) Sim, independente de qual programa fosse
- (B) Sim, apenas se a programa fosse de fácil uso
- (C) Não
- (D) Não, procedimentos técnico-cirúrgicos devem ser exclusivamente práticos

22. Você teria interesse de continuar ajudando nesse projeto?

- (A) Sim
Nome: _____
Email: _____
- (B) Não

APÊNDICE B – Questionário sobre protótipo visual do CyberMedVPS

Universidade Federal da Paraíba



Programa de Pós-Graduação em Informática

Laboratório de Tecnologias para Ensino Virtual e Estatística



Essa é uma entrevista anônima. Por favor, responda essas questões o melhor que puder. Suas respostas ficarão privadas e apenas serão usadas para a pesquisa proposta. Se em qualquer momento se sentir desconfortável ou não quiser continuar a pesquisa você pode abandonar suas respostas e sair sem punições. Quando finalizar, por favor, devolva o questionário ao pesquisador.

A-Informações sobre o usuário

1. Qual é a sua idade?
 - a. ☐ Menos que 20 anos.
 - b. ☐ Entre 21 e 29 anos.
 - c. ☐ Entre 30 e 40 anos.
 - d. ☐ Entre 40 e 50 anos.
 - e. ☐ Entre 50 e 60 anos.
 - f. ☐ Mais de 60 anos.
2. Qual é o seu gênero?
☐ Masculino. ☐ Feminino.
3. Você é:
 - a. ☐ Professor.
 - b. ☐ Profissional de saúde.
 - c. ☐ Estudante da área de saúde.
 - d. ☐ Outro: _____
4. Qual é a sua área de especialidade?

5. Você tem computador em casa?
☐ Sim. ☐ Não.
6. Em média, quantas horas, por semana, você usa o computador **apenas** para fins não-profissionais?
 - a. ☐ Mais do que 15 horas.
 - b. ☐ Entre 12 e 15 horas.
 - c. ☐ Entre 9 e 12 horas.
 - d. ☐ Entre 6 e 9 horas.
 - e. ☐ Entre 3 e 6 horas.
 - f. ☐ Menos de 3 horas.
7. Em média, quantas horas, por semana, você usa o computador **apenas** para o trabalho?
 - a. ☐ Mais do que 15 horas.
 - b. ☐ Entre 12 e 15 horas.
 - c. ☐ Entre 9 e 12 horas.
 - d. ☐ Entre 6 e 9 horas.
 - e. ☐ Entre 3 e 6 horas.
 - f. ☐ Menos de 3 horas.
8. Qual o sistema que você utiliza?
 - a. ☐ Windows.
 - b. ☐ Mac OSX.
 - c. ☐ Linux.
 - d. ☐ Outros: _____
9. Você já programou no computador?
☐ Sim ☐ Não

B- Informações sobre impressões do CyberMedVPS

10. Você já tinha usado um sistema como o CyberMedVPS antes (interface baseada em fluxogramas)?
() Sim. () Não.

11. Você se sentiu confortável ao usar o sistema CyberMedVPS?
() Sim. () Não.

12. Você estaria disposto(a) a utilizar um sistema como CyberMedVPS se ele auxiliasse no seu trabalho (carreira)?
() Sim. () Não.

13. Você conseguiu entender a função dos componentes (caixas) do CyberMedVPS?
() Sim. () Não.

14. Você conseguiu ver as cores? Se não, explique qual(is) objetos você não conseguiu ver.
() Sim.
() Não: _____

15. A função das cores nos componentes é:
() Meramente ilustrativa.
() Ajuda a diferenciar cada um.
() Relacionada com a sua função.
() Não tem função.

16. Você entendeu as funções das duas caixas de textos (Ações e Dicas), localizadas no rodapé da janela?
() Sim. () Não.

17. A caixa na parte inferior direita, referente às dicas, ajudou?
a. () Sim.

b. () Não.
c. () Eu não usei.

18. A caixa na parte inferior esquerda, referente às ações, ajudou no processo de compreensão das suas ações no programa?
a. () Sim.
b. () Não.
c. () Eu não usei.

19. Você conseguiu finalizar a aplicação?
() Sim. () Não.

20. Você conseguiu relacionar os nomes dos componentes, com suas ações realizadas?
() Sim. () Não.

21. Você usou as funções de fazer e desfazer (as setas em azul)?
() Sim. () Não

22. Você achou o tutorial útil? Se não achou, por favor, explique abaixo:
() Sim.
() Eu não usei.
() Não: _____

Sugestões: _____

APÊNDICE C – Tutorial de configuração e execução do CyberMed no Windows

Environment Configuration Tutorial to CyberMed

In CyberMed migration to the Windows OS, it was decided that to be possible to execute this set of libraries have the environment in the following configuration:

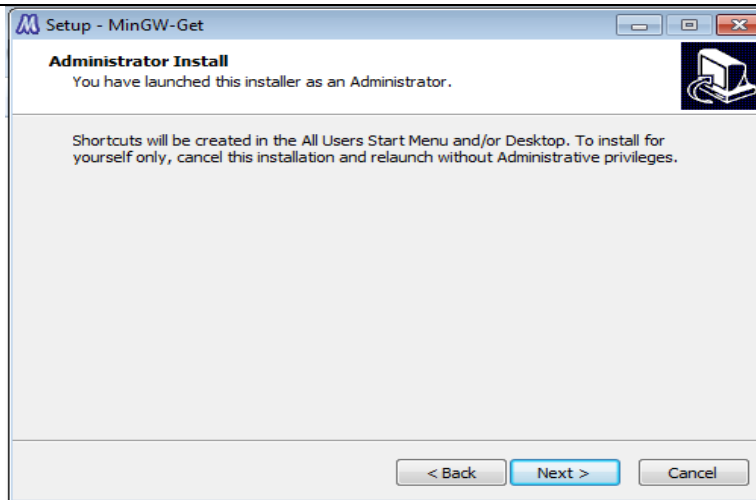
- Have an executor C / C ++ on Windows;
- Emulate the Linux directory structure, specifically in order to use the Makefile file.

