

**Universidade de São Paulo**  
**Escola de Artes, Ciências e Humanidades**

## **Relatório Final**

### **Instanciação automatizada de um *framework* de Realidade Virtual para treinamento médico**

Modalidade: PIBIC/CNPq

Aluna  
Daniela Oliveira Bertolli

Orientadora  
Fátima de Lourdes dos Santos Nunes Marques

Agosto 2010

## Resumo

Este relatório consiste em uma descrição detalhada das atividades realizadas ao longo do projeto de Iniciação Científica “Instanciação automatizada de um framework de realidade virtual para treinamento médico”.

Ele apresenta uma revisão sistemática sobre *frameworks* de realidade virtual e aplicações para auxílio no treinamento médico. Também são apresentados o *framework ViMeT*, um *framework* voltado para a geração de aplicações de realidade virtual para treinamento médico, e sua ferramenta para instanciação automatizada *ViMeT Wizard*.

Ao final é apresentado o trabalho realizado ao longo do projeto, mostrando as modificações feitas na ferramenta *ViMeTWizard* e as novas funcionalidades criadas para ela.

## Sumário

1. Introdução.....	4
1.1 Justifi ca ti vas.....	5
1.2 Disposi ção do Trabalho.....	6
2. Revisão Sis te máti ca .....	7
3. Me to do lo gi a.....	16
3.1 O Framework <i>ViMeT</i> .....	16
3.2 A fe rra men ta <i>ViMeT Wizard</i> .....	18
3.3 Trabalho Realizado .....	21
3.3.1 Fases do Projeto .....	21
3.3.2 Al te ra ções no Banco de Dados .....	23
3.3.3 Al te ra ções na In te rfa ce Grá fi ca .....	26
3.3.4 Al te ra ções no Códi go.....	35
5. Condu são.....	41
6. Apê n di ces.....	42
Apê n di ce 1 – Pro to co lo de Re vi são.....	42
Apê n di ce 2 – For mu lá ri o de Con du ção de Re vi são.....	45
Apê n di ce 3 – For mu lá ri o de Se le ção dos Es tu dos.....	47
Apê n di ce 4 – For mu lá ri o de Ex tra ção de Da dos .....	49
Apê n di ce 5 – mé to dos adi ci o na dos pa ra a ma ni pu la ção do no vo ban co de da dos.....	50
7. Re fe rên ci as Bi bli o grá fi cas.....	54

## 1. Introdução

A Realidade Virtual (RV) é uma técnica que permite a criação de Ambientes Virtuais (AV) gerados pelo computador que simulam ambientes ou objetos reais do “mundo físico”, e criam no usuário a sensação de estar dentro do ambiente (imersão) (Bastos *et al.* 2005).

Atualmente técnicas de Realidade Virtual (RV) vêm sendo empregadas em muitos campos da tecnologia, como por exemplo, em ferramentas para treinamento médico. Os avanços da Realidade Virtual nesse tipo de ferramenta representam um grande benefício para a Medicina segundo Monteiro *et al.* (2006), pois permitem um maior nível de detalhes em exames, no ensino e treinamento de profissionais e na simulação de procedimentos.

Na simulação de procedimentos, é importante que o usuário tenha a sensação de imersão no ambiente, ou seja, o computador deve detectar as entradas do usuário e modificar o Ambiente Virtual (AV) e as ações sobre ele (Bastos *et al.* 2005). Os dispositivos de entrada podem ser mouse, teclado, equipamento háptico, luvas e outros. E como resultado das modificações feitas pelo usuário pode haver saídas visuais, auditivas e táteis.

Atualmente existem linguagens específicas e bibliotecas que permitem o desenvolvimento de sistemas de RV, como Java, C e *Virtual Reality Modeling Language* (VRML). A linguagem Java permite a criação e manipulação de AVs com a utilização da biblioteca de alto nível e gratuita Java3D. A linguagem C disponibiliza a biblioteca de alto nível *WorldToolkit*, mas apresenta a desvantagem de essa biblioteca ser proprietária, exigindo uma licença para utilizá-la. A linguagem VRML também permite a criação de Ambientes Virtuais, porém não dá suporte a dispositivos não convencionais (Delfino, 2007).

Uma maneira de se criar esses ambientes de Realidade Virtual são os *frameworks*. *Frameworks* orientados a objetos são estruturas de classes interligadas que formam uma

aplicação não-acabada. A extensão (instanciação) dessa estrutura é que produz aplicações, proporcionando o reuso (Freiberger, 2002).

O *framework ViMeT* é um *framework* para a criação de ambientes de realidade virtual. Ele foi desenvolvido usando a linguagem Java e a *Application Programming Interface* (API) Java 3D. Por meio da instanciação de suas classes é possível criar aplicações que simulam órgãos e instrumentos médicos. Essa instanciação pode ser feita de duas maneiras: manualmente ou através da *ViMeT Wizard* (Oliveira, 2007).

Devido ao fato de o *framework ViMeT* ter como função gerar ambientes de treinamento médico para serem utilizados em sala de aula, é interessante uma forma de instanciação automatizada onde o usuário não necessita de grandes conhecimentos de informática e programação para gerar uma simulação. A *ViMeT Wizard* é uma ferramenta construída anteriormente para executar estas tarefas, porém ainda eram necessárias várias correções e um aperfeiçoamento das suas funcionalidades.

O objetivo geral do projeto foi criar uma nova versão da ferramenta *ViMeT Wizard* que facilitasse a implementação de aplicações para treinamento médico utilizando as classes do *framework ViMeT*, além de oferecer maior flexibilidade na criação de aplicações usando o *framework ViMeT*, realizando modificações na estrutura do banco de dados, na interface gráfica, e na estrutura da ferramenta.

## **1.1 Justificativas**

Aplicações médicas de Realidade Virtual voltadas para o aprendizado e treinamento tanto de alunos quanto de profissionais da área representam um grande ganho para a medicina. Elas permitem que um procedimento possua alto grau de realismo e que ele seja feito inúmeras vezes.

A proposta do *framework ViMeT* também é permitir a geração de aplicações que atuem como ambientes de treinamento médico, para serem utilizadas em sala de aula, portanto é interessante uma forma de instanciação automatizada na qual o usuário não necessite de grandes conhecimentos de informática e programação para gerar uma simulação.

O *ViMeT* já possuía uma ferramenta para realizar a instânciação automatizada de suas classes, a *ViMeTWizard*, porém ela ainda necessitava de algumas modificações para adaptar-se a novas funcionalidades. Com as correções e novas funcionalidades implementadas no presente projeto, o *framework* passa a ser mais abrangente, e com a modificação da interface gráfica da ferramenta *Wizard* sua utilização torna mais fácil.

## 1.2 Disposição do Trabalho

O conteúdo desse relatório divide-se da seguinte forma, além da presente seção:

Seção 2 – Revisão Bibliográfica/Revisão Sistemática: apresenta breve descrição sobre o que é uma Revisão Sistemática e apresenta a revisão sistemática realizada no projeto.

Seção 3 – Apresentação do *ViMeT* e da ferramenta *ViMeTWizard*: apresenta descrição sobre o *framework ViMeT* e suas funcionalidades, e sobre a ferramenta de instânciação automatizada denominada *ViMeTWizard*.

Seção 4 – Metodologia: apresenta as atividades realizadas no projeto, incluindo a nova interface gráfica para a ferramenta *ViMeTWizard*, o novo banco de dados do *ViMeT* e as modificações realizadas no código.

Seção 5 – Resultados: apresenta os resultados finais obtidos no projeto.

Seção 6 – Considerações Finais: apresenta as considerações finais sobre as atividades realizadas no projeto.

## 2. Revisão Sistemática

Revisão Sistemática é um tipo de Revisão Bibliográfica documentada, onde cada passo da pesquisa é registrado e é feito seguindo rigorosos critérios. Segundo Kitchenham (2004), a Revisão Sistemática é um meio de avaliar, identificar e interpretar todas as pesquisas relevantes disponíveis em uma determinada área com base em uma questão de pesquisa.

Ela pode ser dividida basicamente em três fases: Planejamento da Revisão, Condução da Revisão e Análise de Resultados. O planejamento da revisão consiste em determinar os dados iniciais que guiarão toda a revisão, como quais são os objetivos da pesquisa, as fontes de busca, as palavras-chave para busca e os critérios de inclusão ou exclusão de documentos na pesquisa. Esses dados são registrados em um documento denominado protocolo de revisão. Durante a fase de condução da revisão as palavras-chave são buscadas em cada fonte determinada no protocolo de revisão, e o resultado de cada busca é registrado em um formulário de condução de revisão, onde são armazenadas a data da busca, a fonte, a palavra-chave utilizada, o título e autor(es) de cada documento encontrado. Então é feita uma análise de cada documento e eles são incluídos ou excluídos de acordo com os critérios previamente estipulados. Os documentos incluídos são reunidos em uma única lista, e então começa a fase de Análise dos resultados. Nessa fase, cada documento é lido e preenche-se para ele um formulário de extração de dados que contém informações sobre ele e um resumo feito pelo pesquisador. Após a extração dos dados é feita uma síntese dos resultados e conclusão sobre a pesquisa realizada.

A Revisão Sistemática desse relatório foi feita com base no protocolo de revisão (Apêndice 1), no formulário de condução da revisão (Apêndice 2), no formulário de seleção dos estudos (Apêndice 3) e no formulário de extração de dados (Apêndice 4) obtidos em (Mafra *et al.* 2006) e em (Oliveira *et al.* 2009).

A condução da revisão sistemática ocorreu de Outubro/2009 a Novembro/2009. Para cada fonte de busca foi registrada uma listagem com todos os documentos obtidos (formulários de condução de revisão), e a partir da análise com base nos critérios

definidos no protocolo, os artigos foram incluídos ou excluídos. Foram também analisadas as referências bibliográficas de cada documento a fim de verificar se havia mais alguma referência relevante a ser incluída na revisão.

A partir dos documentos selecionados na listagem, foram feitos formulários de extração de dados para cada um dos documentos, nos quais foram armazenados dados importantes para a definição do projeto a ser implementado. A seguir é apresentada a síntese geral da análise dos estudos primários.

*Frameworks* orientados a objetos são estruturas de classes interligadas que formam uma aplicação não-acabada. A extensão (instanciação) dessa estrutura é que produz aplicações, proporcionando o reúso (Freiberger, 2002). O reúso não se refere apenas ao aproveitamento de trechos de código, mas também a reutilização dos esforços de todas as fases envolvidas no desenvolvimento de *software* (análise de requisitos, projeto de software, implementação e testes) (Braga, 2002).

Diversos tipos de *frameworks* já foram desenvolvidos para gerar aplicações de Realidade Virtual voltadas para o treinamento médico nas mais diversas áreas. Na área da ortodontia, por exemplo, foi criado um *framework* para simulação de tratamento dentário, voltado tanto para estudantes quanto para profissionais experientes. Seu objetivo é ajudar ortodontistas a prever e a lidar com problemas ortodônticos, além de sugerir soluções para os tratamentos. Sua proposta é ser um software de domínio público com livre acesso ao código. Porém, o *framework* não foi usado sistematicamente e ainda não foi completamente modelado para explorar todos os aspectos mecânicos envolvidos na ortodontia. Também não possui ferramenta de instanciação automatizada (Rodrigues *et al.* 2004). Na área da radiologia foi criado o *Cyclops 3D Environment – C3DE* (Silva *et al.* 2009), que é uma coleção de bibliotecas (*framework*) cujo objetivo também é gerar aplicações médicas. Uma *Workstation 3D* foi criada no “topo” do *framework* utilizando suas classes.

Outro *framework* é apresentado por Ma *et al.* (2008), que foi criado para a avaliação dinâmica de fadiga muscular (ou seja, quando o músculo não é mais capaz de suportar a força de trabalho requerida). Com esse *framework*, a tecnologia de Realidade



Virtual é utilizada para gerar um ambiente de trabalho que pode interagir com interfaces hápticas e com um sistema óptico de captura de movimento. Seu objetivo é avaliar o trabalho humano e prever dinamicamente potenciais riscos de desordem músculo-esquelética, especialmente para fadiga muscular. Também pretende avaliar independentemente o trabalho mecânico humano, induzindo fadiga, conforto e outros aspectos. Nele, há três módulos fundamentais principais: “*Motion Tracking*” (utilizado para digitalizar a movimentação do trabalhador e preparar os dados para futuro processamento), simulação de realidade virtual (responsável por simular o movimento do trabalhador no ambiente de trabalho virtual, prover visualização da simulação da performance do trabalhador e visualização das interações entre trabalhador e os objetos virtuais), e avaliação de trabalho (que avalia objetivamente o processo de trabalho, fazendo o papel de ergonômista). Na Figura 1 é apresentada uma imagem do movimento do trabalhador no ambiente real e sua respectiva representação no mundo virtual.

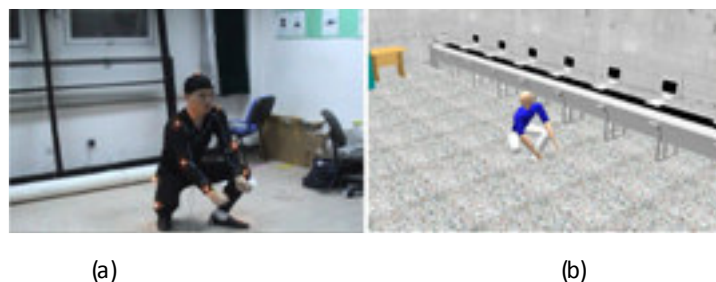


Figura 1 - Resultado do software de simulação: (a) ação do trabalhador real; (b) ação do trabalhador virtual.  
(MA *et al.*, 2008)

Por sua vez, o *framework* *Piavca* é voltado para a criação de personagens virtuais interativos. Ele permite unificar e combinar diversos estilos de animação e de interação com um personagem, utilizando uma única apresentação de função abstrata. Um participante humano pode utilizar simultaneamente os seguintes tipos de interação com um personagem: verbal, não-verbal, movimento e controle. O comportamento do participante humano é capturado com típicos sensores de ambientes de Realidade Virtual imersiva (como um microfone e um “head tracker”). Alguns comportamentos do

personagem virtual são influenciados pelo participante “real”, e outros não exercem influência (como o piscar de olhos, por exemplo) (Gillies, 2008). Na Figura 2 são mostradas telas da interação entre o personagem humano e o personagem virtual.



