UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

SIMULADOR DE ANIMAIS VIVOS: MEIOS ALTERNATIVOS

THOMAS DA ROSA

BLUMENAU
2008

2008/2-25

THOMAS DA ROSA

SIMULADOR DE ANIMAIS VIVOS: MEIOS ALTERNATIVOS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, M.sc. - Orientador

BLUMENAU
2008

2008/2-25

SIMULADOR DE ANIMAIS VIVOS: MEIOS ALTERNATIVOS

Por

THOMAS DA ROSA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, M.Sc. – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Dr. – FURB
Membro:	Prof. Paulo Cesar Rodacki Gomes, Dr. – FURB

Blumenau, 11 de fevereiro de 2009

Dedico este trabalho a meus pais que estiveram ao meu lado, e a todos os amigos que, de alguma forma, me auxiliaram na realização deste.

AGRADECIMENTOS

Aos meus pais, Emílio Medeiros da Rosa e Marilene da Rosa, que sempre me direcionaram no caminho dos estudos, prestando seu apoio e carinho.

Aos meus irmãos, Thiago da Rosa e Thaís Scharianne da Rosa, por me ajudarem a superar os estreasses que passamos em nossas vidas.

Aos amigos da FURB, que me "aturam" todos os dias, pessoal do APUS e da Computação. Estes que também que me cobravam muito, querendo ver o simulador, me impulsionando a progredir.

Ao meu orientador Dalton Solano dos Reis, que sempre que estive perdido me indicou o caminho, e sempre que estive desesperado mostrou-me que ainda havia esperança.

Ao professor Silvio Luiz Negrão e as acadêmicas Ana Julia Girardi e Gisele Floriani, que colaboraram com a parte técnica da Medicina Veterinária.

A todos vocês, muito obrigado.

Somos do tamanho dos nossos sonhos.

Fernando Pessoa

RESUMO

Este trabalho apresenta o desenvolvimento de um simulador de animais vivos para a automatização da aula de Indução da Dor seguida da Administração de Antiinflamatórios (IDAA) da disciplina de Farmacologia do curso de Medicina Veterinária da FURB. Para isto são mapeados procedimentos para a execução desta aula em laboratório e sua relevância ética. Sobre as técnicas utilizadas são efetuados efeitos de *blend* para mistura de camadas de imagens, técnicas de *floodfill* para preenchimento e inundação das artérias do animal e técnicas de desenho em JOGL. Também são possibilitadas aplicações de elementos químicos e geração de relatório como auxílio ao aprendizado.

Palavras-chave: Simulador. Medicina veterinária. Animação 2D. *Blend. Floodfill.*

ABSTRACT

This work describes the development of a simulator of life animals to automate the Induction of the Pain next Anti-inflammatory Administration of the Pharmacology lesson of the Veterinary Medicine course of the FURB. To do this are mapped procedures of this lesson laboratory execution and it's ethics relevance. About the techniques are used blend effects to mix image layers, technique of floodfill to filling and inundation of animal arteries and draw techniques in JOGL. Also are possible chemical elements administration and report generation, everything to aid the learning.

Key-words: Simulator. Veterinary medicine. 2D animation. Blend. FloodFill.

LISTA DE ILUSTRAÇÕES

Figura 1 – Efeito de transparência com refração 20
Figura 2 – Efeito de transparência com reflexão 20
Figura 3 – Tipos de verificação conexa 23
Figura 4 – Exemplo de inundação *floodfill* básico 23
Figura 5 – Exemplo de inundação *floodfill* complexo 23
Figura 6 – Verificação da vantagem *pixels* 8-conexos 24
Figura 7 – Simulação dos efeitos de drogas no coração de um sapo 27
Figura 8 – Simulação software Isolated Phrenic Nerve – Diaphragm 28
Figura 9 – Tutorial sobre o estudo dos reflexos de córneas 29
Figura 10 – Diagrama de casos de uso 31
Quadro 1 – Caso de uso Visualizar camadas 32
Quadro 2 – Caso de uso Visualizar *hitens* 32
Quadro 3 – Caso de uso Aplicar componente químico no animal 33
Quadro 4 – Caso de uso Exportar imagem da simulação 34
Quadro 5 – Caso de uso Exportar imagem da simulação 34
Figura 11 – Diagrama de classes. 35
Figura 12 – Diagrama de classes. 37
Figura 13 – Diagrama de transição de estados 38
Quadro 6 – Trecho do código da *thread* do eletro 40
Quadro 7 – Trecho do código para desenhar linha do eletro 40
Quadro 8 – Trecho do código para desenhar texto do eletro 41
Quadro 9 – Trecho do código para desenhar texto do eletro 41
Quadro 10 – Trecho do código para desenhar texto do eletro 42
Quadro 11 – Trecho do código da responsável pelo preenchimento todos os *pixels* 42
Quadro 12 – Trecho do código responsável pela leitura do RGB do *pixel* 43
Quadro 13 – Trecho do código responsável pelo teste de tolerância da cor vermelho 43
Quadro 14 – Trecho do código responsável pela leitura do RGB do *pixel* 44
Figura 15 – Inserção do peso do cachorro 45
Figura 16 – Início da simulação 46
Figura 19 – Imagem pontos dos *hitens* 47
Figura 20 – Imagem *hiten* esmaecido 47

Figura 21 – Exportar imagem para arquivo 48
Figura 22 – Selecionar elemento químico 48
Figura 23 – Parametrizar quantidade de elemento químico 49
Figura 24 – Animação do elemento químico formalina 49
Figura 25 – Animação do elemento químico dexametasona 50
Figura 26 – Pré-visualização do relatório de simulação 50
Figura 27 – Exemplo de relatório gerado em arquivo PDF 52

SUMÁRIO

1 INTRODUÇÃO 12
1.1 OBJETIVOS DO TRABALHO 13
1.2 ESTRUTURA DO TRABALHO 13
2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 SIMULADORES 15
2.1.1 Simulações na Farmacologia – Medicina Veterinária 16
2.1.2 Simulação da aula prática IDAA 16
2.1.3 Questões Éticas das simulações em Farmacologia 18
2.2 MULTIMÍDIA 19
2.2.1 Técnicas de tratamento de imagens 19
2.2.1.1 Blend de imagens 20
2.2.1.2 Técnica de preenchimento por inundação – floodfill 21
2.2.2 Manipulação de áudio 24
2.2.3 Manipulação de vídeo 25
2.3 TRABALHOS CORRELATOS 25
2.3.1 Expharm version T1.00 26
2.3.2 Isolated phrenic nerve – diaphragm 27
2.3.3 Microlabs for pharmacologists 28
3 DESENVOLVIMENTO 30
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 30
3.2 ESPECIFICAÇÃO 31
3.2.1 Diagrama de casos de uso 31
3.2.2 Diagrama de classes 34
3.2.3 Diagrama de seqüência 36
3.2.4 Diagrama de transição de estados 38
3.3 IMPLEMENTAÇÃO 39
3.3.1 Técnicas e ferramentas utilizadas 39
3.3.1.1 Desenho das linhas e texto do eletro 39
3.3.1.2 Blend das imagens 41
3.3.1.3 Preenchimento – floodFill 42
3.3.2 Operacionalidade da implementação 44

3.3.2.1 Iniciando a simulação 45
3.4 RESULTADOS E DISCUSSÃO 53
4 CONCLUSÕES 55
4.1 EXTENSÕES 56
REFERÊNCIAS BIBLIOGRÁFICAS 57

1 INTRODUÇÃO

Atualmente na Medicina e Veterinária são movidos grandes esforços para aprimoramento de técnicas e procedimentos. Esses esforços levam ao desenvolvimento de novos medicamentos e pesquisa de novos meios de tratamentos. Dentre seus avanços destacam-se a utilização do estetoscópio[1], termometria[2], medição da pressão arterial e Farmacologia[3].

No caso da Farmacologia, realizam-se testes em animais para analisar a utilização de substâncias químicas para tratamento, cura e prevenção de doenças. Há uma preocupação emocional nestes testes, dado ao fato de serem executados em animais vivos. Por outro lado, há um grande investimento financeiro para o desenvolvimento de medicamentos, visando resolução de doenças como por exemplo: o mal da vaca-louca, gripe do frango e até malária.

Muitos centros de pesquisa e universidades abordam a Farmacologia, efetuando os mais diversos testes em animais. Dentre eles o curso de Medicina Veterinária da Universidade Regional de Blumenau (FURB) executa, na disciplina de Farmacologia, a aula prática de "Indução da Dor seguida da Administração de Antiinflamatórios (IDAA)". Esta aula prática de IDAA gera sérios problemas éticos em função da dor infligida aos animais e a possibilidade de morte dos mesmos. Vários defensores dos animais lutam pela criação de políticas para proibição desta prática e "taxam" de "barbárie" os atos cometidos a estes animais.

Uma forma alternativa para este cenário é a adoção de simuladores. Os simuladores trabalham recursos gráficos de imagem, áudio e vídeo na tentativa de reproduzir algum sistema ou fenômeno o mais próximo possível da vida real. Atualmente existem inúmeros simuladores disponíveis no mercado, cada um com suas vantagens e desvantagens. Mas, o que se constata é que não existe um simulador que atenda as necessidades da aula prática IDAA de Farmacologia.

É importante ressaltar também que a possibilidade de aliar os experimentos de uma aula prática em laboratório com uma aula teórica com explicações destes experimentos torna

---

[1] Aparelho utilizado para amplificar sons corporais, como sons cardíacos e sons dos pulmões.

[2] Estudo da medida do grau de agitação molecular. Essa agitação transforma-se em calor que é utilizado para estudar as alterações das temperaturas corpóreas (SPINOSA; GÓRNIAK; BERNARDI, 2006, p. 117).

[3] Ciência que estuda a origem, as ações e as propriedades das substâncias químicas nos organismos vivos (RANG; DALE, 1993, p. 3).

o simulador uma ferramenta de potencial para o ensino local e a distância. Seu auxílio na educação pode trazer ao conhecimento dos alunos equipamentos que não se poderia estudar por falta de recursos. A manipulação dos animais inicialmente utilizando o simulador pode abordar ferramentas não comuns em laboratório, observando os efeitos gerados por sua manipulação, de forma próxima da real e com a melhor didática possível. Outro fator que impulsiona a utilização de simuladores é evitar riscos aos animais, podendo evitar traumas psicológicos aos seus utilizadores.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é implementar um simulador para a aula prática de IDAA da disciplina de Farmacologia do curso de Medicina Veterinária da FURB.

Os objetivos específicos do trabalho são:

a) mapear procedimentos da aula prática de IDAA, bem como seus conceitos teóricos, viabilizando padrões de ações e reações dos animais utilizados;
b) transformar as ações, reações e procedimentos da aula prática de IDAA para a forma gráfica, utilizando recursos de multimídia e computação gráfica;
c) disponibilizar um simulador para a aula IDAA para ser utilizado na disciplina de Farmacologia do curso de Medicina Veterinária da FURB.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em quatro capítulos. O segundo capítulo contém a fundamentação teórica, onde são verificados os tópicos necessários para o entendimento e desenvolvimento do simulador.

A seção 2.1 concentra-se a apresentação do conceito geral de simuladores, passando na seção 2.1.1 pelas simulações farmacológicas na Medicina Veterinária. Após na seção 2.1.2 é apresentada a aula prática de IDAA assim como seu roteiro em laboratório para após abordar na seção 2.1.3 as questões éticas com relação a estas simulações.

Na seção 2.2 são apresentados conceitos de multimídia e computação gráfica. A seção

2.2.1 apresenta fundamentação técnica de tratamento de imagens, tendo a subseção 2.2.1.1 abordagem do *blend* de imagens e a subseção 2.2.1.2 a técnica de preenchimento por inundação (*floodfill).* Na seção 2.2.2 são abordados conceitos de manipulação de áudio e na seção 2.2.3 para fechamento da seção conceitos de manipulação de vídeo.

A seção 2.3 tem o foco em mostrar as ferramentas e simuladores correlatos onde são apresentados os softwares 2.3.1 Expharm version T1.00, 2.3.2 Isolated phrenic nerve – diaphragm e 2.3.4 Microlabs for pharmacologists.

O capítulo 3 apresenta a especificação e implementação do simulador. Tendo as seções 3.2.1 o diagrama de casos de uso, 3.2.2 diagrama de classes, 3.2.3 diagrama de seqüência e 3.2.4 diagrama de estados. Na seção 3.3 é mostrada a implementação do simulador com suas técnicas utilizadas, a operacionalidade do simulador, assim como os resultados e discussão.

No capítulo 4 são apresentadas as conclusões deste trabalho bem como possíveis extensões em futuros trabalhos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns conceitos teóricos e da aula prática IDAA que irão permitir um melhor entendimento do trabalho, a começar por uma abordagem geral de simuladores. Também são abordados conceitos de simulações farmacológicas, dentro da medicina veterinária, os procedimentos existentes atualmente para a realização da aula prática de IDAA e as questões éticas que envolvem testes em animais. Na seqüência são pesquisadas rotinas de computação gráfica e conceitos técnicos de multimídia, tratamento de imagens, vídeo e manipulações de áudio, assim como *frameworks*. Por fim são listados alguns trabalhos correlatos.

2.1 SIMULADORES

Simulador, do latim *simulatore,* consiste em simular, aquilo que simula ou para questões físicas um aparelho de instrução e treino que simula as condições de funcionamento de outro aparelho (MICHAELIS, 2007). Na área da computação um simulador consiste basicamente em criar um cenário virtual que propicie sua execução o mais próximo possível do mundo real e que aborde o maior número de variáveis reais possíveis (SCHULTER, 2007).

Atualmente uma grande variedade de empresas e governos vem cogitando e implementando soluções que envolvem a utilização de simuladores. Esta necessidade se confirma pelo fato de que seus projetos são muito complexos, custosos e de alto risco, especialmente a vida humana.

“Suponha um simulador de vôos, onde o piloto adquire experiência sem o risco, tempo e custo associados com o equipamento real. Ao piloto é permitido errar em ambientes simulados, pois com os erros em terra aprende-se e evita-se erros no ar. De modo semelhante pode-se aproximar, dos estudantes ou funcionários de uma empresa, a realidade dos sistemas de manufatura, aumentando a compreensão e ganhando muito tempo.” (CORNÉLIO FILHO, 1998).

2.1.1 Simulações na Farmacologia – Medicina Veterinária

Pouco depois do século XIX, quando o homem fez seu primeiro teste farmacológico, ganhou vida a ciência que estuda a administração de componentes químicos em seres vivos, a Farmacologia. De fato, ela limita-se a estudar a aplicação de drogas em seres vivos e os efeitos ocasionados por elas, sejam estes efeitos para fins terapêuticos ou não. Graças a estes estudos, hoje sabe-se que um analgésico pode aliviar a dor e um antiinflamatório pode tratar uma infecção (SCHEINDLIN, 2001).

Para entender as ações de uma droga, é necessário considerar os efeitos produzidos no sistema biológico e, para isso, cursos de Medicina Veterinária oferecem a disciplina de Farmacologia e laboratórios para a execução de testes com drogas em animais.

Um método alternativo ao aprendizado em laboratório é o emprego de simuladores, onde torna-se possível “simular” e criar “novas situações sobre as quais se tenha pouco conhecimento e experiência podem ser tratadas, de tal forma que se possa ter, teoricamente, alguma preparação diante de futuros eventos” (VIU, 2008). Outros fatores importantes a serem observados são os custos com laboratórios, riscos envolvidos nas simulações, necessidade de conhecimentos conceituais em sala de aula e uma boa dose de prática em laboratório, os quais podem ser automatizados em um simulador (CORNÉLIO FILHO, 1998).

Devido a sua versatilidade, didática e facilidade de uso, principalmente com programas orientados a objetos, diversas áreas têm utilizado simuladores como ferramentas de projeto, análise e ensino. Com possibilidades de que estudantes, em apenas alguns semestres de seus cursos, adquiram conhecimento suficiente para a execução de testes em animais vivos.