1- MinGW Installation:

- a. Make a download: [click here](#) (Has the MinGW and msys);
- b. After the download, execute an installation file called of MinGW-5.1.4.exe, which is into of **Compilador-Gcc-forwindows** package;
- c. The first windows is only to greeting, click the **Next** option:

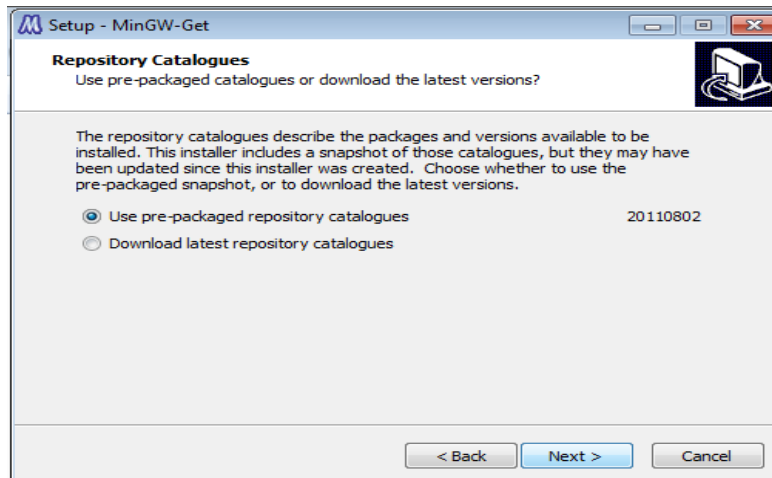


- d. The second window is about the administrator privileges:

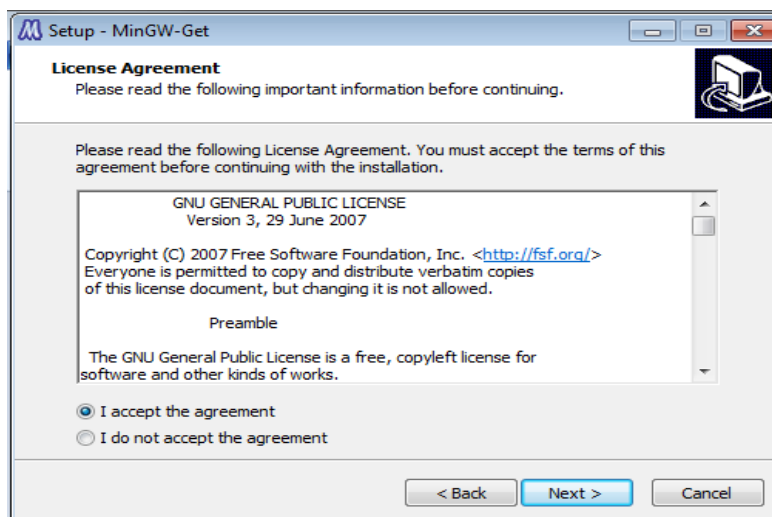


- e. The third window, we have two options: a) Use pre-package repository catalogues and b) Download latest repository catalogues. This tutorial choice **Use pre-package repository catalogues**.

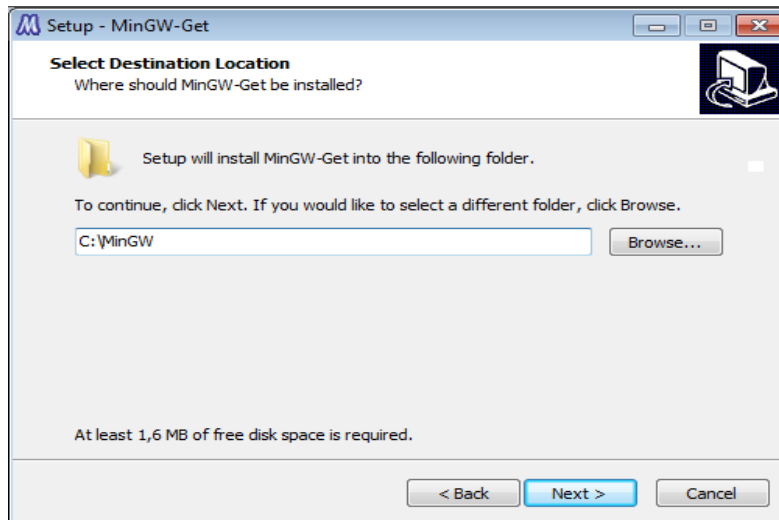
- f. After that click the **Next** option:



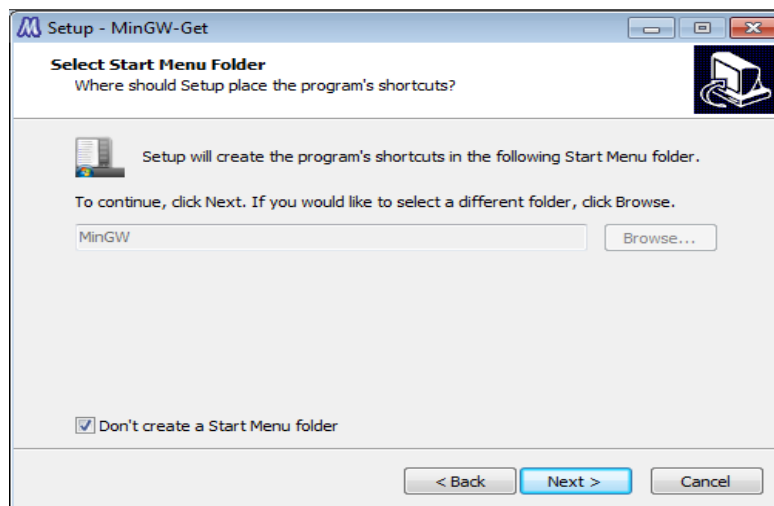
- g. After, it is showed an Agreement. Read and click on **I Agree** option:



h. The next screen asks about the place of MinGW will be saved:

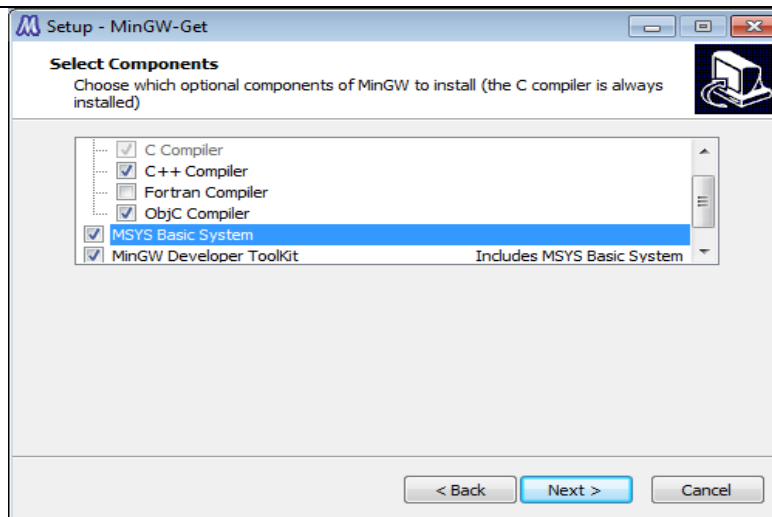


i. Then, ask about if you want a program's shortcut:



j. Then, will be showed a screen which asks about what want we install. Select **MinGW base tools**, **C++ compiler**, **ObjC compiler**, **MSYS Basic System** and **MinGW Developer Toolkit**.