Figura 2 - Personagem real e personagem virtual interagindo em um ambiente de realidade virtual imersivo.  
(Gillies, 2008)

Uma das dificuldades no uso de *frameworks* está no fato de que o desenvolvedor de aplicações (usuário do *framework*) precisa conhecê-lo a fundo para conseguir utilizá-lo corretamente (Freiberger, 2002). Nesse sentido, é interessante que um *framework* tenha uma ferramenta para instânciação automatizada, facilitando seu uso. Segundo Bastos *et al.* (2005), flexibilidade e simplicidade são qualidades primordiais em um *framework* de Realidade Virtual, pois se for complexo demais, ele pode tornar-se inadequado para projetos mais simples ou pode comprometer a qualidade da aplicação.

Pensando nessas características foi criado o *framework* ViRAL (*Virtual Reality Abstraction Layer*), cujo objetivo é facilitar a criação de aplicações extensíveis operadas por interfaces WIMP (*Windows, Icons, Menus and Pointers*) no desenvolvimento de aplicações de realidade virtual baseadas em componentes gráficos. Além disso, cada aplicação permite simular múltiplos ambientes virtuais, visitados por múltiplos usuários nos mais diversos sistemas de Realidade Virtual. (BASTOS *et al.* 2005) O ViRAL pode ser utilizado como uma biblioteca, ou pode gerar aplicações por meio de uma aplicação padrão. Nessa aplicação padrão há uma janela com diversas abas, uma para cada subsistema, através das quais é possível gerar aplicações. Instanciando e configurando componentes, podem ser criadas aplicações que são capazes de simular diversos mundos virtuais ao mesmo tempo, além de salvar e restaurar ambientes utilizando arquivos. Dentre outras funcionalidades, o ViRAL permite salvar e resumir uma simulação do ponto

em que ela foi interrompida, permite que sejam adicionados novos dispositivos através de *plugins* e possui seus próprios *widgets* para facilitar a criação de interfaces. Na Figura 3 é apresentada uma visão da aplicação padrão e um exemplo de visão de cena.

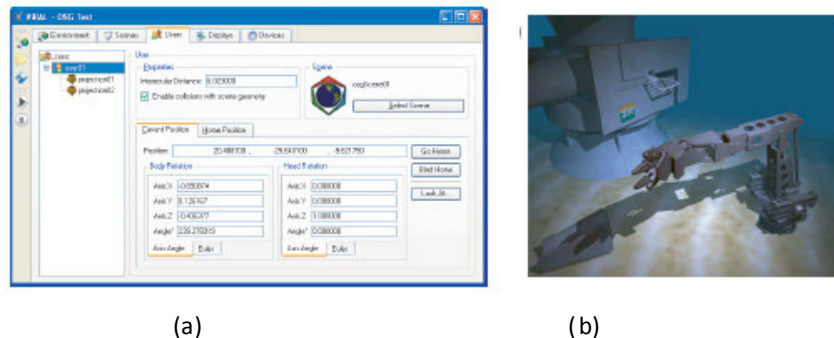
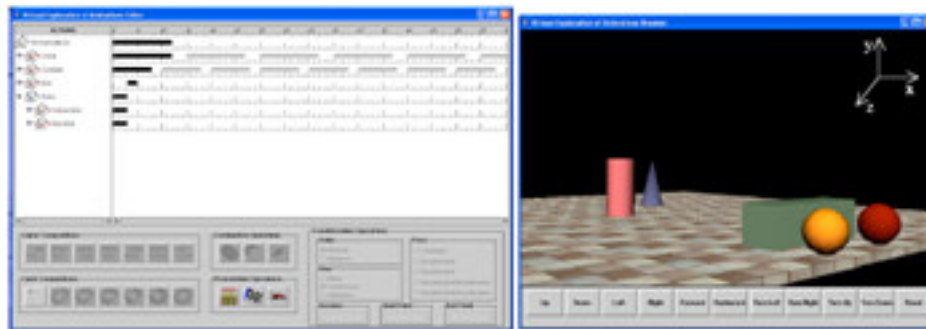


Figura 3 – *Framework ViRAL*: (a) interface da aplicação padrão; (b) exemplo de uma visão de cena. (Bastos et al. 2005)

O *framework* “*Virtual Exploration of Animations*” não é voltado para a área médica, porém possui ferramenta de interface gráfica com o usuário. Ele é composto de tipos de dados abstratos e uma interface de usuário que permite que usuários não experientes controlem, manipulem, analisem e definam o comportamento de objetos geométricos em um ambiente de realidade virtual. O *framework* oferece duas interfaces gráficas do tipo GUI (*Graphical User Interfaces*): um editor de animação que permite ao usuário manipular as características temporais das ações dos objetos e alguns parâmetros da apresentação da animação, além de um localizador de animação que permite ao usuário explorar a representação tridimensional (3D) do ambiente no qual os objetos podem ser vistos realizando suas ações, ambos apresentados na Figura 4 (Campos 2004).



(a)

(b)

Figura 4 – *Framework* “*Virtual Exploration of Animations*”: (a) editor para exploração virtual de animações; (b) localizador de animação. (Campos 2004)

Bellotti *et al.* (2008) apresentam um *framework* extensível que pode ser utilizado para implementar aplicações “*edutainment*” (que mistura educação e entretenimento) com diferentes tipos de desafios/aventuras baseadas em ambientes 3D. A proposta é um jogo interativo educativo (*TiE – Tour in Europe*) onde o usuário pode se divertir e aprender ao mesmo tempo. Ele consiste basicamente de uma reconstrução em um ambiente 3D realístico de uma cidade e região, com a qual o usuário pode interagir. Para tornar o aspecto de customização mais acessível a educadores não familiarizados com computadores, está em desenvolvimento o *Authoring Tool*, nomeado *TiE Creative ToolKit*. Ele ajudará a instanciar o *template*, e a especificar os parâmetros configuráveis, criando jogos para uma dada cidade/área utilizando seus próprios conteúdos. A Figura 5 apresenta a visualização de uma região e um Quiz Visual gerados a partir do *framework*.



(a)

(b)

Figura 5- *Tour in Europe* : (a) snapshot da reconstrução feita pelo TiE 3D da “Strada Nuova” no centro da cidade de Genoa; (b) Quiz visual sobre “De Ferrarisquare” em Genoa (Bellotti *et al.* 2008)

A revisão sistemática realizada também abordou *frameworks* cujos objetivos não são a geração de aplicações de Realidade Virtual, porém apresentam ferramenta para instanciação automatizada. Um deles é o GREN, *framework* para a gestão de recursos de negócios como aluguel, comércio ou manutenção de recursos (Cagnin, 2005) baseado em uma linguagem de padrões denominada GRN. O GREN é um *framework* bem completo, possui um manual de instanciação denominado “Cookbook do GREN”, além da *GREN Wizard*, ferramenta que funciona como um gerador de aplicações, e tem como objetivo automatizar os passos da instanciação do *framework*. Com essa ferramenta, cuja interface é apresentada na Figura 6, o usuário responde a diversas perguntas, e então o código (classes e métodos) é gerado automaticamente. Em seu desenvolvimento houve diversas avaliações com usuários, para testar e identificar pontos do projeto a serem melhorados (Braga, 2002).



Figura 6 - Exemplo da GUI do GREN-Wizard. (Braga, 2002)

Outro exemplo é o *Framework Qd+*, desenvolvido com base na linguagem de padrões LV cujo objetivo é o apoio ao desenvolvimento de sistemas de informação baseados na web. Ele possui três camadas: camada de apresentação (navegador na

máquina cliente), camada de aplicação (servidor de *web* ou servidor *http*), e camada de persistência (servidor de banco de dados). A camada de apresentação é a única visível aos usuários, contém todas as classes de interface, e é a que mais sofre alterações. É ela que permite a entrada de dados, navegação, propagação de eventos, requisição de páginas e ativação de tarefas. A camada de aplicação é a camada intermediária onde as requisições são interpretadas e processadas. E a camada de persistência é responsável pela manipulação de dados e pela conexão com o Sistema Gerenciador de Banco de Dados (SGBD). A interface com o usuário utiliza código *HTML* junto com linguagem *Smalltalk*, produzindo páginas *HTML* dinâmicas, como a que pode ser vista na Figura 7. O *Framework* Qd+ também pode apoiar a instanciação de novas aplicações no domínio dos sistemas de gestão de leilões virtuais e pode ser integrado ao *framework* GREN citado anteriormente (Ré, 2002).



Figura 7 - Página principal do Sistema de Leilões Virtuais instanciada. (Ré, 2002)

Como se pode perceber, existem muitas possibilidades de aplicações de *frameworks*, que vão desde aplicações de negócios até jogos para entretenimento, passando por aplicações para treinamento médico. Dentre os *frameworks* voltados para aplicações na área médica utilizando RV, são poucos os que apresentam alguma ferramenta para instanciação automatizada. Na Tabela 1 é apresentado um resumo das

características dos *frameworks* abordados na revisão bibliográfica disponibilizada nesta seção.

Tabela 1 – Resumo das características dos *frameworks* abordados na revisão sistemática.

<i>Framework</i>	Realidade Virtual?	Possui ferramenta de instanciação automatizada?	Linguagem	Objetivo
Simulação de Tratamento Dentário	Sim	Não	-	Ajudar ortodontistas a prever e a lidar com problemas ortodônticos, além de sugerir soluções para os tratamentos
Cyclops 3D Environment – C3DE	Sim	Não	C++	Gerar aplicações de radiologia
Avaliação dinâmica de fadiga muscular	Sim	Não	-	Avaliar o trabalho humano e prever dinamicamente potenciais riscos de desordem músculo-esquelética, especialmente para fadiga muscular
Piavca	Sim	Não	-	Criação de personagens virtuais interativos
<i>Virtual Reality Abstraction Layer - ViRAL</i>	Sim	Sim	C++	Facilitar a criação de aplicações extensíveis operadas por interfaces WIMP
<i>Virtual Exploration of Animations</i>	Sim	Sim	-	Controlar o comportamento de objetos geométricos em um ambiente de realidade virtual
<i>Tour in Europe - TiE</i>	Sim	Não	-	Reconstrução em um ambiente 3D realístico de uma cidade e região, com a qual o usuário pode interagir
GREN	Não	Sim	Smalltalk	Gestão de recursos de negócios

Qd+	Não	Sim	Smalltalk	Gerar páginas HTML dinâmicas
<i>ViMeT</i>	Sim	Sim	Java com API Java3D	Gerar ambientes de treinamento médico

Os *frameworks* voltados para geração de aplicações para treinamento médico são opções muito viáveis no treinamento e aprendizado tanto de estudantes quanto de profissionais, pois muitas vezes podem substituir o uso de laboratórios e até mesmo testes em pessoas reais. Na próxima seção é apresentado um *framework* desse tipo, o *ViMeT*, e sua ferramenta de instanciação *ViMeTWizard*, que são o foco principal desse projeto.

### 3. Metodologia

Nesta seção será descrita a metodologia utilizada para desenvolver o projeto. Serão apresentados o *framework ViMeT* e sua ferramenta de instanciação automatizada *ViMeT Wizard*, as tecnologias utilizadas para sua implementação e o trabalho que foi desenvolvido nesse projeto.

#### 3.1 O Framework *ViMeT*

O *framework* orientado a objetos *ViMeT* (“*Virtual Medical Training*”) foi criado para simulação de procedimentos médicos com o objetivo de treinar profissionais e estudantes da área. (OLIVEIRA, 2009) Ele foi implementado com a linguagem de programação Java e utilizando também a API Java3D. Essa linguagem foi escolhida devido à sua gratuidade, possibilitando sua utilização em locais que não dispõem de muitos recursos financeiros, além de ser compatível com variados sistemas operacionais. A API Java3D, por sua vez, possui diversas classes que auxiliam na visualização tridimensional,



na interação e na implementação de ambientes de Realidade Virtual. Esses ambientes de Realidade Virtual são construídos pela API usando grafos de cena, com os quais é possível criar grandes ambientes, ricos em detalhes e que podem interagir com gráficos tridimensionais (Sun, 2010). O banco de dados escolhido para o armazenamento dos dados foi o *Apache Derby*.

A partir da instanciação das classes do *ViMeT*, é possível gerar uma aplicação com um objeto tridimensional que representa um órgão humano (mama ou pernas, por exemplo), e outro objeto que representa um instrumento médico (como uma seringa), a fim de simular um procedimento de exame de biópsia em ambiente de Realidade Virtual. O *framework* possui como funcionalidades detecção de colisão com precisão, deformação, interação com equipamentos convencionais e não convencionais, interface gráfica, estereoscopia, modelagem de objetos tridimensionais e AVs (OLIVEIRA e NUNES, 2009).

A Figura 8 apresenta um exemplo de aplicação que foi gerada através da instanciação manual do *framework* (Oliveira, 2006). Nessa aplicação em específico a mama foi escolhida como órgão humano e a seringa foi o objeto escolhido para ser o instrumento médico. As funcionalidades de colisão, deformação e estereoscopia por anaglifos foram induzidas.

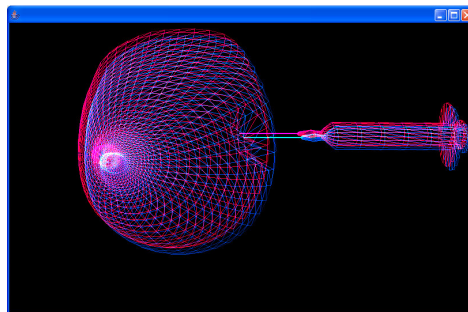


Figura 8 – Exemplo de Aplicação gerada pela instanciação manual do *ViMeT* (Oliveira, 2006)

A manipulação dos instrumentos pode ser feita por meio de dispositivos convencionais (como o teclado e o mouse), ou pode ser feita através de dispositivos não convencionais (como luva de dados e dispositivo háptico) (Corrêa, 2008) proporcionando maior realismo nas simulações.

Um dos itens de extrema importância na criação de um *framework* é sua documentação, pois é através dela que o usuário poderá compreender como funciona o *framework* e como ele é capaz de gerar as aplicações que lhe interessa. No ViMeT a documentação foi feita por meio de um *Cookbook* e pela documentação do tipo *javadoc*, onde são disponibilizadas algumas de suas classes, métodos e recursos (Oliveira, 2006).

Na Figura 9 está representado como era o diagrama de classes do *framework*.