2.1.2 Simulação da aula prática IDAA

A aula prática IDAA é uma simulação em laboratório oferecida pelo curso de Medicina Veterinária da FURB e tem como objetivo o estudo das ações e reações de analgésicos e antiinflamatórios em organismos vivos, mais especificamente em camundongos e cachorros.

Para e execução desta aula prática se faz necessário seguir o roteiro da aula prática de IDAA, a fim de descobrir os efeitos destes antiinflamatórios para o alívio da dor. Segundo a

associação Internacional para o Estudo da Dor, define-se a dor como sendo “uma experiência sensorial ou emocional desagradável associada à lesão tissular[4] real ou potencial, ou descrita em termos de tal lesão” (HELLEBREKERS, 2002, p. 49). “Uma vez que a dor está relacionada à forma pela qual a sensação desagradável ou aversiva é experimentada pelo indivíduo, seria mais correto limitar o uso deste termo em animais” (HELLEBREKERS, 2002, p. 69). Através desta sensação desagradável, o sistema nervoso reconhece através dos receptores sensoriais e emite sinais relacionados a dano tissular. Estes sinais são interpretados pelo cérebro e levam a variadas reações, que pode-se descrever (sentir).

Os animais por sua vez não conseguem descrever sua dor, o que necessita de métodos de observação que atentem a qualquer comportamento manifestado pelo animal. Estes métodos devem avaliar o comportamento normal do animal, o comportamento da espécie do animal, a experiência do observador e por fim o comportamento do animal frente à dor. Submeter o animal a testes de reflexo auxilia a traçar um perfil comportamental do animal, geralmente obtido por estímulo doloroso agudo, elétrico, térmico ou mecânico.

Desta forma, a aula prática de IDAA consiste basicamente em observar os efeitos das drogas sobre um camundongo ou cachorro. Uma droga comum utilizada é a dexametasona. Essa droga é um glicocorticóide[5] usado principalmente por seus potentes efeitos antiinflamatórios. O efeito principal deste medicamento é a profunda alteração promovida na resposta imune linfocitária, devido à ação antiinflamatória e imunossupressora, podendo prevenir ou suprimir processos inflamatórios de várias naturezas. Outro produto químico utilizado para a aula prática de IDAA é a formalina, uma mistura composta por gás formol (37%) e água (63%). Esta substância emite desconforto e sensação de queimadura na área da aplicação.

Durante uma aula prática de IDAA, inicialmente o camundongo ou cachorro é pesado. Com a pesagem pode-se determinar as quantidades de produto químico necessárias para aplicar sobre ele. Por exemplo, a dexametasona é um produto que necessita uma aplicação de 50mg por quilo. Em seguida o camundongo ou cachorro é colocado em uma ambiente calmo para sua estabilização. Após 10 minutos é administrada dexametasona (intraperitoneal[6]). Espera-se 25 minutos e em seguida administra-se 20µl de formalina na pata. Por fim é

[4] Termo referente a tecido celular. Comumente utilizado na medicina para referenciar o órgão chamado “pele”.

[5] Hormônio desenvolvido em laboratório para afetar o metabolismo de carboidratos, proteínas e gorduras. Dentre seus efeitos além de antiinflamatórios pode alterar funções motoras do corpo, funções neuronais dentre outras (RANG; DALE, 1993, p. 312).

[6] Administração através do peritoneo. O peritoneo é uma pequena e transparente membrana que delimita e engloba os órgãos intestinais, como estômago e intestino (BOWEN, 2006).

marcado o tempo de lambida da pata durante 20 minutos com o animal dentro do funil.

Como resultado observa-se que o camundongo ou cachorro lambe a pata durante 40 segundos nos 10 primeiros minutos e 16 segundos nos 10 minutos seguintes. Neste caso constata-se que a injeção de dexametasona teve um efeito de diminuição da lambida na pata ocasionado pela formalina. Este efeito deu-se pela administração de dexametasona que afeta quase todas as células, influenciando o metabolismo e agindo sobre as infecções e atenuando a dor.

2.1.3 Questões Éticas das simulações em Farmacologia

Muitas organizações não governamentais e sociedades protetoras de animais lutam contra a utilização de animais na execução de testes farmacológicos e sua principal alegação é que todo o ser vivo tem direito a vida e principalmente que a vida deve ser respeitada.

O que sabe-se é que os testes em animais vem de longa data, através dos estudos de Hipócrates (450 a.C), e tem se intensificado ao longo dos anos em função dos avanços ocasionados especialmente na área da saúde. Isso tornou-se um fator de constante preocupação no meio acadêmico, forçando a criação de normas e princípios orientadores para o uso de testes em animais, criadas por diversas instituições nacionais e internacionais com o intuito de orientar seus pesquisadores. (RAYMUND; GOLDIM, 2002)

Atualmente o Brasil segue aprovando várias leis que protegem os animais de testes que vão contra a ética e moral, um exemplo é a Lei de Crimes Ambientais de 1998 que além de impedir maus-tratos, mutilações e ferimentos em qualquer tipo de animais sob pena de detenção e multa. A Lei também no primeiro parágrafo diz que “incorre nas mesmas penas quem realiza experiência dolorosa ou cruel em animal vivo, ainda que para fins didáticos ou científicos, quando existirem recursos alternativos” (ANTEPROJETO de lei, 1998). O código de proteção aos animais, Lei Nº 11.977, 25 de Agosto de 2005 do estado de São Paulo, afirma que “A experimentação animal fica condicionada ao compromisso moral do pesquisador ou professor, firmado por escrito, responsabilizando-se por evitar sofrimento físico e mental ao animal, bem como a realização de experimentos cujos resultados já sejam conhecidos e demonstrados cientificamente” (ANTEPROJETO de lei, 2005).

A regulamentação de leis que proíbam a experimentação científica está forçando, especialmente as universidades, a busca de meios alternativos aos apresentados atualmente e embora se escreva muito sobre o *status* moral dos animais ainda não existe um consenso sobre

a verdadeira posição que os eles ocupam em relação aos seres humanos (RAYMUND; GOLDIM, 2002).

2.2 MULTIMÍDIA

Para o desenvolvimento de simuladores é necessário o emprego de diversos recursos gráficos. Alguns destes recursos consistem na utilização de imagens, áudio e vídeo.

2.2.1 Técnicas de tratamento de imagens

Dentre os recursos de imagem existem dois tipos, imagem vetorial e imagem *raster*. As imagens vetoriais são geradas a partir de geometria, já as imagens *raster* são formadas por uma matriz que contém a descrição de cada *pixel* na tela. Estes dois tipos de imagem, *raster* e vetorial, contam com o sistema de combinação de cores *Red-Green-Blue* (RGB). Este sistema consiste nas três cores primárias, o vermelho, o verde e o azul, que podem ser combinados formando novas cores (FERNANDES, 2002, p. 36).

Existem atualmente vários tipos de formatos de imagens em arquivos e muitos softwares para manipulação destas imagens. Estes formatos podem variar de acordo com a capacidade de armazenar a imagem. Dentre os formatos tem-se: *Joint Pictures Expert Group* (JPEG), orientado para imagens *raster* e existe possibilidade de compressão com perdas; *Portable Network Graphics* (PNG) orientado para imagens *raster* com compressão sem perdas; *Computer Graphics Metafile* (CGM), formato independente de fabricante e orientado para imagens vetoriais de padrão bidimensional.

Com relação aos softwares para manipulação de imagens tem-se como exemplo: Corel Photo-Paint, software pago; Adobe PhotoShop, software pago e Gimp, software livre. Estes softwares permitem realizar vários tipos de processamento, tornando-se recursos muito importantes na manipulação da imagem e possibilitando alterações desde pincéis para retoque, escalonamento de imagem, realce de bordas até efeitos de anti-aliasing[7] entre outros (PAULA FILHO, 2000, p. 130).

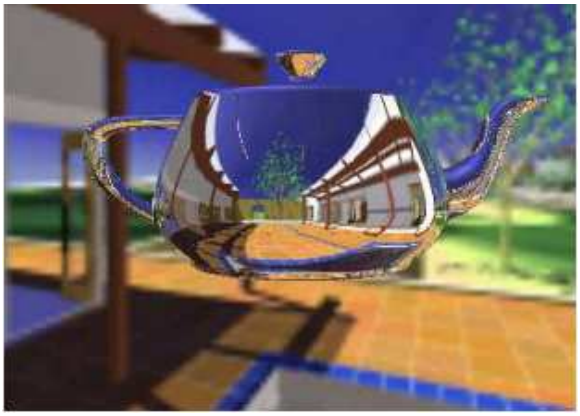
2.2.1.1 Blend de imagens

Uma técnica muito utilizada na computação gráfica é o *Blend.* Esta técnica consiste em exibir objetos transparentes através de uma combinação das cores de seus objetos. Segundo (OPENGL, 2008) transparência é um “objeto translúcido e sem reflexão de luz em sua superfície.” Também afirma que “Um material em perfeita transparência é completamente invisível” e que “todo material transparente reflete alguma luz que passa por ele, tornando o material exatamente após este visível”, ou seja, a luz reflete nos materiais dependendo do nível de transparência do mesmo. Desta forma podemos determinar diversos tipos de transparência, sendo a transparência de um plástico diferente da transparência de um vidro.

Objetos curvos também determinam outros tipos de transparência, como a refração (figura 1) e a reflexão (figura 2).

Fonte: Fernandes, 2006.

Figura 1 – Efeito de transparência com refração



Fonte: Fernandes, 2006.

Figura 2 – Efeito de transparência com reflexão

A imagem da esquerda demonstra o efeito de refração que acontece porque um feixe de luz, incidindo obliquamente, muda de direção quando passa de um meio transparente para outro transparente. Esta mudança de direção apresenta velocidades da luz diferentes, um pequeno atraso na mudança, e o fenômeno chamado de refração (EDUCAR, 2003). Já a imagem da direita apresenta efeito de reflexão onde a luz é refletida formando um espelho, mas nesta imagem existe uma reflexão diferente chamada de reflexão difusa, ou seja, ocorre numa superfície irregular com seus raios espalhados desordenadamente em todas as direções (LASALLE, 2003).

---

[7] Efeito que remove o serrilhado da imagem. Observado no escalonamento de imagens (PAULA FILHO, 2000, p. 132).

Alguns destes efeitos de *blend* em OpenGL estão disponíveis na biblioteca JOGL, e a combinação de cores que o efeito de *blend* possibilita, é possível apenas quando desenhados dois objetos um na frente do outro. A transparência se dará baseada no valor a*lpha*, nas cores do objeto e na função de *blend* que está sendo usada. O valor *alpha* geralmente é o quarto atributo da cor e seu valor pode variar de 0.0 para um objeto transparente e 1.0 para um objeto opaco. Em OpenGL é chamado com a função `GL_RGBA`, já no JOGL é chamado pelo método `glColor4f()` da classe `GL` (NEHE, 2008).

Este método `glColor4f()` pode ser utilizado para realizar o *blend* de imagens, mas para o sucesso deste método deve-se contar com outra função OpenGL que é o `glBlendFunc`. Esta função recebe dois parâmetro: o primeiro define o peso da cor do novo ponto e o segundo o peso da cor do ponto que já está na tela. Este pesos, em OpenGL, são sempre uma função do nível de transparência do objeto, ou seja, de seu valor *alfa* (quadro 1).

```
gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);
//Define que a nova cor do ponto na tela será:
//NovaCorNaTela=CorDoObjeto*AlfaDoObjeto+CorAntigaNaTele*(1-AlfaDoObjeto);
gl.glColor4f(0f,1f,0f,0.5f);
//Definição da cor verde com transparência de 50%.
```

Quadro 1 – Função *blend* do JOGL

A função glBlendFunc possui duas constantes. `GL_SRC_ALPHA` define que o peso da cor do objeto que está sendo desenhado é o próprio valor *alfa* de sua cor, já a constante `GL_ONE_MINUS_SRC_ALPHA` define que o peso da cor que já está na tela é de um menos *alfa*, onde *alfa* é o nível de transparência do objeto que está sendo desenhado. A definição da cor, `glColor4f(0f,1f,0f,0.5f),` faz com que qualquer objeto desenhado após ela fique com uma cor verde e com uma transparência de 50%.

2.2.1.2 Técnica de preenchimento por inundação – floodfill

Preenchimento por inundação é uma técnica muito comum utilizada para a retransmissão de informações, onde cada objeto se encarrega de transmitir determinada informação para seus vizinhos. Existem certas especificações e cuidados que deve-se observar na utilização desta técnica de inundação, tanto na preservação de sua informação anterior quanto a nova informação que está sendo atribuída.

Sua aplicação é encontrada em várias áreas (ALGORITMO, 2008):

a) redes de computadores: utilizada para distribuir informação para os nós de uma rede conectada (programas de compartilhamento de arquivos e protocolos de

roteamento);

b) matemática: utilizada para resolver vários problemas, inclusive problemas de labirinto e os envolvidos com a teoria de grafos. O algoritmo de inundação eventualmente passa por todos os pontos do grafo, a partir de qualquer ponto, desde que haja um caminho válido para tal;
c) computação gráfica: com sua aplicação é possível determinar a área conectada de um determinado nó. Considerando-se uma imagem *bitmap* como sendo o vetor bi-dimensional, e um *pixel* como sendo o nó, é possível utilizar o algoritmo para preencher toda uma área delimitada por uma cor com uma outra cor, a partir de um ponto inicial. Sua aplicação é geralmente utilizada por softwares de edição de imagem, através de ferramentas como a lata de tinta.

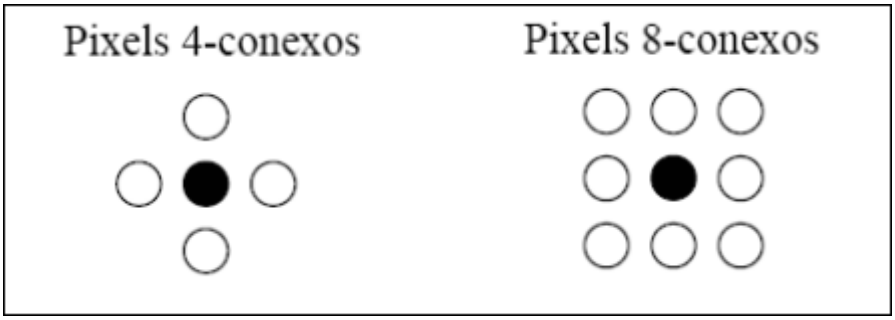
Na computação gráfica é comumente utilizada para verificação de cores em imagens, tanto *raster* quanto vetorial. Sua idéia básica é começar num determinado ponto reconhecidamente dentro da região a ser preenchida, chamado de semente, e a partir daí percorrer e preencher seu interior.

Para sua utilização na computação gráfica é necessário especificar duas variáveis (FERREIRA; REGOBELLO, 2007):

a) seu interior: deve-se fornecer um *pixel* no interior da região, que servirá como semente. Este *pixel* determinará a região conexa, ou seja, todos os *pixels* que estiverem conectados a ele, e que satisfaçam a fronteira definida, serão afetados;
b) sua fronteira: deve-se fornecer os atributos dos *pixels* da imagem, estes atributos serão verificados para determinarão a região conexa através desta fronteira.

Para este algoritmo fornecem-se como dados de entrada as coordenadas (x,y) de um *pixel* semente na imagem, a cor deste determinado *pixel*, e a cor da fronteira ou faixa de tolerância desta cor. Na seqüência o processo de preenchimento começa, a partir do ponto semente, a verificar todos os *pixels* vizinhos em busca de *pixels* que satisfaçam as condições da fronteira. Desta forma se a cor estiver dentro da faixa especificada o *pixel* será pintado.

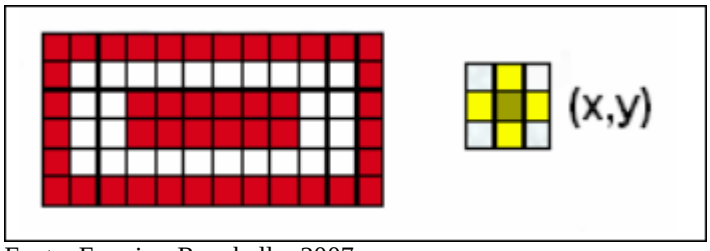
Define-se *pixels* conexos onde haja conectividade entre eles, ou seja, para qualquer par de *pixels* no conjunto deve existir uma seqüência que que se inicia no primeiro e termina no segundo. Desta forma e por agir no preenchimento dos *pixels* conexos, define-se duas formas de implementar o *floodfill*. *Pixels* 4-conexos verifica apenas quatro vizinhos, já o *pixels* 8-conexos verifica oito vizinhos (figura 3).