k. After that, click **Next** option:



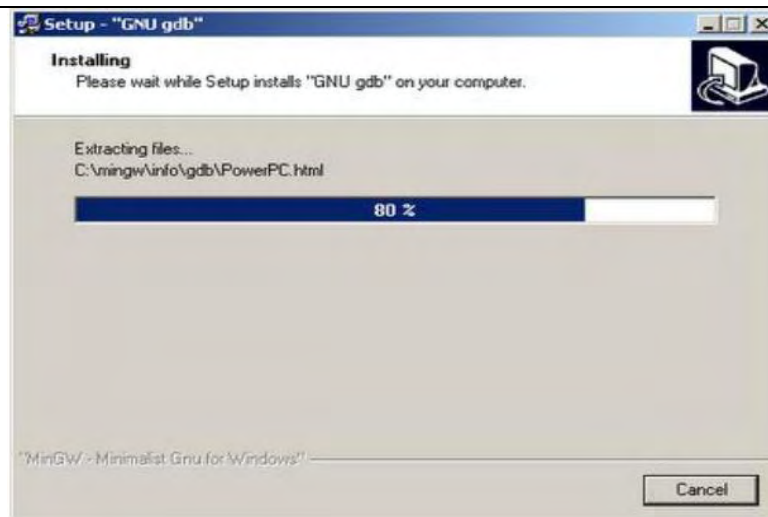
- l. After these configurations, you will click on **Install** option;
- m. When finish the download, you will click on **Next** option and **Finish** option.

2- gdb Installation:

- a. Make a download: click here ;
- b. Execute a file installation called of gdb-5.2.1-1.exe;
- c. Will be display a confirmation screen. Click on **Sim** option;



- d. The installer will open a first greeting screen. Click on the **Next** option;
- e. After that, Will be showed a gdb Agreement. Read and Click on **Yes** option;
- f. Will be showed a message of portability responsible. Click on the **Next** option;
- g. The installer will ask about where would you want install the gdb. Install at same place where the MinGW is;
- h. After that, select a directory and click on the **Next** option. Then, will display a confirmation window. Click on **Install** option and wait an installation be finalized:



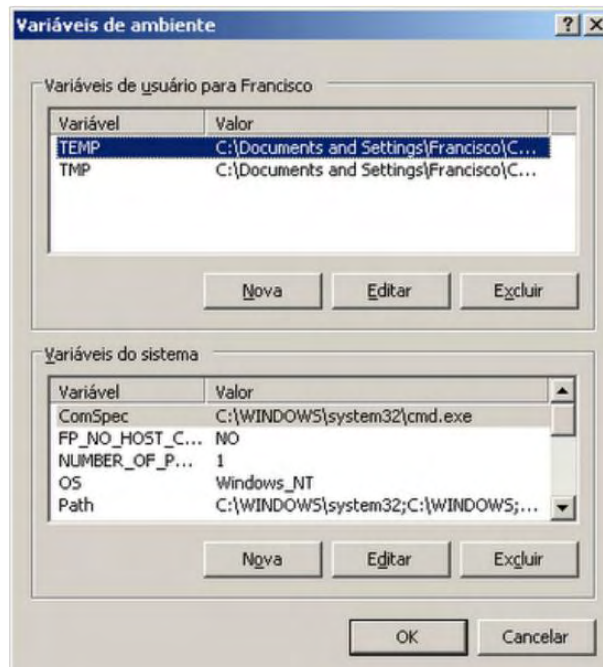
- i. Finished an installation, click on **Finish** option.

3- Environment Variable Configuration

- a. In order to configure an Windows environment variables you need go:

Start -> Control Panel -> System and Security -> System -> Click on the lateral link “Advanced system settings” -> Select a tab “Advanced” -> Click on the “Environment Variables...” tab.

- b. This screen will showed:



- c. Click on the **New** button, at a superior rectangle (user variables for ...);
- d. At **Variable name** section write *MinGW*;
- e. Then, at **Variable value** section, put a location of MinGW until bin directory. This tutorial put

C:/MinGW/bin :



f. Click on **OK** button and click again at superior tab;

g. Create other variable called of MSYS (**Variable name**);

h. At **variable value** area, put a path to MSYS until a *bin* directory. This tutorial put *C:\msys\1.0\bin*;

i. Find another variable called **Path**, but in the rectangle called of **System variables**.

j. After this, click on the **Edit** option;

k. At variable value, **go to the final of the values** and put:

;%MINGW%;%MSYS%

l. Click on **OK** option and again.

OBS: If you want, you can go to Windows Command Prompt and execute an instruction: “gdb – version”, i.e..

4- Posix Installation:

a. Download of POSIX: click here;

b. Install the POSIX and execute;

c. Click on the **Extract** button;

d. Then click on the **Done** button. Will be generated two folders:

- Pre-built: folder with precompiled libraries.

- Pthread: folder with a source code with Pthread library;

e. Enter at **Pre-built\lib** folder;

f. Copy **all files** to lib folder and paste to *C:\MinGW\lib*;

g. Copy all **.dll** to *C:/WINDOWS/System32* ;

h. Come back to Pre-built folder;

- i. Enter to **include** folder;
- j. Copy **all files** of include folder and paste to *C:\MinGW\include*;

5- OpenGL Installation:

- a. Download of OpenGL: click here;
- b. Unzip a glut-3.7.6-bin.zip file;
- c. Enter in **glut-3.7.6-bin** folder;
- d. Copy a **glut.dll** to *C:\Windows\System32*;
- e. Copy a **glut32.lib** to *C:\MinGW\lib*;
- f. Copy a **glut.h** to *C:\MinGW\include\GL*;

6- Copy:

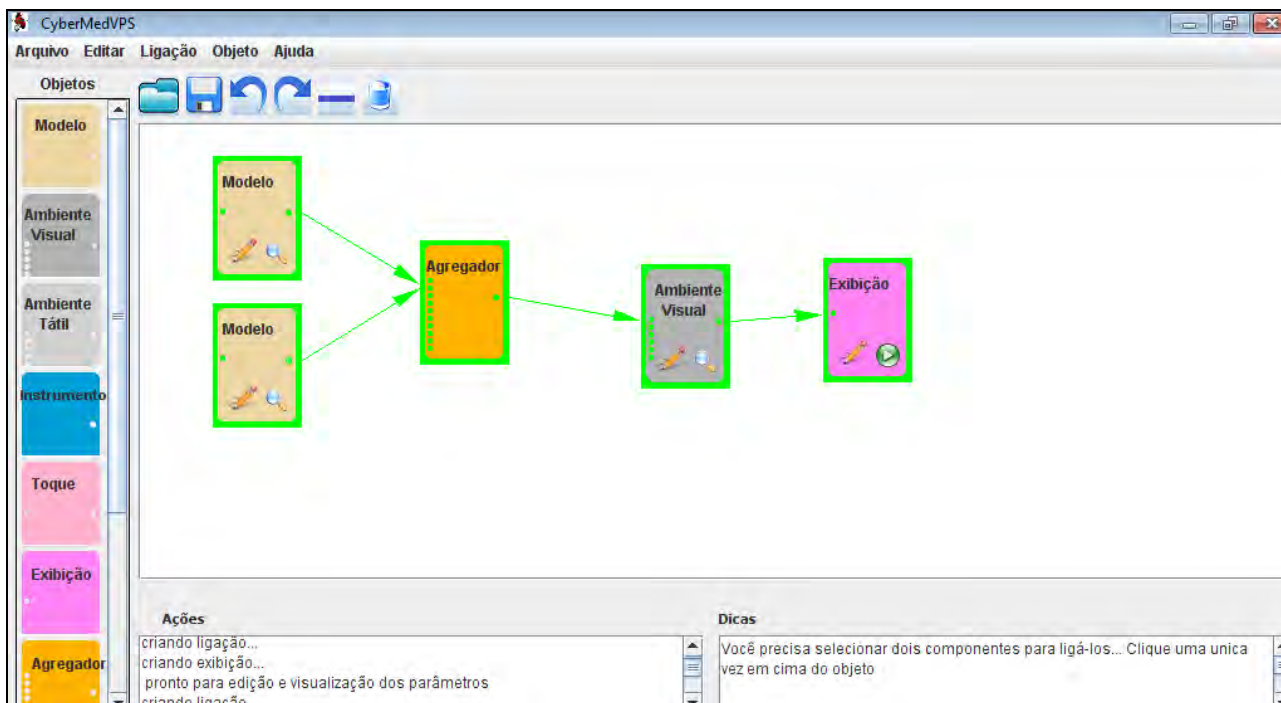
- a. *C:\MinGW\bin\libgcc_s_dw2-1.dll* to *C:\WINDOWS\System32*;
- b. *C:\MinGW\bin\libstdc++-6.dll* to *C:\WINDOWS\System32*;
- c. All **.dll** files of *C:\MinGW\bin* to *C:\WINDOWS\System32*;

OBS: if your MinGW hasn't these dlls, make a download:
https://www.dropbox.com/s/oqr8pedqbkd8nmi/dlls_MinGW.zip

APÊNDICE D – Resultados obtidos com os cenários de validação do CyberMedVPS

Cenário 1: Aplicação de visualização monoscópica de um objeto

a) Fluxograma:



b) XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<component>
  <model id="0">
    <modelName>C:/Documents and Settings/Aline/Meus
documentos/NetBeansProjects/SimpleVisualization/agulha_unida2.wrl</modelName>
    <positionX>1000</positionX>
    <positionY>1000</positionY>
    <positionZ>1000</positionZ>
    <rotationX>1000</rotationX>
    <rotationY>1000</rotationY>
    <rotationZ>1000</rotationZ>
    <translationX>1000</translationX>
    <translationY>1000</translationY>
    <translationZ>1000</translationZ>
    <visibility>0.5</visibility>
  </model>

  <visual_scene>
    <number>1</number>
    <pXi>-1</pXi>
    <pYi>-1</pYi>
```

```

        <pXf>-1</pXf>
        <pYf>-1</pYf>
        <viewType> monoscopic</viewType>
    </visual_scene>
    <exhibition>
        <name>teste</name>
    </exhibition>
</component>

```

c) Aplicação de validação pelo CyberMed:

```

#include <cybermed/CybDataObtainer.h>;
#include <cybermed/cybViewMono.h>;

int main() {
    char *fileName = "C:/Documents and Settings/Aline/Meus
documentos/NetBeansProjects/SimpleVisualization/agulha_unida2.wrl";
    int numIterator = 0;

    int numLayer = 1;
    CybDataObtainer<cybTraits> data(numLayer, numIterator);
    CybParameters *cybCore = CybParameters::getInstance();
    data.loadModel(0, fileName);
    data.startParameters(numLayer);

    cybCore->setColor(0,1,1,0.7,0.5);
    CybViewMono view;
    view.setWindowName("teste");
    view.init();
}

```

d) Makefile para o main.cpp:

```

CC      =    g++
CFLAGS + =    -DNDEBUG -O3
CONFIG + =    console
LFLAGS  =    -LC:/MinGW/lib
LIBS    =    $(LFLAGS) -lcybutil -lcybcore -lcybview
INCLUDES =    -IC:/MinGW/include/cybermed/of

TARGET  =    main.exe
SRCS=    main.cpp
OBJS =    $(SRCS:.cpp=.o)