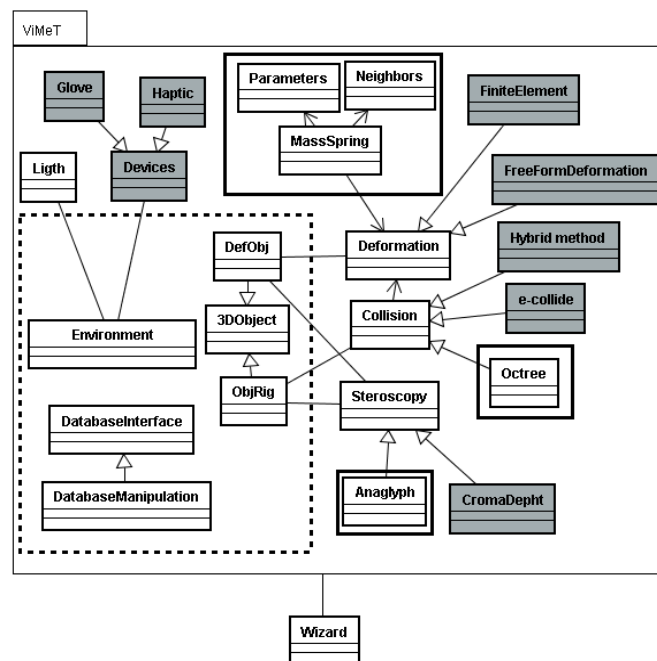


Figura 9 – Diagrama de Classes do *ViMeT* (Oliveira, 2006)

### 3.2 A ferramenta ViMeT Wizard

Além da instanciação manual do *framework*, também foi desenvolvida a ferramenta *ViMeTWizard*, quem tem como objetivo automatizar e, conseqüentemente, simplificar e facilitar a instanciação das classes e a geração de aplicações.

Por meio da ferramenta, o usuário (desenvolvedor de aplicações) não necessita ter prévio conhecimento sobre as classes do *framework*. Ele só precisa escolher as opções desejadas para que a ferramenta gere o código fonte (arquivo com extensão .java) e a

aplicação (arquivo compilado com extensão .class), armazenando as opções selecionadas em um banco de dados (Oliveira, 2006). Ela permite salvar aplicações para serem utilizadas posteriormente, permite que sejam feitas edições nas mesmas, e também é possível que o usuário defina características dos objetos (como escala, translação e rotação) além de definir quais funcionalidades eles terão. Na Figura 10 é apresentada uma visão da interface inicial da versão anterior da ferramenta *ViMeTWizard*.



Figura 10– Visão geral da versão antiga da ferramenta *ViMeTWizard* (Oliveira, 2006)

Na figura 11 é apresentada a guia *Loader*. Por meio dela era possível que selecionar um órgão e um instrumento médico de suas respectivas listas, tendo-os cadastrados previamente no Banco de Dados. Depois de inseridos os objetos, se fosse necessário modificar algum bastava selecioná-lo e inserir novos valores para sua escala, translação ou rotação e acionar o botão *OK* para aplicar as transformações.

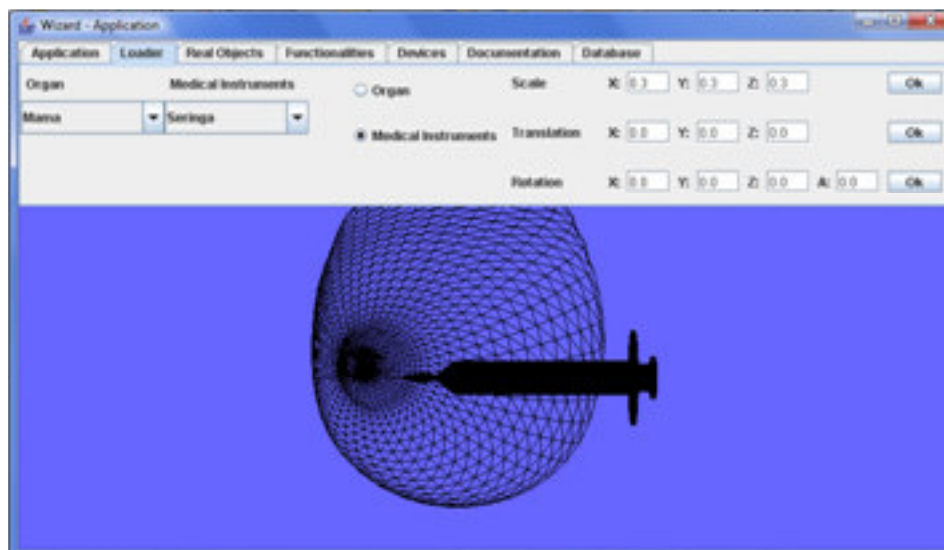


Figura 11 – Guia *Loader* da versão antiga da ferramenta *ViMeTWizard*.

Na guia *Functionalities*, apresentada na Figura 12, era possível definir as funcionalidades da aplicação (deformação, detecção de colisão e estereoscopia), e personalizar algumas delas. Após feitas as modificações desejadas, basta acionar o botão *Save* para gravar as alterações ou *Cancel* para descartar as alterações.

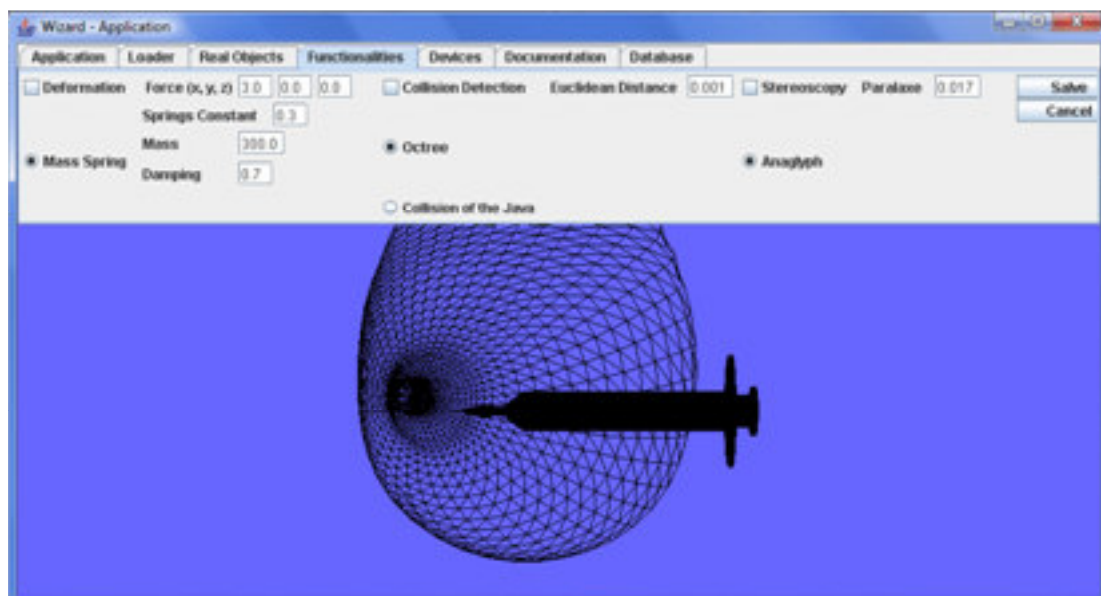


Figura 12 – Guia *Functionalities* da ferramenta *ViMeTWizard*.

Essa versão da ferramenta ViMeT Wizard possuía alguns itens problemáticos que necessitavam ser modificados. Não havia padronização da linguagem, havia textos tanto em português quanto em inglês. Não era possível realizar a manutenção das tabelas do Banco de Dados separadamente. Não havia a possibilidade de adicionar mais de um objeto representando um órgão humano, e um objeto representando um instrumento médico. Foram esses os principais alvos de mudanças do projeto, cujos resultados podem ser visualizados na próxima seção.

### 3.3 Trabalho Realizado

A seguir é apresentado como o projeto foi desenvolvido. São descritas suas fases de execução, as alterações que foram realizadas no banco de dados, na interface gráfica e no código da ferramenta *ViMeT Wizard*.

#### 3.3.1 Fases do Projeto

O presente projeto foi dividido em diversas fases para facilitar o entendimento do assunto e prover os conhecimentos necessários para sua execução. A Figura 13 representa a ordem de execução dessas fases.

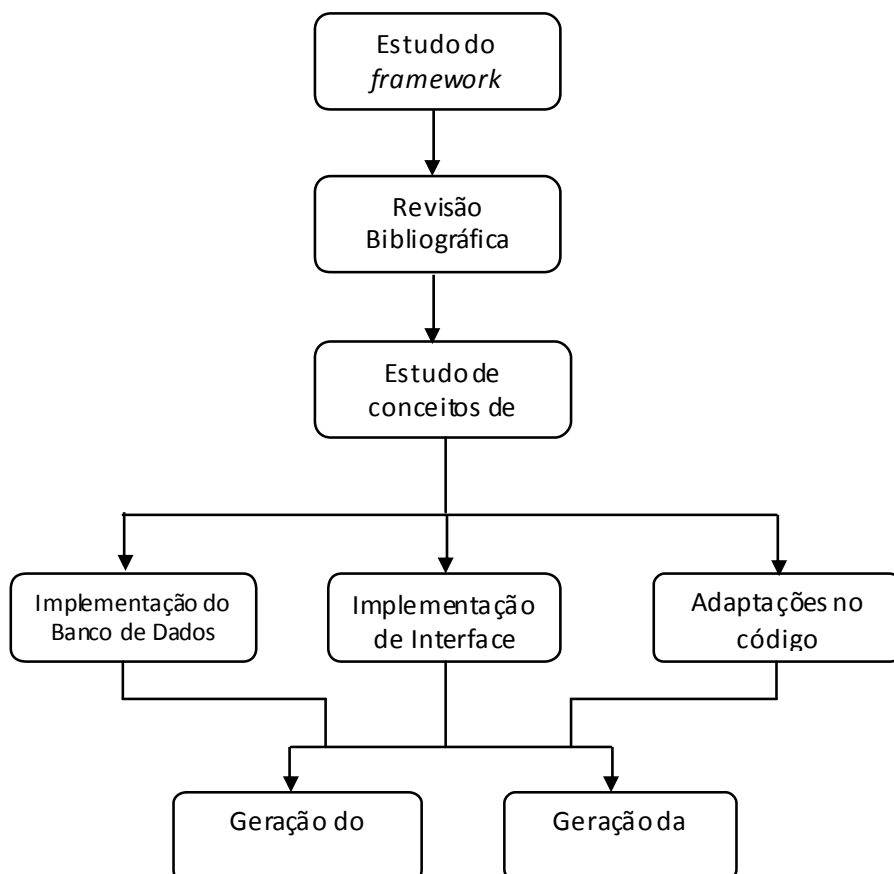


Figura 13 – Fases de execução do projeto

A primeira fase desenvolvida foi a de “Estudo do *framework ViMeT*” e teve duração de aproximadamente um mês. Nesse período foram estudados o comportamento das aplicações geradas pelo *framework*, o funcionamento da ferramenta *ViMeT Wizard* e quais recursos ela possuía, consistindo de um processo de familiarização e adaptação com os aplicativos que seriam utilizadas no projeto. Também foram realizadas leituras de artigos e dissertações escritas pelos desenvolvedores do *ViMeT* para entender como ele foi implementado e quais eram os seus propósitos, além de leituras de artigos de congressos e simpósios que falavam sobre *frameworks*, realidade virtual e ferramentas para treinamento médico.

A segunda fase foi a de “Revisão Bibliográfica”, onde foram estudados os conceitos de Revisão Sistemática para, através desse método, identificar o que já havia sido feito sobre o assunto abordado. A Revisão Sistemática desse relatório foi feita utilizando-se como base o protocolo de revisão, formulários de condução de revisão, formulários de seleção dos estudos e formulários de extração de dados, tendo duração de aproximadamente dois meses. Os resultados obtidos nessa revisão bibliográfica podem ser encontrados na seção 2 desse relatório. Essa fase foi muito importante, pois foi através dela que ocorreu uma maior familiarização com o assunto e foi onde ocorreu um ganho muito grande de conhecimentos sobre outros projetos científicos que haviam sido feitos até o momento. Outro ponto interessante a destacar é que a revisão sistemática foi um método interessante para realizar pesquisa bibliográfica, pois ela é um método que

traça uma espécie de roteiro de pesquisa a ser seguido, ajudando a não perder o foco no assunto pesquisado e sendo ao mesmo tempo uma maneira bastante abrangente de obtenção de dados.

Na terceira fase, “Estudo de conceitos de tecnologia” foram estudadas as tecnologias utilizadas na implementação do *ViMeT*, como a biblioteca Java3D e seus recursos, conceitos de banco de dados, além da realização de testes para entender o funcionamento e a integração do SGBD Derby com a linguagem Java. Nessa fase começou o estudo mais aprofundado sobre as classes e métodos da ferramenta *ViMeT Wizard*.

Na quarta fase foram realizadas simultaneamente três principais atividades: “Implementação do Banco de Dados”, “Implementação da Interface Gráfica” e “Adaptações no código”. Essas atividades foram se mesclando na quarta fase, pois as alterações feitas em uma delas necessitavam que fossem feitas alterações nas outras. Foi desenvolvido um novo banco de dados, foi criada uma nova aparência na ferramenta *Wizard*, e foram realizadas adaptações no código para que a *Wizard* pudesse adaptar-se às suas novas funcionalidades. Muitos métodos foram modificados e outros foram criados para que fosse possível a integração da nova interface gráfica com o novo banco de dados. Essas alterações realizadas serão explicadas mais detalhadamente nas seções 3.3.2, 3.3.3 e 3.3.4.

A quinta fase foi o período de refazer a geração automática do código da aplicação que se deseja construir, a partir dos dados definidos pelo usuário na ferramenta *Wizard* (objetos, seus parâmetros e funcionalidades). Gerando esse código é possível então gerar a aplicação em si com as necessidades do usuário.

### **3.3.2 Alterações no Banco de Dados**

Na antiga versão do *ViMeT* era possível gerar somente aplicações que continham um único órgão e um único instrumento médico. Uma das propostas do projeto era ampliar essas opções para que pudessem ser inseridos mais de um órgão, podendo assim

formar um objeto composto de várias partes. Com essa modificação, o *framework* ficaria mais genérico e poderia gerar um maior número de tipos de aplicações.

Para atingir esse objetivo foi necessária uma reestruturação geral do banco de dados. Na Figura 14 é apresentado como era a estrutura anterior do banco de dados do *ViMeT*. Nessa versão eram necessárias somente duas tabelas, uma para armazenar os objetos e outra para armazenar os dados específicos de cada aplicação. Não era necessária uma tabela intermediária entre essas duas, pois o número de objetos de cada aplicação era fixo, portanto as informações sobre eles eram armazenadas na própria tabela “Aplicações”.