Fonte: Ferreira; Regobello, 2007.

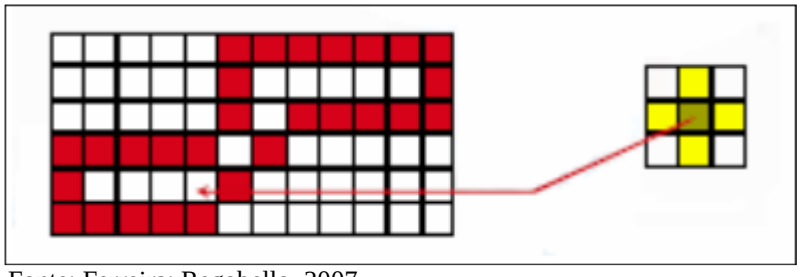
Figura 3 – Tipos de verificação conexa

O conjunto de *pixels* 4-conexos é recomendado para formas simples, como a figura 4, mas apresenta dificuldades em formas mais complexas como da figura 5. A implementação de *pixels* 8-conexos mostra-se mais eficiente porque possibilita percorrer os *pixels* na diagonal (figura 6). Vale ressaltar que a verificação *pixels* 8-conexos requer a visita em oito visinhos tornando o processo mais lento e "pesado". Como exemplo, é possível verificar na figura 4 a existência de dois ambientes conexos, um representado por uma cor clara e outro representado por uma cor escura. Observa-se que, se for executada a técnica pixels 4-conexos para preencher a cor clara com outra cor, é possível ser preenchida facilmente. Já a figura 5 é possível observar que o pixels 4-conexos não consegue preencher totalmente a figura, parando no pixel apontado pela flecha e impossibilitando o preenchimento na diagonal. Observa-se também que a técnica pixels 8-conexos consegue este preenchimento é possível verificar na figura 6 e enfatizados pelas setas.



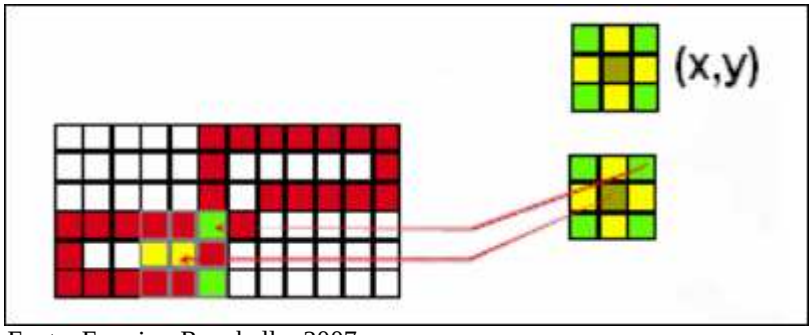
Fonte: Ferreira; Regobello, 2007.

Figura 4 – Exemplo de inundação *floodfill* básico



Fonte: Ferreira; Regobello, 2007.

Figura 5 – Exemplo de inundação *floodfill* complexo



Fonte: Ferreira; Regobello, 2007.

Figura 6 – Verificação da vantagem *pixels* 8-conexos

As técnicas de preenchimento apesar de serem robustas também apresentam certos problemas que devem ser avaliados e corrigidos para sua implementação. Um risco muito importante que deve-se observar é o de implosão. Como o *pixel* sempre percorre todos os seus vizinhos, ele não considera se o vizinho já foi percorrido e atualizado por outro *pixel*, fazendo com que ocorra risco de *loops* infinitos. Além disto, a visita a um *pixel* que já tenha sido percorrido também causa um desperdício de recursos do sistema (GOUSSEVSKAIA, 2005).

2.2.2 Manipulação de áudio

Já para os recursos de áudio existem dois tipos de representação, o analógico e o digital. Atualmente o mais utilizado é o digital, embora o sistema auditivo só perceba o som analógico, por meio de suas vibrações. Esta forma digital de representação foi desenvolvida para utilizar o som em dispositivos eletrônicos, como aparelhos de som e computadores. O que possibilitou o desenvolvimento de softwares para manipulação e criação de efeitos. Um dos softwares de manipulação de áudio é o Audacity (AUDACITY, 2008), o qual possui gravação de até 16 canais, edição, criação de efeitos com filtros para remover ruídos indesejados, alteração de graves e agudos além de compressão em diversos tipos de formatos.

Os formatos de arquivos de áudio existentes hoje dependem da compressão utilizada. Estas compressões são importantes para determinar tamanho de arquivos e qualidade de áudio disponível. Dentre os formatos mais populares tem-se: formato *WAVE* (WAV), formato-padrão de arquivo de áudio não comprimido da Microsoft e International Business Machines (IBM) e *MPEG-1/2 Audio Layer 3* (MP3), que é um formato popular de compressão com perdas quase imperceptíveis (PAULA FILHO, 2000, p. 189).

2.2.3 Manipulação de vídeo

Por fim os recursos de vídeo nada mais são do que várias imagens apresentadas numa ordem ou seqüência para criar uma ilusão de movimento (ilusão ótica) decorrente do fenômeno de persistência da visão. A esta seqüência dá-se o nome de vídeo ou animação. Existe vários tipos de animação: Animação Convencional (AC), muito utilizada para desenhos, onde devem ser desenhados os quadros para em seqüência formar a animação; Animação Bidimensional, a qual refere-se a usar somente 2D para desenhar, mas como resultado pode-se ter uma aproximação 3D; Animação Tridimensional (3D), sendo não só o resultado, mas toda a modelagem é em 3D; Animação por *Frame* (*key frames*), onde uma tela é mostrada após a outra de acordo com a velocidade do computador ou do formato do vídeo (PAULA FILHO, 2000, p. 189).

Com relação aos formatos há certa dependência da ferramenta que estiver utilizando. Dentre os formatos mais populares estão: *Windows Media Vídeo* (WMV) para compressão de vídeo com perda, desenvolvido pela Microsoft; *Audio Video Interleave* (AVI), sendo um formato para compactação de áudio e vídeo desenvolvido pela Microsoft e MPEG-4 Part 14 (MP4), o qual é um padrão de container de áudio e vídeo que é parte da especificação MPEG-4.

Para a utilização dos formatos apresentados, existem ferramentas livres, com código aberto e com código fechado, possibilitando manipulação de vídeo, captura de vídeo, divisão em canais, criação e alteração de animações. Dentre as ferramentas mais utilizadas de vídeo estão: VirtualDub, ferramenta de licença livre que possui diversos filtros de ruído e captura de vídeo; Blender, ferramenta de código aberto que possui recursos para criação de animações de jogos e animações de vídeo em geral e Windows Movie Maker, ferramenta disponível com o Windows para criação de vídeos com efeitos de transição (PAULA FILHO, 2000, p. 288).

2.3 TRABALHOS CORRELATOS

A seguir são apresentados dois simuladores de animais vivos na área de Medicina Veterinária e uma ferramenta de auxilio ao aprendizado de farmacologia baseada em tutoriais.

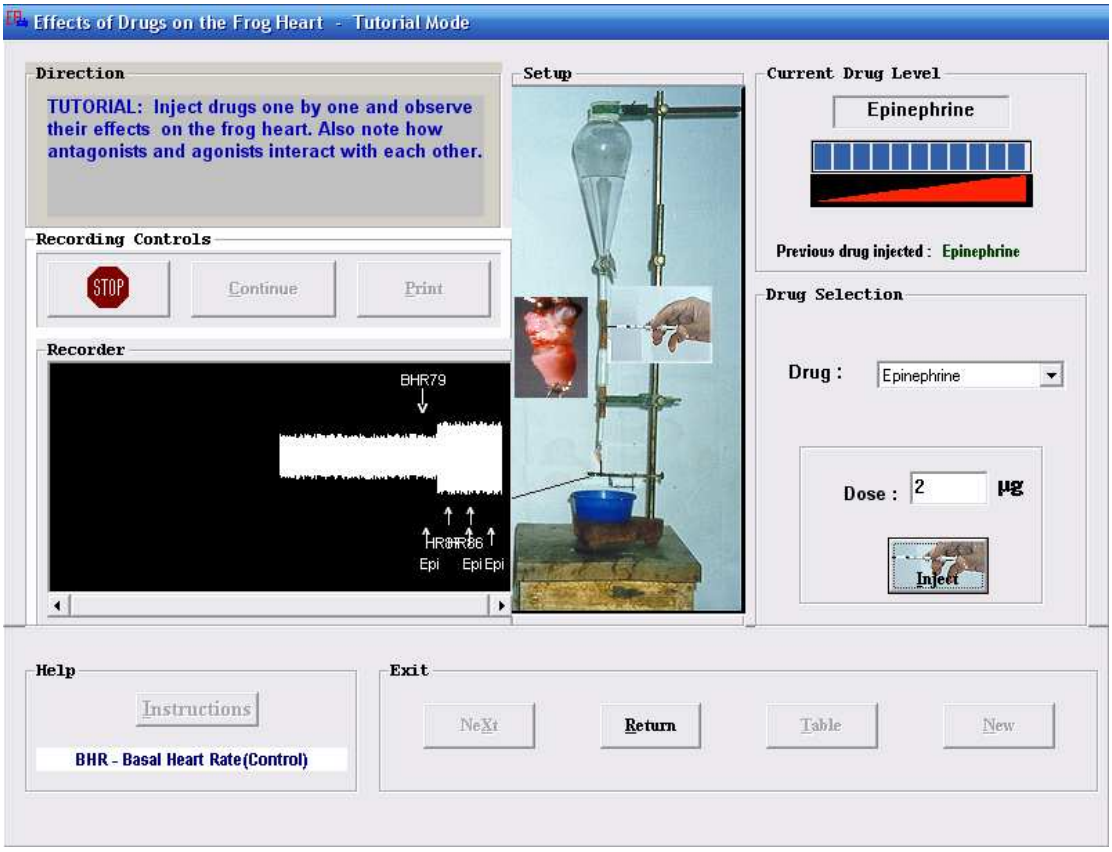
2.3.1 Expharm version T1.00

O simulador Expharm version T1.00 (RAVEENDRAN; SHASHINDRAN; KALATHI, 2005) é um software educacional que simula experimentos farmacológicos em animais. Consiste no desenvolvimento de quatro experimentos com simulação dos efeitos de drogas no sistema sensorial, sistema circulatório e sistema respiratório dos seguintes animais:

a) olhos: efeitos de drogas sobre os olhos de um coelho;
b) coração: efeitos de drogas sobre o coração de um sapo;
c) íleo: bioensaio da histamina sobre o íleo do porco de guiné;
d) esôfago: efeitos das drogas sobre o esôfago de sapos.

Estas simulações podem ser executadas apenas no modo tutorial, mas um modo de examinação está em desenvolvimento para esta ferramenta. O simulador também possui instruções detalhadas em cada experimento. A configuração dos equipamentos é apresentada para o estudante alterá-la, forçando assim o aprendizado sobre os equipamentos utilizados nas simulações.

O estudante também pode injetar a droga a partir de uma lista, previamente cadastrada, e em seguida observar seus efeitos, que são mostrados em seqüências animadas e simuladas em tempo real (Figura 7). Durante todo o experimento o estudante pode tomar notas e observações no livro de trabalhos, visando à documentação de detalhes. Já o futuro modo de examinação, as drogas serão apresentadas e o estudante poderá identificá-las, auxiliando no estudo das mesmas.



Fonte: Raveendran; Shashindran e Kalathi (2005).

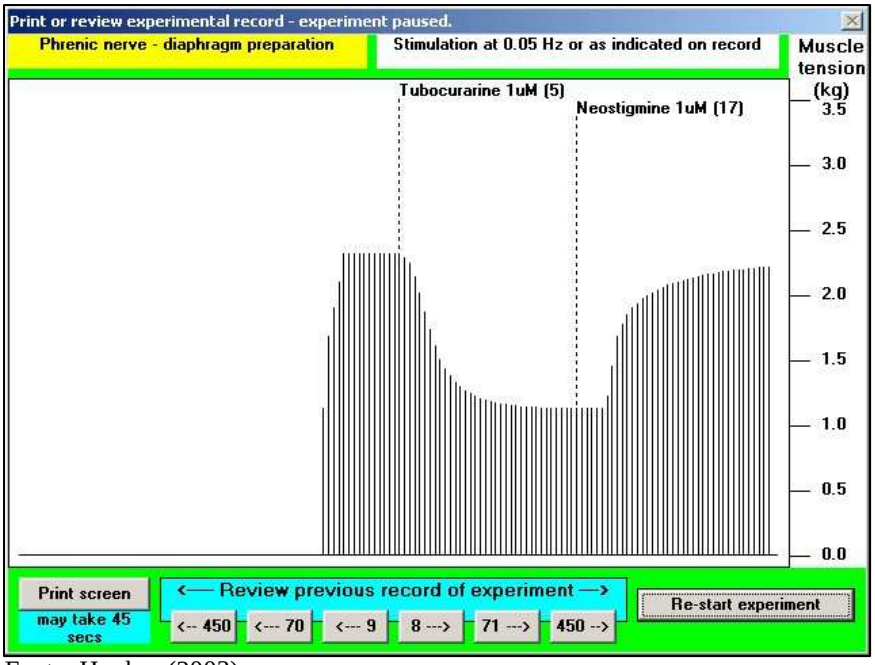
Figura 7 – Simulação dos efeitos de drogas no coração de um sapo

Após a conclusão do experimento é apresentado um questionário ao estudante. O simulador também possui uma interface de utilização amigável, intuitiva e totalmente operável via *mouse* (é mínima a necessidade de utilização do teclado).

2.3.2 Isolated phrenic nerve – diaphragm

O simulador Isolated phrenic nerve - diaphragm (HUGHES, 2003) tem como objetivo principal simular os efeitos de drogas sobre o esqueleto muscular e sistema nervoso motor de um animal de laboratório, por exemplo um rato. Para isso a estimulação tetânica[8] é simulada podendo-se aplicar drogas em áreas do corpo do animal e observar a tensão muscular em quilos oferecida por cada tipo de droga, como pode-se ver na Figura 8.

[8] Na neurobiologia a estimulação tetânica consiste na estimulação de um neurônio em seqüências de alta freqüência (SPINOSA; GÓRNIAK; BERNARDI, 2006).



Fonte: Hughes (2003).

Figura 8 – Simulação software Isolated Phrenic Nerve – Diaphragm

A estimulação também pode ser configurada das seguintes formas: desligada, a 0,05 Hz, a 0,5 Hz ou para 5 segundos a 30 Hz. As drogas disponíveis são: acetilcolina, lignocaine, gallamine, tubocurarina, pancurônio, atracúrio, fazadimium, hexamethonium, triethylcholine, colina, 4-aminopyridine, succinilcolina, edrophonium, neostigmina, atropina, carbacol, estreptomicina, physostigmine, dantroleno e decamethonium.

2.3.3 Microlabs for pharmacologists

Microlabs for pharmacologists é uma ferramenta (WILGENBURG, 1997) de auxílio ao aprendizado de farmacologia, consistindo basicamente em uma série de tutoriais desenvolvidos em computador. Sua principal finalidade é auxiliar o aprendizado e a realização de experiências farmacológicas. Dentre as mídias utilizadas estão:

a) tutoriais: com execução passo a passo;
b) vídeos: capturados com procedimentos a serem realizados;
c) imagens: capturadas e manipuladas com detalhes de experimentos;
d) áudio: capturados com as reações dos animais;
e) animações: pequenas seqüências de desenhos.

A partir destas mídias é possível combinar fundamentação teórica, imagem, vídeo e áudio em tutoriais animados que disponibilizam ao aluno uma série de procedimentos, passo a

passo, para a realização de experimentos (Figura 9).