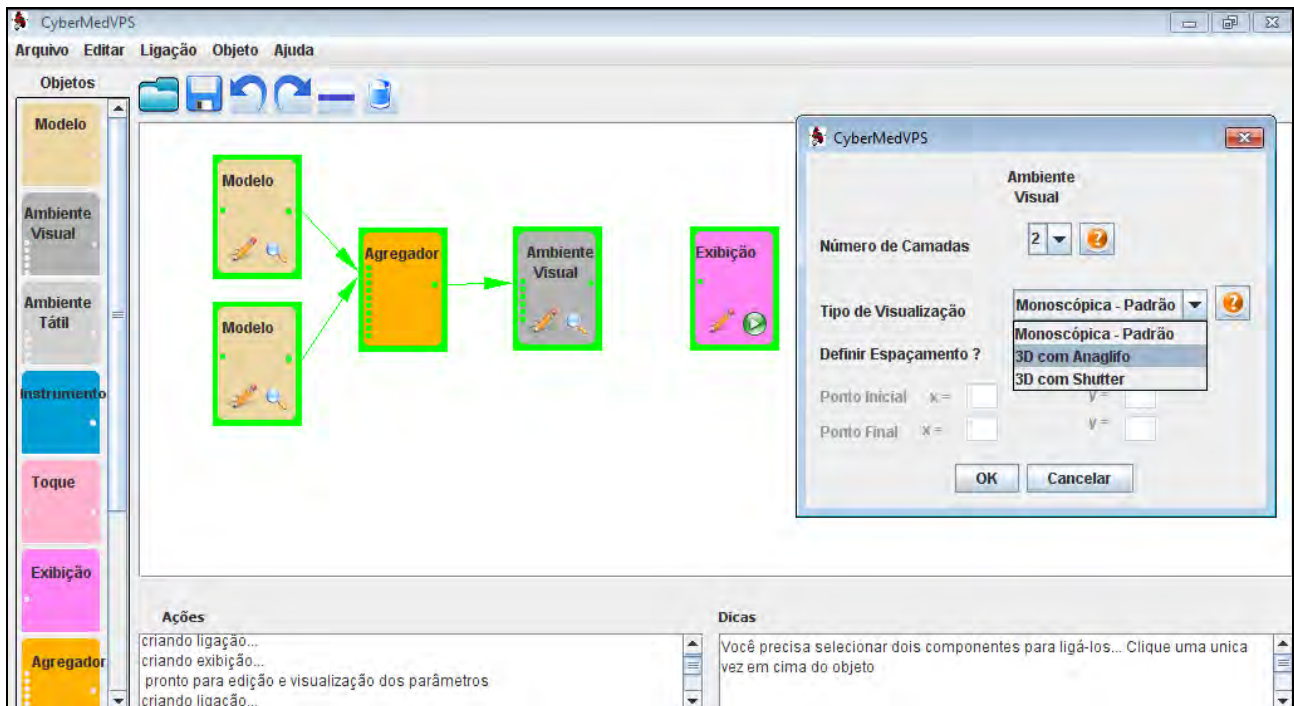
all: $(TARGET)
$(TARGET): $(SRCS)
    $(CC) $(INCLUDES) -o $@ $(SRCS) $(LIBS)

clean:
    rm -f $(OBJS) $(TARGET)

```

Cenário 2 : Aplicação de visualização com estereoscopia

a) Fluxograma:



b) XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<component>
  <model id="0">
    <modelName>C:/Documents and Settings/Aline/Meus
documentos/NetBeansProjects/SimpleVisualization/Bacia - Pele.wrl</modelName>
    <positionX>1000</positionX>
    <positionY>1000</positionY>
    <positionZ>1000</positionZ>
    <rotationX>1000</rotationX>
    <rotationY>1000</rotationY>
    <rotationZ>1000</rotationZ>
    <translationX>1000</translationX>
    <translationY>1000</translationY>
    <translationZ>1000</translationZ>
    <visibility>0.5</visibility>
  </model>
  <visual_scene>
    <number>1</number>
    <pXi>-1</pXi>
    <pYi>-1</pYi>
    <pXf>-1</pXf>
    <pYf>-1</pYf>
    <viewType>stereoscopic </viewType>
  </visual_scene>
```

```
<exhibition>
  <name> validação2</name>
</exhibition>
</component>
```

c) Aplicação em de Validação com o CyberMed:

```
#include <cybermed/CybDataObtainer.h>;
#include <cybermed/cybViewAnaglyph.h>;

int main() {
    char *fileName = "C:/Documents and Settings/Aline/Meus
documentos/NetBeansProjects/SimpleVisualization/Bacia - Pele.wrl";
    int numInterator = 0;

    int numLayer = 1;
    CybDataObtainer<cybTraits> data(numLayer, numInterator);
    CybParameters *cybCore = CybParameters::getInstance();
    data.loadModel(0, fileName);
    data.startParameters(numLayer);

    cybCore->setColor(0,1,1,0.7,0.5);
    CybViewAnaglyph view;
    view.setWindowName("validação2");
    view.init();
}
```

d) Makefile do main.cpp:

```
CC      =    g++
CFLAGS + =    -DNDEBUG -O3
CONFIG + =    console
LFLAGS  =    -LC:/MinGW/lib
LIBS    =    $(LFLAGS) -lcybutil -lcybcore -lcybview
INCLUDES =    -IC:/MinGW/include/cybermed/of

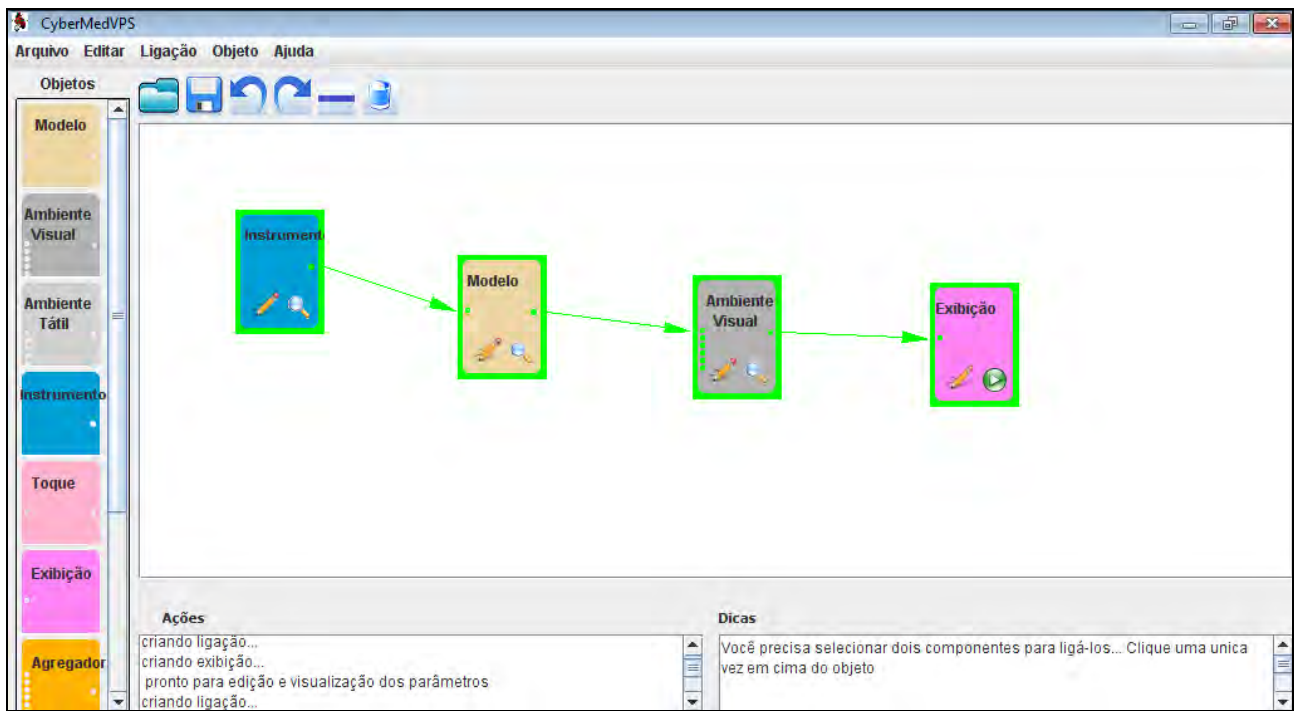
TARGET   =    main.exe
SRCS=    main.cpp
OBJS =    $(SRCS:.cpp=.o)
```

```
all: $(TARGET)
$(TARGET): $(SRCS)
    $(CC) $(INCLUDES) -o $@ $(SRCS) $(LIBS)
```

```
clean:
    rm -f $(OBJS) $(TARGET)
```

