FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
MANTENEDORA DO CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA–UNIVEM
BACHARELADO DE CIÊNCIA DA COMPUTAÇÃO

**GUILHERME OELSEN FRANCHI JUNIOR**
**RAFAEL EUCLIDES DA SILVA**

# PROJETO E IMPLEMENTAÇÃO DE JOGOS EDUCATIVOS NA ÁREA DE QUÍMICA

MARÍLIA
2006

GUILHERME OELSEN FRANCHI JUNIOR
RAFAEL EUCLIDES DA SILVA

# PROJETO E IMPLEMENTAÇÃO DE JOGOS EDUCATIVOS NA ÁREA DE QUÍMICA

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília - UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador:
Prof. Dr. Ildeberto Aparecido Rodello

MARÍLIA
2006

FRANCHI JUNIOR, Guilherme Oelsen; SILVA, Rafael Euclides da.
Projeto e implementação de jogos educacionais na área de Química / Guilherme Oelsen Franchi Junior, Rafael Euclides da Silva; orientador: Prof. Dr. Ildeberto Aparecido Rodello.
Marília, SP: [s.n.],2006.
101f.

Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípedes de Marília, Fundação de Ensino Eurípedes Soares da Rocha.

1. Elementos Químicos 2. *Engines* 3. Jogos Educacionais 4. Moléculas e Substâncias Químicas 5. *Softwares* Educacionais.

CDD: 006

GUILHERME OELSEN FRANCHI JUNIOR

# PROJETO E IMPLEMENTAÇÃO DE JOGOS EDUCATIVOS NA ÁREA DE QUÍMICA

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação, do UNIVEM,/F.E.E.S.R., como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Resultado: Aprovado

ORIENTADOR: Prof. Dr. Ildeberto Aparecido Rodello

1º EXAMINADOR: Prof. Dr. José Remo Ferreira Brega

2º EXAMINADOR: Prof. Dr. Antonio Carlos Sementille

Marília, 01 de dezembro de 2006.

RAFAEL EUCLIDES DA SILVA

# PROJETO E IMPLEMENTAÇÃO DE JOGOS EDUCATIVOS NA ÁREA DE QUÍMICA

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação, do UNIVEM,/F.E.E.S.R., como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Resultado: Aprovado

ORIENTADOR: Prof. Dr. Ildeberto Aparecido Rodello

1º EXAMINADOR: Prof. Dr. José Remo Ferreira Brega

2º EXAMINADOR: Prof. Dr. Antonio Carlos Sementille

Marília, 01 de dezembro de 2006.

# AGRADECIMENTOS

*Agradeço primeiramente a Deus por todas as bênçãos que me têm concedido. Obrigado, Senhor!*

*Aos meus pais, Guilherme Oelsen Franchi e Verlaine Sarlo Rocha Franchi, que, com muito amor, carinho e compreensão me apoiaram em toda a minha vida, inclusive na conclusão deste trabalho, acreditando e sempre dando o maior apoio em todos os meus sonhos. Pai e Mãe, amo muito vocês.*

*Ao meu Prof. Dr. Ildeberto Aparecido Rodello pela orientação, ajuda e incentivo durante todo o período de desenvolvimento deste trabalho. Beto, obrigado por tudo!*

*A todos os professores do curso de Bacharelado de Ciência da Computação pelo incentivo e contribuição na minha formação. Obrigado a todos vocês.*

*À minha namorada, Laís Danielli A. Soto, que sempre esteve ao meu lado me incentivando e me apoiando em tudo. Sempre me lembrando de que para Deus nada é impossível, e por estar sempre comigo nos momentos mais difíceis, e também nos mais alegres. Lá, te amo demais.*

*Aos meus amigos Thiago Tobias, Thiago Donófrio e Rafael Euclides, que desempenharam papéis muito especiais durante todo o período de desenvolvimento deste trabalho, me apoiando e ajudando com dicas, criatividades e soluções. Valeu galera.*

*Guilherme*

# AGRADECIMENTOS

*Agradeço a Deus por ter me dado força e sabedoria para vencer essa etapa da minha vida.*

*Aos meus pais, Benedito Teodoro da Silva e Maria Luzinete Euclides, e aos meus irmãos, Renato e Rodrigo, que fizeram possível a realização desse trabalho, apoiando-me em todos os momentos.*

*Ao Prof. Dr. Ildeberto Aparecido Rodello pela dedicação, orientação, incentivo, fundamentais para o desenvolvimento deste trabalho.*

*A todos os professores do curso de Bacharelado de Ciência da Computação, que muito contribuíram para a ampliação dos meus conhecimentos.*

*Aos meus amigos que estiveram comigo durante a graduação, em especial a Thiago Tobias e Guilherme Oelsen, pelo apoio e amizade.*

*Rafael*

FRANCHI JUNIOR, Guilherme Oelsen; SILVA, Rafael Euclides da. **Projeto e implementação de jogos educacionais na área de Química.** 2006, 101f. Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípedes de Marília, Fundação de Ensino Eurípedes Soares da Rocha, Marília, 2006.

## RESUMO

Este trabalho tem como finalidade o desenvolvimento de jogos educativos capazes de auxiliar na fixação dos tópicos Moléculas e Substâncias Químicas e Elementos Químicos para alunos do ensino fundamental, utilizando *engines* gratuitas, específicas para o desenvolvimento de jogos. Este trabalho aborda um estudo sobre *softwares* educacionais, incluindo os jogos educacionais, e apresenta as *engines jMonkey* e *Xith3D*.

**Palavras-chave:** Elementos Químicos. *Engines*. Jogos Educacionais. Moléculas e Substâncias Químicas. *Softwares* Educacionais.

FRANCHI JUNIOR, Guilherme Oelsen; SILVA, Rafael Euclides da. **Projeto e implementação de jogos educacionais na área de Química.** 2006, 101f. Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípedes de Marília, Fundação de Ensino Eurípedes Soares da Rocha, Marília, 2006.

## ABSTRACT

This work has as purpose the development of educational games capable to assist in the fixation of Chemistry's Molecules and Substances and Chemistry's Elements topics for students of Fundamental Education, using free engines, specifics to the development of games. This work approaches an study on educational softwares, including educational games, and presents the engines jMonkey and Xith3D.

**Keywords:** Chemistry's Elements. Chemistry's Molecules and Substances. Educational Games. Educational Softwares. Engines.

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

**API** – *Application Programming Interface*

**BSD** – *Berkeley Software Distribution*

**CAI** – *Computer Aided Instructio*n

**CVS –** *Concurrent Versions System*

**EAD** – Educação à Distância

**FPS** – *Frames Per Second*

**ILE** – *Interactive/Intelligent Learning Environment*

**ITS** – *Intelligent Tutoring Syste*m

**jME** – *jMonkey Engine*

**JDK –** *Java Development Kit*

**JOGL** - *Java bindings for OpenGL*

**LWJGL** – *Lightweight Java Game Library*

**MIT** – *Massachusetts Institute of Technology*

**RPG –** *Role Playing Game*

**SVN –** *Subversion*

**VBL** – *Virtual Biochemistry Laboratory*

# SUMÁRIO

# INTRODUÇÃO

Os computadores têm se tornado de extrema importância quando se diz respeito ao aprendizado. São consideradas ferramentas poderosas no qual todas as suas potencialidades podem ser utilizadas com propósitos educacionais. Os computadores também permitem a criação e desenvolvimento de *softwares* e jogos educacionais, e a utilização destes *softwares* em sala de aula.

Martins et al. (2002) citam os jogos educacionais como um "catalisador", ou seja, os jogos educacionais tendem a melhorar a motivação, aumentando a produtividade do aluno e, conseqüentemente, a qualidade do ensino.

Para o desenvolvimento dos jogos existentes, os desenvolvedores criam e utilizam *engines*, que são bibliotecas para as mais diversas linguagens de programação e que possuem a maioria dos recursos necessários para a criação de um jogo.

O surgimento de *engines* cada vez mais robustas e completas permite ao desenvolvedor utilizá-las na criação de *softwares* com fins educacionais. Os jogos educacionais tornam-se mais interessantes quando é possível visualizar o ambiente como se fosse uma representação do mundo real.

Com isso, a finalidade de criar jogos educacionais é ativar o interesse do aluno em querer aprender mais e mais de uma maneira divertida e interessante.

## 1.1 Objetivo

Nesse contexto, o objetivo deste trabalho é criar dois jogos educacionais, utilizando as características de *softwares* educacionais e a utilização de *engines* gratuitas específicas para

a construção de jogos. Os jogos a serem desenvolvidos utilizarão as *engines* de distribuições gratuitas *jME* e *Xith3D*, e irão auxiliar no aprendizado de Substâncias e Moléculas Químicas e também os Elementos químicos presentes na Tabela Periódica, tópicos apresentados na matéria de Ciências do Ensino Fundamental.

## 1.2 Estrutura do trabalho

O trabalho está dividido em seis Capítulos, incluindo o Capítulo introdutório.

O Capítulo 2 apresenta a definição de um *Software* Educacional, mostrando sua classificação e como este tipo de *software* auxilia no ensino educacional, com destaque para os jogos educacionais, apresentando sua definição e classificações, além de dois exemplos de jogos educacionais.

O Capítulo 3 apresenta as *engines Xith3D* e *jMonkey*, estudados para serem utilizados na implementação dos jogos educacionais, mostrando os recursos que cada uma dessas *engines* possuem.

Os Capítulos 4 e 5 descrevem o processo de desenvolvimento e a implementação do jogo E.C.A.S. e do Jogo dos Elementos, respectivamente, utilizando as *engines jME* e *Xith3D* e os resultados obtidos durante a implementação. Nestes capítulos também são fornecidas as informações a respeito dos jogos e suas jogabilidades.

O trabalho é concluído no Capítulo 6, avaliando os aspectos gerais, a viabilidade dos projetos e trabalhos futuros.

Os Anexos A e B, que se encontram no final deste trabalho, explicam como obter e instalar as *engines jME* e *Xith3D*, respectivamente.

# CAPÍTULO 2 - SOFTWARE EDUCACIONAL

Este Capítulo apresenta a definição de *Software* Educacional, mostrando as suas classificações e como estes tipos de *softwares* ajudam no ensino educacional. Neste Capítulo também são apresentadas as classificações de jogos educacionais, e suas respectivas definições. Ao final, são apresentados dois exemplos de Jogos Educacionais.

## 2.1 Definição e Caracterização

O computador tem sido utilizado na educação não apenas para auxiliar o ensino sobre computação, mas também como ferramenta para ensinar praticamente qualquer assunto (ensino por meio do computador). No ensino de computação, o aluno utiliza o computador para adquirir conceitos computacionais, como princípios de funcionamento do computador, noções de programação e implicações sociais do computador na sociedade. Por outro lado, o ensino por meio do computador auxilia o aluno a ter condições de adquirir e fixar conceitos sobre qualquer campo do conhecimento (VALENTE, 1995).

Para que computadores possam ser utilizados no ensino, se faz necessário o desenvolvimento de *softwares* específicos para esta área. Para a utilização destes *softwares*, certas características adquiridas por métodos pedagógicos precisam ser empregadas.

Portanto, o *software*, sozinho, não implica em nenhuma mudança no processo educacional se não for utilizado dentro de um contexto que envolva o projeto político-pedagógico da Instituição no qual o *software* é empregado. O *Software* Educacional deve ser visto como uma ferramenta que facilite o fazer em um ambiente pedagógico. O seu conteúdo deve ser apresentado de forma objetiva, priorizando a interatividade e a criatividade e

fornecendo sempre o retorno necessário. Por outro lado, ele deve estimular, provocar e desafiar o aluno, a fim de prender a sua atenção (VALENTE, 2002).