Fonte: Wilgenburg (1997).

Figura 9 – Tutorial sobre o estudo dos reflexos de córneas

Todos os experimentos são realizados em ratos de laboratório e abaixo seguem algumas de suas funcionalidades:

a) anestesia local ou total;
b) atividades de reflexo motor;
c) reflexo do olho e córneas;
d) coleta de amostras de sangue, urina e fezes;
e) análise de sintomas e indicativo da substância a ser utilizada;
f) estudos de casos (exercícios).

3 DESENVOLVIMENTO

As informações existentes neste capítulo demonstram as etapas do desenvolvimento do simulador de animais vivos. São apresentados os requisitos, especificação da UML, implementação das técnicas utilizadas e resultados e discussões.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O simulador de animais vivos deverá:

a) permitir ao usuário a inspeção de várias camadas (visual externo, visual interno órgãos e visual interno sistema arterial) do fenômeno a ser simulado na aula prática de IDAA (Requisito Funcional – RF);
b) permitir ao usuário parametrizar atributos da cobaia, no caso cachorro (RF);
c) permitir ao usuário parametrizar os elementos químicos a serem aplicados à cobaia (RF);
d) permitir a simulação em tempo real dos comportamentos esperados como reação da cobaia (RF);
e) permitir ao usuário exportar imagem da aula prática de IDAA no formato *Portable Network Graphics* (PNG) (RF);
f) permitir ao usuário gerar pré-visualização e exportar relatório em formato *Portable Document Format* (PDF) com principais informações simuladas na aula prática (RF);
g) permitir ao usuário monitorar os batimentos cardíacos do animal a ser simulado em um eletrocardiograma virtual que fará parte da simulação (RF);
h) permitir ao usuário parar a aplicação do elemento químico a qualquer momento da simulação (RF);
i) permitir ao usuário a visualização de *hitens* explicativos durante simulação (RF);
j) ser implementado usando a análise orientada a objetos (Requisito Não Funcional - RNF);
k) ser implementado usando as bibliotecas gráfica Java OpenGL (JOGL) (RNF).

3.2 ESPECIFICAÇÃO

Para melhor entendimento do simulador a especificação foi desenvolvida utilizando a notação *Unified Modeling Language* (UML) (UML, 2002). Da notação são explanados os diagramas de casos de uso, de classes, de seqüência e de estados.

3.2.1 Diagrama de casos de uso

O simulador constitui, basicamente, em uma aula prática que mostra para o aluno os efeitos das drogas no organismo de mamíferos, no caso um cachorro. Para esta aula especificou-se as principais interações, de acordo com a aula feita em laboratório, e o resultado mostra-se na figura 4:

a) visualização das camadas dos órgãos do cachorro;
b) visualização de balões *hitens* com breves explicações;
c) aplicação dos elementos químicos no animal;
d) exportação de uma imagem da simulação;
e) relatório da simulação.

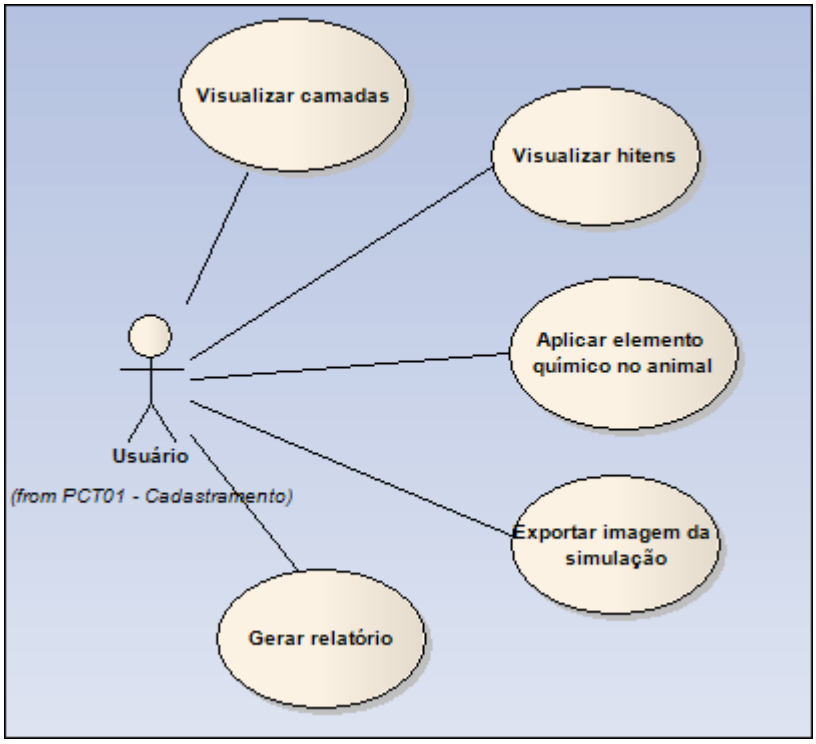


Figura 10 – Diagrama de casos de uso

O primeiro caso de uso, `Manipular camadas` (Quadro 1), descreve como o usuário pode visualizar as camadas de órgãos do cachorro. Durante a visualização ele poderá verificar três tipos de camadas com informações diferentes. Este caso possui um cenário principal e três alternativos, não possuindo exceções.

Visualizar camadas: possibilita ao usuário visualizar as diferentes camadas de órgãos do corpo do cachorro e suas informações.	
Pré-condição	Um animal deve estar carregado no simulador.
Cenário principal	1) Usuário arrasta *drag button* para uma das posições oferecidas. 2) O simulador faz *blend* (transparência) entre as três imagens enquanto o usuário arrasta o botão. 3) Usuário solta *drag button* em algum ponto. 4) Simulador mantém *blend* na posição desejada pelo usuário.
Cenário alternativo	No passo 3. Se usuário desejar soltar botão na posição `sistema arterial` o simulador manterá aquela posição.
Cenário alternativo	No passo 3. Se usuário desejar soltar botão na posição `órgãos` o simulador manterá aquela posição.
Cenário alternativo	No passo 3. Se usuário desejar soltar botão na posição `imagem externa` o simulador manterá aquela posição.
Pós-condição	*Blend* efetuado e posição mantida.

Quadro 1 – Caso de uso Visualizar camadas

O segundo caso de uso, `Visualizar hitens` (Quadro 2), explana como utilizar os *hitens* da aplicação. Esses *hitens* contém informações de determinados pontos da imagem, estas informações podem auxiliar no entendimento da aula. Além do cenário principal existem mais três cenários alternativos com os diferentes *hitens* visualizados.

Visualizar camadas: possibilita ao usuário visualizar *hitens* com informações da aula.	
Pré-condição	Um animal deve estar carregado no simulador.
Cenário principal	1) Usuário ativa opção *hitens* na barra de ferramentas ou no menu. 2) O simulador exibe imagem com pontos coloridos de *hitens*. 3) Usuário move mouse sobre os pontos coloridos. 4) Simulador faz *blend* do *hiten* na posição próxima ao mouse. 5) Usuário desativa *hitens*.
Cenário alternativo	No passo 3. Se usuário move o mouse o sobre o ponto verde determinado *hiten* é mostrado.
Cenário alternativo	No passo 3. Se usuário move o mouse o sobre o ponto azul determinado *hiten* é mostrado.
Cenário alternativo	No passo 3. Se usuário move o mouse o sobre o ponto vermelho determinado *hiten* é mostrado.
Pós-condição	Visualiza com *blend* o *hiten* escolhido pelo usuário.

Quadro 2 – Caso de uso Visualizar *hitens*

O terceiro caso de uso, `Aplicar elemento químico no animal` (Quadro 3), demonstra como o usuário pode executar a aplicação de elementos químicos no cachorro. Esta funcionalidade permite o usuário administrar, quantas desejar, doses de formalina e

dexametasona e observar os efeitos destas drogas. O usuário deve selecionar a opção elemento químico na barra de ferramentas ou no menu, selecionar a droga, informar a quantidade e aplicar na veia do animal. Essa aplicação pode ser repetida quantas vezes o usuário desejar e independente da droga utilizada. O resultado desta funcionalidade é uma animação do elemento "navegando" pelo sistema arterial do animal e modificação do comportamento do batimento cardíaco. Foi utilizado um algoritmo de *floodfill* (seção 2.2.1.2) na criação do caminho do elemento. Esta aplicação reflete diretamente no batimento cardíaco do animal podendo levá-lo a morte. Além do cenário principal existem três cenários alternativos e um cenário de exceção.

Aplicar elemento químico no animal: possibilita ao usuário administrar elementos químicos no cachorro e observar seus efeitos.	
Pré-condição	Um animal deve estar carregado visualizando sistema arterial.
Cenário principal	1) Usuário ativa opção aplicar elemento na barra de ferramentas ou no menu. 2) O simulador exibe uma janela solicitando ao usuário qual elemento ele deseja aplicar. 3) Usuário seleciona elemento formalina. 4) Simulador apresenta tela solicitando quantidade a ser aplicada. 5) Usuário insere quantidade a ser aplicada, dose de 20µl. 6) Simulador armazena quantidade. 7) Usuário aplica elemento em alguma artéria do animal. 8) Simulador inicia aniamação da aplicação do elemento. 9) Depois de certo tempo de aplicação simulador atualiza eletro.
Cenário alternativo	No passo 3. Se usuário selecionar elemento dexametasona, a dose é de 1500mg/kg.
Cenário alternativo	No passo 5. Se usuário aplicar uma dose muito alta de formalina o coração do cachorro pode fibrilar e morrer no passo 9.
Cenário alternativo	No passo 5. Se usuário aplicar uma dose muito alta de dexametasona o coração do cachorro fibrilar e morrer no passo 9.
Cenário de exceção	No passo 5. Caso usuário insira informação não *float*, sistema apresenta mensagem de erro.
Pós-condição	Simulador anima o caminho do elemento no sistema arterial do cachorro.

Quadro 3 – Caso de uso Aplicar componente químico no animal

O quarto caso de uso, `Exportar imagem da simulação` (Quadro 4), tem como funcionalidade exportar uma imagem da simulação. Esta imagem pode ser gerada no momento que o usuário desejar e é gerada na extensão PNG. Este caso de uso possui apenas cenário principal não possuindo cenários alternativos e nem cenários de exceção.

O quinto e último caso de uso, `Gerar relatório` (Quadro 5), oferece ao usuário a opção de efetuar um relatório automático da simulação efetuada. Para a execução deste relatório é primeiramente apresentado ao usuário uma pré-visualização dos procedimentos

executados na simulação e caso o usuário deseje ainda é possível gerar um arquivo PDF com as informações. Foi utilizada a biblioteca iText (ITEXT, 2008) para a execução do relatório PDF e este caso de uso possui apenas cenário principal e um cenário alternativo. Não existem pré-condições e havendo solicitação de geração de relatório por parte do usuário sem existir simulação concluída ou simulação em andamento o relatório sairá com informações omitidas.

Exportar imagem da simulação: possibilita ao usuário exportar uma imagem da simulação em formato PNG.	
Pré-condição	Um animal deve estar carregado no simulador.
Cenário principal	1) Usuário seleciona opção exportar imagem na barra de ferramentas ou no menu. 2) O simulador exibe uma janela solicitando nome para o arquivo já na extensão PNG. 3) Usuário entra com informações de nome e pasta para salvar. 4) Simulador salva arquivo PNG.
Pós-condição	Imagem PNG salva em arquivo.

Quadro 4 – Caso de uso Exportar imagem da simulação

Gerar relatório: possibilita ao usuário gerar relatório da simulação em formato PDF.	
Pré-condição	Não existem pré-condições.
Cenário principal	1) Usuário seleciona a opção gerar relatório na barra de ferramentas ou no menu. 2) O simulador exibe uma janela de pré-visualização com os dados até o momento da geração do relatório e solicita a criação do arquivo PDF. 3) Usuário solicita criação de arquivo PDF. 4) O simulador exibe uma janela solicitando nome para o arquivo já na extensão PDF. 5) Usuário entra com informações de nome e pasta para salvar 6) Simulador salva arquivo PDF.
Cenário alternativo	No passo 3. Caso usuário não solicite criação de arquivo PDF simulador encerra relatório.
Pós-condição	Relatório gerado em arquivo PDF .

Quadro 5 – Caso de uso Exportar imagem da simulação

3.2.2 Diagrama de classes

Para melhor apresentar como as classes estão implementadas no simulador especificou-se um diagrama de classes. Na figura 5 estão as oito principais classes implementadas, para preservar a legibilidade do diagrama foram omitidos os atributos, os métodos e algumas classes pouco utilizadas.

O início do simulador é dado através da classe `Frame`, a qual carrega as funcionalidades da barra de ferramentas, menus e demais botões. Após isso a classe carrega

mais dois objetos *canvas* das classes `Universo` e `Eletro`. Além disso a classe *`Frame`* também implementa métodos responsáveis pela sincronização entre as informações da classe `Universo` e as informações da classe `Eletro`.

A classe `Universo` é a principal classe do simulador e é onde o usuário terá a maioria de sua interação com o simulador. Nela são carregadas as imagens das camadas dos órgãos do cachorro através do método `drawAnimal` que é chamado para cada imagem que se deseja carregar no simulador, e neste método também está programado para efetuar o *blend* (seção 2.2.1.1) de todas as imagens carregadas. Nesta classe também existe o método `drawHitens`, este por sua vez carrega uma imagem de fundo e após escreve informações da simulação com o método drawStr, o texto contido nela. Sua exibição é suavizada com o *blend* do *hiten* fazendo com que seu surgimento seja esmaecido. Esta classe também é responsável pela exportação da foto com o método `saveFrameAsPNG` e efetuar o desenho do caminho que a droga percorre no sistema com o método `FloodFill`.

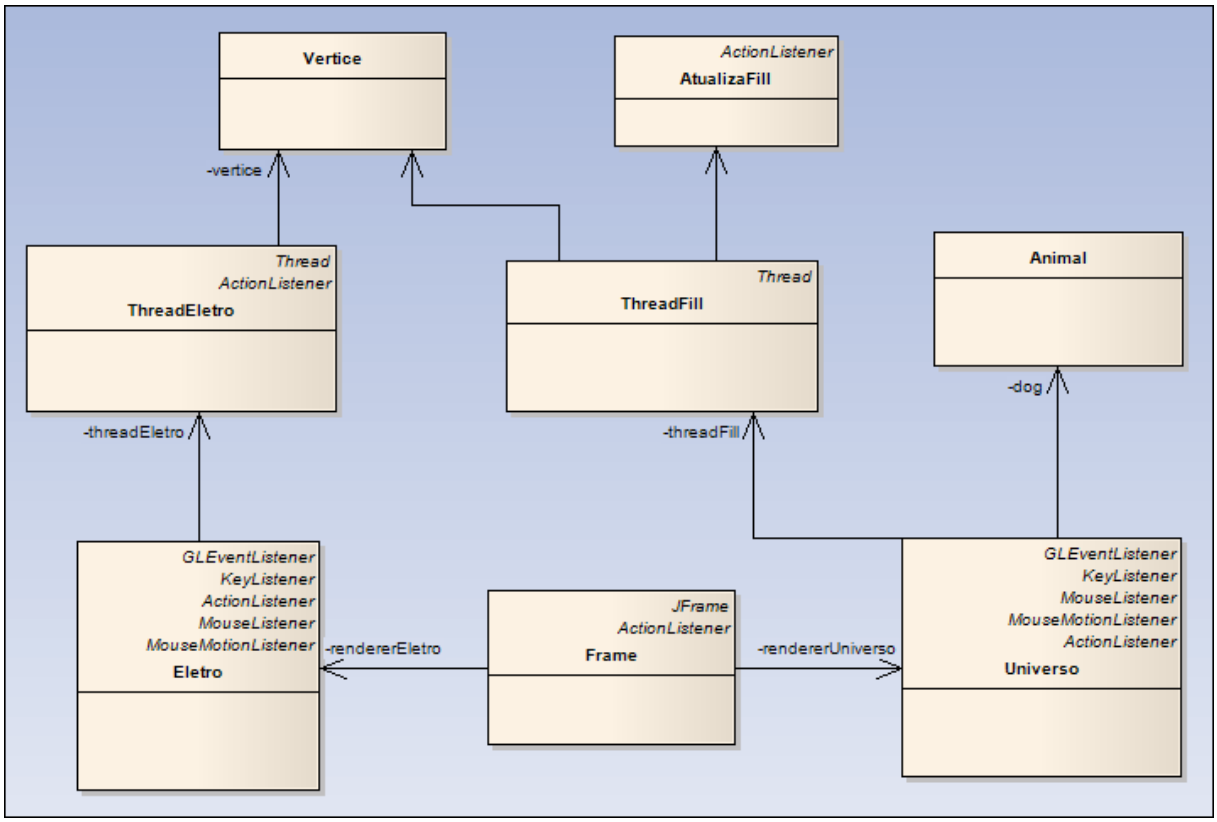


Figura 11 – Diagrama de classes

A classe `Eletro` é uma classe muito importante no simulador porque além de fazer interação com o usuário ela implementa todas as funções do eletro. Dentre as funções estão a de carregar a imagem do eletro, ligar/desligar eletro, ativar/desativar o som do eletro e mostrar/apagar linha do eletro. Esta classe também desenha a linha dos batimentos cardíacos com o método `GL.GL_LINE_STRIP`, atualiza o número dos batimentos cardíacos com o

método `drawStr`, faz a interface de aplicação dos elementos no cachorro e também gera a pré-visualização com o método `previewRelatorio` e arquivo PDF com o método `generatePDF` utilizando a biblioteca *iText*.