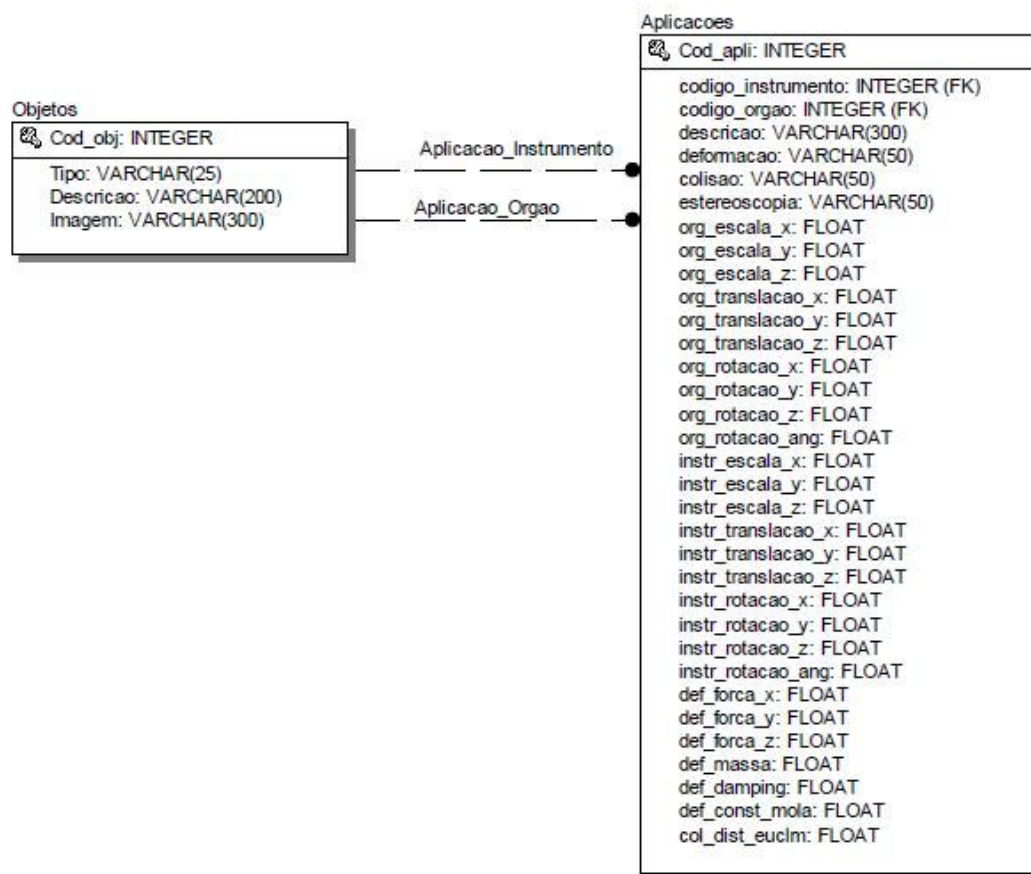


Figura 14 – Banco de Dados da versão anterior do *ViMeT*

Com a nova proposta da *ViMeT Wizard*, foi necessário modificar esse banco de dados, criando novas tabelas e novas relações. Foram desenvolvidas várias versões diferentes, pois eram necessárias adaptações conforme a interface gráfica era modificada.



Na Figura 15 é apresentada a versão final do Banco de Dados do *ViMeT*. Foi adicionada uma tabela “Categoria” para ajudar a diferenciar e identificar os objetos, uma vez que agora é possível adicionar vários objetos diferentes. Também foi criada uma tabela “Apli\_Obj”, pois cada Aplicação poderá ter um número variável de objetos, necessitando dessa tabela intermediária. Essa tabela armazena as características específicas de cada objeto que o usuário deseja ter em determinada aplicação, como escala, translação, rotação, transparência, entre outras. A tabela “Objeto” continua armazenando os dados dos objetos, com a diferença de que agora ela também armazena a categoria à qual esse objeto pertence. A tabela “Aplicacao” continua existindo e armazena as características gerais da aplicação, como a descrição, parâmetros de deformação, estereoscopia e colisão.

Foram criadas mais três tabelas: “Estereoscopia”, “Deformacao” e “Colisao”. Atualmente a *ViMeT* possui apenas um método de deformação, de um método de estereoscopia e dois métodos de detecção de colisão, mas essas tabelas foram adicionadas pensando em alterações futuras a serem feitas no *ViMeT*, pois facilitam a inclusão de novos tipos de métodos para executar as funcionalidades de deformação, de detecção de colisão e estereoscopia.

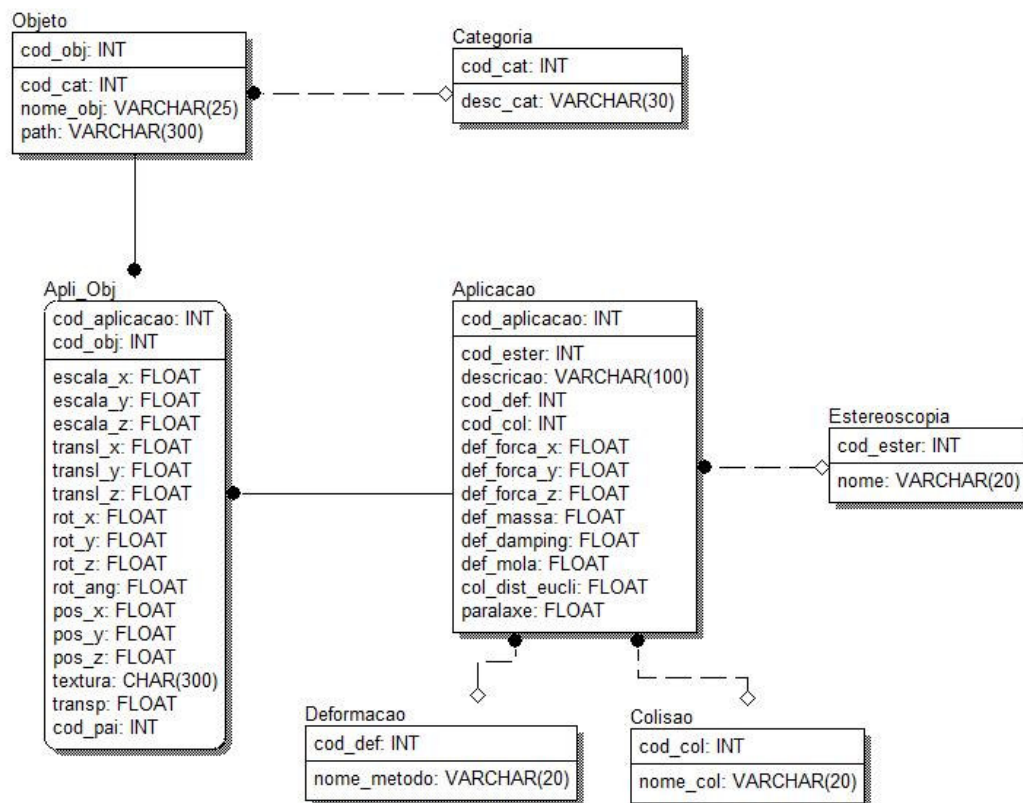


Figura 15 –Versão final do banco de dados do ViMeT

Esse novo modelo de banco de dados foi implementado utilizando o SGBD Derby, assim como o banco de dados antigo, pois ele é um SGBD com distribuição gratuita e que supre todas as necessidades requeridas pelo programa.

### 3.3.3 Alterações na Interface Gráfica

As primeiras alterações realizadas na *ViMeT Wizard* foram feitas com base no trabalho realizado por Frata (2009), denominado “Avaliação heurística da aplicação *Wizard* do *framework ViMeT*”. Nesse trabalho foram identificadas algumas características da aplicação que necessitavam ser melhorados. A primeira mudança foi em relação ao idioma da aplicação. A ferramenta ainda apresentava algumas telas, alertas e componentes gráficos cujo idioma era o português, então eles foram modificados para o inglês, visando à sua maior divulgação, inclusive em sites e artigos estrangeiros.

A interface gráfica da *ViMeT Wizard* também passou por uma série de adaptações e transformações. O intuito principal do projeto era fazer com que fosse possível carregar um número variável de objetos na aplicação, a fim que aumentar o realismo das aplicações geradas. Um órgão humano virtual, quando modelado sinteticamente, é composto por diversos objetivos virtuais (por exemplo: um braço virtual pode ser composto por pele, músculo e osso). Esses objetos devem exercer influência nos processos de colisão e deformação e, por isso, a ferramenta de instanciação deve gerar o código fonte considerando os diversos objetos a fim de que, futuramente suas características sejam consideradas nas funcionalidades citadas.

Para isso, além da reestruturação do banco de dados, foi necessária uma reestruturação da interface gráfica da ferramenta *ViMeTWizard*. Na Figura 16 é apresentada a guia “Loader” antiga. Na Figura 17 é apresentado um dos protótipos desenvolvidos para a guia “Loader”. E na Figura 18 é apresentada a versão final da guia “Loader”. Essa guia foi a que sofreu as mudanças mais significativas.

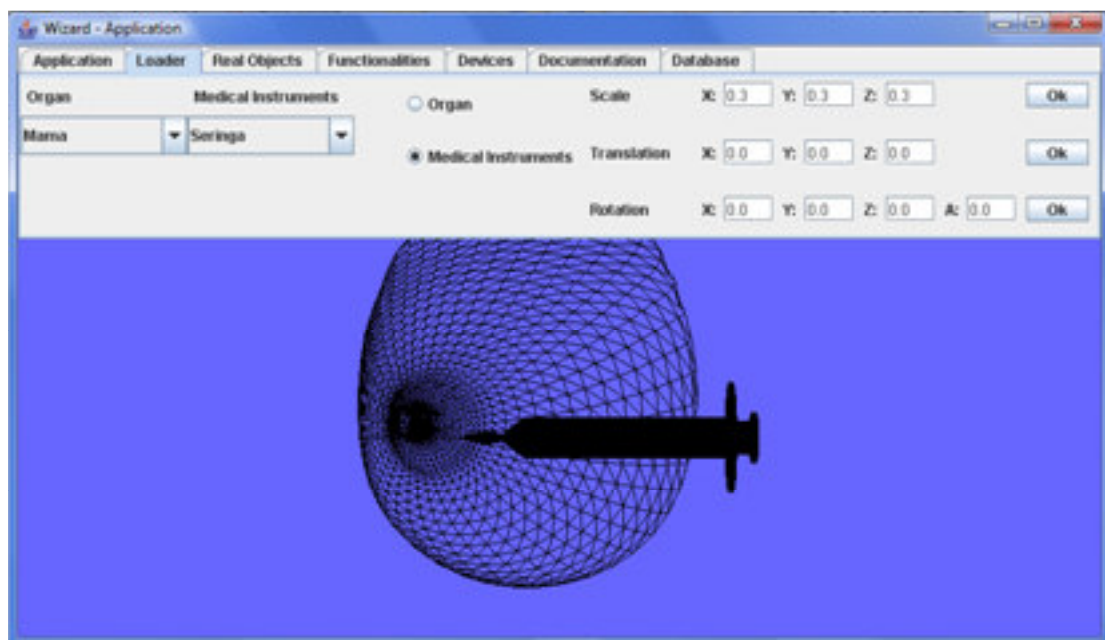


Figura 16 – Guia “Loader” antiga.

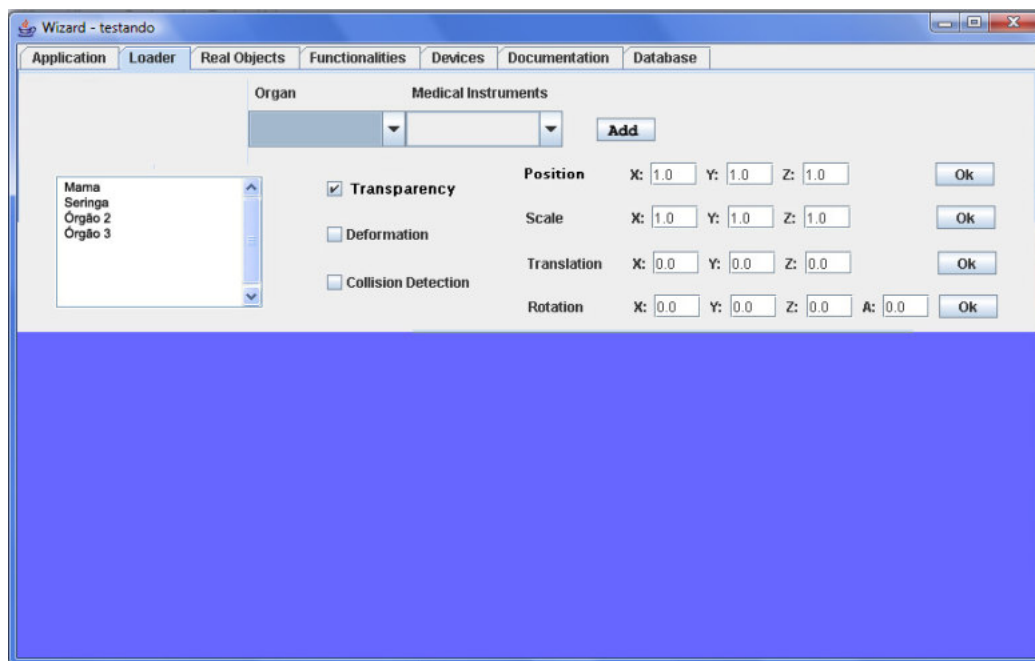


Figura 17 – Protótipo de modificação da guia “Loader”.