Os *softwares* existentes no mercado para promover o ensino, mostram que a tarefa do professor se torna mais fácil de ser desempenhada com o auxílio do computador e, talvez, aumentando até a sua eficiência. (VALENTE, 2002)

## 2.2 Classificação de Softwares Educacionais

A classificação de *Softwares* Educacionais apresentada nessa seção está relacionada a uma escala evolutiva (ordem em que os diversos tipos de *softwares* educacionais foram surgindo), começando por sistemas onde o usuário obtinha uma única resposta correta por meio do computador, obtendo um único caminho a seguir (Sistemas de Instrução Assistida por Computador), evoluindo até chegar aos Sistemas com Ambientes Inteligentes e Interativos de Aprendizagem, onde os *softwares* educacionais obtêm respostas utilizando técnicas de Inteligência Artificial Distribuídas (cooperativismo) e, no qual, também são incorporadas técnicas de Realidade Virtual.

Barros e Araújo (1998) classificam os *softwares* educacionais da seguinte maneira:

1. Sistemas de Instrução Assistida por Computador (CAI – *Computer Aided Instructio*n)
2. Micromundos
3. Simuladores e Jogos Educacionais
4. Sistemas Tutores Inteligentes (ITS – *Intelligent Tutoring Syste*m)
5. Sistemas com Ambientes Inteligentes e Interativos de Aprendizagem

Os CAI são sistemas que adotam a filosofia de ensino pelo computador, sem personalizações de conteúdos e sem considerar as necessidades individuais de cada aluno.

Os micromundos têm proposta pedagógica contrária a dos CAIs, adotando a filosofia de aprendizagem pela ação, onde os alunos têm liberdade para definir os caminhos de construção do conhecimento.

Os simuladores permitem ao usuário testar, de forma virtual, as reações de um determinado ambiente, de acordo com suas ações. Os jogos incorporam aspectos lúdicos ao processo de aprendizagem. Em termos pedagógicos podem, dependendo do tipo de *software*, ter a filosofia operacional mais próxima dos CAIs ou dos micromundos.

Os jogos educacionais tem como característica fixar o aprendizado obtido em sala de aula pelo aluno. Este tipo de *software* será discutido na Seção 2.3.

Os ITS nascem da incorporação de recursos de inteligência artificial e de conceitos cognitivos na área de psicologia e educação. A arquitetura modular destes sistemas reflete as questões: o que ensinar (modelo de domínio ou do especialista), a quem ensinar (modelo do estudante) e como ensinar (modelo pedagógico ou do tutor).

Os sistemas com ambientes inteligentes e interativos de aprendizagem são semelhantes aos ITS, porém com algumas melhorias como a incorporação de recursos de trabalho cooperativo e recursos de inteligência artificial distribuída, com base no conceito de multi-agentes, aumentando mais a capacidade destes sistemas em termos de interação/cooperação.

## 2.3 Jogos Educacionais

Esta seção apresenta a definição de jogos educacionais e suas classificações em função da interface, objetivos e número de usuários. Também são apresentados dois exemplos de jogos educacionais.

### 2.3.1 Definição

Os jogos educacionais são *softwares* que vêm se tornando muito úteis nos dias de hoje com o intuito de prover melhor aprendizado, desenvolvimento de habilidades e também para um melhor acompanhamento dos resultados dos alunos.

Os jogos educacionais desempenham uma maior motivação, desenvolvendo a vontade de sempre tentar novos desafios e novas dificuldades nos alunos. Estes jogos constituem para as crianças, adolescentes e jovens, a maneira mais fácil e divertida de aprender, além de, segundo Tarouco (2004), se basearem na abordagem auto-dirigida, ou seja, fazer com que o indivíduo aprenda praticamente sozinho, por meio da descoberta de relações e da interação com o programa ou *software*.

Além de aumentar a motivação, os jogos educacionais podem aumentar também a produtividade do aluno, e conseqüentemente, a qualidade do ensino (MARTINS et. al, 2002).

Schank (1997, apud MARTINS 2006) diz que, para a aprendizagem por meio dos jogos educacionais, alguns pontos são de extrema importância. São eles:

- Aprender errando;
- Aprendizagem de modo personalizado (situações alteradas de acordo com o usuário, como nível de dificuldade, tempo, etc...);
- Não humilhação pelos erros;

Muitas vezes os professores sentem grande dificuldade em realizar um acompanhamento do aluno em relação aos seus erros mais freqüentes e à ordem que o aluno executa determinadas tarefas. Esse acompanhamento pode ser realizado pelo computador de uma forma muito mais detalhada. Outro fator muito importante é que os jogos educacionais e sistemas computacionais em geral podem se utilizar de recursos multimídia, como cores, animações e sons, de uma forma que o professor dificilmente poderá apresentar utilizando

somente o giz e o quadro negro (mesmo utilizando giz colorido e ser um comunicador exemplar).

É por via dos jogos educacionais que o aluno aprende melhor, quando ele é livre para descobrir novos caminhos por ele mesmo, ao invés de ser explicitamente ensinado (VALENTE, 1995).

Algo importante a ser considerado, é que os jogos tendem a desviar a atenção do usuário do conceito envolvido no jogo através da competição. A maioria dos jogos não tem a capacidade de diagnosticar as falhas do jogador. Por isso é importante que estes problemas sejam contornados fazendo com que o aprendiz, após uma jogada errada, reflita sobre a causa do erro e tome consciência do seu erro, e, se necessário, fazer o aprendiz tentar novamente (VALENTE, 1995).

É por meio destes erros e na ação do jogo que o indivíduo pode conhecer-se, estabelecendo o limite de sua competência e reavaliando o que precisa ser trabalhado, praticando e desenvolvendo suas potencialidades, para que se evite um próximo erro ou derrota.

Por isso, o computador deve ser visto como uma ferramenta que, de modo geral, auxilia o aprendizado, sendo que o professor tem o importante papel de acompanhar o desenvolvimento e o desempenho do aluno e intervir positivamente, vindo a realizar questões ao aluno sempre que necessário.

### 2.3.2 Classificação de Jogos Educacionais

A classificação de jogos educacionais define algumas de suas características de interface e de implementação. A seguir são apresentadas as classificações em função da interface, dos objetivos e do número de usuários.

## a) Classificação em função da interface

A interface define uma forma particular de atuação do usuário no universo do jogo. Uma possibilidade é o usuário atuar em terceira pessoa (o usuário se vê na cena) ou, atuar em primeira pessoa (a cena exibida representa o que os olhos do usuário vêem).

Em uma interface em terceira pessoa, o personagem que representa o usuário na cena é denominado de ator, é parte integrante da cena e deve ser bem caracterizado. Numa interface em primeira pessoa, o personagem é denominado de avatar. O avatar simplesmente marca o usuário na cena.

A interface também pode variar em termos de projeção gráfica. Utilizam-se projeções planares ortográficas (planta, vista lateral, vista frontal), ou projeções perspectivas, que são utilizadas quando a noção de profundidade é requerida.

## b) Classificação em função dos objetivos

Em função dos objetivos, os jogos podem ser de vários tipos, embora a tendência hoje seja combinar características de cada tipo para propor jogos diferentes. Os principais tipos são os seguintes (TAROUCO et al., 2004):

- Ação
- Estratégia
- Aventura
- Simulação
- *Role Playing Game* (RPG)
- Infantil

Os jogos de ação caracterizam-se por apresentar uma ação ininterrupta e veloz, onde são testadas as principais habilidades do jogador, como tempo de reação e coordenação motora. Segundo Tarouco (2004), estes tipos de jogos, considerando um jogo educacional, auxiliam no desenvolvimento psicomotor, desenvolvendo reflexos, coordenação olho-mão e auxiliando no processo de pensamento rápido frente a uma situação inesperada.

Os jogos de estratégia são normalmente de âmbito tático, onde o objetivo principal é a tomada de decisões que possibilita um desafio no nível intelectual. Segundo Tarouco (2004), esse tipo de jogo, para fins educacionais, pode proporcionar uma simulação em que o usuário aplica conhecimentos adquiridos em sala de aula, percebendo uma forma prática de aplicá-los.

Os jogos de aventura combinam ações baseadas em raciocínio e reflexo, onde o objetivo principal é ultrapassar estágios completando objetivos. Estes tipos de jogos, também com fins educacionais, podem fazer com que o aluno vivencie e administre situações que não podem ser vividas em salas de aula, como, por exemplo, ajudar a combater o desmatamento da Mata Atlântica, ou até mesmo controlar, por meio de missões, a destruição da camada de ozônio (TAROUCO et al., 2004).

Os jogos de simulação têm como objetivo inserir o usuário em um ambiente virtual semelhante ao real. Considerando em um jogo educacional, é possível fazer com que aluno possa ir a lugares impossíveis de se estar no mundo real, como, por exemplo, entrar dentro da corrente sanguínea, ou até mesmo realizar experimentos extremamente perigosos, também impossíveis de serem realizados no mundo real, tal como, estar próximo ao ponto de impacto de uma bomba nuclear e poder visualizar os estragos causados.

Tarouco (2004) define RPG como um jogo em que o usuário controla um personagem em um ambiente. Nesse ambiente, seu personagem encontra outros personagens e com eles interage. Dependendo das ações e escolhas do usuário, os atributos dos personagens podem ir

se alterando, construindo dinamicamente uma história. Esse tipo de jogo é complexo e difícil de desenvolver e pode oferecer um ambiente cativante e motivador para o usuário.

Os jogos infantis enfocam histórias e objetivos simples com a finalidade de divertir o usuário, caracterizam-se por apresentar gráficos bem coloridos, próximos aos dos desenhos animados.

### c) Classificação em função do número de usuários

Em função do número de usuários os jogos se dividem em monousuários e multiusuários.

Por questões de limitações tecnológicas nas áreas de *software*, hardware e de redes de comunicação, os jogos por computador em sua maioria são monousuários, ou seja, suportam apenas um jogador.

Os jogos multiusuários oferecem maior interatividade e realismo, despertando muito mais interesse por parte dos alunos do que um simples jogo monousuário. Isto se deve ao fato de que um jogo distribuído transmite a idéia de competição e relacionamento entre os alunos, proporcionando maior interatividade e interesse por parte dos mesmos.

## 2.3.3 Exemplos de Jogos Educacionais

Considerando que os métodos de ensino-aprendizagem vem evoluindo e adaptando-se as novas tecnologias aliado a crescente necessidade de desenvolver criatividade e autonomia nas crianças e jovens, os jogos por computador passam a serem usados como uma eficiente ferramenta de ensino.

O jogo *Supercharged!* (SUPERCHARGED!, 2006) foi escolhido por ter sido desenvolvido por pesquisadores do MIT com o objetivo de transformar a maneira de como os jogos computacionais são utilizados em sala de aula. O jogo *The Virtual Chemistry Laboratory* (VBL, 2006) foi escolhido por ter se destacado entre os jogos educacionais na área de química, premiados pelo NobelPrize (NOBELPRIZE, 2006).

A seguir são apresentados dois exemplos de jogos por computador como ferramentas educacionais.

## a) Supercharged

No projeto "*The Education Arcade*", coordenado por pesquisadores do MIT (*Massachusetts Institute of Technology*) em parceria com a Universidade de *Wisconsin,* foi desenvolvido o jogo "*Supercharged*". Na Figura 1 é possível visualizar a tela inicial do jogo.


**Figura 1** – Tela Inicial do Jogo *Supercharged*

O jogo coloca o aluno em um ambiente tridimensional onde ele navega em uma nave espacial controlando a carga elétrica da nave, colocando partículas de carga pelo espaço. O aluno deve planejar cuidadosamente sua trajetória por cada nível por meio do controle do

campo que emana dos objetos carregados, e nesse processo, desenvolve o seu entendimento sobre como interagem partículas carregadas. A Figura 2 é uma imagem obtida durante a execução do jogo.