A classe `ThreadFill` é reponsável exclusivamente pela construção do caminho que a droga vai percorrer no sistema arterial do animal. Desta forma ela faz a leitura dos *pixels* da imagem, a tradução destes *pixels* e armazena numa lista de objetos. As informações de RGB destes *pixels* também são armazenadas, assim como de cada *pixel* da imagem. Ela também instancia um objeto da classe `AtualizaFill`.

`ThreadEletro` é uma classe exclusiva para o cálculo da linha do eletro. Essa linha é calculada dependendo da freqüência dos batimentos cardíacos e sua intensidade. Cada posição da linha calculada é armazenada numa lista de vértices.

A classe `Animal` é responsável por ler as imagens do animal em arquivo. Estas imagens podem ser de formato PNG, JPG, GIF, BMP ou RAW, sendo traduzidas suas informações para OpenGL. O método utilizado para isto é o `readTexture`.

`Vertice` é a menor classe do simulador e é responsável apenas como meio de armazenar as posições das coordenadas X e Y na tela e o RGB destas posições, para recuperá-las quando necessário. Objetos desta classe são instanciados pelas classes `Universo`, `ThreadEletro`, `ThreadFill`, `AtualizaFill` e são acessados por quase todas as classes do simulador.

Finalmente a classe `AtualizaFill` é responsável pela atualização das cores dos objetos que a classe `ThreadFill` instancia. De acordo com o elemento químico aplicado esta classe fica responsável por mudar a cor da artéria gradativamente, e de acordo com a velocidade que o elemento deve percorrer no sistema arterial.

3.2.3 Diagrama de seqüência

O caso de uso `Aplicar elemento químico no animal` envolve o maior número de troca de mensagens do simulador, principalmente entre as classes `Frame`, `Universo` e `Eletro`, desempenhando um papel fundamental na sincronização entre as informações destas classes. Para o entendimento desta troca de mensagens, foi implementado o diagrama de seqüência (Figura 12), onde pode-se observar o envolvimento das classes. Por questões de visibilidade a classe `Vértice`, qual armazena as informações de coordenadas e cor foi,

omitida do diagrama.

A troca de mensagens se inicia, basicamente, com a solicitação do usuário a classe `Frame` para aplicar o elemento químico. Em seguida, após o usuário parametrisar o elemento, a classe `Frame` avisa a classe `Universo` que droga está pronta para ser injetada e o usuário está livre para injetar o elemento químico no sistema arterial do cachorro. Na seqüência é iniciada então a *thread* de preenchimento e a classe `Frame` verifica se foi injetada, avisando então a classe `Eletro` para atualizar seus dados. Este procedimento acontece simultaneamente aos *loops* do simulador que são responsáveis por desenhar na tela o preenchimento e a linha do eletro.

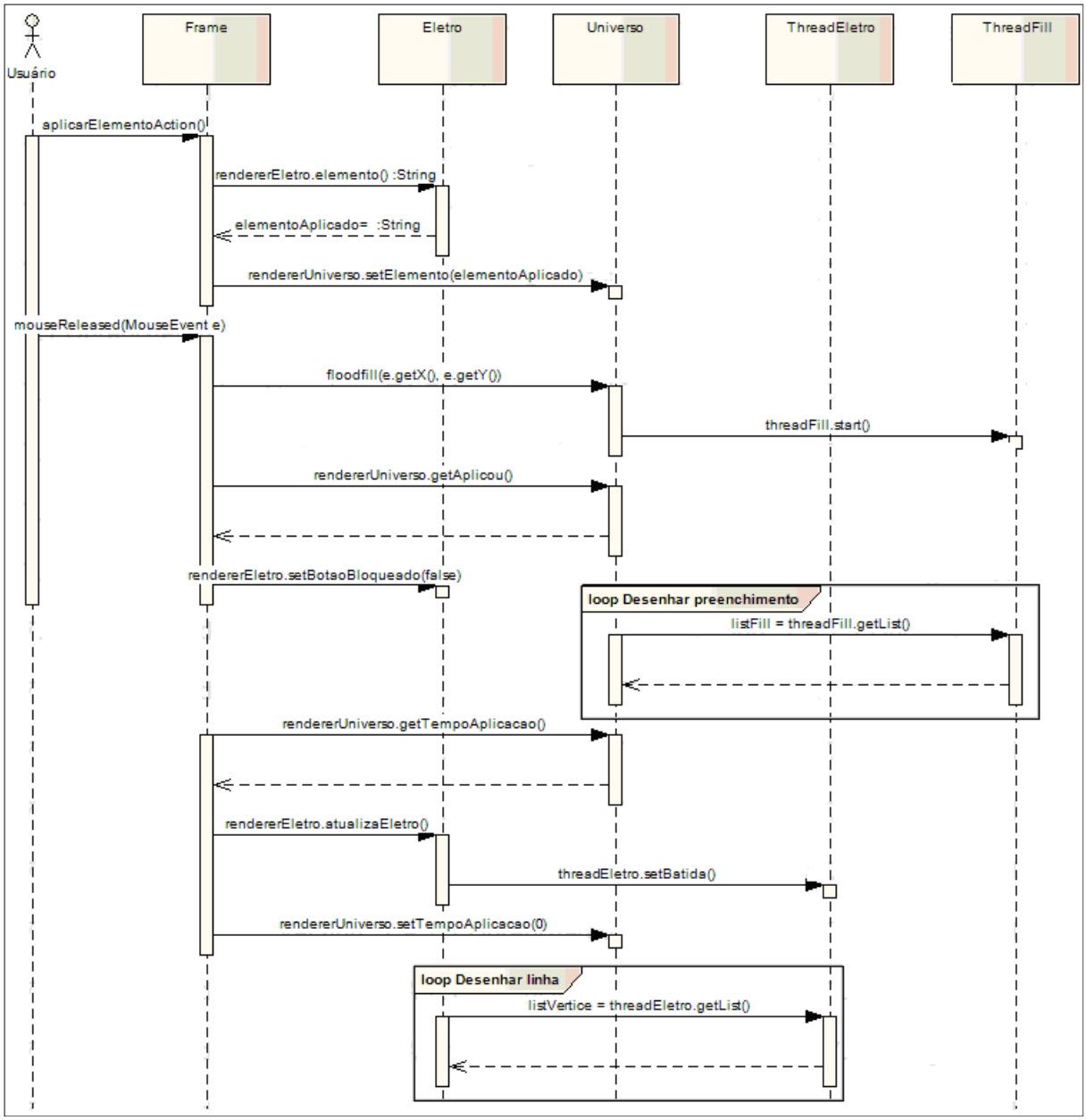


Figura 12 – Diagrama de seqüência

3.2.4 Diagrama de transição de estados

Para um melhor entendimento dos estados e possibilidades que o simulador oferece na aplicação dos elementos químicos, foi implementado o seguinte diagrama de estados (Figura 13). A ação e a reação que a simulação deve exercer vão depender diretamente de qual estado se encontra no momento.

Primeiramente o estado inicial é batimento normal a 75 Batimentos Por Minuto (BPM), iniciado após o usuário entrar com o peso da cobaia. Em seguida o usuário pode optar por dois caminhos, a aplicação de formalina leva a batimento acelerado, sendo superior a 75BPM, e a aplicação de dexametasona que leva a batimento desacelerado, sendo inferior a 75BPM. Os estados em que o animal pode se encontrar no sistema varia de acordo com cada elemento aplicado, resultando em um estado diferente. O que pode acontecer é que o usuário aplique uma dose superior ao “normal” levando a fibrilação do coração e conseqüentemente a morte do animal. Vale lembrar que o usuário pode a qualquer momento finalizar a simulação.

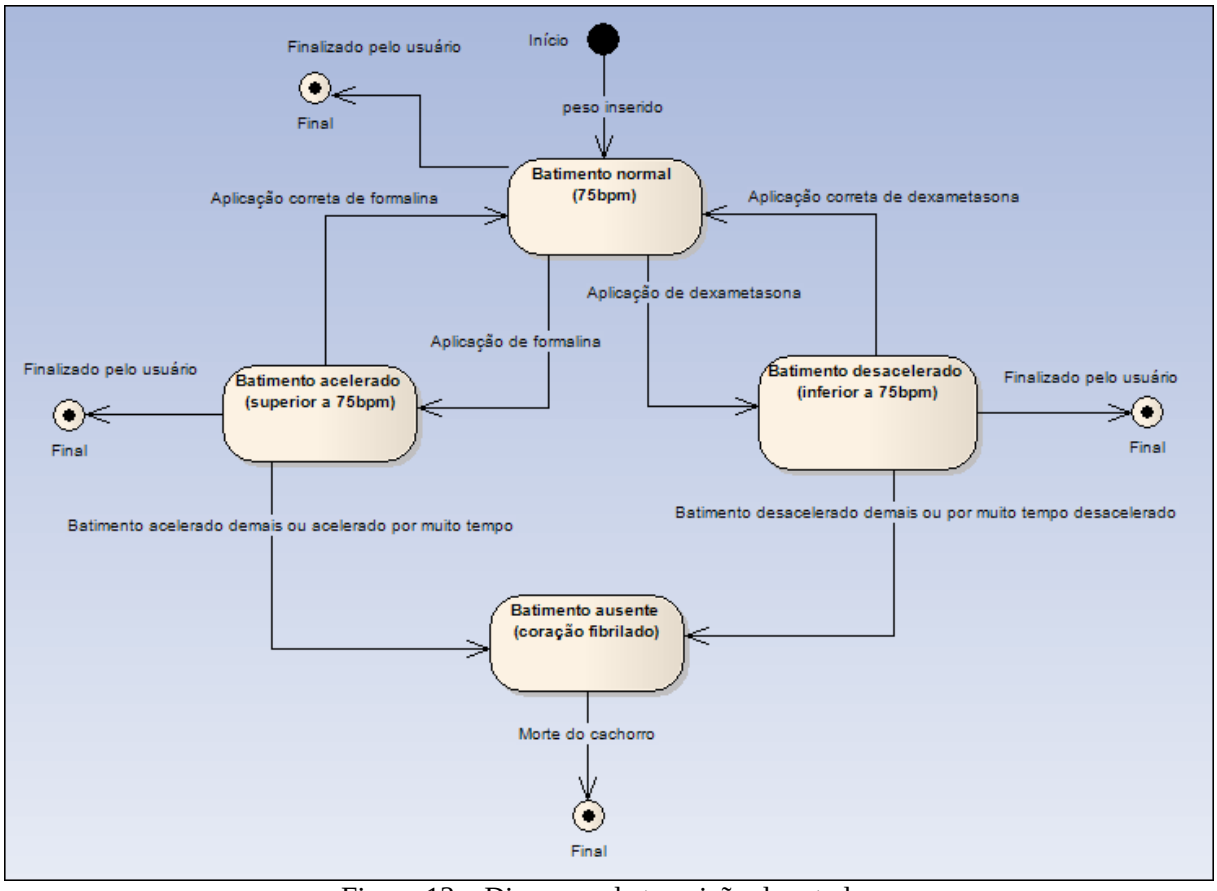


Figura 13 – Diagrama de transição de estados

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentados os aspectos a respeito da implementação do simulador, bem como as técnicas e ferramentas utilizadas. Por último, é descrita a operacionalidade da implementação através de um manual do usuário.

3.3.1 Técnicas e ferramentas utilizadas

Para a implementação do simulador na linguagem Java, foi utilizado o ambiente de desenvolvimento integrado Eclipse 3.3.1. Também foi utilizada a ferramenta Adobe Photoshop CS2 para a correção e adaptação das imagens às necessidades do simulador.

3.3.1.1 Desenho das linhas e texto do eletro

Para o desenho das linhas do eletro foi implementada uma classe, estendida de *thread*, que fica calculando os próximos “picos” de batimento. No quadro 6 é possível observar que nela existem três métodos principais, dois métodos que calculam a subida e a descida da linha e um método que determina o tamanho do “pico” de subida e descida da linha. Todos os cálculos são executados constantemente e armazenados em uma lista, qual é recuperada pela classe `Eletro`.

No quadro 7, é possível observar a recuperação da lista de pontos da linha e seu desenho na tela utilizando as técnicas da classe `GL` do JOGL.

```
private void addPoints(){
	if (leituraY < batida){
		leituraY+= 1f;
		sobe = true;
		if (estabiliza){
			batida = 13;
			desce = 13;
			estabiliza = false;}
	}
	else{
		if(leituraY > batida){
			if (sobe){
				addDesce();
				estabiliza = true;
				return;}
			batida = desce;
			leituraY-= 1f;}
	}
	vertice = new Vertice();
	vertice.setX(leituraX);
	vertice.setY(leituraY);
	...
}
private void addDesce() {
	if(leituraY > batida){
		batida = 12f;
		leituraY -= 1;
	}
	vertice = new Vertice();
	vertice.setX(leituraX);
	vertice.setY(leituraY);
	this.listVertice.add(vertice);
	leituraX+= 0.05;
...
}
public void setBatida(){
	if (delay){
		batida = 14.6f;
		delay = false;
		desce = 12.5f;
	}
...
}
```

Quadro 6 – Trecho do código da *thread* do eletro

```
gl.glLineWidth(3f);
gl.glColor3f(0f, 1f, 0f);
ArrayList<Vertice> listVertice = threadEletro.getList();
for (int count = 0; count < listVertice.size(); count++){
	gl.glBegin(GL.GL_LINE_STRIP);
	gl.glVertex2d(listVertice.get(count).getX(),
			listVertice.get(count).getY());
	if(listVertice.size() > count+1)
		gl.glVertex2d(listVertice.get(count+1).getX(),
				listVertice.get(count+1).getY());
	gl.glEnd();
}
```

Quadro 7 – Trecho do código para desenhar linha do eletro

Outra técnica utilizada na classe eletro é para desenhar textos com o método `drawStr` da classe `GLUT` também do JOGL (quadro 8).

```
private void drawStr(float x, float y, String texto, int font) {
	if (font == 7)
		font = GLUT.BITMAP_HELVETICA_12;
	else
		font = GLUT.BITMAP_HELVETICA_18;
	gl.glRasterPos2f(x, y);
	//desenha letra por letra na tela
	try {
		for(int i=0; i < texto.length(); i++)
			glut.glutBitmapCharacter(font, texto.charAt(i));
	} catch(Exception e) {}
}
```

Quadro 8 – Trecho do código para desenhar texto do eletro

3.3.1.2 *Blend* das imagens

Para o *blend* (seção 2.2.1.1) e carregamento das imagens foi implementada a técnica presente no quadro 9.

```
private void drawAnimal(Animal imagem, float blend) {
	gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);
	gl.glEnable(GL.GL_BLEND);
	gl.glColor4f(1f, 1f, 1f, blend);
	int texture = imagem.getTexture();
	gl.glTranslatef(imagem.getPositionX(),imagem.getPositionY(),-4.0f);
	gl.glScalef(imagem.getEscalaX(), imagem.getEscalaY(), 0f);
	gl.glBindTexture(GL.GL_TEXTURE_2D, texture);
	gl.glBegin(GL.GL_QUADS);
	// Desenha imagem fundo
		gl.glTexCoord2f(0.f, 0.0f);
		gl.glVertex2f(-1.0f, -1.0f);
		gl.glTexCoord2f(1.0f, 0.0f);
		gl.glVertex2f(1.0f, -1.0f);
		gl.glTexCoord2f(1.0f, 1.0f);
		gl.glVertex2f(1.0f, 1.0f);
		gl.glTexCoord2f(0.0f, 1.0f);
		gl.glVertex2f(-1.0f, 1.0f);
	gl.glEnd();
	gl.glDisable(GL.GL_BLEND);
}
```

Quadro 9 – Trecho do código para desenhar texto do eletro

O atributo *blend* do tipo `float` é responsável pela transparência da imagem. Se este valor for igual a um a imagem é opaca, se este valor for zero a imagem desaparece. Todos os valores entre zero e um farão a tranparência de fraca intensidade ou forte intensidade. Esta técnica é usada também na exibição dos *hitens* através de uma instrução `for` observada no quadro 10.

```
lerRGB("imagens/imagem_pontos_ler.png",e.getY(),e.getX());
if(iR == 1f && iG == 0f && iB == 0f){
	if(janela1.getBlenadJanela() == 0){
		for(float i = 0.0f; i < 1; i+=0.03f){
			janela1.setBlendJanela(i);
			glDrawable.display();
		}
		janela1.setBlendJanela(1);
	}
...
}
```

Quadro 10 – Trecho do código para desenhar texto do eletro

Neste caso, se o mouse estiver passando sobre a cor vermelha de valor igual a um e o valor de azul e verde for igual a 0 o *hiten* é esmaecido, ficando na marca d'água.