Cenário 3: Aplicação de visualização com a participação de um interador

a) Fluxograma:



b) XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<component>
<model id="0">
  <modelName>C:/Users/HP/Documents/monkey.wrl</modelName>
  <positionX>1000</positionX>
  <positionY>1000</positionY>
  <positionZ>1000</positionZ>
  <rotationX>1000</rotationX>
  <rotationY>1000</rotationY>
  <rotationZ>1000</rotationZ>
  <translationX>1000</translationX>
  <translationY>1000</translationY>
  <translationZ>1000</translationZ>
  <visibility>0.5</visibility>
</model>
<visual_scene>
  <number>2</number>
  <pXi>-1</pXi>
  <pYi>-1</pYi>
  <pXf>-1</pXf>
  <pYf>-1</pYf>
  <viewType>monoscopic</viewType>
</visual_scene>
<instrument id="0">
```

```

<typeInt>MOUSE_INTERATOR</typeInt>
<name>C:/Users/HP/Documents/esfera.wrl</name>
</instrument>
<exhibition>
  <name>app</name>
</exhibition>
</component>

```

c) Aplicação em de Validação com o CyberMed:

```

#include <cybermed/CybDataObtainer.h>;
#include <cybermed/cybInterator.h>
#include <cybermed/cybInteratorFactory.h>
#include <cybermed/cybViewMono.h>;

int main(){

char *fileName = "C:/Users/HP/Documents/monkey.wrl";
int numInterator =1;

int numLayer = 1;

CybDataObtainer<cybTraits> data(numLayer, numInterator);
CybParameters *cybCore = CybParameters::getInstance();

CybInterator* interator = CybInteratorFactory::getInterator(MOUSE_INTERATOR);
interator->create(0);

data.loadInteratorModel(0, 0,"C:/Users/HP/Documents/esfera.wrl");
data.startInterator(0, 0);
data.loadModel(0, fileName);
interator->setColor(1, 0, 0, 1);
data.startParameters(numLayer);
cybCore->setColor(0,1,1,0.7,0.5);

CybViewMono view;
view.setWindowName("app");
view.init();
}

```

d) Makefile:

```

CC      =  C:/MinGW/bin/g++.exe
CFLAGS + =  -DNDEBUG -O3
CONFIG + =  console
LFLAGS  =  -LC:/MinGW/lib
LIBS    =  $(LFLAGS) -lcybutil -lcybcore -lcybview
INCLUDES =  -IC:/MinGW/include/cybermed/of

TARGET =  main.exe
SRCS   =  app.cpp
OBJS   =  $(SRCS:.cpp=.o)

all: $(TARGET)

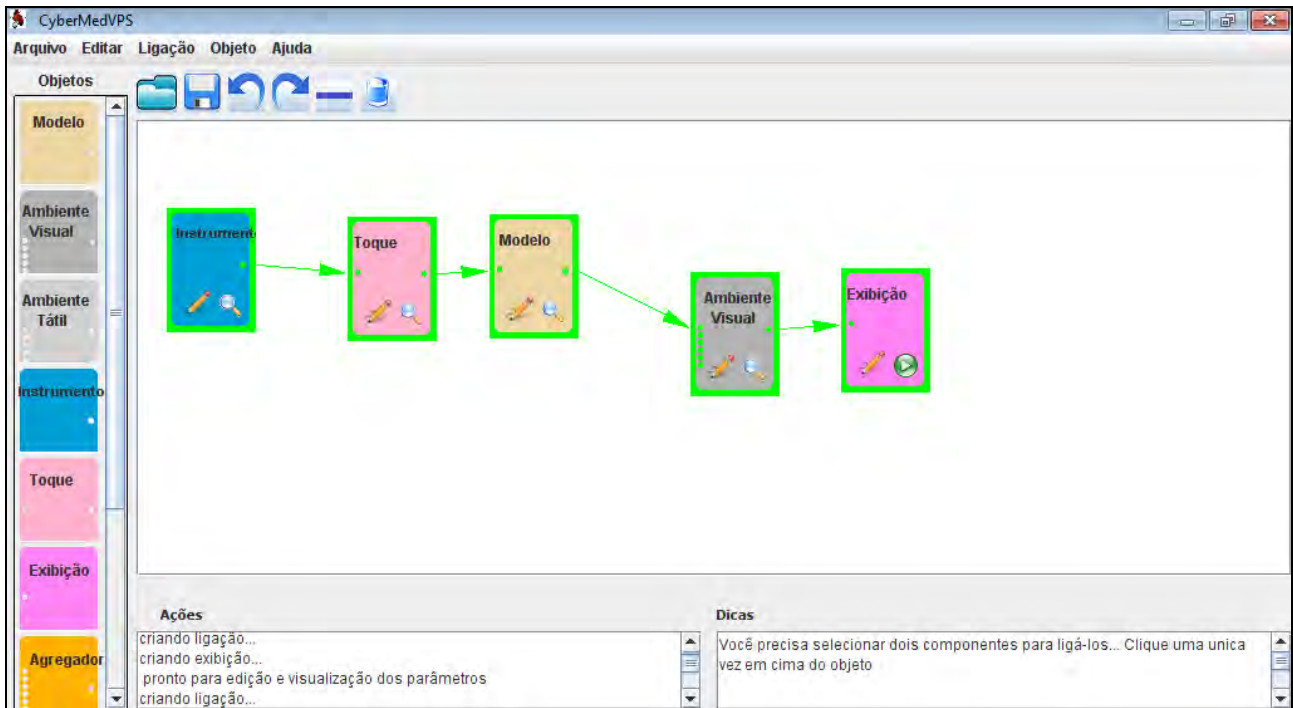
```

```
$(TARGET): $(SRCS)
$(CC) $(INCLUDES) -o $$@ $(SRCS) $(LIBS)