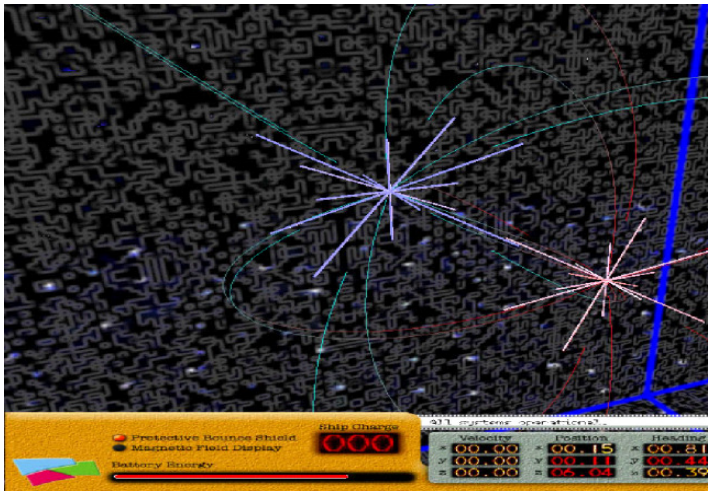


**Figura 2 –** Imagem obtida durante a execução do jogo

## b) The Virtual Biochemistry Laboratory

O projeto *"The Virtual Biochemistry Laboratory"* foi desenvolvido com a cooperação dos professores Dr. Inger Carlberg (Departamento de Bioquímica da Universidade de Stockholm), Prof. Sture Forsén (Universidade de Lund), Dr. Peter Lundberg (Departamento de Diagnóstico de Radiologia do Hospital Universitário de Linköping) e o Prof. Mikael Oliveberg (Universidade de Umea). O jogo é on-line e gratuito, tendo como requisito mínimo um navegador com conexão para Internet e o plugin *Shockwave Player 9* para usuários da plataforma Macintosh e *Shockwave Player 8.5.x* para demais plataformas.

O projeto tem o objetivo de tornar o conhecimento de bioquímica visível através do uso de computadores e Internet como ferramentas para o aprendizado interativo, e ser um caminho para o entendimento dos princípios e teorias por trás dos Prêmios Nobel em química.

O projeto também pretende contribuir para um maior interesse em química no geral e, como parte do projeto *Wallenberg Young Scholar's*, entre os estudantes do ensino médio também.

No ambiente do laboratório é possível mover-se ao redor e interagir com o equipamento do laboratório em uma interface com estilo de jogo. É possível também aprender as lições e realizar experimentos com a assistência de funções de ajuda inteligentes. Por se focar constantemente no usuário final do produto, o projeto visa fornecer uma experiência estimulante e agradável em que o aprendizado ativo do usuário é promovido.

O laboratório consiste de várias salas, cada qual focando em um método particular usado em bioquímica: "Ressonância Nuclear Magnética", "Seqüenciamento de proteína", "Difração de Raio-X" e "Manipulação de Proteína". Os diferentes métodos ajudam os cientistas a descobrir mais sobre moléculas, por exemplo, proteínas.

A Figura 3 mostra a tela de *loading* do jogo, mostrando que o jogo está sendo carregado, e, enquanto isso, é possível responder a um questionário fornecidos pelo próprio jogo.

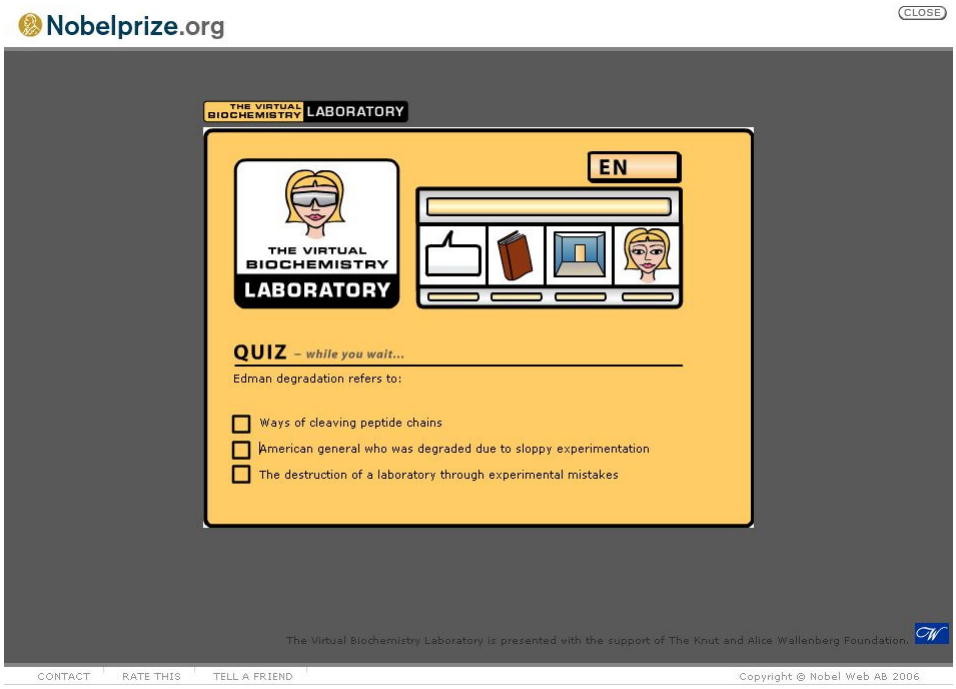


**Figura 3 –** Página de *loading* do jogo

Na Figura 4, é exibida a tela de início do jogo, que pede a entrada de usuário e senha, caso o jogador já possua o seu cadastro, caso contrário, é possível acessar o jogo como visitante (*Guest*).


**Figura 4 –** Tela de *login* necessária para acessar o jogo

Uma das áreas possíveis de estar acessando dentro do jogo, é o mapa (Figura 5), que determina a sua localização e os diversos laboratórios disponíveis (como, por exemplo, o laboratório de difração de raios-X, exibido na Figura 6).

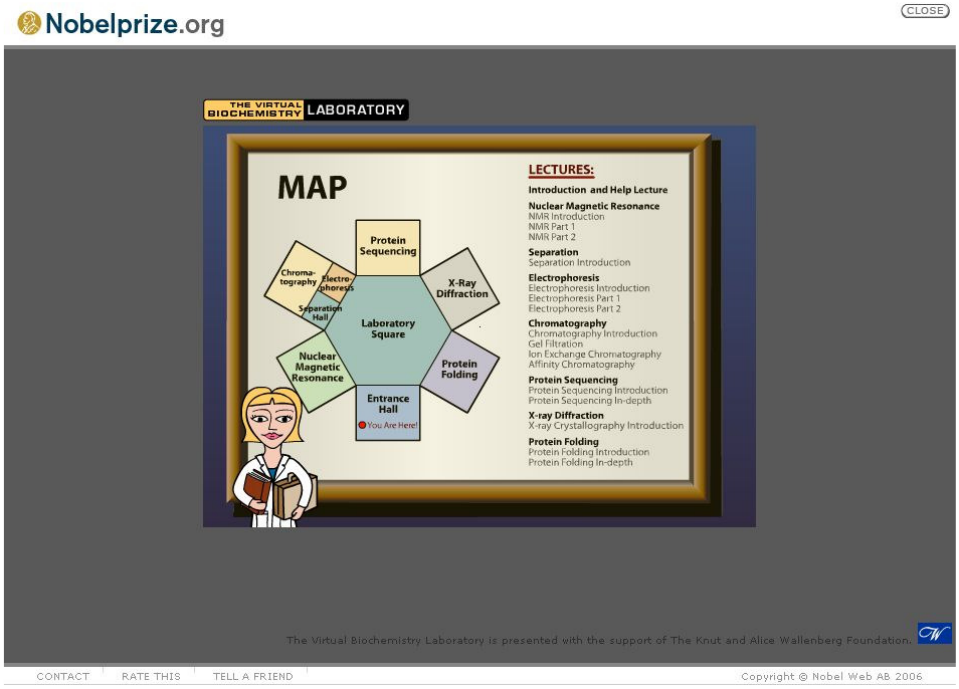


**Figura 5** – Mapa contendo a localização atual e os diversos laboratórios

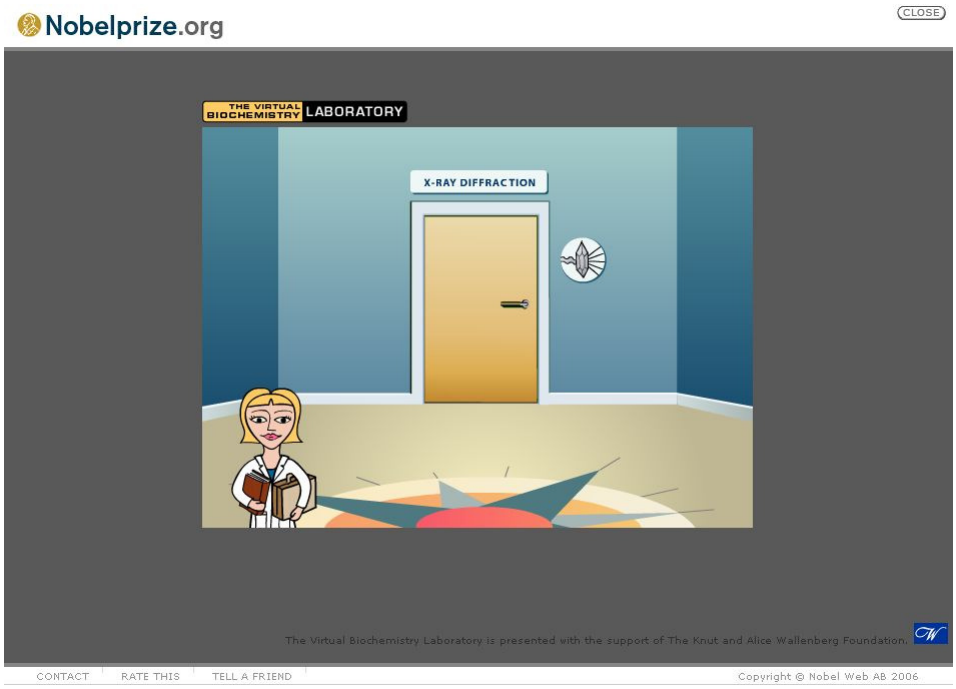


**Figura 6 –** Entrada para o Laboratório de Difração de Raio-X

Por último, a Figura 7 mostra como é dentro de um laboratório, no caso, o laboratório de cromatografia. Nas Figuras 5, 6 e 7 há, do lado inferior esquerdo, uma personagem, que representa a assistente do jogo. Por ela, é possível, ao colocar o mouse na

frente de seu rosto, obter informações muito significativas no decorrer do jogo, como por exemplo, que ações devem ser feitas em determinados laboratórios.