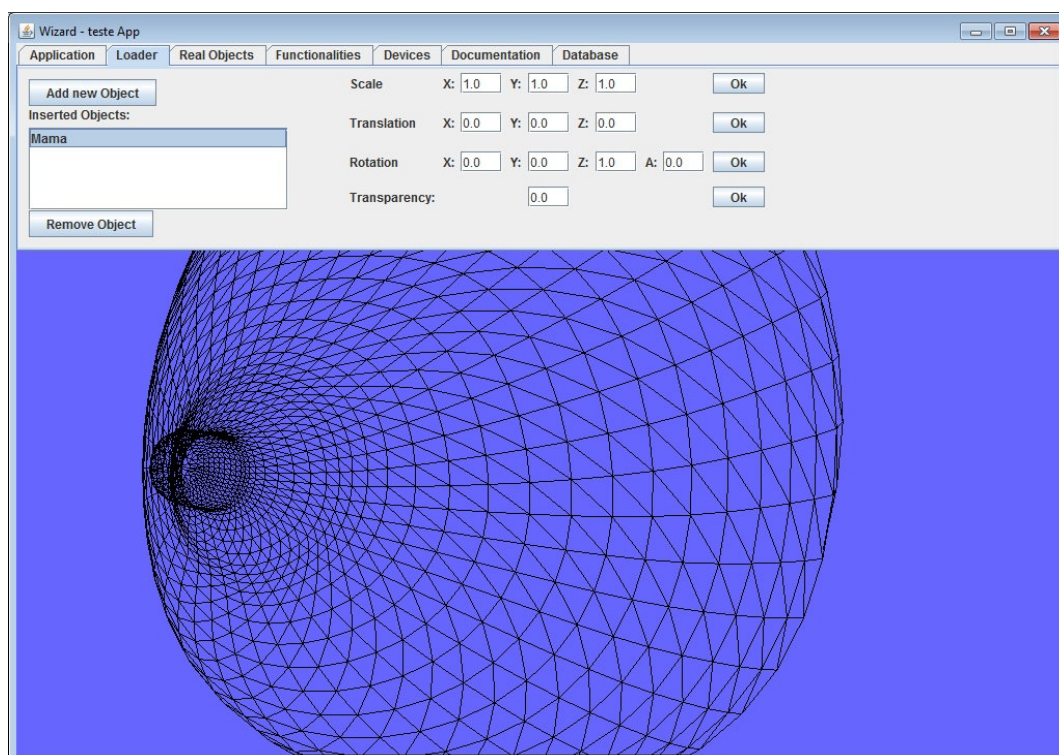


Figura 18 – Versão Final da guia *Loader*.

Na versão anterior havia dois componentes do tipo *comboBox* destinados para a adição de um órgão e de um instrumento médico, pois esse número era invariável. Na versão nova essa adição de objetos é feita acionando-se o botão “Add New Object”, o qual faz abrir uma janela onde é possível escolher a categoria do objeto desejado e, em seguida, são carregados no próximo componente *comboBox* os objetos que pertencem àquela categoria. Essa janela está representada na Figura 19. Então basta o usuário escolher o objeto desejado e clicar em “Add”. O objeto é adicionado na tela e em uma lista de objetos adicionados dentro da guia “Loader”.

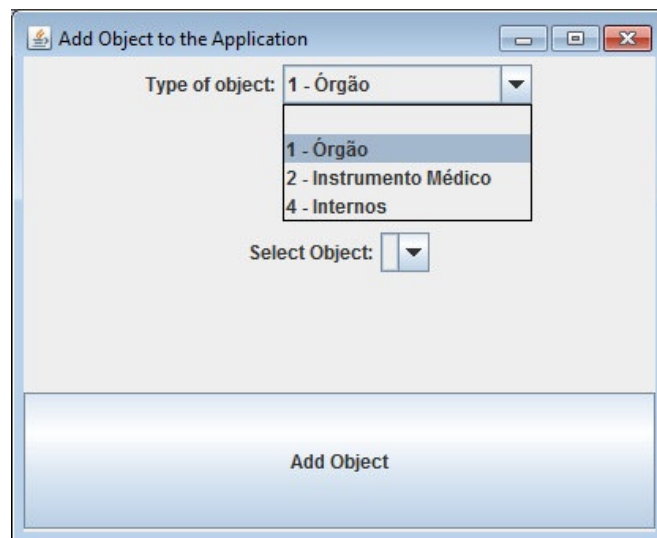


Figura 19 – Tela de adição de novo Objeto.

Depois de adicionado é possível mudar as características desses objetos. Para isso basta selecionar o nome do objeto alvo das mudanças dentro da lista de objetos adicionados, e estipular os novos parâmetros de escala, translação, rotação ou transparência.

Com a possibilidade da adição de diversos objetos foi necessário acrescentar um novo atributo “Transparência”, pois assim é possível visualizar vários objetos mesmo que eles estejam sobrepostos. Ele também pode ser modificado através da guia “Loader”. O valor da transparência pode variar entre 0.0 e 1.0, sendo que quanto mais próximo de zero

mais sólido é o objeto, e quanto mais próximo de um mais transparente é o objeto. Na Figura 20 é possível visualizar um exemplo de adição de três objetos com níveis diferentes de transparência.

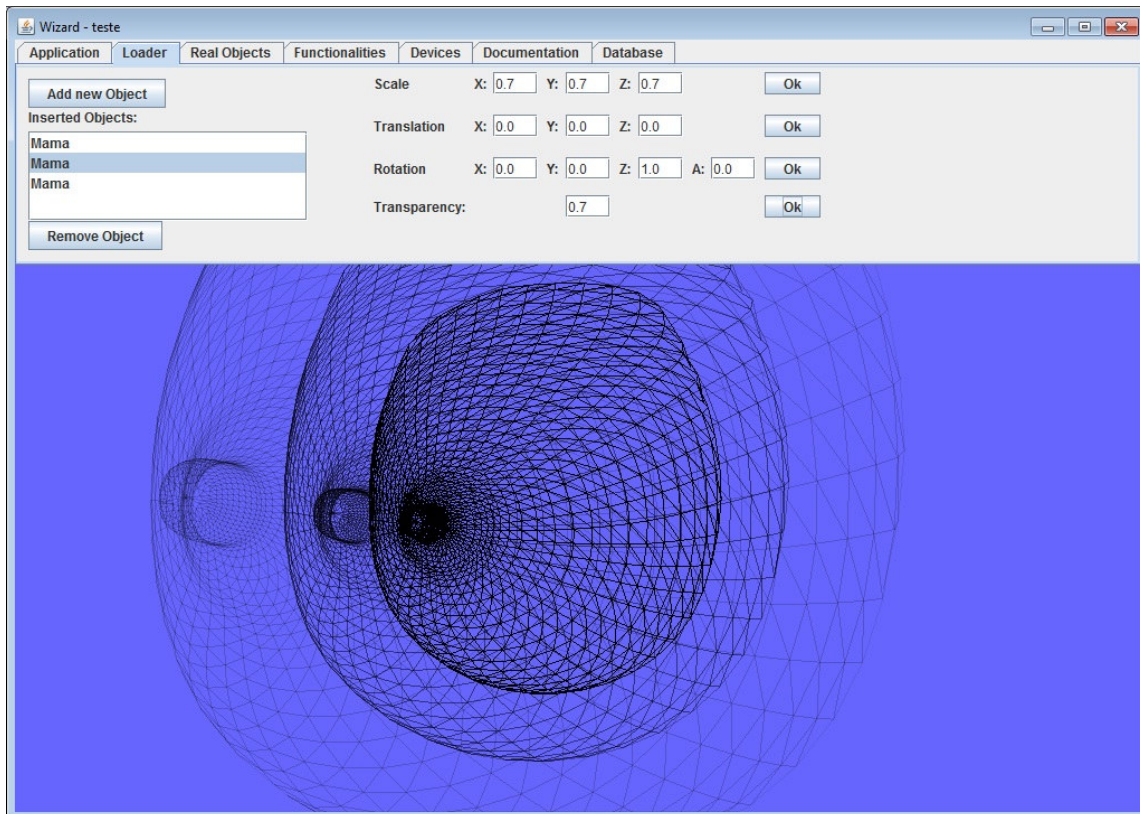


Figura 20 – Adição de três objetos com diferentes níveis de transparência.

Depois de adicionados os objetos, também é possível removê-los. Para isso deve-se selecionar o nome do objeto a ser removido e depois em “Remove Object”, localizado abaixo da lista de objetos adicionados. Como consequência o objeto é removido da tela de visualização e também da lista de objetos selecionados.

Outra guia que sofreu várias alterações foi a guia “Database”, pois antes o banco de dados possuía apenas duas tabelas e agora possui diversas. Foi então necessário desenvolver telas para a manutenção das tabelas separadamente. Na Figura 20 é apresentada a antiga versão dessa guia, e na Figura 21 é apresentada a nova versão.



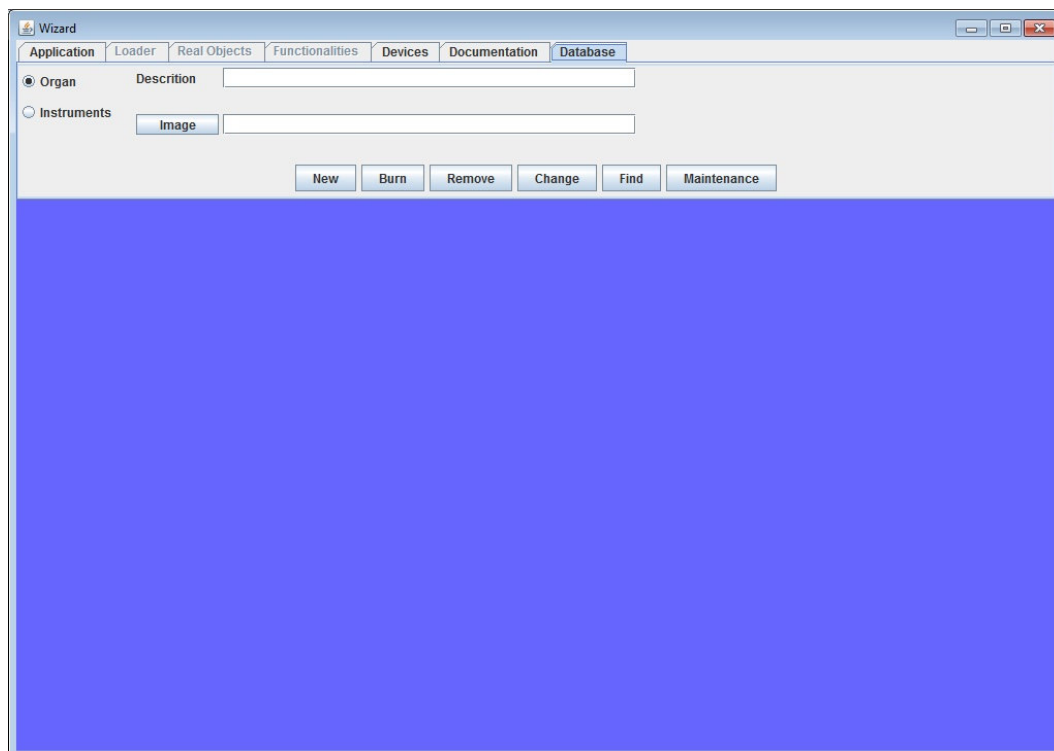


Figura 20 –Antiga guia “Database”.

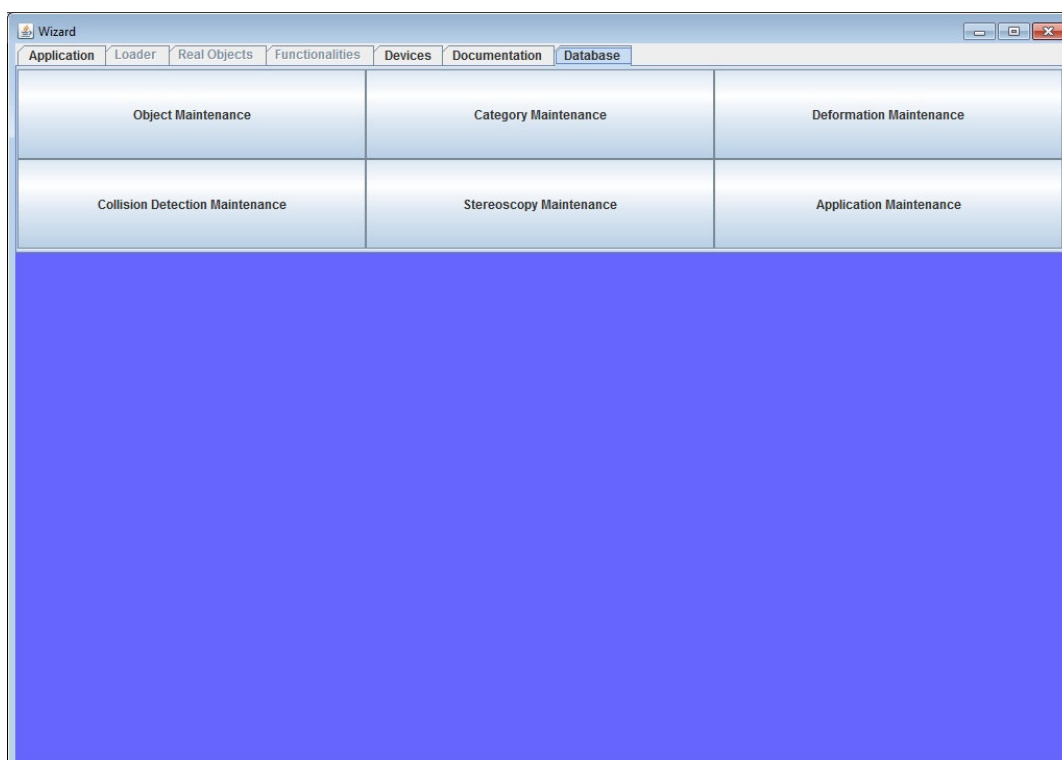


Figura 21 – Nova guia “Database”.

Acionando cada um dos botões exibidos na tela da Figura 21, abrem-se as telas de manutenção de cada uma das tabelas do banco de dados. Na Figura 22 é apresentada a tela de manutenção da tabela “Objeto” do banco de dados. Esta tela permite que seja adicionado um novo objeto, que seja editado um objeto já existente ou que um objeto seja removido.