clean:
rm -f $(OBJS) $(TARGET)
```

Cenário 4: Aplicação de visualização com detecção de colisão

a) Fluxograma:



b) XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<component>
<model id="0">
  <modelName>C:/modelos/Bacia_Pele.wrl </modelName>
  <positionX>1000</positionX>
  <positionY>1000</positionY>
  <positionZ>1000</positionZ>
  <rotationX>1000</rotationX>
  <rotationY>1000</rotationY>
  <rotationZ>1000</rotationZ>
  <translationX>1000</translationX>
  <translationY>1000</translationY>
  <translationZ>1000</translationZ>
  <visibility>0.5</visibility>
</model>
<visual_scene>
  <number>2</number>
```

```

    <pXi>-1</pXi>
    <pYi>-1</pYi>
    <pXf>-1</pXf>
    <pYf>-1</pYf>
    <viewType>monoscopic</viewType>
</visual_scene>
<instrument id="0">
    <typeInt> MOUSE_INTERATOR </typeInt>
    <name>C:/modelos/agulha_unida2.wrl</name>
</instrument>
<collision id="0">
    <typeCol>SphereTriangle</typeCol>
    <information>true</information>
</collision>
<exhibition>
    <name>Janela Teste</name>
</exhibition>
</component>

```

c) Aplicação em de Validação com o CyberMed:

app.cpp:

```

#include <cybermed/CybDataObtainer.h>;
#include <cybermed/cybInterator.h>
#include <cybermed/cybInteratorFactory.h>
#include <cybermed/cybViewMono.h>;
#include <cybermed/cybSphereTriangle.h>
#include "collisionCheck.h"

int main(int argc, char** argv){

    char *fileName = "C:/modelos/Bacia_Pele.wrl";
    int numLayer = 1;
    int numInterator =1;

    CybDataObtainer<cybTraits> data(numLayer, numInterator);

    CybInterator* interator =
CybInteratorFactory::getInterator(MOUSE_INTERATOR);
    CybParameters *cybCore = CybParameters::getInstance();

    data.loadModel(0, fileName);

    data.loadInteratorModel(0, 0,"C:/modelos/agulha_unida2.wrl");
    data.startInterator(0,0);
    interator->setColor(0, 1, 0, 1);
    data.startParameters(numLayer);
    cybCore->setColor(0,1,1,0.7,0.5);

    CybViewMono view;
    view.setWindowName("Janela Teste");
    cybCore->createMaterialPropertyContext(numLayer);

```

```
collisionCheck *collCheck = new collisionCheck(0, 0);
collCheck->getCollisionInstance()->init();
collCheck->init();
view.init();
```

collisionCheck.h

```
class collisionCheck : public CybThread {
```

```
private:
```

```
    CybSphereTriangle *collisionObj;
    CybParameters *cybCore;
    int count;
```

```
public:
```

```
    collisionCheck(int layerID, int interatorId){
        collisionObj = new CybSphereTriangle (layerID, interatorId);
```

```
count = 0;
```

```
cybCore = CybParameters::getInstance();
```

```
this->setTime(50);
```

```
    }
```

```
    ~collisionCheck(){
```

```
delete collisionObj;
```

```
}
```

```
void run(){
```

```
    CybThread::lock();
```

```
if(collisionObj->getCollisionStatus()){
```

```
    cybCore->setColor(0,1,1,0,1);
```

```
}else
```

```
    cybCore->setColor(0,1,0,1,1);
```

```
CybThread::unlock();
```

```
}
```

```
CybSphereTriangle *getCollisionInstance() {
```

```
    return collisionObj;
```

```
}
```

```
};
```

d) Makefile:

```
CC      =    C:/MinGW/bin/g++.exe
```

```
CFLAGS + =    -DNDEBUG -O3
```

```
CONFIG + =    console
```

```
LFLAGS  =    -LC:/MinGW/lib
```

```
LIBS    =    $(LFLAGS) -lcybutil -lcybcore -lcybview -lcybcollision
```

```
INCLUDES =    -IC:/MinGW/include/cybermed/of
```

```
TARGET =    main.exe
```

```
SRCS    =    app.cpp
```

```
OBJS    =    $(SRCS:.cpp=.o)
```

```
all: $(TARGET)
```

```
$(TARGET): $(SRCS)
```

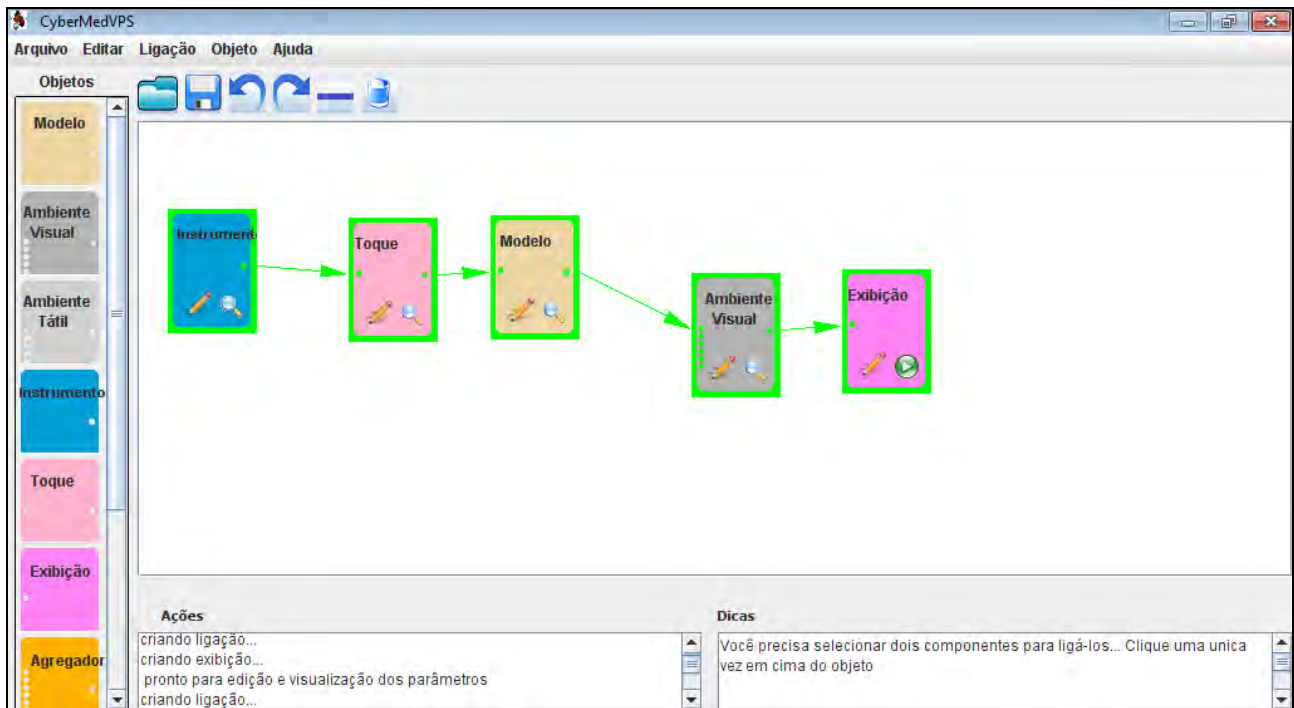
```
$(CC) $(INCLUDES) -o $@ $(SRCS) $(LIBS)
```