3.3.1.3 Preenchimento – *floodFill*

Para o preenchimento da artéria com o elemento químico foi utilizada a técnica de *floodfill* (seção 2.2.1.2), esta técnica consiste em distribuir informação, para todos os *pixels* da imagem a partir de um *pixel*, chamado semente. Este primeiro *pixel* semente transmite esta informação para seus quatro vizinhos (cima, baixo, direita, esquerda) e cada um destes vizinhos re-transite esta informação para seus vizinhos sucessivamente. Desta forma foi implementado uma classe `ThreadFill` que é estendida de *thread* que armazena todos os *pixels* do preenchimento numa lista de vértices. No quadro 11 temos o núcleo das chamadas efetuadas pela `ThreadFill`, ficando responsável pela distribuição da informação semente.

```
public void run(){
	while (fica){
	lerRGB("imagens/sistema_arterial_modificado.png",posicaoY*0.5395f,
                                                        posicaoX*0.5474f);
	adiciona(posicaoX, posicaoY);
	for(int count = 0; count < listFill.size(); count++)
		floodfill(listFill.get(count).getFrameX(),
                                        listFill.get(count).getFrameY());
		proximo = true;
		while (proximo){
			try {
				Thread.sleep(tempo);
			} catch (InterruptedException e) {
				//    TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
	}
}
```

Quadro 11 – Trecho do código da responsável pelo preenchimento todos os *pixels*

O método *`run()`* da *`thread`* permite que várias aplicações de elementos químicos

sejam efetuadas. Se o algoritmo terminar o preenchimento de todos os *pixels* a *thread* fica "dormindo" esperando pela próxima aplicação. Após o usuário clicar no *pixel* semente, é utilizado o método `lerRGB` (quadro 12) que lê o *pixel* na posição indicada e o guarda em três constantes `iR`, `iG` e `iB`. Na seqüência é acionado o método `adiciona(posicaoX, posicaoY)` (quadro 13), este método utiliza as constantes armazenadas pelo método `lerRGB` e verifica se este *pixel* está na faixa de vermelho aceitável, se não estiver na faixa aceitável o método `adiciona(posicaoX, posicaoY)` é abortado. A construção *`for`* existente dentro deste método é de extrema importância, porque ela verifica se este *pixel* já existe dentro da lista, evitando assim um *loop* infinito que aconteceria se o mesmo *pixel* fosse adicionado várias vezes.

```
private void lerRGB(String animal, float y, float x) {
        TextureReader.Texture texture = null;
        try {
            texture = TextureReader.readTexture(animal);
        } catch (IOException e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
        makeRGBTexture(texture, y, x);
}
private void makeRGBTexture(TextureReader.Texture img,float row,float col)
                                                                        {
        img.setRowCol((int)row, (int)col);
        iR =  img.getR();
        iG =  img.getG();
        iB =  img.getB();
}
```

Quadro 12 – Trecho do código responsável pela leitura do RGB do *pixel*

```
public void adiciona(int posicaoX, int posicaoY){
     if (iR > 0.50 && iG < 0.50 && iB < 0.50){
         atualizaPosicao(posicaoX, posicaoY);
         //armazena em uma lista os vétices a serem preenchidos
         Vertice vertice = new Vertice();
         vertice.setX(x);
         vertice.setY(y);
         vertice.setFrameX(posicaoX);
         vertice.setFrameY(posicaoY);
         vertice.setRGB(iR, iG, iB);
         for(int count = 0; count < listFill.size(); count++){
            if (listFill.get(count).getFrameX()==vertice.getFrameX()
                 && listFill.get(count).getFrameY()==vertice.getFrameY()){
                  return;
            }
         }
         this.listFill.add(vertice);
      }else
         return;
}
```

Quadro 13 – Trecho do código responsável pelo teste de tolerância da cor vermelho

A existência do *loop* *`for`*, já vista no quadro 11, vai chamar o método `floodfill` para

cada item existente na lista. Isso fará com que os vizinhos de cada *pixel* sejam testados e adicionados na lista, permitindo que o preenchimento seja executado.

Por fim temos o método `floodfill` (quadro 14). Este método vai ler o RGB de cada um dos vizinhos do *pixel* atual, tentando adicionar cada um dos vizinhos logo em seguida, isso acontecerá caso o RGB dos vizinhos estejam dentro da faixa de vermelho aceitável.

Pode-se observar também que na chamada do método `lerRGB` está a posição da coordenada X e Y mais o número dois. Isto faz com que um vizinho seja "pulado", e que para corrigir este pulo é desenhado de um ponto maior na hora da execução do desenho. Esta alteração se fez necessária para melhorar a performance do preenchimento.

```
private void floodfill(int posicaoX, int posicaoY){
      if (cancelou)
            return;
      //ler para cima
      lerRGB("imagens/sistema_arterial_modificado.png",(posicaoY+2)*0.5395f,
                                                      posicaoX*0.5474f);
      //manda adicionar
      adiciona(posicaoX, posicaoY+2);
      //ler para baixo
      lerRGB("imagens/sistema_arterial_modificado.png",(posicaoY-2)*0.5395f,
                                                      posicaoX*0.5474f);
      //manda adicionar
      adiciona(posicaoX, posicaoY-2);
      //ler para direita
      lerRGB("imagens/sistema_arterial_modificado.png",posicaoY*0.5395f,
                                                  (posicaoX+2)*0.5474f);
      //manda adicionar
      adiciona(posicaoX+2, posicaoY);
      //ler para esquerda
      lerRGB("imagens/sistema_arterial_modificado.png",posicaoY*0.5395f,
                                                  (posicaoX-2)*0.5474f);
      //manda adicionar
      adiciona(posicaoX-2, posicaoY);
}
```

Quadro 14 – Trecho do código responsável pela leitura do RGB do *pixel*

3.3.2 Operacionalidade da implementação

Esta seção tem por objetivo mostrar a operacionalidade da implementação do simulador em nível de usuário. Nas próximas seções serão abordadas as funcionalidades do simulador.

3.3.2.1 Iniciando a simulação

Na figura 14 apresenta-se a tela principal do sistema. Logo em seguida o usuário pode iniciar a simulação clicando no menu `Arquivo`, `Novo` ou diretamente no botão `Novo` na barra de ferramentas.

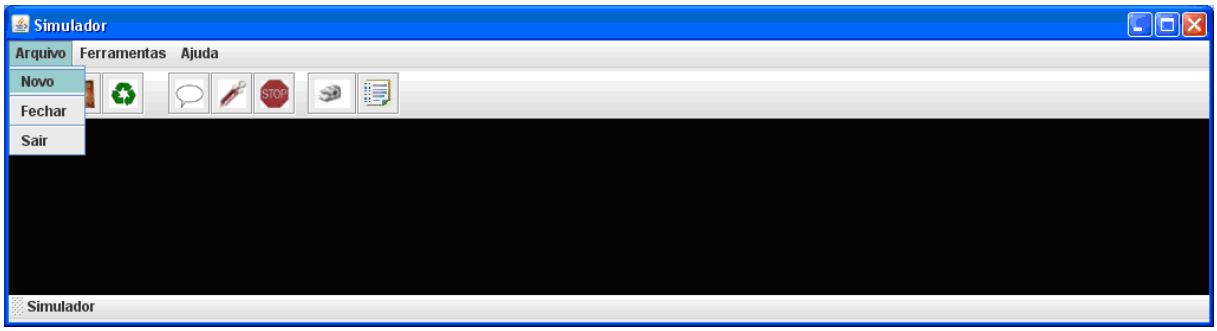


Figura 14 – Tela principal do simulador

Ao iniciar a simulação será apresentada ao usuário a tela de inserção do peso do cachorro (Figura 15).

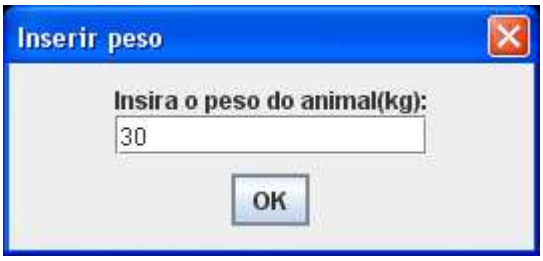


Figura 15 – Inserção do peso do cachorro

Após inserir o peso do cachorro é apresentado o animal e as informações do eletro (Figura 16).

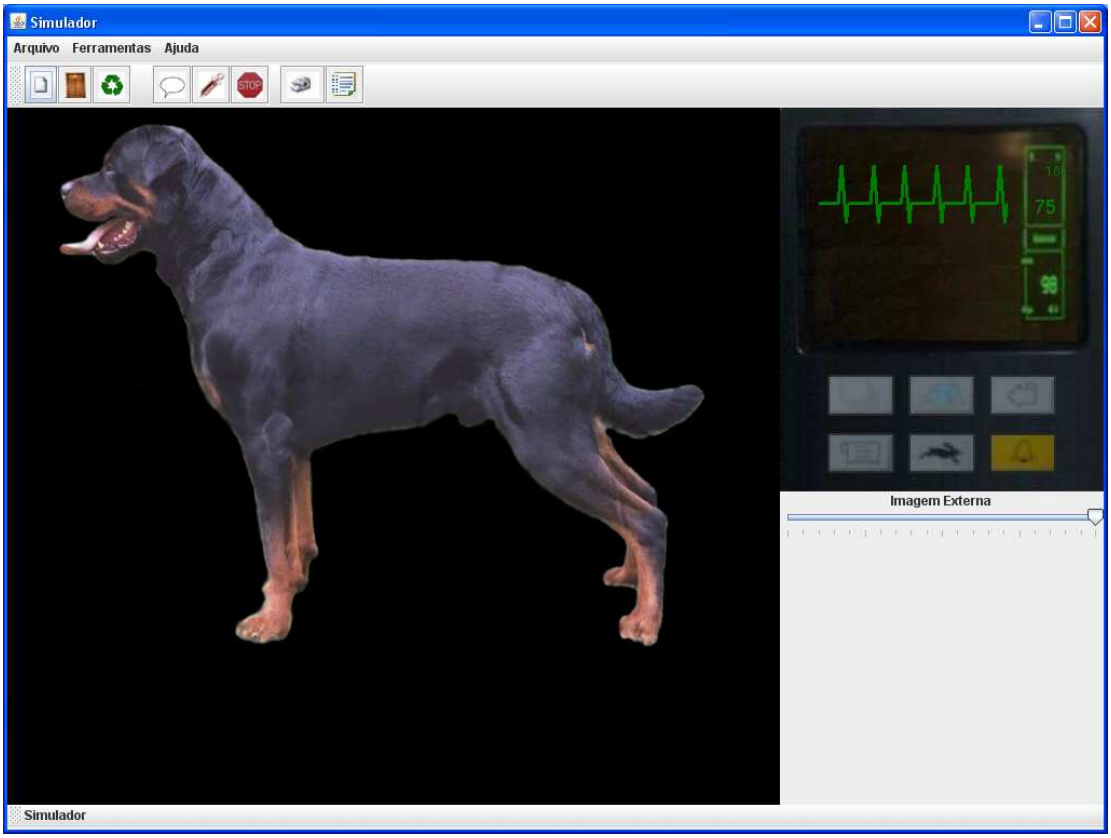


Figura 16 – Início da simulação

Com o início da simulação o usuário está liberado para fazer as inspeções das camadas do animal. Para isto, basta o componente *drag button* ser arrastado da esquerda para a direita ou da direita para a esquerda. As imagens apresentadas são: imagem externa apresentada na figura 17; imagem interna (órgãos) (figura 18) e imagem interna (sistema arterial) (figura 19).

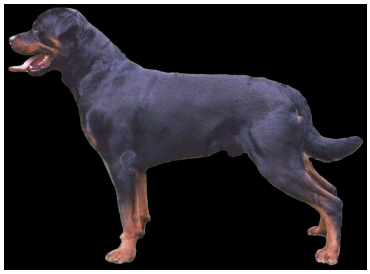


Figura 17 – Externa

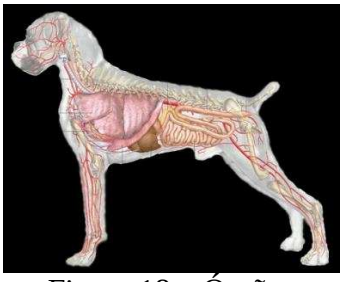


Figura 18 – Órgãos

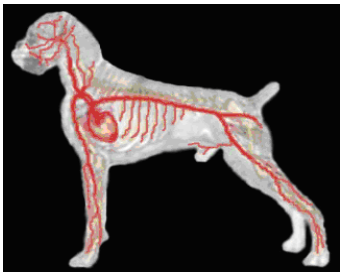


Figura 19 – Arterial

Além da inspeção de imagens o usuário poderá visualizar *hitens*, que são balões esplicativos. Para isto basta clicar no menu `Ferramentas`, `Exibir` *`hitens`* ou clicar no botão *hitens* na barra de ferramentas. A imagem que aparecerá é a da figura 19.