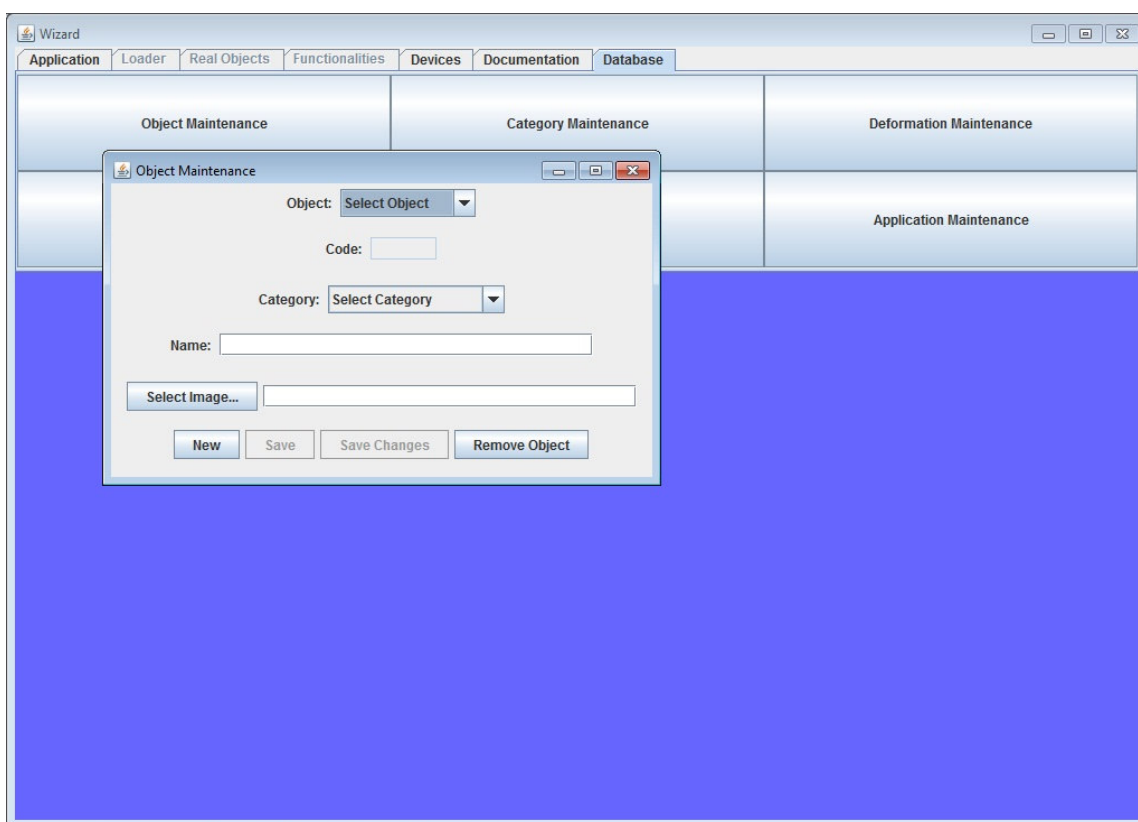
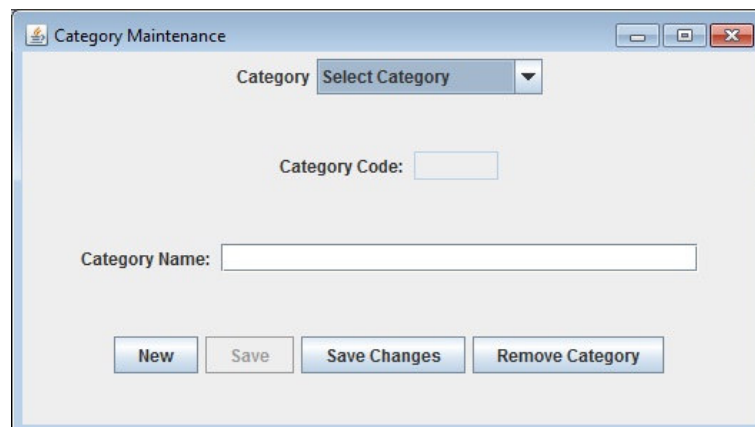


Figura 22 – Tela de manutenção da tabela Object.

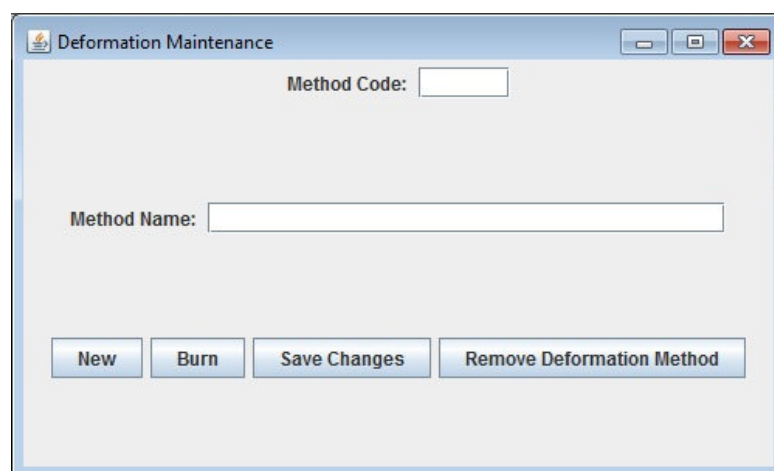
As outras telas de manutenção “Category Maintenance”, “Deformation Maintenance”, “Collision Maintenance” e “Stereoscopy Maintenance” apresentam essas mesmas funcionalidades e podem ser visualizadas nas figuras Figura 23, Figura 24, Figura 25 e Figura 26, respectivamente.





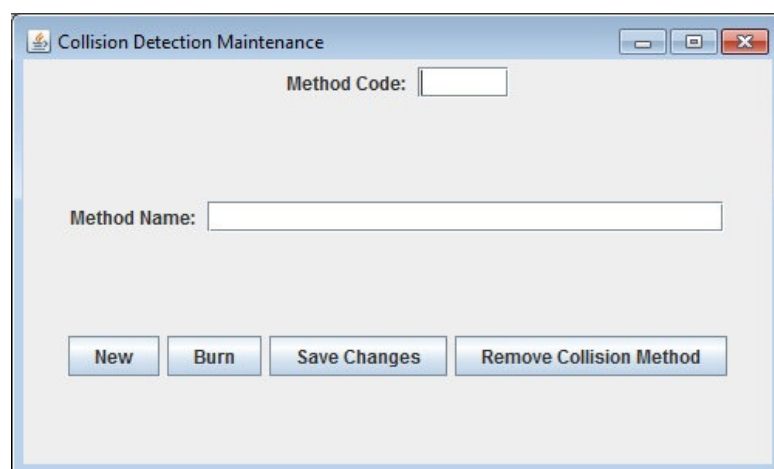
The 'Category Maintenance' window features a title bar with standard Windows controls. Below the title bar, there is a 'Category' label followed by a dropdown menu currently showing 'Select Category'. Underneath this is a 'Category Code:' label with an adjacent text input field. Further down is a 'Category Name:' label with a larger text input field. At the bottom of the window, there are four buttons: 'New', 'Save', 'Save Changes', and 'Remove Category'.

Figura 23 – Tela de manutenção da tabela “Category Maintenance”.



The 'Deformation Maintenance' window has a title bar with standard Windows controls. It contains a 'Method Code:' label with a text input field. Below this is a 'Method Name:' label with a larger text input field. At the bottom, there are four buttons: 'New', 'Burn', 'Save Changes', and 'Remove Deformation Method'.

Figura 24 – Tela de manutenção da tabela “Deformation Maintenance”.



The 'Collision Detection Maintenance' window features a title bar with standard Windows controls. It includes a 'Method Code:' label with a text input field. Below this is a 'Method Name:' label with a larger text input field. At the bottom, there are four buttons: 'New', 'Burn', 'Save Changes', and 'Remove Collision Method'.

Figura 25 – Tela de manutenção da tabela “Collision Maintenance”.

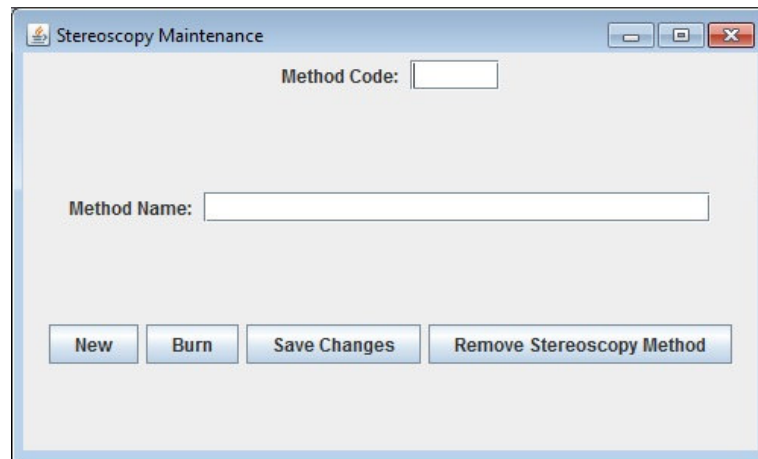


Figura 26 – Tela de manutenção da tabela “Stereoscopy Maintenance”.

A tabela “Aplicacao” também necessitou que fosse feita uma tela específica – Figura 27 – para sua manutenção. Através dela é possível remover uma aplicação já existente no banco de dados.

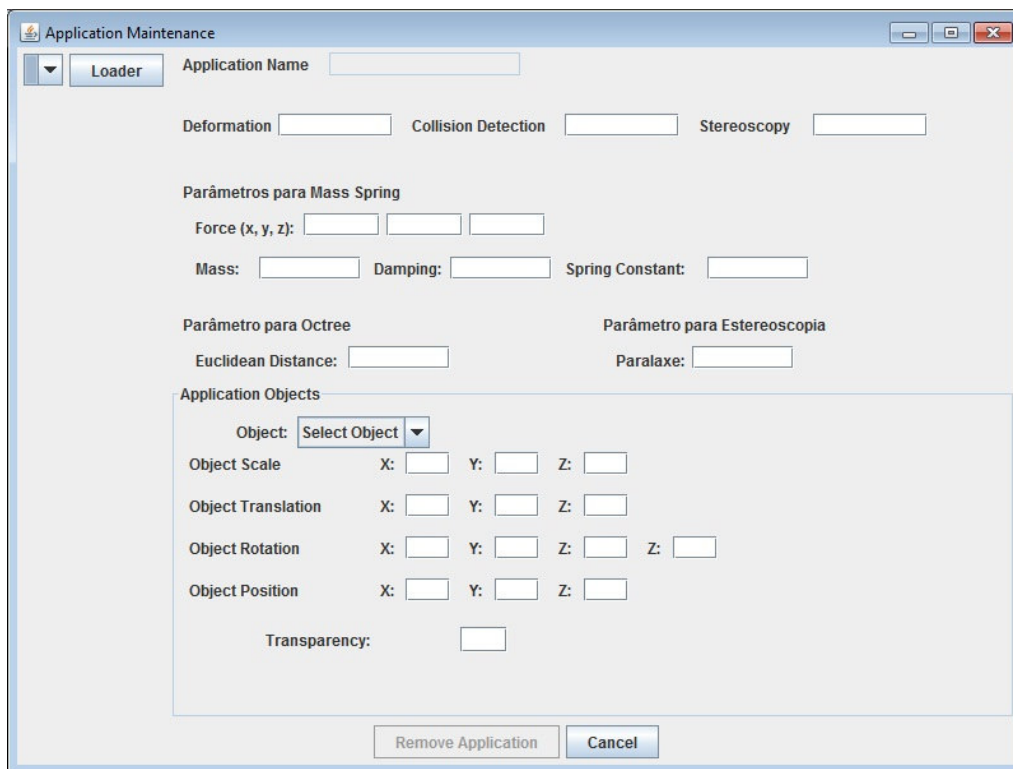


Figura 27 – Tela para manutenção da aplicação.

### 3.3.4 Alterações no Código

A maioria dos métodos da antiga versão da *ViMeT Wizard* foi concebida para manipular apenas dois objetos: um que representa o órgão e outro que representa um instrumento médico. Com a possibilidade de adicionar uma quantidade variável de objetos que representam órgãos compostos de várias camadas, havendo a conseqüente mudança da interface gráfica e do banco de dados, foi necessário que esses métodos também fossem adaptados.

Grande parte das variáveis que eram únicas tiveram de ser transformadas em vetores. O primeiro objeto adicionado é colocado na posição 0 do vetor, e ele será o objeto “pai” dos outros objetos adicionados posteriormente. Os métodos que anteriormente recebiam um objeto deformável como parâmetro agora receberão esse objeto da posição 0 do vetor. Um exemplo pode ser visto na Figura 28. A parte (a) mostra como eram algumas variáveis da versão anterior, que representam as características da translação, rotação e posição do órgão e do instrumento médico. A parte (b) mostra os

vetores que foram utilizados para substituí-las. Da mesma maneira, muitas outras variáveis únicas ao longo do código foram substituídas por variáveis em forma de vetores.

<pre>Vector3d to, eo; Vector3d ti, ei; AxisAngle4d ro, ri;</pre>	<pre>Vector3d [] tObj = new Vector3d [30]; //translacao Vector3d [] eObj = new Vector3d [30]; //escala AxisAngle4d [] rObj = new AxisAngle4d [30]; //rotação</pre>
--	--

(a)

(b)

Figura 28 – (a) Antigas variáveis. (b) Novas variáveis em forma de vetor.

Ao elemento “list” da interface gráfica foi adicionada uma ação para que quando algum de seus itens (objetos) fossem selecionados, suas respectivas características fossem carregadas nos devidos campos dentro da guia “Loader”. Esse evento do componente “list” pode ser verificado na Figura 29.

```

list.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        int index = list.getSelectedIndex();
        if(eObj[index].x > 0 && eObj[index].y > 0 && eObj[index].z > 0){
            xTextoEscala.setText("" + eObj[index].x);
            yTextoEscala.setText("" + eObj[index].y);
            zTextoEscala.setText("" + eObj[index].z);
        }
        else{
            xTextoEscala.setText("1.0");
            yTextoEscala.setText("1.0");
            zTextoEscala.setText("1.0");
        }

        xTextoTransalacao.setText("" + tObj[index].x);
        yTextoTransalacao.setText("" + tObj[index].y);
        zTextoTransalacao.setText("" + tObj[index].z);

        xTextoRotacao.setText("" + rObj[index].x);
        yTextoRotacao.setText("" + rObj[index].y);
        zTextoRotacao.setText("" + rObj[index].z);
        aTextoRotacao.setText("" + rObj[index].angle);

        textoTransp.setText("" + transp[index]);
        System.out.println("cliqueu o mouse " + index);
    }
});

```

Figura 29– Ação associada ao clique de mouse do componente list

Com a mudança das variáveis mostradas na Figura 28, os métodos que as utilizavam também passaram por adaptações. Exemplos de métodos que sofreram adaptações em consequência da alteração dessas variáveis foram o mudaEscala(), mudaTranslacao() e mudaRotacao(). A nova versão do método mudaEscala() é mostrada na Figura 30.

```

public void mudaEscala()
{
    int index = list.getSelectedIndex();
    eObj[index].x = Double.parseDouble(xTextoEscala.getText());
    eObj[index].y = Double.parseDouble(yTextoEscala.getText());
    eObj[index].z = Double.parseDouble(zTextoEscala.getText());
    Transform3D t = new Transform3D();
    t.setScale(eObj[index]);
    t.setRotation(rObj[index]);
    t.setTranslation(tObj[index]);
    tgObj[index].setTransform(t);
}

```

Figura 30 – Nova versão do método mudaEscala().