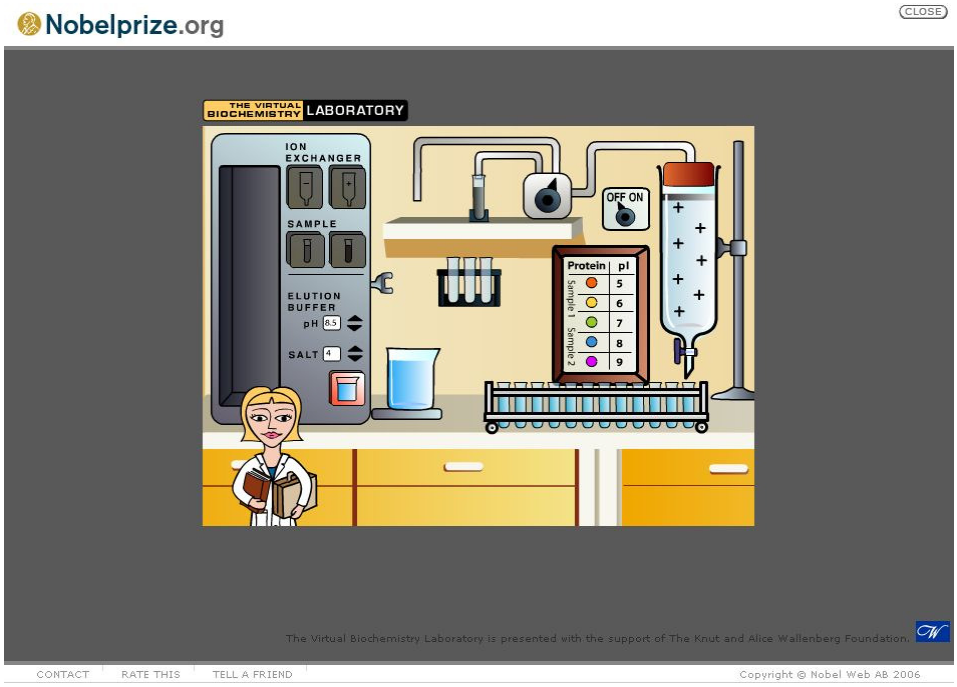


**Figura 7 –** Instrumentação do Laboratório de Cromatografia

# CAPÍTULO 3 - ENGINES PARA JOGOS

Neste Capítulo são apresentados as *engines Xith3D* e *jMonkey*, estudados para serem utilizados na implementação do jogo educacional, mostrando os recursos que cada um desses motores possuem.

Outra *engine* estudada para o desenvolvimento de jogos foi o projeto enJine (ENJINE, 2006). Esta *engine* não foi utilizada por ser uma biblioteca em fase inicial de desenvolvimento e, por isso, não dispor de todos os recursos necessários para criação de jogos robustos e de alta performance.

## 3.1 *Xith3D*

O *Xith3D* (XITH3D, 2006) é uma API (*Application Programming Interface*) de programação 3D baseado em um grafo de cena (*scenegraph*) e focado na criação de jogos. Ela não faz parte do núcleo do *Java Runtime Environment* (SUN, 2006). É um projeto de código aberto (*open-source*), liberado sob uma licença modificada do BSD (*Berkeley Software Distribution*).

O *Xith3D* consiste de um pequeno número de pacotes, nos quais os mais importantes são o grafo de cena e renderizador. O renderizador, implementado nas APIs Jogl (JOGL, 2006) e LWJGL (LWJGL, 2006), é uma camada de abstração que permite ao *Xith3D* funcionar com diferentes APIs. O renderizador é projetado para suportar vários efeitos, como iluminação de alta qualidade, sombreamento de volumes e transparência.

O grafo de cena permite que os dados dos jogos sejam organizados em uma estrutura de árvore, onde um nó pai pode conter qualquer número de nós filhos, mas um nó filho pode

conter um único nó pai. Tipicamente, estes nós são organizados espacialmente para que uma certa quantidade de nós sejam descartados do processamento. Por exemplo, se for construído um grafo de forma que todos os objetos em um cômodo compartilhem um pai (no caso, o cômodo), e todos os cômodos compartilharem um pai (andar de um prédio), no qual todos os andares são compartilhados por um único edifício pai, supõe-se que nosso personagem está no cômodo um do primeiro andar.

Pode-se então rapidamente descartar o nó “segundo andar” (o qual descarta todo cômodo no segundo andar e todos os objetos contidos nos cômodos do segundo andar). Os nós do “primeiro andar” da árvore podem, então, serem processados. Todos os cômodos que não fazem parte do cômodo um são descartados (incluindo todos os objetos dentro destes cômodos). O cômodo um, incluindo seus objetos, são então processados.

A Figura 8 apresenta os componentes básicos do núcleo do *Xith3D*, como o grafo de cena, o renderizador, os componentes para importar sons, texturas e modelos 3D e os componentes para interface com som.

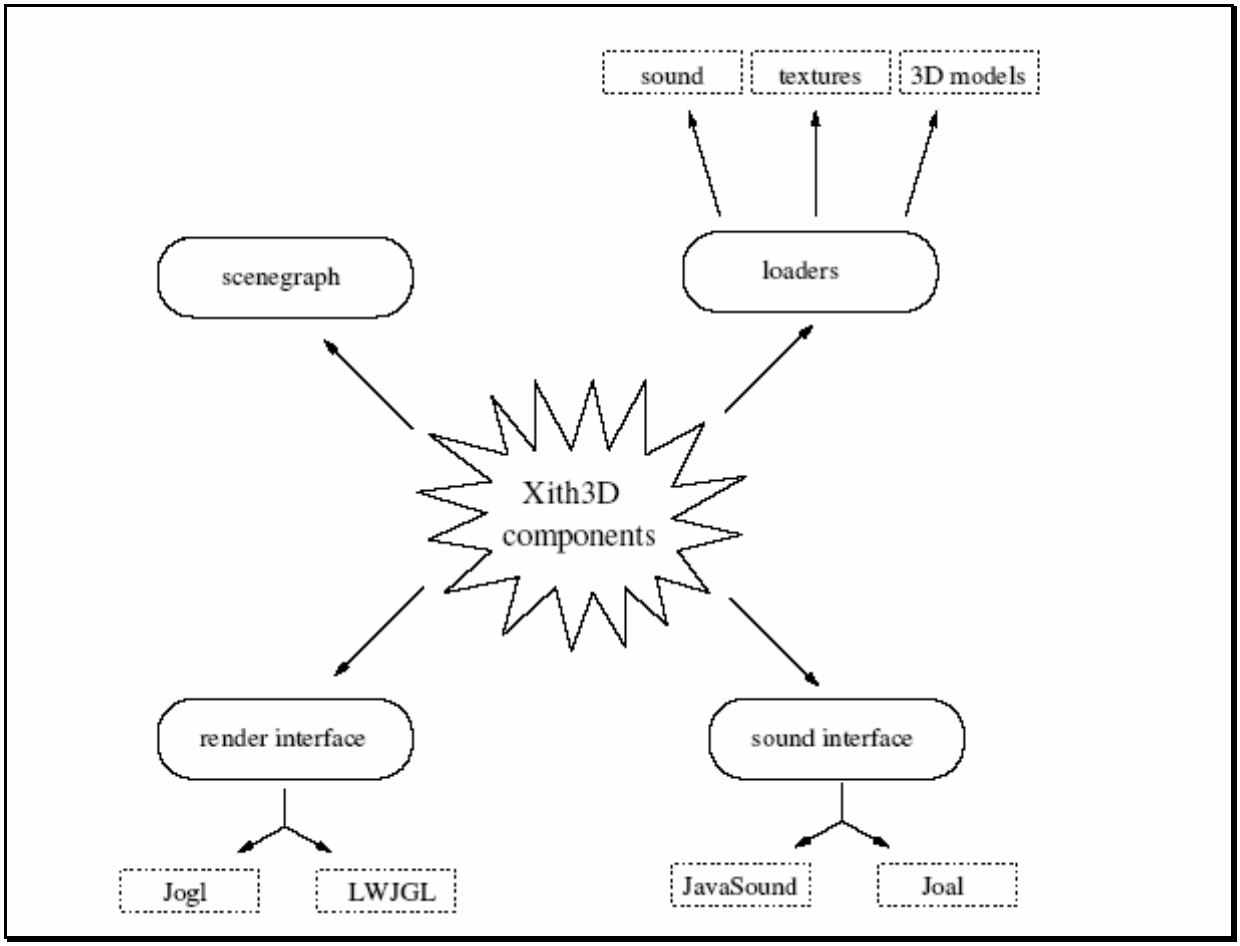


**Figura 8 –** Componentes básicos do núcleo *Xith3D*
(LEHMANN J., 2004)

O *Xith3D* apresenta como desvantagem o fato de atualmente a biblioteca não dispor de uma API de comportamentos, que facilitaria o desenvolvimento de aplicações que envolvam animação e interação do usuário com o ambiente 3D, e também o fato de não possuir uma classe que implemente as colisões entre os objetos da cena.

### 3.3.2 Exemplo de programação

A Figura 9 apresenta o código de um programa utilizando *Xith3D*.

```
1  import java.io.File;
2  import javax.vecmath.Color3f;
3  import net.jtank.input.KeyCode;
4  import org.xith3d.geometry.Cube;
5  import org.xith3d.loaders.texture.TextureLoader;
6  import org.xith3d.loaders.texture.TextureStreamLocatorFile;
7  import org.xith3d.render.base.Xith3DEnvironment;
8  import org.xith3d.render.canvas.Canvas3DWrapper;
9  import org.xith3d.render.loop.RenderLoop;
10 import org.xith3d.scenegraph.Appearance;
11 import org.xith3d.scenegraph.BranchGroup;
12 import org.xith3d.scenegraph.Material;
13 import org.xith3d.scenegraph.Texture;
14
15
16 public class SceneWithCube extends RenderLoop {
17     public void onKeyReleased(int key) {
18         switch (key) {
19             case KeyCode.VK_ESCAPE:
20                 System.exit( 0 );
21                 break;
22         }
23     }
24
25     private BranchGroup createScene() {
26         Cube cube = new Cube( 3, true );
27
28         BranchGroup bg = new BranchGroup();
29         bg.addChild( cube );
30
31         return( bg );
32     }
33
34     public SceneWithCube() {
35         super( 128L );
36
37         Xith3DEnvironment env = new Xith3DEnvironment( this );
38
39         Canvas3DWrapper canvas = Canvas3DWrapper.createStandalone(
40                 Canvas3DWrapper.Resolution.RES_800X600,
41                 "Box" );
42         this.registerKeyboardAndMouse( canvas );
43
44         env.addCanvas( canvas );
45
46         env.addBranchGraph( createScene() );
47
48         this.begin();
49     }
50
51     public static void main(String args[]){
52         SceneWithCube scene = new SceneWithCube();
53     }
54 }
```

**Figura 9 -** Código de um programa *Xith3D*

Pode-se observar pela Figura 9 que na linha 26 é criado um cubo de tamanho três com preenchimento de cor. A seguir este cubo é adicionado ao *BranchGroup* (nó raiz da cena) e posteriormente (linha 46) é adicionado ao ambiente por meio do método *addBranchGraph().*

O método *onKeyReleased()*, na linha 17, é herdada da classe *RenderLoop* e é responsável pelo gerenciamento das ações tomadas quando uma tecla é liberada.

A linha 25 refere-se ao construtor da superclasse e limita o número máximo de quadros por segundo em 128. Na linha 37 é criado o ambiente do *Xith3d* (*Xith3dEnviroment*), que cria e gerencia os métodos básicos para a execução do um programa *Xith3D*. A seguir, na linha 39 é criada a área onde será desenhada a cena.

Na linha 42 é criado o *listener* para entradas de mouse e teclado. Este método é necessário para que a função *onKeyReleased* funcione.

Pode-se observar na linha 44 que o *canvas* criado anteriormente é adicionado ao ambiente. Finalmente na linha 48 é inicializado o *loop* de renderização por meio do método *begin()*.

A Figura 10 apresenta o objeto gerado na execução do código da Figura 9.