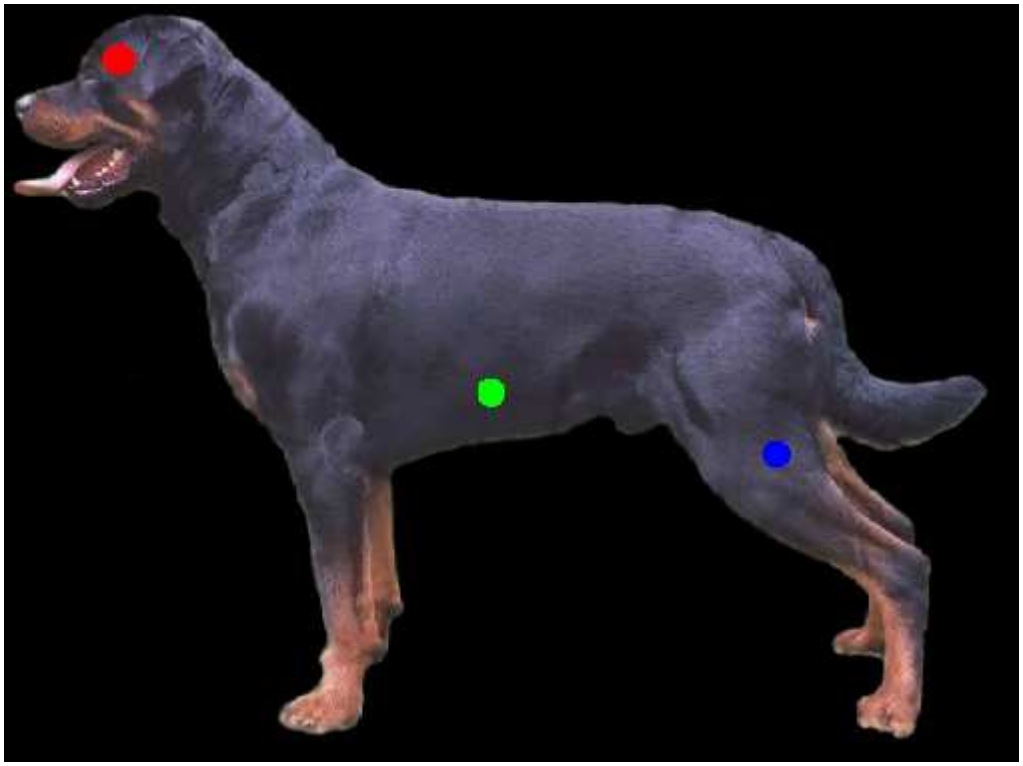


Figura 19 – Imagem pontos dos *hitens*

Ao visualizar esta imagem o usuário pode arrastar o mouse sobre os pontos coloridos, fazendo com que apareça uma imagem da *hiten* esmaecida (figura 20).

Figura 20 – Imagem *hiten* esmaecido

Além da exibição dos *hitens* o simulador também oferece a funcionalidade de exportar uma imagem do que está na tela neste momento. Para isso basta clicar em `Ferramentas`, `Exportar Imagem` (figura 21).

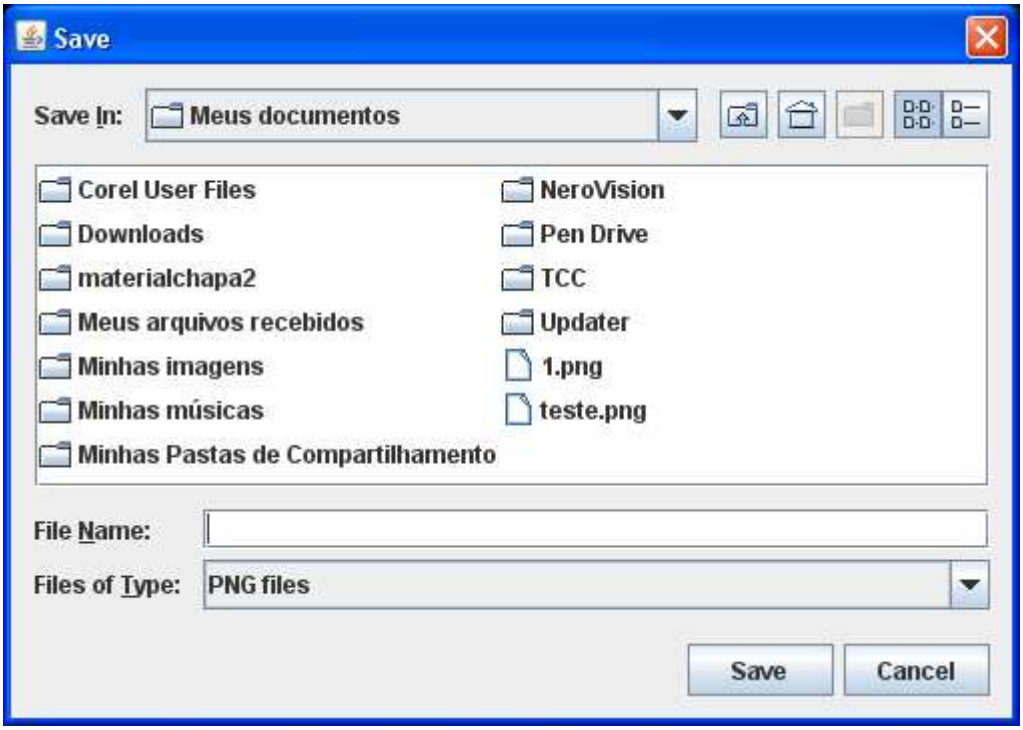


Figura 21 – Exportar imagem para arquivo

Para a funcionalidade de aplicar elemento químico o usuário poderá selecionar no menu `Ferramentas`, `Aplicar elemento`. Após esta seleção aparecerá uma janela onde o usuário poderá escolher qual elemento químico deseja administrar (Figura 22).

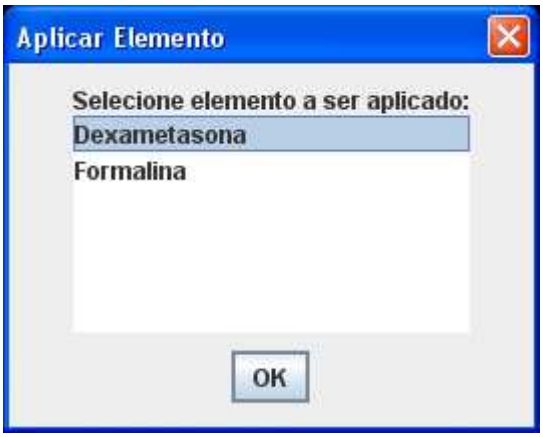


Figura 22 – Selecionar elemento químico

Ao escolher uma das opções ao usuário aparece uma nova janela que solicitará a quantidade de elemento que deseja-se aplicar (figura 23).

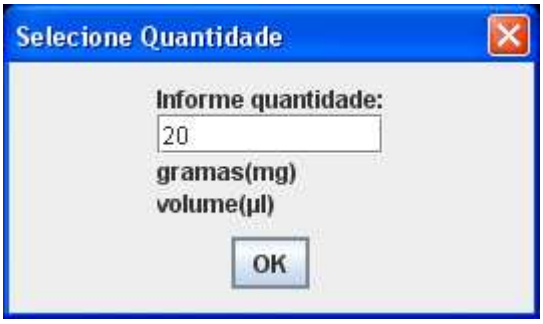


Figura 23 – Parametrizar quantidade de elemento químico

A quantidade de elemento influenciará diretamente no batimento cardíaco do animal e possivelmente o colocará em risco. Para o elemento formalina é recomendado a aplicação da dose normal, ou seja, 20µl. Já para o elemento dexametasona varia de acordo com o peso do animal, sendo 1500mg/kg (mil e quinhentos miligramas por quilo). O resultado da aplicação da formalina está na figura 24 e o resultado da dexametasona na figura 25.

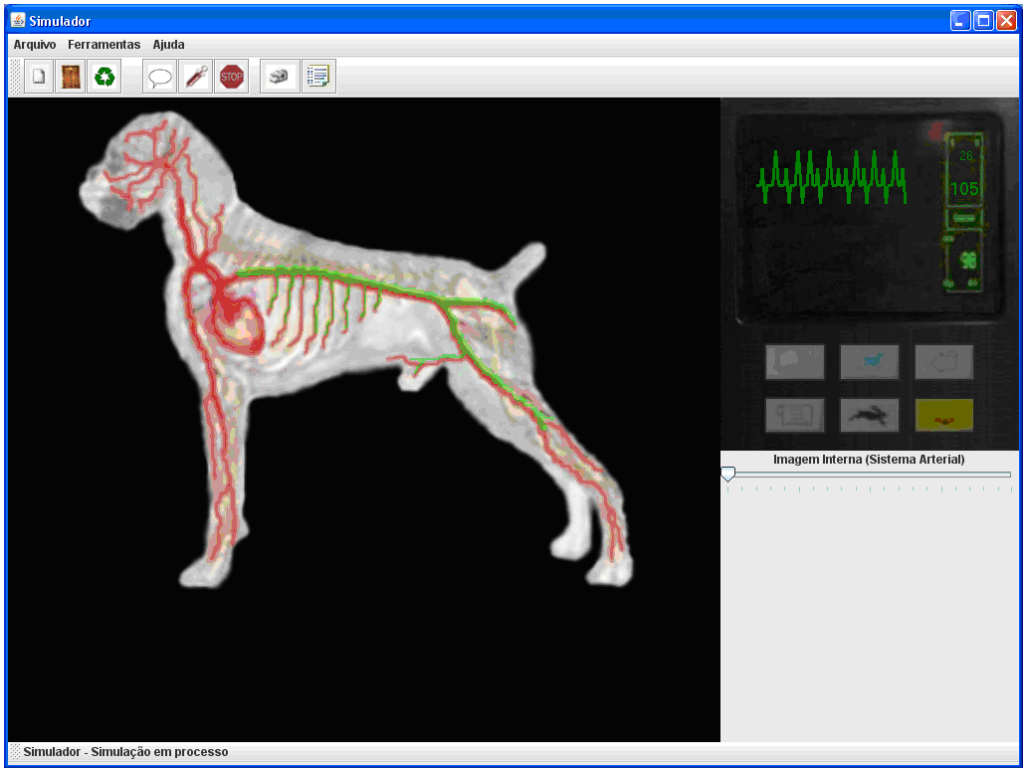


Figura 24 – Animação do elemento químico formalina

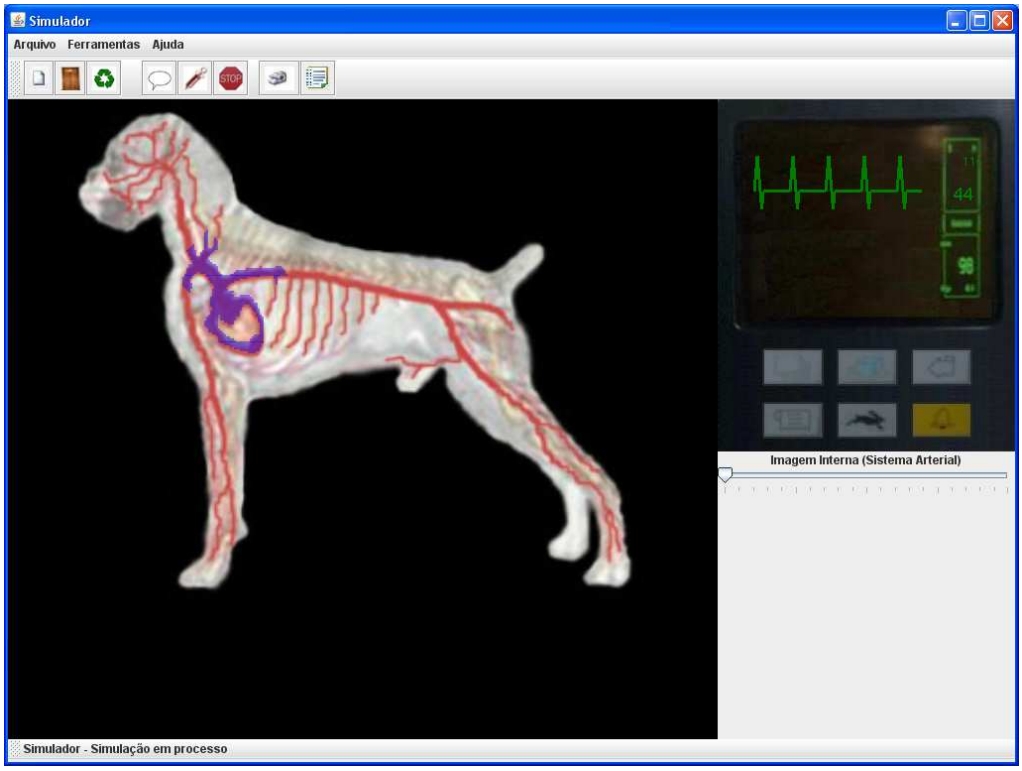


Figura 25 – Animação do elemento químico dexametasona

A última funcionalidade que o sistema tem a oferecer é o relatório automático. Para executar e exportar o relatório para arquivo PDF basta selecionar a opção `Ferramentas`, `Relatório` ou na barra de ferramentas a opção `Relatório`. Ao clicar nesta funcionalidade o simulador apresenta a seguinte tela (figura 26).

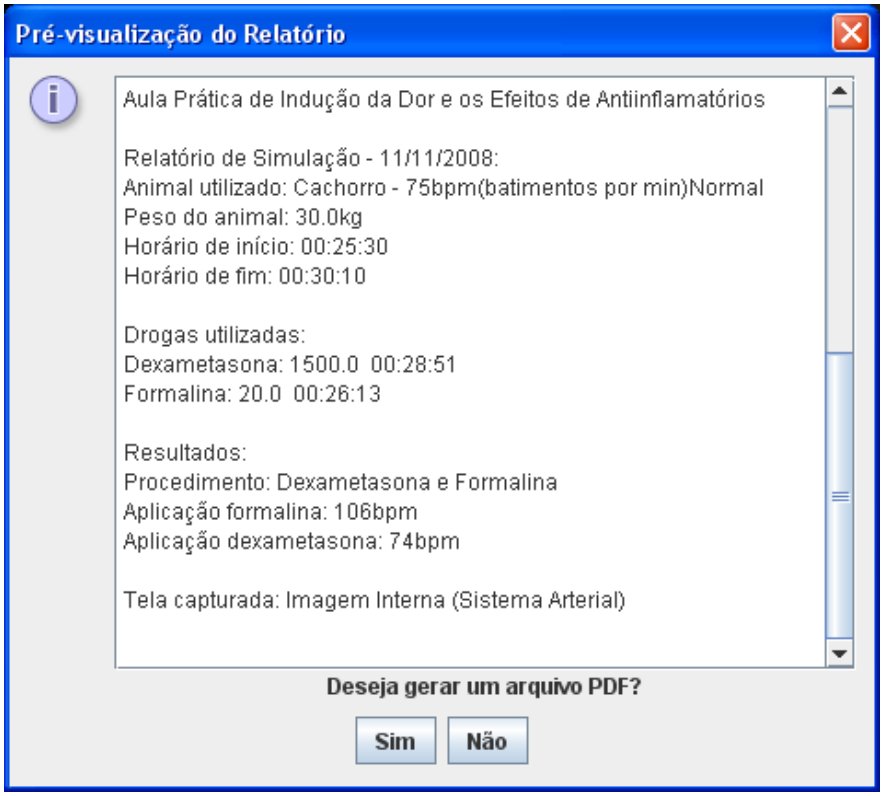


Figura 26 – Pré-visualização do relatório de simulação

Esta tela mostra uma pré-visualização do relatório até o momento e faz a solicitação ao usuário se deseja gerar um arquivo PDF deste relatório. Selecionando sim ele abre uma janela para salvar o arquivo semelhante a figura 21, caso contrário, nenhuma ação é realizada.

O relatório em arquivo PDF é gerado como na figura 27.