clean:

```
rm -f $(OBJS) $(TARGET)
```

Cenário 5: Aplicação de visualização com deformação

a) Fluxograma:



b) XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<component>
<model id="0">
  <modelName>C:/modelos/osso5x.wrl</modelName>
  <positionX>1000</positionX>
  <positionY>1000</positionY>
  <positionZ>1000</positionZ>
  <rotationX>1000</rotationX>
  <rotationY>1000</rotationY>
  <rotationZ>1000</rotationZ>
  <translationX>1000</translationX>
  <translationY>1000</translationY>
  <translationZ>1000</translationZ>
  <visibility>0.5</visibility>
</model>
<visual_scene>
  <number>2</number>
  <pXi>-1</pXi>
```

```

    <pYi>-1</pYi>
    <pXf>-1</pXf>
    <pYf>-1</pYf>
    <viewType>monoscopic</viewType>
</visual_scene>
<instrument id="0">
    <typeInt> MOUSE_INTERATOR </typeInt>
    <name>C:/modelos/agulha_unida2.wrl</name>
</instrument>
<collision id="0">
    <typeCol>SphereTriangle</typeCol>
    <information>true</information>
    <reaction>STIFFNESS</reaction>
    <viscosity>10</viscosity>
    <resistence>12</resistence>
</collision>
<exhibition>
    <name>teste</name>
</exhibition>
</component>

```

c) Aplicação em de Validação com o CyberMed:

```

#include <cybermed/CybDataObtainer.h>;
#include <cybermed/cybInterator.h>
#include <cybermed/cybInteratorFactory.h>
#include <cybermed/cybMassSpring.h>
#include <cybermed/cybViewMono.h>;
#include <cybermed/cybSphereTriangle.h>
#include "collisionCheck.h"

int main(int argc, char** argv){

    char *fileName = "C:/modelos/osso5x.wrl";
    int numLayer = 1;
    int numInterator =1;

    CybDataObtainer<cybTraits> data(numLayer, numInterator);

    CybInterator* interator = CybInteratorFactory::getInterator(MOUSE_INTERATOR);
    CybParameters *cybCore = CybParameters::getInstance();

    data.loadModel(0, fileName);

    data.loadInteratorModel(0, 0,"C:/modelos/agulha_unida2.wrl");
    data.startInterator(0,0);
    interator->setColor(0, 1, 0, 1);
    data.startParameters(numLayer);
    cybCore->setColor(0,1,1,0.7,0.5);

    CybViewMono view;
    view.setWindowName("teste");
    cybCore->createMaterialPropertyContext(numLayer);

```

```

cybCore->setMaterialPropertyValue(0,STIFFNESS, 0.5);
CybSphereTriangle *collisionObj = new CybSphereTriangle(0, interator);

CybMassSpring *deformationObj = new CybMassSpring(collisionObj, DEF_GO_AND_BACK);
deformationObj->update();
deformationObj->init();
view.init();
}

```

d) Makefile:

```

CC      =   C:/MinGW/bin/g++.exe
CFLAGS + =   -DNDEBUG -O3
CONFIG + =   console
LFLAGS  =   -LC:/MinGW/lib
LIBS    =   $(LFLAGS) -lcybutil -lcybcore -lcybview -lcybcollision -lcybdeformation
INCLUDES =   -IC:/MinGW/include/cybermed/of

TARGET =   main.exe
SRCS    =   app.cpp
OBJS    =   $(SRCS:.cpp=.o)

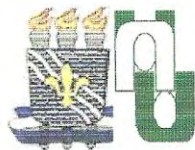
all: $(TARGET)
$(TARGET): $(SRCS)
    $(CC) $(INCLUDES) -o $@ $(SRCS) $(LIBS)

clean:
    rm -f $(OBJS) $(TARGET)

```

Anexo

ANEXO A - Aprovação do projeto CyberMedVPS no CEP do HULW



UNIVERSIDADE FEDERAL DA PARAIBA - UFPB
HOSPITAL UNIVERSITÁRIO LAURO WANDERLEY - HULW
**COMITÊ DE ÉTICA EM PESQUISA COM SERES
HUMANOS - CEP**

CERTIDÃO

Com base na Resolução nº 196/96 do CNS/MS que regulamenta a ética da pesquisa em seres humanos, o Comitê de Ética em Pesquisa do Hospital Universitário Lauro Wanderley - CEP/HULW, da Universidade Federal da Paraíba, em sua sessão realizada no dia 26/10/2010, após análise do parecer do relator, resolveu considerar APROVADO o projeto de pesquisa intitulado CYBERMEDVPS - PROGRAMAÇÃO VISUAL PARA UM FRAMEWORK DE DESENVOLVIMENTO DE APLICAÇÕES MÉDICAS BASEADAS EM REALIDADE VIRTUAL. Protocolo CEP/HULW nº. 614/10, Folha de Rosto nº 373810, CAAE nº 0488.0.126.000-10, da pesquisadora ALINE MARQUES DE MORAIS.

Ao final da pesquisa, solicitamos enviar ao CEP/HULW, uma cópia desta certidão e da pesquisa, em CD, para emissão da certidão para publicação científica.

João Pessoa, 27 de outubro de 2010.

Iaponira Cortez Costa de Oliveira
Coordenadora do Comitê de Ética
em Pesquisa - CEP/HULW

Profª Drª Iaponira Cortez Costa de Oliveira
Coordenadora do Comitê de Ética em Pesquisa-HULW

Endereço: Hospital Universitário Lauro Wanderley-HULW - 4º andar. Campus I - Cidade Universitária.
Bairro: Castelo Branco - João Pessoa - PB. CEP: 58051-900 CNPJ: 24098477/007-05
Fone: (83) 32167302 — Fone/fax: (083)32167522 E-mail - cepulw@hotmail.com

Figura 52. Aprovação do Comitê de ética do HUWL.