Uma das funcionalidades adicionadas à ferramenta *ViMeT Wizard* foi a transparência. Foi então necessário criar um método para essa característica pudesse ser aplicada aos objetos mostrados na tela. Esse método é apresentado na Figura 31, e a parte responsável por mudar a transparência do objeto é a linha após o comentário.

```

public void mudaTransp(){
    BranchGroup bgAux=null;
    String nome_arquivo = ap.CaminhoImagem("1");
    ObjectFile obj1 = new ObjectFile();
    try
    {
        bgAux = obj1.load(nome_arquivo).getSceneGroup();
    }
    catch(Exception e){
        System.out.println("Exceção no método mudarTransp");
    }
    Shape3D temp = (Shape3D) bgAux.getChild(0);
    Shape3D shape = new Shape3D (temp.getGeometry(),temp.getAppearance());

    int indice = list.getSelectedIndex();
    String item = list.getSelectedValue().toString();

    bgObj[indice].detach();
    bgObj[indice] = new BranchGroup();
    bgObj[indice].setCapability(BranchGroup.ALLOW_DETACH);

    TransparencyAttributes transp;
    transp = new TransparencyAttributes();
    float transparency = Float.parseFloat(textoTransp.getText().toString());
    // "transparency" pode variar entre 0.0 e 1.0 quanto maior, mais claro
    transp.setTransparency(transparency);
    transp.setCapability(TransparencyAttributes.ALLOW_MODE_WRITE);
    transp.setTransparencyMode(TransparencyAttributes.BLENDED);

    Appearance appearancell = new Appearance();
    appearancell.setCapability(Appearance.ALLOW_MATERIAL_READ);
    appearancell.setCapability(Appearance.ALLOW_MATERIAL_WRITE);

```

```

PolygonAttributes attributes11 = new PolygonAttributes();
attributes11.setPolygonMode(PolygonAttributes.POLYGON_LINE);
attributes11.setCullFace(PolygonAttributes.CULL_BACK);
appearancell.setPolygonAttributes(attributes11);

Color3f objColor = new Color3f(0.0f, 0.0f, 0.0f);
Color3f black = new Color3f(0.0f, 0.0f, 0.0f);
Color3f white = new Color3f(1.0f, 1.0f, 1.0f);

appearancell.setMaterial(new Material(objColor, black, objColor, white, 64.0f));
appearancell.setTransparencyAttributes(transp);
shape.setAppearance(appearancell);

TransformGroup tgAux = new TransformGroup();
Transform3D aux = new Transform3D();
aux.setScale(0.027f);
tgAux.setTransform(aux);

tgObj[indice] = new TransformGroup();
tgObj[indice].setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
tgObj[indice].setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

bgObj[indice].addChild(tgAux);
tgAux.addChild(tgObj[indice]);
tgObj[indice].addChild(shape);

bgObj[indice].addChild(tgAux);
tgAux.addChild(tgObj[indice]);
tgObj[indice].addChild(shape);
Shape3D shapeAux = (Shape3D) tgObj[indice].getChild(0);
Shape3D shapeAux2 = new Shape3D (shapeAux.getGeometry(),shapeAux.getAppearance());
shapeAux2.setAppearance(appearancell);
tgObj[indice].setChild(shapeAux2, 0);

myLocale.addBranchGraph(bgObj[indice]);
mudaEscala();
mudaTranslacao();
mudaRotacao();
}

```

Figura 31 – Método que permite a mudança da transparência do objeto.

Diversos métodos para auxílio na utilização do banco de dados foram criados ou modificados para se adaptarem à nova estrutura de tabelas. No Apêndice 5 podem ser visualizados alguns desses métodos que foram criados.



## 5. Conclusão

A Realidade Virtual vem sendo cada vez mais aplicada para gerar aplicações que possam simular o corpo humano e possam servir como ferramentas de aprendizado e treinamento tanto para estudantes quanto para profissionais da área, trazendo inúmeros benefícios para toda a sociedade.

Uma das ferramentas que se encaixam nesse grupo é o *framework ViMeT*, e sua ferramenta de instanciação automatizada *ViMeTWizard*. A tradução de alguns de seus componentes do português para o inglês contribuiu para permitir que sua utilização seja feita em vários países além do Brasil.

Ao melhorar a interface gráfica do *ViMeTWizard* foi oferecida uma contribuição para a sua facilidade de uso (instanciação), o que segundo os estudos realizados é muito benéfico para o usuário. Ao acrescentar a opção de carregar mais de dois objetos na mesma cena, foram expandidas as possibilidades do *framework* em questão. Dessa maneira, outras funcionalidades poderão lhe ser acrescentadas futuramente como, por exemplo, a geração de um atlas virtual para o estudo da anatomia humana.

Não houve tempo suficiente durante o projeto para que fossem feitas todas as alterações que eram necessárias. A geração do código fonte e a geração da aplicação são exemplos de atividades que fazia parte do escopo inicial, mas não houve tempo suficiente para executá-las. Porém, as alterações realizadas e os recursos adicionados descritos nesse relatório representam um grande avanço para a ferramenta.

Como trabalhos futuros, seria interessante que fossem feitas a geração automática do código fonte e a geração da aplicação. Outro trabalho que seria de grande importância é a adaptação da ferramenta *ViMeTWizard* para que ela possa fazer a geração de atlas virtuais.

## 6. Apêndices

### Apêndice 1 – Protocolo de Revisão

**Objetivos:** Identificar e analisar estudos primários que abordem frameworks para ambientes de realidade virtual (voltados para aplicações na área médica). Identificar e analisar estudos primários que abordem a instanciação automatizada de frameworks.

**Questão de Pesquisa:** *O que já foi feito a respeito da instanciação automatizada de frameworks de realidade virtual usando ferramentas Wizard?*

**CrITÉRIOS de Seleção de Fontes:** As fontes utilizadas serão selecionadas principalmente via web, mas também poderão haver fontes selecionadas via anais de eventos científicos disponíveis em outros meios, como CD-ROM.

**Métodos de Busca de Fontes:** Artigos, periódicos, trabalhos de conclusão de curso, iniciação científica ou pós-graduação (mestrado, doutorado, outros) que contenham as palavras chave definidas, e estejam disponíveis via web ou em anais de eventos científicos.

**Palavras-chave:** (framework AND instantiation) OR (virtual reality framework) OR (wizard tool) OR (automated code generation) OR (code generation) OR (product line)

#### Listagem de Fontes:

Biblioteca digital do IEEE (<http://www.ieee.org/portal/site>)

Biblioteca digital da ACM (<http://portal.acm.org/portal.cfm>)

CITeseer (<http://citeseer.ist.psu.edu/>)

Biblioteca digital da Sociedade Brasileira de Computação (SBC)  
(<http://www.sbc.org.br/bibliotecadigital/>)

Periódicos Capes (<http://www.periodicos.capes.gov.br/portugues/index.jsp>)

SCIELO (<http://www.scielo.org/php/index.php>)

IBICT(<http://www.ibict.br/>)

Teses da USP (<http://www.teses.usp.br/>)

Site da capes de teses e dissertações (<http://servicos.capes.gov.br/capesdw/>)

Anais de eventos científicos

**Tipo dos Trabalhos:** Artigos, periódicos, trabalhos de conclusão de curso, iniciação científica e pós-graduação.

**Idioma dos Artigos:** Português e Inglês.

**Critérios de Inclusão e Exclusão dos documentos:**

- A) Os documentos devem possuir formato digitalizado ou impresso
- B) Os documentos devem estar escritos em inglês ou em português
- C) Os artigos devem ter, no máximo, 5 anos, mas trabalhos de Iniciação Científica, teses e dissertações são considerando em um período de 10 anos.
- D) Os documentos devem conter alguma palavra-chave dentre as definidas.
- E) Serão desconsiderados documentos que apresentam aplicações simples de RV sem contemplar frameworks.
- F) Os frameworks devem abordar pelo menos realidade virtual, ou pelo menos instânciação automatizada.
- G) Serão desconsiderados artigos que não apresentem data de publicação
- H) Serão desconsiderados documentos que não estejam inteiramente disponíveis

**Processo de seleção dos Estudos Primários:** Serão executadas buscas nas fontes selecionadas com base nas palavras-chave definidas.

Os artigos encontrados serão analisados e verificados se enquadram-se nos critérios de inclusão ou exclusão.

De acordo com esses critérios, os artigos serão incluídos ou não na revisão sistemática.

**Estratégia de extração de informação:** Para cada documento induído na Revisão Sistemática, será preenchido um formulário de extração de dados objetivando destacar os pontos importantes que podem contribuir para o projeto.

**Sumarização dos resultados:** Após a extração das informações importantes, os resultados serão tabulados e será elaborado um relatório crítico para a sintetização dos dados obtidos.

## Apêndice 2 – Formulário de Condução de Revisão

N	Nome	Autores	Data Publicação	Critério de Inclusão	Fonte
1	A Framework for Orthodontic Treatment Simulation	Maria Andréia F. Rodrigues , Rafael G. Barbosa , Milton E. B. Neto	2004, 31 maio a 4 junho	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
2	Desenvolvimento e Implementação de Framework de Realidade Virtual para Treinamento Médico	Ana Cláudia M. T. G. de Oliveira , Fátima L. S. Nunes , Larissa Pavarini , Leonardo C. Botega	2006 29 mai. 03 jun	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
3	Framework for Interactive Medical Imaging Applications	André Ferreira Bem Silva, Tiago H. C. Nobrega , Diego Dias Bispo Carvalho , Renan Teston Inácio, Aldo Von Wangenheim	2009 jul. 22-23	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
4	Suporte ao Uso de Frameworks Orientados a Objetos com Base no Histórico do Desenvolvimento de Aplicações	Evandro César Freiberger , Ricardo Pereira e Silva	2004, 31 maio a 4 junho	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
5	Um Framework para o Desenvolvimento de Aplicações de Realidade Virtual Baseadas em Componentes Gráficos	Thiago de A. Bastos , Alberto B. Raposo , Marcelo Gattass	2005 jul. 22-29	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
6	A DATA MODEL FOR EXPLORATION OF TEMPORAL VIRTUAL REALITY GEOGRAPHIC INFORMATION SYSTEMS	Jorge Alberto Prado de Campos	2004-08-00	F	Biblioteca Digital Brasileira de Teses e Dissertações ( <a href="http://bdtd2.ibict.br/index.php?option=com_wrapper&amp;Itemid=40">http://bdtd2.ibict.br/index.php?option=com_wrapper&amp;Itemid=40</a> )
7	Framework for dynamic evaluation of muscle fatigue in manual handling work	Liang Ma; Bennis, F.; Chablat, D.; Wei Zhang;	21-24 April 2008	F	IEEE ( <a href="http://www.ieee.org/portal/site">http://www.ieee.org/portal/site</a> )
8	The Research and Development of Fitness Equipment Embedded System Based on Distributed Virtual Environment	Xiaojun Li; Mingyan Li;	25-26 July 2009	F	IEEE ( <a href="http://www.ieee.org/portal/site">http://www.ieee.org/portal/site</a> )
9	Um processo para construção e instanciação de frameworks baseados em uma linguagem de padrões para um domínio específico.	Braga, Rosana Teresinha Vaccare	2002	F	Teses da USP ( <a href="http://www.teses.usp.br/">http://www.teses.usp.br/</a> )

10	"PARFAIT: uma contribuição para a reengenharia de software baseada em linguagens de padrões e frameworks"	Cagnin, Maria Istela	2005	F	Teses da USP ( <a href="http://www.teses.usp.br/">http://www.teses.usp.br/</a> )
11	"Um processo para construção de frameworks a partir da engenharia reversa de sistemas de informação baseados na Web: aplicação ao domínio dos leilões virtuais"	Ré, Reginaldo	2002	F	Teses da USP ( <a href="http://www.teses.usp.br/">http://www.teses.usp.br/</a> )
12	Piavca: A Framework for Heterogeneous Interactions with Virtual Characters	José Eduardo Mendonça Xavier , Hansjörg Andreas Schneebeli	2008	F	IEEE ( <a href="http://www.ieee.org/portal/site">http://www.ieee.org/portal/site</a> )
13	Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds	Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, Victor Zappi	2008	F	Portal ACM ( <a href="http://portal.acm.org/portal.cfm">http://portal.acm.org/portal.cfm</a> )
14	The mapping problem back and forth: customizing dynamic models while preserving consistency	Tim Derckx, Kris Luyten, Karin Coninx	2004	F	Portal ACM ( <a href="http://portal.acm.org/portal.cfm">http://portal.acm.org/portal.cfm</a> )
15	basho - A Virtual Environment Framework	André Hinkenjann1 , Florian Mannuß	2004	F	<a href="http://cg.inf.fh-bonn-rhein-sieg.de/wp-content/uploads/2007/10/l.pdf">http://cg.inf.fh-bonn-rhein-sieg.de/wp-content/uploads/2007/10/l.pdf</a>
16	VR Juggler: A virtual platform for virtual reality application development	Allen Douglas Bierbaum	2000	F	<a href="http://oldsite.vrjuggler.org/pub/Bierbaum-VRJuggler-MastersThesis-final.pdf">http://oldsite.vrjuggler.org/pub/Bierbaum-VRJuggler-MastersThesis-final.pdf</a>