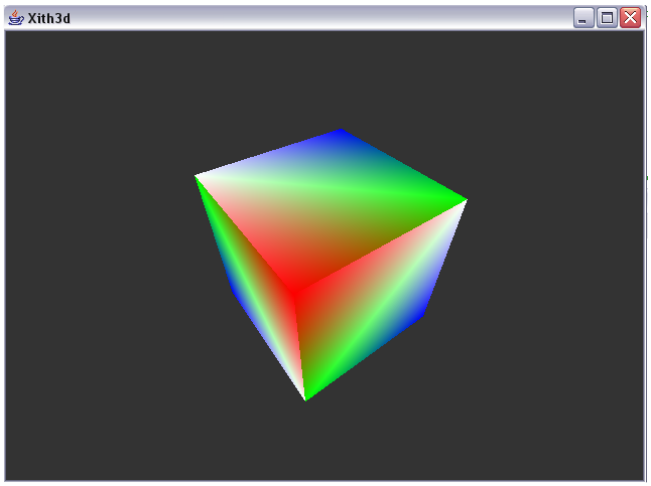


**Figura 10 –** Tela do cubo gerado

## 3.2 *jMonkey*

O *jME* (JME, 2006), ou *jMonkey Engine*, é uma *engine* para grafo de cena de alto desempenho utilizando APIs gráficas. Ele foi construído para suprir a falta de *engines* gráficas de alta qualidade escritos utilizando a tecnologia Java. Usando uma camada de

abstração, esta *engine* permite que qualquer sistema de renderização seja integrado a ele. Atualmente, o LWJGL é suportado, mas há planos para que, num futuro próximo, o JOGL (*Java bindings for OpenGL*) seja suportado. O *jME* possui seu código completamente aberto sobre a licença BSD.

Esta *engine*, assim com o *Xith3D*, também possui uma arquitetura baseada em grafo de cena, no qual, quando um nó não é mais utilizado, ele é descartado do grafo de cena.

Descartar pode significar diversas coisas, mas o mais significativo para a programação gráfica é o chamado *culling* de dados. O *culling* é um processo no qual é feita uma seleção do que será ou não renderizado ou visto pelo espectador (por exemplo, um cubo que será visualizado somente sua parte frontal, não precisa ter sua parte traseira renderizada, por isso é feita a seleção do que será renderizado na tela). Isto permite que cenas complexas sejam renderizadas rapidamente, pois a maior parte das cenas não são visualizadas todas de uma vez.

Os nós fim (também chamados de "folhas") da estrutura de árvore do gráfico de cena são geometrias que serão renderizadas para serem mostradas. Há diversos tipos de geometrias suportadas pelo *jME*, incluindo: caminhos *bezier,* linhas, pontos, modelos (*Milkshape, MD2, ASE, 3DS, COLLADA,* etc), terrenos, nível de detalhamento, e outros. O *jME* também suporta muitos efeitos de alto nível, como: renderização para textura, mapeamento ambiental, reflexos de luzes, tinturas, sistemas de partículas, etc.

### 3.2.1 Exemplo de Programação

Para um melhor entendimento de como programar utilizando o *jME*, é apresentado na Figura 11 um programa básico, explorando as bibliotecas *SimpleGame, Box,* e também o

nó raiz *rootNode*. Este programa de ser gravado com a extensão ".java" (LINDAMOOD, 2005).

```
1  import com.jme.app.SimpleGame;
2  import com.jme.scene.shape.Box;
3  import com.jme.math.Vector3f;
4
5  public class HelloWorld extends SimpleGame{
6   public static void main(String[] args) {
7    HelloWorld app = new HelloWorld(); // Cria uma nova aplicação HelloWorld
8    // Sinal para mostrar a janela de Propriedades
9    app.setDialogBehaviour(SimpleGame.ALWAYS_SHOW_PROPS_DIALOG);
10   app.start(); // Inicia a execução do Programa
11  }
12  protected void simpleInitGame() {
13   // Cria um novo Box (Caixa)
14   Box b = new Box("MinhaCaixa", new Vector3f(0,0,0), new Vector3f(1,1,1));
15   rootNode.attachChild(b); // Adiciona a caixa no grafo de cena (no nó raiz)
16  }
17 }
```

**Figura 11 –** Código fonte do Programa HelloWorld

A classe herdada *SimpleGame*, na linha 5, efetua diversas inicializações necessárias, como a movimentação que o mouse e o teclado produzirão. Ela é ideal para prototipações e testes.

A linha 9 permite que uma tela de preferências seja exibida toda vez que o programa entra em execução. Esta tela define qual as preferências do usuário como, por exemplo, a resolução da tela de exibição. A Figura 12 exibe a tela de preferências.


**Figura 12 –** Tela de Preferências

A função *start()*, localizada na linha 10 do código fonte, é um *loop while*. Primeiro, ele inicializa o sistema *jME*. Então, o *loop* executa duas coisas por iteração: primeiro, ele informa ao programa tudo o que precisa mover, e em segundo, ele renderiza tudo (cria os objetos na tela, para poderem ser visualizados). Basicamente, ele faz o jogo "rodar".

A função *simpleInitGame()* da linha 12, é abstrata na classe *SimpleGame*, então é necessário implementá-la toda vez que a classe *SimpleGame* for herdada. Observando o código, é possível ver que duas coisas acontecem. Primeiro foi criado um cubo na linha 14, e na linha 15, este cubo é adicionada ao nó raiz. O objeto *rootNode* pertence a classe *Node,* criado pela classe *SimpleGame.* Todos os objeto criados são anexados ao *rootNode* ou a um de seus "filhos".

É possível observar que para a caixa ser criada são passados 3 parâmetros: uma *String* e dois objetos *Vector3f*. Todo objeto *Node, Box, Circle, Person* ou qualquer outro objeto no grafo de cena possui um nome. Normalmente os nomes são específicos para cada objeto. O cubo criado no programa *HelloWorld* foi chamada de "Minha Caixa". Os próximos 2 parâmetros especificam os vértices do *Box*. Ele possui o canto inferior esquerdo na origem e o canto superior direito em x = 1, y = 1 e z = 1.

Depois do cubo ter sido criado, é necessário dizer ao programa que este cubo precisa ser renderizado. Este é o motivo pelo qual ele é anexado ao objeto *rootNode*. O grafo de cena estará basicamente como a Figura 13.

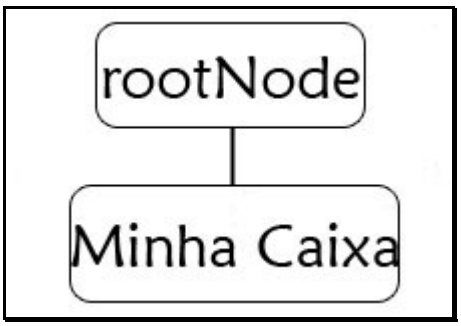


**Figura 13 –** Grafo de Cena do programa *HelloWorld*

O objeto *rootNode* estará no topo e "Minha Caixa" abaixo. Portanto, quando a classe *SimpleGame* tentar desenhar o *rootNode*, a "Minha Caixa" também será desenhado na tela.

## 3.3 Comparando *Xith3D* e *jME*

A Tabela 1 mostra uma comparação entre as *engines Xith3D* e *jME*, mostrando suas diferenças e principais características.

Ambas *engines* suportam os sistemas de som multiplataforma OpenAL (OPENAL, 2006). O *Xith3D* suporta também o sistema JavaSound.

| ***Engine*** | **Bibliotecas de Renderização Suportadas** | **Geometrias Suportadas** | **Modelos para importação** | **Arquivos de Textura** |
|---|---|---|---|---|
| *Xith3D* | • JOGL<br>• LWJGL<br>• JSR231 | *Triangle Array, Quad Array, Triangle Fan, Triangle Strip, Indexed Triangle* e *Array.* | ASE, OBJ, 3DS, MD2, Cal3D, BSP e COLLADA 1.4. | Imagens RGB, RGBA e GRAY para formato PNG, JPG, BMP, etc. |
| *jME* | • Permite que qualquer API seja implementada<br>• LWJGL implementada atualmente | *Box, Cylinder, Disk, Hexagon, Octahedron, PQTorus, Pyramid, Quad, Sphere* e *Torus.* | 3DS, Obj, MD2, MD3, Milkshape, ASE e COLLADA 1.4. | BMP, TGA descompactado, JPG, PNG, GIF. |

**Tabela 1–** Comparação entre as engines *Xith3D* e *jME*

# CAPÍTULO 4 - DESENVOLVIMENTO E IMPLEMENTAÇÃO DO JOGO ESTAÇÃO DE CONTROLE AMBIENTAL SECRETA

Este Capítulo descreve as fases de desenvolvimento e implementação, e os resultados obtidos. Neste capítulo também são fornecidas informações a respeito do jogo, como os comandos de controle do jogo e a descrição do jogo.

## 4.1 Objetivos e Público Alvo

Como mencionado anteriormente, o objetivo deste trabalho é o desenvolvimento de um jogo educacional, utilizando a *engine* de programação de jogos *jME*, para ser empregado como ferramenta de auxílio ao aprendizado de moléculas e substâncias químicas existentes em nosso planeta, presente na matéria de Química do Ensino Fundamental.

O público-alvo para este jogo são os alunos do ensino fundamental que tiverem em sua matriz curricular os tópicos de Substâncias e Moléculas Químicas.

## 4.2 Metodologia

A metodologia aplicada no desenvolvimento deste trabalho compreende as seguintes etapas:

- Estudos sobre *Softwares* Educacionais;
- Estudos sobre Jogos Educacionais e Realidade Virtual;
- Estudos de e*ngines* para desenvolvimento de jogos;

- Desenvolvimento do jogo;

## 4.3 Enredo do Jogo

O jogo se passa no século 21 onde uma Estação de Controle Ambiental Secreta (ECAS), localizada na Ilha de Fernando de Noronha, está prestes a causar uma desordem na natureza terrestre. Esta estação de controle ambiental é responsável por manter a ordem entre as moléculas existentes em nosso planeta, respeitando suas equações originais como, por exemplo, manter a molécula da água como sendo $H_2O$.

O Computador Central (CC), responsável pela maior parte das operações presentes na estação, entrou em pane devido a uma falha no gerador de energia que não foi acionado corretamente após uma queda brusca de energia.

Devido a esta pane, a E.C.A.S. entrou em alerta máximo e diversos setores presentes na E.C.A.S. precisam ser restabelecidos resolvendo alguns desafios e enigmas relacionados a moléculas químicas. Estes enigmas estão localizados dentro dos setores que, após serem solucionados, permitem a entrada em novos setores. Após estes setores serem restabelecidos, é necessário acessar a sala de controle geral, onde o computador central está instalado. Ao acessá-lo, as senhas obtidas nos setores concluídos deverão ser digitadas em cada terminal do computador central para que a E.C.A.S. retorne à sua atividade normal novamente. É necessário restabelecer a ordem nos diversos setores e reativar o computador central dentro de um tempo pré-determinado.

### 4.3.1 Personagens

O personagem principal, Dagoberto Jones, é o técnico responsável pelo restabelecimento de situações críticas. Ele é quem irá solucionar os desafios presentes em cada setor e os normalizar. Dagoberto também deverá reativar a sala do Computador Central inserindo as senhas corretas, obtidas nos outros setores, em cada terminal do computador central presente na sala.

Não haverão outros personagens, pois a estação foi evacuada antes da chegada de Dagoberto Jones, por questões de segurança.

### 4.3.2 Setores

Os setores são laboratórios separados localizados dentro da E.C.A.S. que conterão as tarefas a serem solucionadas. No começo do jogo, a maior parte dos setores estarão bloqueados devido ao sistema de alerta acionado pela pane no computador central. Após a solução de determinadas tarefas, setores que antes estavam bloqueados serão abertos.

Ao completar os desafios de cada setor, os novos setores que serão abertos poderão apresentar enigmas com um nível de dificuldade maior, sendo necessário um maior conhecimento a respeito de Química por parte do jogador.

Os enigmas que estarão presentes nas diversas salas do jogo ECAS têm o objetivo de ajudar o aluno a obter um melhor entendimento a respeito das diversas moléculas e substâncias existentes em nosso planeta, demonstrando o seu funcionamento, suas características e suas fórmulas químicas.