UNIVERSIDADE REGIONAL DE BLUMENAU
BACHARELADO EM CIÊNCIAS DA COMPUTAÇAO
SIMULADOR DE ANIMAIS VIVOS - MEIOS ALTERNATIVOS

Aula Prática de Indução da Dor e os Efeitos de Antiinflamatórios sobre Ela

Relatório de Simulação - 11/11/2008:

Animal utilizado: Cachorro - 75bpm(batimentos por minuto) Normal
Peso do animal: 30.0kg
Horário de início: 00:25:30
Horário de fim: 00:37:22
Drogas utilizadas:

DROGA	DOSE	Qtd aplicada	HORÁRIO
Dexametasona	50mg/kg	1500.0mg	00:28:51
Formalina	20µl	20.0µl	00:26:13

Resultados:

Animal	Cachorro	
Procedimento	Dexametasona e Formalina	
Tempo	Formalina	Dexametasona
Valores Obtidos	106bpm	74bpm

Tela capturada:

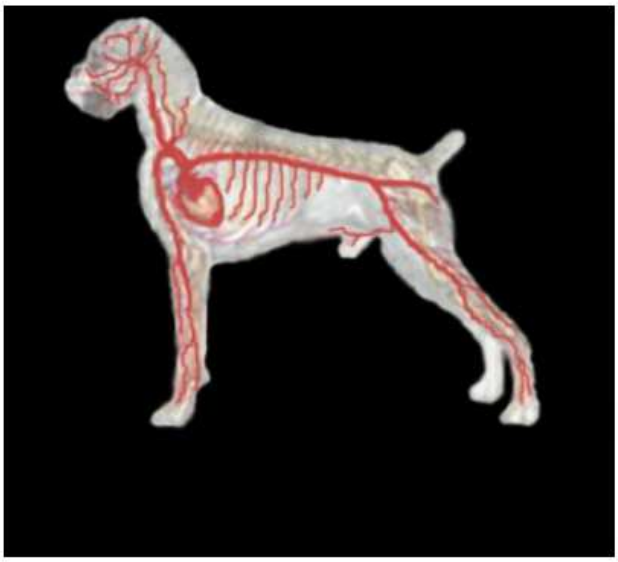


Figura 27 – Exemplo de relatório gerado em arquivo PDF

3.4 RESULTADOS E DISCUSSÃO

Com o desenvolvimento deste simulador pôde-se concretizar a aproximação de duas áreas de tecnologias distintas, a Medicina e a Computação. Inicialmente existia um receio sobre como seriam determinadas as questões da área da Medicina Veterinária, mas com a ajuda do professor Sílvio Luis Negrão e as acadêmicas Ana Júlia Girardi e Gisele Floriani do curso de veterinária da FURB foi possível fazer uma mapeamento básico destas questões. O resultado desta aproximação determinou a construção de um software simulador, de laboratório, que automatiza uma aula, IDAA, que é executada frequentemente no curso de Medicina Veterinária. Hoje os resultados obtidos por esta aula, na maioria das vezes, já são fenômenos conhecidos. Outro fator importante é que as funcionalidades do simulador trouxeram avanços com relação às dificuldades encontradas atualmente com a aula prática em laboratório. Dentre estes avanços pode-se citar a utilização de eletrocardiograma que não era possível ser instalado em todos os experimentos em laboratório, além da possibilidade de verificar a droga progredindo no sistema arterial do cachorro e a geração de relatório automático e preciso.

Ao iniciar o desenvolvimento do simulador procurou-se por materiais referentes a Medicina Veterinária, mais especificamente anatomia canina, em fontes vetoriais e matriciais. Não foi encontrado nenhuma fonte vetorial livre, mas foram encontradas fontes vetoriais pagas em sites de venda de modelagens 3D, criadas em sua maioria por softwares como Blender (BLENDER, 2008) e 3D Studio Max (AUTODESK, 2008). Foram encontrados dois sites para compra (TURBOSQUID, 2008) (FALLINGPIXEL, 2008), onde suas modelagens 3D apresentavam anatomia do sistema circulatório, respiratório, muscular, ósseo entre outros, sendo cachorro completo ou parcial, e seu valor aproximado é de $ 500,00, dependendo da quantidade de sistemas anatômicos desejados. Em função deste problema optou-se por seguir para as fontes matriciais, e observou-se um novo problema relativo à qualidade e escassez deste material, o que prejudicou diretamente a qualidade do simulador.

O material matricial encontrado, sendo imagens em livros e internet, na maioria de suas vezes foi na forma de partes de membros, o que não ajudou no desenvolvimento porque havia necessidade de ter uma imagem do animal por completo. A união destas partes também tornou-se inviável por serem partes de animas diferentes e por falta de todas as partes dos animais. A solução foi utilizar a ferramenta Adobe Photoshop CS2 para tentar corrigir e adequar as imagens.

Dentre os resultados obtidos observou-se que o simulador atende ao quesito didático por parte de alguns usuários que testaram o simulador. Alguns quesitos com relação veracidade dos valores do eletro e as aplicações de elementos químicos são parcialmente hipotéticos e, portanto ficando como sugestão para trabalhos futuros um mapeamento mais profundo de como os elementos químicos se comportam no sistema circulatório dos animais. Um mapeamento com maior quantidade de reações dos animais também torna-se interessante, não deixando apenas o batimento cardíaco como reação. Outra limitação apresentada pelo simulador é a baixa performance do algoritmo de preenchimento, o que acarreta problemas em computadores de baixo porte de processamento.

A aplicação do elemento químico não está completa, faltando algumas consistências na aplicação de um elemento precedido por outro elemento, ficando também para trabalhos futuros. Na tentativa de corrigir este problema foi criada a classe `AtualizaFill`, que seria responsável por atualizar, com a troca de cores das artérias, as aplicações dos elementos. A não finalização desta classe ocasionou a dissipação do elemento muito rapidamente, além de que, quando existe a troca do elemento predominante no sistema arterial ela ocorre imediatamente, o que deveria acontecer de acordo com as características dos componentes, apresentando assim falhas técnicas de Medicina Veterinária.

Com relação ao algoritmo de preenchimento, *floodfill* (seção 2.2.1.2), foi inicialmente implementado de modo recursivo, o que acarretou em baixa performance e comportamento não esperado da animação. Não foi possível determinar a causa que resultou neste problema, e foi decidido implementar de forma não recursiva, aumentando a quantidade de código, mas melhorando o desempenho e o comportamento da animação.

Adquirir mais imagens de anatomia canina e com melhor qualidade seria de grande serventia para trabalhos futuros e tornaria o simulador uma ferramenta mais atrativa para o ensino. Outra opção seria adquirir uma modelagem 3D da anatomia canina, ou criar tal modelagem, com o intuito de torná-lo mais completo através de um módulo 3D, aumentando consideravelmente a motivação para seu uso.

4 CONCLUSÕES

Após o estudo sobre a aula prática IDAA de Farmacologia do curso de Medicina Veterinária, observou-se viável o desenvolvimento de um simulador para esta aula prática. Muitos são os benefícios trazidos com a possibilidade de efetuar uma simulação da aula prática de IDAA, sejam eles por questões éticas, por questões práticas ou por questões didáticas.

Com a simulação da aula prática de IDAA espera-se obter uma análise prévia dos experimentos, podendo assim o acadêmico, avaliar um comportamento no simulador para posteriormente comprová-lo na prática e prevenir futuros cenários de exceção.

Para o desenvolvimento do simulador realizou-se entrevistas com acadêmicos, monitores e professores do curso de Medicina Veterinária da FURB, a fim de estudar elementos químicos e possíveis ações e reações do animal. Deste mapeamento converteu-se vários métodos que foram implementados com recursos gráficos da biblioteca JOGL. Também verificou-se livros e artigos a fim de automatizar a aula prática de IDAA, convertendo-se em animações, como visto em todo o capítulo 2, com o intuito de simular o mais próximo do mundo real a aula prática. Com o simulador também é possível demonstrar uma possibilidade didática alternativa por meio de simuladores.

Quanto aos quesitos didáticos, pode-se avaliar um avanço podendo o usuário ou aluno aprender a utilizar os elementos químicos na aula prática e ao mesmo tempo verificar seu comportamento no sistema arterial do animal qual está sendo simulado, além de monitorar sua freqüência cardíaca com o eletrocardiograma, o que raramente era utilizado em laboratório.

Com relação ao eletrocardiograma é possível o usuário verificar que a maioria das reações referentes às drogas aplicadas ao animal reflete diretamente no eletro, sendo de acordo com a quantidade inserida ou tipo de droga aplicada, o que raramente era vivenciado nas aulas práticas em laboratório.

Também foi possível observar a utilização de técnicas de tratamento de imagens como o *blend*, qual se mostrou muito eficiente para demonstrar as camadas internas do animal e o *floodfill*, demonstrando o comportamento dos elementos químicos no sistema arterial, técnica que também se mostrou eficiente e de várias aplicações na computação especialmente na área gráfica.

Com relação aos trabalhos correlatos apresentados, verifica-se que o simulador obteve algumas das funcionalidades existentes nas ferramentas disponíveis no mercado, levando-se

em conta que a seqüência de simulação e seu comportamento será baseado no simulador ExPahrm, mas com abordagem de foco em áreas diferentes. Criou-se uma interface que possibilita carregar, além da simulação do animal virtual, uma animação com a leitura dos sinais vitais do animal, através do eletrocardiograma. No entanto, a principal diferença em relação aos trabalhos correlatos é que não existe um simulador ideal para a aula prática IDAA de Farmacologia, segundo a entrevista inicial com acadêmicos do curso de Medicina Veterinária da FURB. E este simulador de animais vivos atende aos objetivos e pré-requisitos desta aula. Por fim vale ressaltar que o mercado, principalmente brasileiro, carece de simuladores para fins acadêmicos de medicina e veterinária, o que releva o investimento em desenvolvimento para a área.

4.1 EXTENSÕES

Sugerem-se as seguintes extensões para continuidade do trabalho:

a) permitir ao usuário criar, alterar, salvar e/ou excluir as respostas dos questionários e anotações realizados nas aulas práticas de IDAA;
b) efetuar as consistências necessárias a classe `AtualizaFill` para que a aplicação dos elementos aconteça de forma real;
c) permitir ao usuário exportar vídeo da aula prática de IDAA no formato *Moving Picture Experts Group* versão 4 (MPEG4);
d) implementação de módulos com Java *Media Framework* (JMF) para a integração de tecnologias multimídia;
e) pesquisa de melhores fontes matriciais de imagem a fim de melhorar a qualidade visual do simulador;
f) implementar uma modelagem 3D da anatomia canina a fim de implementação de um módulo 3D do simulador, com possibilidade de reações mecânicas;
g) buscar aprofundamento teórico com relação as reações e ações oferecidas pelo cachorro. Buscar realmente o que acontece, na artéria do animal, quando se insere determinado elemento químico no cachorro, verificando fluxo sanguíneo e velocidade do mesmo.

REFERÊNCIAS BIBLIOGRÁFICAS

ALGORITMO de inundação. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2008. Disponível em: <http://pt.wikipedia.org/wiki/Algoritmo_de_inunda%C3%A7%C3%A3o>. Acesso em: 15 nov. 2008.

ANTEPROJETO de lei. **Código de proteção aos animais do estado.** São Paulo, 2005. Disponível em: <www.cetesb.sp.gov.br/licenciamentoo/legislacao/estadual/leis/2005_Lei_Est_11977.pdf>. Acesso em: 15 nov. 2008.

ANTEPROJETO de lei. C**ondutas e atividades lesivas ao meio ambiente.** Brasília, 1998. Disponível em: <http://www.planalto.gov.br/ccivil_03/LEIS/L9605.htm>. Acesso em: 15 nov. 2008.

AUDACITY. [S. l.], 2008. Disponível em: <http://audacity.sourceforge.net/>. Acesso em: 15 nov. 2008.

AUTODESK. [S.l.], 2008. Disponível em: <http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5659302>. Acesso em: 15 nov. 2008.

BLENDER. [S.l.], 2008. Disponível em <http://www.blender.org/>. Acesso em: 15 nov. 2008.

BOWEN, R. **Peritoneum, mesentery and omentum.** [S.l.], [2006]. Disponível em: <http://www.vivo.colostate.edu/hbooks/pathphys/misc_topics/peritoneum.html>. Acesso em: 13 abr. 2008.

CORNÉLIO FILHO, P. **O modelo de simulação do GPCP-1:** jogo do planejamento e controle da produção. Florianópolis, 1998. Disponível em: <http://www.eps.ufsc.br/disserta98/plinio/index.htm>. Acesso em: 15 nov. 2008.

EDUCAR: refração – fundamentos teóricos. [S. l.], 2003. Disponível em: <http://educar.sc.usp.br/otica/refracao.htm#introducao>. Acesso em: 15 nov. 2008.

FALLINGPIXEL. [S.l.], 2008. Disponível em: <http://www.fallingpixel.com/product.php/5479>. Acesso em: 15 nov. 2008.

FERNANDES, A. C. **Protótipo de visualizador para modelos de cor para medições de objetos em espectrofotômetros por reflectância.** 2002. 76 f. Trabalho de Conclusão de Curso (Bacharel em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FERNANDES, A. R. **Programação de shaders environment mapping:** reflexão e refração. [S. l.], 2006. Disponível em: <http://sim.di.uminho.pt/disciplinas/cg/0607/at/cg08-shadersIV.pdf>. Acesso em: 15 nov. 2008.

FERREIRA, M. D. C.; REGOBELLO, R. **Preenchimento de polígonos.** São Carlos, 2007. Disponível em: <http://cg2007.1.googlepages.com/PreenchimentodePolgonos.pdf>. Acesso em: 15 nov. 2008.

GOUSSEVSKAIA, O. N. **Geração de curvas de roteamento para redes de sensores sem fio.** 2005. 85 f. Dissertação (Mestrado em Ciências da Computação) - Curso de Pós-Graduação em Ciências da Computação, Universidade Federal de Minas Gerais, Belo Horizonte.

HELLEBREKERS, L. J. **Dor em animais.** São Paulo: Manole, 2002.

HUGHES, I. E. **Isolated phrenic nerve – diaphragm.** [S.l.], [2003]. Disponível em: <http://oslovet.veths.no/produkt.aspx?produkt=32>. Acesso em: 13 abr. 2008.

ITEXT. Gent, 2008. Disponível em: <http://www.lowagie.com/iText>. Acesso em: 15 nov. 2008.

LASALLE: reflexão da luz. [S.l.], 2003. Disponível em: <http://www.lasallecaxias.com.br/alunos/fisica/espelhos/eplano.htm>. Acesso em: 15 nov. 2008.

MICHAELIS: moderno. [S.l.], 2007. Disponível em: <http://michaelis.uol.com.br/>. Acesso em: 15 nov. 2008.

NEHE: lession 08. [S.l.], 2008. Disponível em: <http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=08>. Acesso em: 15 nov. 2008.

OPENGL: 15 transparency, translucency, and blending. [S.l.], 2008. Disponível em: <http://www.opengl.org/resources/faq/technical/transparency.htm>. Acesso em: 15 nov. 2008.

PAULA FILHO, W. **Multimídia:** conceitos e aplicações**.** Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2000.

RANG, H. P.; DALE, M. M. **Farmacologia.** 2. ed. Tradução de Penildon Silva, Claudia Lucia Caetano de Araujo, Patricia Lydie Voeux Pinho. Rio de Janeiro: Guanabara Koogan, 1993.

RAVEENDRAN, R.; SHASHINDRAN, C. H.; KALATHI, R. P. **ExPharm version T1.00.** [S.l.], [2005]. Disponível em: <http://oslovet.veths.no/produkt.aspx?produkt=5014>. Acesso em: 13 abr. 2008.

RAYMUND, M. M.; GOLDIM, J. R. Ética da pesquisa em modelos animais. **BIOETICA,** Brasília, v. 10, n. 1, p. 31-44, mar. 2002.

SCHEINDLIN, S. **A brief history of pharmacology.** [S.l.], [2001]. Disponível em: <http://pubs.acs.org/subscribe/journals/mdd/v04/i05/html/05timeline.html>. Acesso em: 13 abr. 2008.

SCHULTER, F. **Simulador de uma partida de futebol com robôs virtuais.** 2007. 90 f. Trabalho de Conclusão de Curso (Bacharel em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SPINOSA, H. S.; GÓRNIAK, S. L.; BERNARDI, M. M. **Farmacologia aplicada a medicina veterinária.** 4. ed. Rio de Janeiro: Guanabara Koogan, 2006.

TURBOSQUID. [S.l.], 2008. Disponível em: <http://www.turbosquid.com/FullPreview/Index.cfm/ID/320296>. Acesso em: 15 nov. 2008.

UML. **Unified modeling language specification:** version 1.4. [S.l.], 2002. Disponível em: <http://www.omg.org>. Acesso em: 15 nov. 2008.

VIU, M. A. O. et al. Emprego de simulação através de modelos bioeconômicos em programas de melhoramento genético animal. **PUBVET,** Londrina, v. 2, n. 5, p. 1-29, fev. 2008.

WILGENBURG, H. **Microlabs for pharmacologists.** [S.l.], [1997]. Disponível em: <http://oslovet.veths.no/produkt.aspx?produkt=5381>. Acesso em: 2 jun. 2008.