### Apêndice 3 – Formulário de Seleção dos Estudos

N	Nome	Autores	Data Publicação	Critério de Inclusão	Fonte
1	A Framework for Orthodontic Treatment Simulation	Maria Andréia F. Rodrigues , Rafael G. Barbosa , Milton E. B. Neto	2004, 31 maio a 4 junho	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
2	Desenvolvimento e Implementação de Framework de Realidade Virtual para Treinamento Médico	Ana Cláudia M. T. G. de Oliveira , Fátima L. S. Nunes , Larissa Pavarini , Leonardo C. Botega	2006 29 mai. 03 jun	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
3	Framework for Interactive Medical Imaging Applications	André Ferreira Bem Silva , Tiago H. C. Nobrega , Diego Dias Bispo Carvalho , Renan Teston Inácio , Aldo Von Wangenheim	2009 jul. 22-23	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
4	Suporte ao Uso de Frameworks Orientados a Objetos com Base no Histórico do Desenvolvimento de Aplicações	Evandro César Freiberger , Ricardo Pereira e Silva	2004, 31 maio a 4 junho	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
5	Um Framework para o Desenvolvimento de Aplicações de Realidade Virtual Baseadas em Componentes Gráficos	Thiago de A. Bastos , Alberto B. Raposo , Marcelo Gattass	2005 jul. 22-29	F	Sociedade Brasileira de Computação ( <a href="http://www.sbc.org.br/bibliotecadigital/">http://www.sbc.org.br/bibliotecadigital/</a> )
6	A DATA MODEL FOR EXPLORATION OF TEMPORAL VIRTUAL REALITY GEOGRAPHIC INFORMATION SYSTEMS	Jorge Alberto Prado de Campos	2004-08-00	F	Biblioteca Digital Brasileira de Teses e Dissertações ( <a href="http://bdtd2.ibict.br/index.php?option=com_wrapper&amp;Itemid=40">http://bdtd2.ibict.br/index.php?option=com_wrapper&amp;Itemid=40</a> )
7	Framework for dynamic evaluation of muscle fatigue in manual handling work	Liang Ma; Bennis, F.; Chablat, D.; Wei Zhang;	21-24 April 2008	F	IEEE ( <a href="http://www.ieee.org/portal/site">http://www.ieee.org/portal/site</a> )
8	The Research and Development of Fitness Equipment Embedded System Based on Distributed Virtual Environment	Xiaojun Li; Mingyan Li;	25-26 July 2009	F	IEEE ( <a href="http://www.ieee.org/portal/site">http://www.ieee.org/portal/site</a> )

9	Um processo para construção e instanciação de frameworks baseados em uma linguagem de padrões para um domínio específico.	Braga, Rosana Teresinha Vaccare	2002	F	Teses da USP ( <a href="http://www.teses.usp.br/">http://www.teses.usp.br/</a> )
10	"PARFAIT: uma contribuição para a reengenharia de software baseada em linguagens de padrões e frameworks"	Cagnin, Maria Istela	2005	F	Teses da USP ( <a href="http://www.teses.usp.br/">http://www.teses.usp.br/</a> )
11	"Um processo para construção de frameworks a partir da engenharia reversa de sistemas de informação baseados na Web: a aplicação ao domínio dos leilões virtuais"	Ré, Reginaldo	2002	F	Teses da USP ( <a href="http://www.teses.usp.br/">http://www.teses.usp.br/</a> )
12	Piavca: A Framework for Heterogeneous Interactions with Virtual Characters	José Eduardo Mendonça Xavier, Hansjörg Andreas Schneebeli	2008	F	IEEE ( <a href="http://www.ieee.org/portal/site">http://www.ieee.org/portal/site</a> )
13	Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds	Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, Victor Zappi	2008	F	Portal ACM ( <a href="http://portal.acm.org/portal.cfm">http://portal.acm.org/portal.cfm</a> )
14	The mapping problem back and forth: customizing dynamic models while preserving consistency	Tim Clerckx, Kris Luyten, Karin Coninx	2004	F	Portal ACM ( <a href="http://portal.acm.org/portal.cfm">http://portal.acm.org/portal.cfm</a> )
15	<i>basho</i> - A Virtual Environment Framework	André Hinkenjann <sup>1</sup> , Florian Mannuß	2004	F	<a href="http://cg.inf.fh-bonn-rhein-sieg.de/wp-content/uploads/2007/10/l.pdf">http://cg.inf.fh-bonn-rhein-sieg.de/wp-content/uploads/2007/10/l.pdf</a>
16	VR Juggler: A virtual platform for virtual reality application development	Allen Douglas Bierbaum	2000	F	<a href="http://oldsite.vrjuggler.org/pub/Bierbaum-VRJuggler-MastersThesis-final.pdf">http://oldsite.vrjuggler.org/pub/Bierbaum-VRJuggler-MastersThesis-final.pdf</a>



#### **Apêndice 4 – Formulário de Extração de Dados**

**Nome do Artigo:**

**Autores:**

**Data de Publicação: Veículo de Publicação:**

**Fonte:**

**Resumo:**

**Referências Relevantes:**

## Apêndice 5 – métodos adicionados para a manipulação do novo banco de dados

```
public Vector Categoria(){
    try
    {
        result = stm.executeQuery("SELECT * FROM Categoria order by
lower(desc_cat) ASC");
        categoria.clear();
        //categoria.add("");
        while ( result.next() )
        {
            String Descricao = result.getString("cod_cat");
            Descricao = Descricao + " - " +
result.getString("desc_cat");
            categoria.add(Descricao);
        }
        return categoria;
    }
    catch (SQLException e)
    {
        System.out.println("Erro no Vetor: " + e.getMessage() +
"\n");
    }
    return orgao;
}

public boolean InserirCategoria(String cod_cat, String desc_cat){
    int linhas = 0;
    try{
        linhas = stm.executeUpdate("insert into Categoria
values("+cod_cat+", '"+desc_cat+"'");
    }
    catch (SQLException e)
    {
        System.out.println("Erro: " + e.getMessage() + "\n");
    }
    finally{
        if (linhas > 0)
            return true;
        else
            return false;
    }
}

public boolean CatExiste(String cod_cat, String descricao){
    try{
        //stm.executeUpdate("delete from Categoria");
        String query = "select * from Categoria where cod_cat="+cod_cat+"
or desc_cat='"+descricao+"'";
        result = stm.executeQuery(query);
        if(result.next()){
            return true;
        }
    }
    else{
```

```

        return false;
    }
}
catch(Exception e){
    System.out.println("erro: " + e);
    return true;
}
}

public boolean removerCategoria(String cod_cat){
    try{
        int linhas = stm.executeUpdate("delete from Categoria where
cod_cat="+cod_cat+"");
        if (linhas > 0) return true;
    }
    catch(SQLException e){
        System.out.println("erro: " + e);
    }
    return false;
}

public boolean salvarCatMudancas(String cod, String desc){
    try{
        int linhas = stm.executeUpdate("update Categoria set desc_cat='" +
desc + "'" where cod_cat="+cod+"");
        if (linhas > 0) return true;
    }
    catch(SQLException e){
        System.out.println("erro: " + e);
    }
    return false;
}

public Vector Objeto(){
    Vector objeto = new Vector(30);
    objeto.clear();
    try
    {
        result = stm.executeQuery("SELECT cod_obj, nome_obj FROM Objeto
order by lower(nome_obj) ASC");
        while ( result.next() )
        {
            String descricao = result.getString("cod_obj");
            descricao = descricao + " - " + result.getString("nome_obj");
            objeto.add(descricao);
        }
        return objeto;
    }
    catch (SQLException e)
    {
        System.out.println("Erro no Vetor: " + e.getMessage() + "\n");
    }
    return null;
}

public boolean ObjExiste(String cod_obj, String nome_obj){

```

```

        try{
            //int i = stm.executeUpdate("delete from Objeto");
            //stm.executeUpdate("delete from Categoria");
            String query = "select * from Objeto where cod_obj="+cod_obj+" or
nome_obj='"+nome_obj+"'";
            result = stm.executeQuery(query);
            if(result.next()){
                return true;
            }
            else{
                return false;
            }
        }
        catch(Exception e){
            System.out.println("erro: " + e);
            return true;
        }
    }

    public boolean InserirObjeto(String cod_obj, String nome_obj, String
cod_cat, String path){
        int linhas = 0;
        try{
            linhas = stm.executeUpdate("insert into Objeto values("+cod_obj+",
 '"+nome_obj+"', '"+path+"', 0, "+cod_cat+"");
        }
        catch (SQLException e)
        {
            System.out.println("Erro: " + e.getMessage() + "\n");
        }
        finally{
            if (linhas > 0)
                return true;
            else
                return false;
        }
    }

    public String getImage(String cod){
        try{
            String query = "select path from Objeto where cod_obj="+cod+"";
            result = stm.executeQuery(query);
            if(result.next()){
                return result.getString("path");
            }
            else{
                return null;
            }
        }
        catch(Exception e){
            System.out.println("erro: " + e);
            return null;
        }
    }

    public String getCategory(String cod){

```

```

        try{
            String query = "select cod_cat from Objeto where cod_obj="+cod+"";
            result = stm.executeQuery(query);
            if(result.next()){
                return result.getString("cod_cat");
            }
            else{
                return null;
            }
        }
        catch(Exception e){
            System.out.println("erro: " + e);
            return null;
        }
    }

    public boolean removerObjeto(String cod_obj){
        try{
            int linhas = stm.executeUpdate("delete from Objeto where
cod_obj="+cod_obj+"");
            if (linhas > 0) return true;
        }
        catch(SQLException e){
            System.out.println("erro: " + e);
        }
        return false;
    }

    public boolean salvarObjMudancas(String cod, String nome, String cat,
String image){
        try{
            int linhas = stm.executeUpdate("update Objeto set cod_cat=" + cat
+ ", nome_obj = '"+nome+"', path = '"+image+"' where cod_obj="+cod+"");
            if (linhas > 0) return true;
        }
        catch(SQLException e){
            System.out.println("erro: " + e);
        }
        return false;
    }
}

```

## 7. Referências Bibliográficas

Bastos, T. A.; Raposo, A. B.; Gattas M. “Um *Framework* para o Desenvolvimento de Aplicações de Realidade Virtual Baseados em Componentes Gráficos”. In: XXV Congresso Da Sociedade Brasileira De Computação, 2005, São Leopoldo.

Bellotti, Francesco; Berta, Riccardo; Gloria, Alessandro De; Zappi, Victor. “Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds”. 2008.

Bierbaum, Allen Douglas. “VR Juggler: A virtual platform for virtual reality application development”. 2000.

Braga, Rosana Teresinha Vaccare. “Um processo para construção e instanciação de *frameworks* baseados em uma linguagem de padrões para um domínio específico”. 2002.

Cagnin, Maria Istela. “PARFAIT: uma contribuição para a reengenharia de software baseada em linguagens de padrões e *frameworks*”. Setembro, 2005

CAMPOS, Jorge Alberto Prado. “A DATA MODEL FOR EXPLORATION OF TEMPORAL VIRTUALREALITY GEOGRAPHIC INFORMATION SYSTEMS”. August, 2004.

Cerckx, Tim; Luyten, Kris; Coninx, Karin. “The mapping problem back and forth: customizing dynamic models while preserving consistency”. 2004.

CORRÊA, C. G. “IMPLEMENTAÇÃO E AVALIAÇÃO DE INTERAÇÃO EM UM *FRAMEWORK* PARA TREINAMENTO MÉDICO”. 2008

Delfino, Sérgio Roberto. “Geração de Estudos de Caso para Treinamento Médico Virtual a partir de Técnicas de Processamento de Imagens e Realidade Virtual”. 2007

DiVerdi, S.; Numi, D.; Hollerer, T. “A *framework* for generic inter-application interaction for 3D AR environments”. 2003

FRATA , ANA CLAUDIA CARREIRA. “Avaliação heurística da aplicação *Wizard* do *framework ViMeT*”. 2009

Freiberger , Evandro César. Suporte ao Uso de *Frameworks* Orientados a Objetos com base no Histórico do Desenvolvimento de Aplicações. 2002. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Santa Catarina, . *Orientador*: Ricardo Pereira e Silva .

Gillies , Marco. “*Piavca: A Framework for Heterogeneous Interactions with Virtual Characters*”. 2008.

Hinkenjann<sup>1</sup>, André ; Mannuß, Florian. “*basho - A Virtual Environment Framework*”. 2004

Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical report, Keele University and NICTA.

LI, Xiaojun; Li, Mingyan. “The Research and Development of Fitness Equipment Embedded System Based on Distributed Virtual Environment”. July, 2009.

LIU, A.; TENDICK, F.; CLEARY, K.; KAUFMANN, C .A Survey of Surgical Simulation: Application, Technology and Education. In: MIT Press, vol. 12, Dezembro, 2003.

MA, Liang; BENNIS , Fouad; CHABLAT, Damien. “*Framework for Dynamic Evaluation of Musde Fatigue in Manual Handling Work*”. April, 2008.

Mafra, S.; Travassos, G.H. (2006) “Estudos Primários e Secundários Apoiando a Busca por Evidência em Engenharia de Software”. Technical Report ES-687/06, PESC/COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MONTEIRO, B. S.; VALDEK, M. C. O.; CUNHA, Í. L., MORAES; R. M., MACHADO, L. S. Anotomi: 3D: Atlas Digital Baseado em Realidade Virtual para Ensino de Medicina. In: SVR 2006 – VII Symposium on Virtual Reality, 2006, Belém. Proceedings... Belém: CESUPA, 2006. p. 13-14.

Nunes, Fátima L. S., “Metodologia Científica – Aula 10”.

OLIVEIRA, Ana Cláudia Melo Tiessi Gomes de ; NUNES, F. L. S. ; PAVARINI, L. ; BOTEGA, L. C. . Desenvolvimento e Implementação de *Framework* de Realidade Virtual para Treinamento Médico. In: WIM'2006 - VI Workshop de Informática Médica, 2006, Vilha Velha. Anais VI Workshop de Informática Médica, 2006.

OLIVEIRA, A. C. M. T. G. “*ViMeT – PROJETO E IMPLEMENTAÇÃO DE UM FRAMEWORK PARA APLICAÇÕES DE TREINAMENTO MÉDICO USANDO REALIDADE VIRTUAL*”. 2007

OLIVEIRA, R. A. P. ; DELAMARO, M. E. ; NUNES, F. L. S. . Oráculos de Teste para Domínios GUI: Uma Revisão Sistemática. In: III Brazilian Workshop on Systematic and Automated Software Testing - SAST 2009, 2009, Gramado - RS - Brasil. Anais do III Brazilian Workshop on Systematic and Automated Software Testing, 2009. v. 3. p. 22-30.

RÉ, Reginaldo. “Um processo para construção de *frameworks* a partir da engenharia reversa de sistemas de informação baseados na Web: aplicação ao domínio dos leilões virtuais”. 2002.

RODRIGUES, M. A. F. ; BARBOSA, R. G. ; BARBOSA NETO, Milton Escossia ; RIBEIRO, Isabel Maria Magalhães P. . A *Framework* for Orthodontic Treatment Simulation. In: iV Workshop de Informática Médica (WIM), em III SBQS, 2004, Brasília-DF. Anais do WIM 2004, 2004.

Silva, André Ferreira Bem, Tiago H. C. Nobrega, Diego Dias Bispo Carvalho, Renan Teston Inácio, and Aldo vonWangenheim. “*Framework* for Interactive Medical Imaging Applications,” July 2009.

SUN (2006). Java 3D API Tutorial. Disponível em: <http://java.sun.com/developer/onlineTraining/java3d/>