### 4.3.3 Finalização

O jogo estará completo após todos os setores tiverem seus enigmas solucionados e todas as senhas necessárias estiverem digitadas no computador central, dentro do tempo estimado. Caso contrário, será exibido um *Game Over* (Fim de Jogo), fazendo com que o jogador comece o jogo novamente.

### 4.3.4 Jogabilidade

O usuário do jogo educativo ECAS irá movimentar e efetuar as ações do personagem principal utilizando o mouse e o teclado. O teclado servirá para movimentá-lo para frente e para trás (utilizando as teclas “W” e “S” respectivamente), e para acessar o menu do jogo (utilizando a tecla “ESC”). Com o mouse será possível mover o personagem para a esquerda e para a direita e, utilizando os botões, o usuário irá efetuar as ações do jogo como abrir as salas, pressionar botões e arrastar e soltar itens.

## 4.4 Ambiente de Desenvolvimento

Para o desenvolvimento e testes no jogo foram utilizados dois computadores com configurações distintas de hardware e software, um composto por processador Atlhon XP 64bits de 3.0 GHz, Placa de vídeo de 256 MB, 512 MB de memória RAM, sistema operacional Microsoft Windows XP e o outro composto por processador Pentium 4 de 2.8

GHz, Placa de vídeo de 64 MB, 512 MB de memória RAM, também com sistema operacional Microsoft Windows XP.

A utilização do primeiro se refere a um computador pessoal, onde foram instaladas as ferramentas gratuitas *jMonkey Engine* e Netbeans (NETBEANS, 2006). O segundo computador mencionado foi em razão de pertencer à Univem e possuir uma cópia licenciada da ferramenta 3D Studio Max 6 (AUTODESK, 2005), utilizado para modelagem do cenário.

## 4.5 Interface do Jogo

Ao iniciar o jogo, o sistema apresenta uma tela de configurações iniciais (Figura 14), no qual o usuário pode definir suas configurações desejadas. As configurações que esta tela apresenta são: Resolução da Tela (por exemplo, 800x600), Qualidade da Cor (por exemplo, 32 bits), a Freqüência do Monitor (por exemplo, 72 Hz) e se deseja ou não executar o jogo em Tela cheia. Após a escolha destas definições, o usuário deverá clicar em OK para iniciar o jogo.

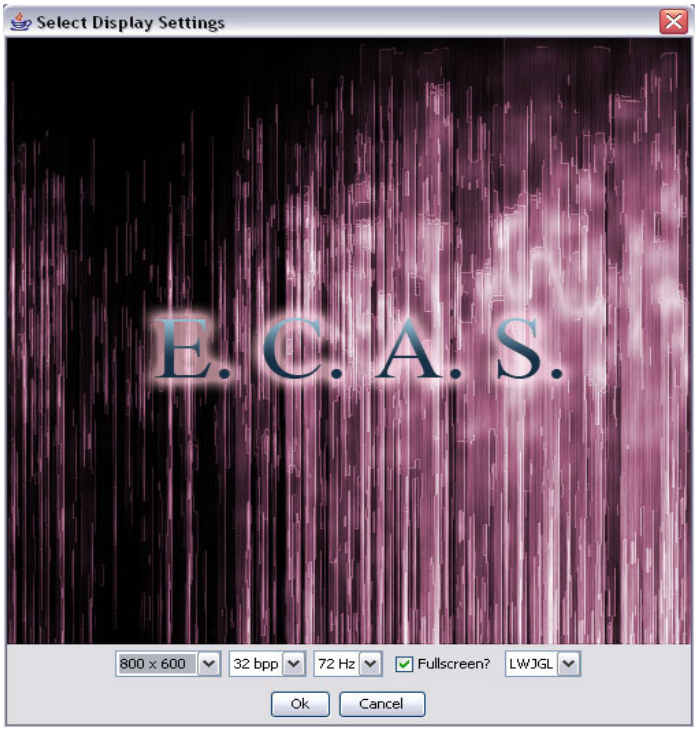


**Figura 14 -** Janela de configuração inicial

A Figura 15 exibe a tela de Menu Inicial do Jogo. Nesta tela o usuário terá a opção de pressionar a tecla ESC para sair do jogo, a tecla F1 para obter as instruções de jogo (Figura 16) e a tecla F2 para iniciar o jogo.


**Figura 15 -** Menu Inicial do Jogo


**Figura 16 -** Tela de Instruções do E.C.A.S.

Após o usuário pressionar F2 para iniciar o jogo, o sistema começará a carregar os modelos 3D (isto pode levar algum tempo, em computadores mais lentos), e também iniciar o contador do tempo. A Figura 17 mostra a primeira cena visível pelo usuário ao entrar na Estação de Controle Ambiental Secreta.


**Figura 17 -** Primeira Cena do E.C.A.S.

Após entrar no jogo, o usuário deverá acessar o laboratório de entrada. Neste laboratório, ele deverá completar o primeiro desafio, que é montar uma molécula de água na lousa (Figura 18). Para isso ele deverá obter as bolinhas corretas que estão presentes em cima das mesas, e inseri-las nos espaços corretos na lousa.

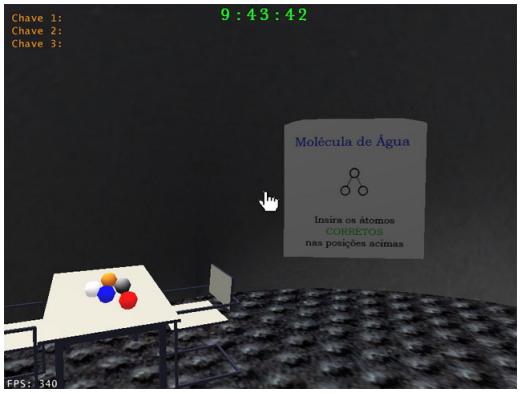


**Figura 18** - Primeiro Desafio

Feito isto, o desafio estará completo, fechando a porta de entrada e abrindo a porta dos fundos para que o usuário tenha acesso aos outros setores da E.C.A.S.

Ao usuário acessar a parte externa da E.C.A.S., ele deverá acessar o setor oeste para completar o 2 desafio, que é levar para um computador *iMac* (Figura 19) a seqüência da fórmula do “Combustível do Futuro” ($H_2$). Para isso, o usuário deverá voltar ao laboratório de entrada e trazer para a tela do computador os cubos corretos, que representam os elementos da fórmula.


**Figura 19 -** Segundo Desafio

Após a conclusão do segundo desafio, o setor leste se abrirá e o usuário deverá prosseguir para este setor para a conclusão do terceiro desafio. Neste desafio serão projetadas diversas afirmações na tela de projeção e ele deverá puxar a alavanca vermelha para afirmações falsas e a alavanca verde para afirmações verdadeiras. Depois de puxar uma determinada alavanca, será projetado se ele acertou ou se errou. Caso tenha errado, o desafio três será reiniciado e, caso tenha acertado, a próxima afirmação será exibida. O desafio será concluído com a projeção da tela de reativação do setor leste (Figura 20).


**Figura 20 -** Desafio três completado

Quando cada desafio é completado, o usuário ganha uma chave que está localizada no canto superior esquerdo da tela. Estas chaves serão necessárias para a reativação do computador central. O computador central se encontra no setor do meio, que será aberto após a conclusão do desafio 3.

Este computador central é composto por três terminais (Figura 21), nos quais deverão ser inseridas as chaves corretas. Para a inserção das chaves o usuário deverá clicar em qualquer um dos terminais e digitar a chave que este terminal solicita.


**Figura 21 -** Terminais do computador central

Assim que o usuário inserir as três chaves corretamente, o jogo estará finalizado. Neste instante é exibida uma tela parabenizando o usuário por ter conseguido finalizar o jogo (Figura 22).


**Figura 22 -** Jogo Finalizado

O usuário perde quando não consegue acessar o setor do computador central e digitar as chaves dentro do tempo estipulado. Será exibida uma tela dizendo que o usuário não conseguiu finalizar o jogo a tempo (Figura 23).


**Figura 23 -** Tempo do Jogo Excedido

## 4.6 Implementação

Este capítulo aborda as partes mais importantes da implementação do sistema e as principais funções utilizadas para a criação do jogo E.C.A.S. utilizando o *jME.*

Para a implementação do jogo foi utilizada a ferramenta Netbeans. Com o Netbeans é possível desenvolver softwares utilizando a linguagem de programação Java.

Os modelos e textos que estarão presentes graficamente no jogo são criados e anexados a Nós que representarão o grafo de Cena do jogo. As texturas podem ser anexadas aos modelos e aos textos.

### 4.6.1 Inicialização e Display

Assim como descrito anteriormente, o *jME* permite que o usuário acesse uma janela de propriedades da tela, antes do início do jogo. Essa janela de propriedades é definida pelo programador para que ela seja exibida ou não. Caso a opção seja de ser exibida, há ainda a possibilidade de que esta janela de propriedades seja mostrada somente na primeira vez em que o usuário entra no jogo ou todas as vezes antes de entrar no jogo. As propriedades que definidas pelo usuário podem afetar a qualidade e o desempenho do jogo.

Após a seleção das propriedades, a *engine* obtêm essas informações e inicializa uma janela para a exibição dos gráficos do jogo. Essa janela é criada com a chamada do método abstrato *initSystem(),*herdado da classe *BaseGame*. Este método deve ser sobrescrito na classe Principal.

A Figura 24 exibe as linhas de programação necessárias para a criação da Janela do Jogo.

```
92      protected final void initSystem() {
93          try {
94
95              display = DisplaySystem.getDisplaySystem(properties.getRenderer());
96
97              display.createWindow(
98                      properties.getWidth(),
99                      properties.getHeight(),
100                     properties.getDepth(),
101                     properties.getFreq(),
102                     properties.getFullscreen());
103
104         }
105         catch (JmeException e) {
106
107             e.printStackTrace();
108             System.exit(1);
109         }
110
111
112         timer = Timer.getTimer();
113
114     }
```

**Figura 24 –** Linhas de Programação para a criação do Display

A variável *display*, que é uma instância da classe *DisplaySystem* do *jME* e é herdada pela classe pai *BaseGame,* é inicializada através da chamada do método g*etDisplaySystem(properties.getRenderer())*. Em seguida, são passados como parâmetros, para o método *createWindow()*, as informações fornecidas pelo jogador no instante em que ele clicou em *OK* na janela de propriedades (essas informações são obtidas através da variável *properties*) para que seja criada a janela, onde serão renderizados todos os objetos e modelos que estão presentes no jogo. Para este método são passados os seguintes parâmetros, na seguinte ordem:

- Largura da Janela;
- Altura da Janela;
- Qualidade de Cor;
- Freqüência do Monitor;
- Tela Cheia;

Outro aspecto importante na inicialização do sistema do jogo é a definição de um *Timer*, que irá controlar os *Frames Per Second* e também o cronômetro do jogo. Este *Timer* também será responsável pelas animações presentes no jogo, como por exemplo, a movimentação dos raios da cerca elétrica. Sua criação é definida com a inicialização da variável *timer* através da chamada do método *Timer.getTimer()*. A variável *timer* é uma variável que instancia a classe *Timer* da biblioteca do *jME*. Esta variável é definida dentro da própria classe Principal do E.C.A.S.

Após a obtenção das propriedades da janela, sua criação e a definição de um *timer,* é necessário configurar uma câmera para a exibição de modelos tridimensionais ou até mesmo bidimensionais. É também na configuração da câmera que será definida a técnica de *Frustrum Culling*, que já vêm implementada nas classes do *jME*. *Frustum Culling* é uma técnica utilizada para que tudo que esteja fora do alcance de uma "pirâmide de visualização", definida pelo programador, não seja renderizada. A Figura 25 exemplifica uma visualização *Frustum* bidimensional.

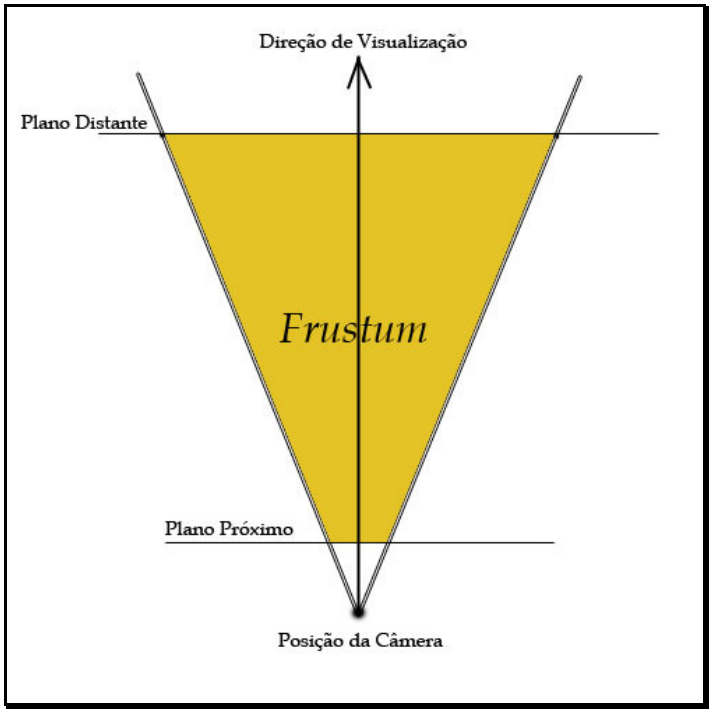


**Figura 25 -** Visualização *Frustum* bidimensional

### 4.6.2 Estados do Jogo

O E.C.A.S. é controlado através de estados pela classe *GameState* do *jME*. A classe *GameState* é utilizada para encapsular um certo estado do jogo como, por exemplo, o estado "em jogo", "menu principal" ou "instruções". O jogo possui três classes além da classe Principal (responsável pela inicialização do Display e do Timer, e também pela definição do estado inicial do jogo):

- A classe *InGameState* que é responsável pela parte "jogável";
- A classe *MenuState* que é a classe responsável pela exibição do Menu Principal do jogo.
- E a classe *InstrState* que é a classe responsável pela exibição de uma tela com as Instruções de como jogar e Objetivos a serem completados.

Tanto a classe *InGameState* quanto as classes *MenuState* e *InstrState* herdam a classe *StantardGameState* da biblioteca do *jME*. Esta classe é responsável pela criação de um estado de jogo que inicializa um nó *rootNode*, uma câmera e um *ZBufferState*. O *ZBufferState* é responsável pelo controle do *buffer* de profundidade do gráfico. Ele não permite que objetos que estejam mais longes da câmera sobreponham objetos mais próximos da câmera. Este controle é feito pela profundidade no eixo Z, ou distância entre o *pixel* fonte e a câmera.

Para o controle de qual estado o jogo deve assumir (Menu, Em Jogo ou Instruções) foi criada a classe *MenuHandler* que é responsável por "escutar" o teclado quando se está no Menu do Jogo, sabendo que o estado Menu será o primeiro estado a ser iniciado pelo jogo. Caso ocorra alguma ação com alguma das teclas pré-definidas, esta classe será responsável pela transição entre o estado Jogo, Menu e Instruções. A Figura 18 exibe o código fonte da classe *MenuHandler*.

```
public class MenuHandler extends InputHandler {
    private GameState meuState;

    //Quad do Menu
    private Quad menu;

    public MenuHandler( GameState myState, Quad menu ) {
        setKeyBindings();
        this.meuState = myState;

        this.menu = menu;
    }

   private void setKeyBindings() {
        KeyBindingManager.getKeyBindingManager().set("saida", KeyInput.KEY_ESCAPE);
        addAction( new ExitAction(), "saida", false );

        KeyBindingManager.getKeyBindingManager().set("entrar", KeyInput.KEY_F2);
        addAction( new EnterAction(), "entrar", false );

        KeyBindingManager.getKeyBindingManager().set("instrucoes", KeyInput.KEY_F1);
        addAction( new InstrAction(), "instrucoes", false );
    }

    private static class ExitAction extends InputAction {
        public void performAction( InputActionEvent evt ) {
            Principal.exit();
        }
    }

    private class EnterAction extends InputAction {
        public void performAction( InputActionEvent evt ) {
            if(GameStateManager.getInstance().getChild("InGame") != null)
            {
                //Volta para o Menu State
                GameStateManager.getInstance().activateChildNamed("InGame");
                //Desativa o Menu (por enquanto)
                meuState.setActive(false);
            }
            else
            {
                //Cria o estado de "Em Jogo"
                GameState ingame = new InGame("InGame");
                //Anexa-o ao GameStateManager
                GameStateManager.getInstance().attachChild(ingame);
                //Seta-o como ativo
                ingame.setActive(true);
                //Desativa o Estado de Menu
                meuState.setActive(false);
            }

        }
    }

    private class InstrAction extends InputAction {
        public void performAction( InputActionEvent evt ) {
            //Volta para o Menu State
            GameStateManager.getInstance().activateChildNamed("Instrucoes");
            //Desativa o Menu (por enquanto)
            meuState.setActive(false);
        }
    }
}
```

**Figura 26 -** Código fonte da classe *MenuHandler*

Nesta classe existem cinco métodos, o primeiro método é o método construtor *MenuHandler()* que chama o método *setKeyBindings()* e define um estado inicial para o jogo. O método *setKeyBindings()* define as teclas que serão responsáveis pela chamada das ações *ExitAction, EnterAction* e *InstrAction*. As classes internas *ExitAction, EnterAction* e InstrAction são classes que herdam a classe *InputAction*, para controle de ações efetuadas no teclado. A classe *ExitAction* faz com que o jogo seja finalizado ao clicar na tecla ESC, a classe *EnterAction* cria um novo estado "*InGame"* (Em Jogo), case ele já não tenha sido criado, quando a tecla F2 é pressionada, setando-o como ativado e desativando o estado Menu, e a classe *InstrAction* seta o estado "Instruções" como o estado atual e desativa o Menu.

Após a definição destes métodos, o sistema estará pronto para a criação dos objetos que irão compor o cenário do jogo no estado "InGame". Estes objetos e geometrias são definidos através do Grafo de Cena da arquitetura do *jME*.

### 4.6.3 Grafo de Cena

Como mencionado anteriormente, o *jME* possui sua arquitetura baseada em grafo de cena. O grafo de cena do *jME* contém um nó raiz, chamado de *"rootNode"* e é a partir deste nó que são definidos os filhos que contem as geometrias presentes no jogo.

Foi definido também um nó separado do *rootNode* que contém as informações de Texto presentes na tela. Este nó foi definido como *fpsNode*, pois além de conter as informações de Texto do jogo (como o Timer, por exemplo), também contém as informações dos Quadros por Segundo (veja a seção 4.7).

A Figura 27 apresenta o grafo de cena do Jogo E.C.A.S.

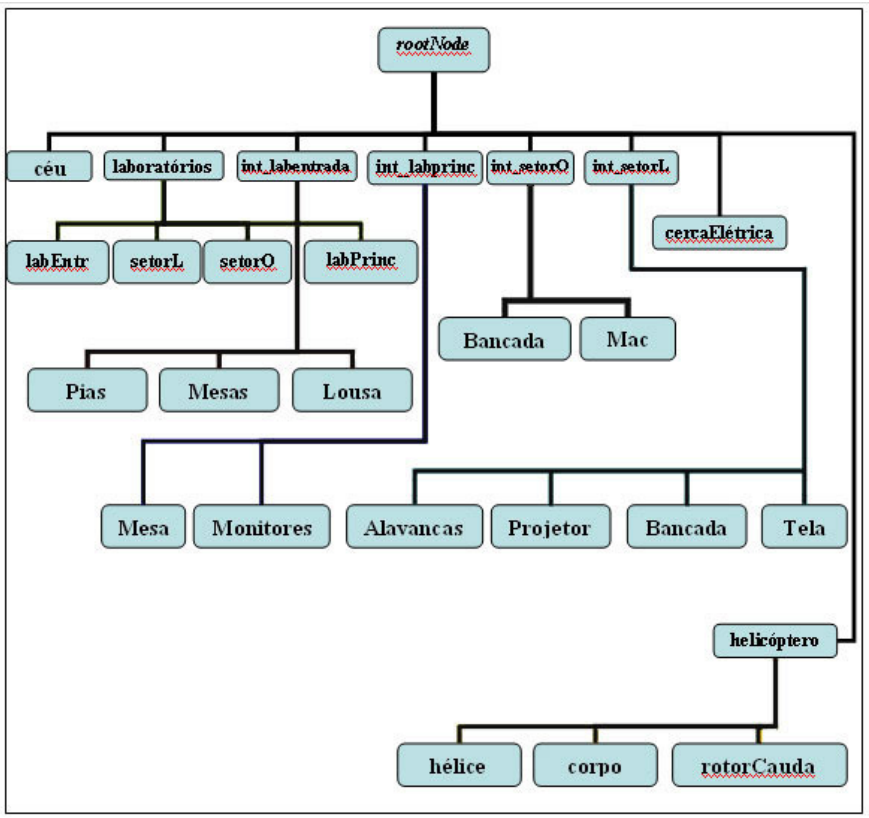


**Figura 27 -**Grafo de Cena do jogo E.C.A.S.

No topo (ou raiz) da árvore da Figura 27 está definido o nó raiz, chamado rootNode. Este nó já é definido por padrão pelo *jME*, e é a partir deste nó que partem os nós filhos que contém as geometrias necessárias para visualização do cenário do jogo.

Para a criação do jogo E.C.A.S., somente os nós terminais é que foram definidos para conter as geometrias do cenário. Como por exemplo, o nó int_labentrada é um nó que foi instanciado somente para ser "pai" de outros nós, sendo que este nó não possui nenhuma geometria filha diretamente, ele apenas contém os nós que estarão no interior do Laboratório de Entrada, ou seja, o nó Pias que contém as geometrias das pias do Laboratório, o nó Mesas, que contém as geometrias das duas mesas que compõem este laboratório e o nó Lousa, que contém a geometria da Lousa, o qual conterá uma textura para informar ao jogador o que deverá ser feito no Laboratório de Entrada. Para a criação de um nó no *jME* basta instanciar uma variável da classe Node.

Com essa definição é possível que, por questões de performance e rapidez, caso as geometrias Pias, Mesas e Lousa não necessitem mais serem visualizadas pelo jogador, basta desanexar todos os filhos do nó int_labentrada, obtendo assim uma melhor performance, pois estes objetos não serão mais renderizados.

A Figura 19 mostra o grafo de cena do nó *fpsNode*. Este nó independe do nó rootNode.

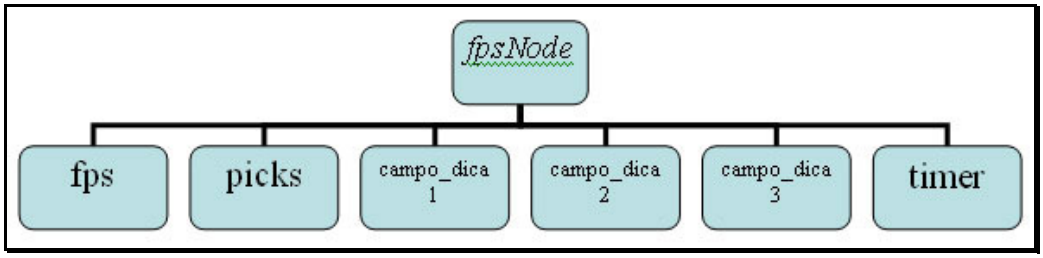


**Figura 28 –** Grafo de Cena do nó fpsNode

O grafo de cena da Figura 28 apresenta os nós que conterão textos que serão mostrados diretamente na tela. O nó fps contém o texto localizado no canto inferior esquerdo da tela mostrando os quadros por segundo atual do jogo. O nó picks contém o Texto que só será exibido quando algo for obtido pelo clique do mouse do jogador, como por exemplo, a representação do nome de uma molécula obtida no primeiro Laboratório.

Os nós campo_dica1, campo_dica2 e campo_dica3 são textos que serão atualizados e exibidos no canto superior esquerdo da tela após a conclusão dos enigmas contidos em cada um dos laboratórios. Estes textos conterão as dicas para inserção do Laboratório do Computador Central.

E por fim, o nó timer contém o texto que será exibido na tela com o tempo atual restante do jogador.

### 4.6.4 *Picking*

A obtenção e ação nos objetos presentes no cenário do jogo E.C.A.S. são controlados através de *Picking*. O *Picking* funciona da seguinte maneira: primeiramente o sistema detecta se houve alguma ação com o botão direito do *mouse*; Em seguida, é criado um "segmento de reta" (*Ray*) que tem como origem a posição da câmera e destino, o infinito, enquanto o botão do mouse estiver pressionado. A direção deste segmento de reta segue a mesma direção da câmera, no momento em que o botão do *mouse* foi pressionado.

Todos as caixas de contorno (*BoundingBox*) dos objetos em que este segmento de reta intercepta são registrados em uma variável do tipo *PickResults.* A reta só interceptará os objetos que estão dentro de um nó que é passado como parâmetro para a variável de *PickResults*.

Em seguida é feita uma verificação para determinar a distância da câmera com o objeto em que a reta intercepta. Caso o objeto for o objeto desejado e a distância for a distância mínima permitida para a ocorrência da ação, a ação é então efetuada.

Há cinco tipos de ações, ou classes, para *Picking* definidas no jogo E.C.A.S. São elas:

- *MousePick*;
- *BolinhasPick*;
- *FormulaPick*;
- *AlavancasPick*;
- *ChavesPick*;

A ação efetuada pela classe *MousePick* é para a abertura de portas pelo usuário. São passados como parâmetros para esta classe os nós lab_entrada e setorO do grafo de cena, pois estes são os únicos modelos em quem o usuário pode efetuar a ação de abrir as portas. As

portas dos outros laboratório são abertas automaticamente após a conclusão de determinados desafios.

As classes *BolinhasPick* e *FormulaPick* possuem praticamente a mesma funcionalidade, a única diferença é que a classe *BolinhasPick* é responsável pelas ações do desafio um (obter as bolinhas e colocá-las na lousa) e a classe *FormulaPick* é responsável pelas ações do desafio dois (obter os cubos dos elementos da fórmula H2 para inseri-los na tela do monitor do setor oeste). A diferença dessas classes de *Pick* para com as outras é que toda vez que uma bolinha ou um cubo é obtido, o nome desta bolinha ou cubo, representando o nome do elemento, é anexado ao nó *picks* do nó *fpsNode* (Figura 29).

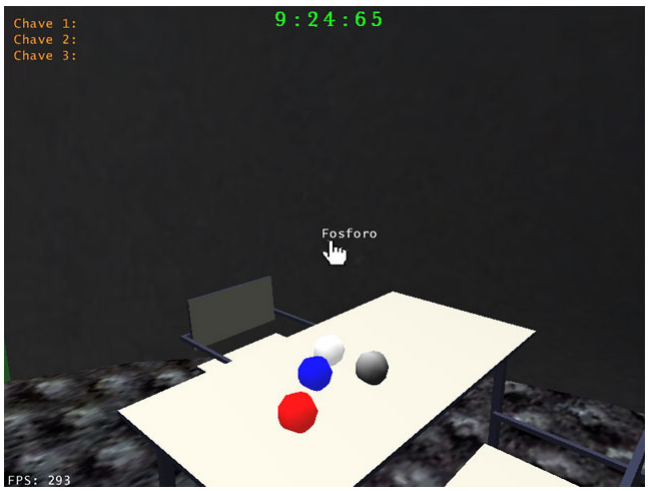


**Figura 29 -** Obtenção de uma bolinha de Fósforo

Assim que a bolinha é obtida, ela também é retirada da renderização da cena através da chamada do método *setCullMode*(*Spatial.CULL_ALWAYS*). O parâmetro *Spatial.CULL_ALWAYS* diz que o objeto que chama o método *setCullMode* não deverá ser renderizado pela *engine.*

Quando a bolinha correta for colocada no campo correto da lousa, o nome desta bolinha é desanexado do nó *picks* do *fpsNode* e sua posição passa a ser o espaço correto na

lousa e ela volta a ser renderizada, passando agora o parâmetro *Spatial.CULL_NEVER* para o método *setCullMode* da bolinha.

A classe *AlavancaPick* têm como funcionalidade ativar o desafio três (clicando na tela de projeção do setor leste) e ativar as animações das alavancas selecionadas.

E a classe *ChavesPick* aciona os campos para digitação das chaves de cada terminal, assim que o usuário clicar em algum dos terminais.

## 4.7 Avaliação

Os Quadros por Segundo (*Frames Per Second*), ou FPS, se referem ao número de quadros (imagens) que são desenhados e criados a cada segundo (Sendo que este não é sempre o número de quadros que é exibido na tela, bem como a placa de vídeo do usuário pode ser capaz de criar muito mais quadros que pode ser exibido em um segundo). Um FPS mínimo geralmente aceitável para um jogo é de 30 quadros por segundo, ou 30 FPS. O número de Quadros por Segundo é inteiramente dependente na complexidade dos gráficos, as capacidades do Hardware da máquina do jogador e da qualidade subjacente da *engine*. O *jME* foi desenvolvido para se obter altas performances e dispor ao usuário o maior número possível FPS para seus jogos.

### 4.7.1 Testes de FPS

Os testes foram efetuados nas máquinas citadas no Ambiente de Desenvolvimento.

Em testes efetuados no computador Atlhon XP 64bits, o E.C.A.S. executou com uma resolução variando de 280 a 400 FPS.

No Pentium IV, o E.C.A.S. executou com uma resolução variando de 88 a 112 FPS.

Ambos resultados demonstraram um FPS de alta qualidade, produzindo alta performance na execução do jogo, sem travamentos no controle do personagem e do cenário.

# CAPÍTULO 5 - DESENVOLVIMENTO E IMPLEMENTAÇÃO DO JOGO DOS ELEMENTOS

Este Capítulo descreve as fases de desenvolvimento e implementação do jogo educativo denominado “Jogo dos Elementos”, discutindo os problemas encontrados, bem como os resultados obtidos.

## 5.1 Objetivos e Público-alvo

Como mencionado o objetivo deste trabalho é o desenvolvimento de um jogo educacional utilizando *Xith3D*, para ser empregado como ferramenta de auxílio ao ensino de elementos químicos, matéria obrigatória no aprendizado de ciências no ensino fundamental.

O público-alvo desta ferramenta são os alunos do ensino fundamental que tiverem a matéria de elementos químicos em sua matriz curricular.

## 5.2 Metodologia

A metodologia aplicada no desenvolvimento deste trabalho compreende as seguintes etapas:

- Estudos sobre *Softwares* Educacionais;
- Estudos sobre Jogos Educacionais;
- Estudos de *engines* para desenvolvimento de jogos;
- Desenvolvimento do jogo;

Primeiramente foi realizado um estudo sobre os *Softwares* Educacionais, conforme discutido no Capítulo 2, citando suas características e definições.

Em seguida foi realizado um estudo específico sobre Jogos Educacionais, dentro do contexto de Software Educacional, citando os tipos de jogos educacionais existentes e também exemplos de Jogos Educacionais conforme abordado no Capítulo 2.

Na seqüência foi realizado um estudo e um levantamento dos motores específicos para a criação e desenvolvimento de jogos (Capítulo 4).

E por fim, foi desenvolvido o Jogo Educacional, denominado "Jogo dos Elementos" utilizando a *engine* para desenvolvimento de jogos *Xith3D*.

## 5.3 Enredo do Jogo

O jogo se ambienta em um laboratório de química, onde o jogador é um aluno que está fazendo um desafio com seu professor. Seu objetivo é entregar ao professor uma série de dez frascos contendo elementos químicos requisitados por este em um dado intervalo de tempo. Os elementos são requisitados por meio de dicas que serão suas propriedades, como ponto de fusão, ponto de ebulição e massa atômica além de suas principais aplicações na indústria e comércio.

O jogador deverá se dirigir ao professor para começar o jogo. Em seguida o professor dará a primeira dica sobre o elemento requisitado e começará a contagem regressiva do tempo. A cada intervalo de tempo, este receberá uma nova dica, até que receba o número máximo de dicas, acabe o tempo, ou encontre o elemento correto.

Depois de encontrado o frasco do elemento o aluno o trará ao professor, a fim de checar se é o elemento correto ou não. Caso tenha acertado, é enviada uma mensagem

indicando o acerto e a pista do próximo elemento será exibida. Se o aluno trouxer o frasco incorreto receberá uma mensagem indicando o erro e continuará sua busca até que o tempo seja zero ou até que entregue ao professor o frasco correto.

Haverá vários frascos de diferentes elementos dispersos pelo ambiente, incluindo frascos que não serão requisitados pelo professor. Os frascos podem estar sobre bancadas, bem como dentro de armários.

Para auxiliar o jogador, o laboratório conterá uma tabela periódica que poderá ser consultada quando necessário durante o jogo.

A pontuação do jogador é dada de acordo com o tempo que este leva para encontrar cada frasco, ou seja, quanto menor o tempo que usar maior será sua pontuação. Após o término do jogo o jogador poderá ver sua pontuação final.

## 5.4 Jogabilidade

O usuário do jogo irá movimentar e efetuar as ações do personagem principal utilizando o mouse e o teclado. O teclado serve para movimentá-lo para frente e para trás (utilizando as teclas “W” e “S”, respectivamente), para a esquerda e direita (utilizando as teclas “A” e “D”, respectivamente) e para agachar (utilizando a tecla “CTRL”). A movimentação também pode ser executada pelas setas do teclado.

A movimentação do mouse rotaciona a câmera. Utilizando o botão esquerdo do mouse, o usuário irá efetuar as ações do jogo como pegar ou entregar um frasco e abrir ou fechar uma porta.

## 5.5 Ambiente de desenvolvimento

Para o desenvolvimento e testes no jogo foram utilizados dois computadores com configurações distintas de hardware e *software*, um composto por processador Pentium 4 de 2.4 GHz, Placa de vídeo de 128 MB, 512 MB de memória RAM, sistema operacional Microsoft Windows XP e o outro composto por processador Pentium 4 de 2.8 GHz, Placa de vídeo de 64 MB, 512 MB de memória RAM, também sistema operacional Microsoft Windows XP.

A utilização do primeiro se refere a um computador pessoal, onde foram instaladas as ferramentas gratuitas *Xith3D* e Netbeans. O segundo computador mencionado foi em razão de pertencer ao Univem e possuir uma cópia licenciada da ferramenta 3D Studio Max 6, utilizado para modelagem do cenário.

## 5.6 Grafo de Cena e Interface

Como mencionado, o *Xith3D* possui uma arquitetura baseada em um grafo, no qual cada nó filho só pode ter um nó pai. Na raiz desse grafo encontra-se o ambiente *Xith3D*. É a partir do ambiente que serão adicionados os nós filhos. Um nó é criado a partir da instanciação de uma variável da classe *Node* ou de outra classe que herda a classe *Node*.

No jogo, foram adicionados três nós à raiz, como é observado na Figura 30. O nó Tela de Carregamento é responsável por exibir na tela a interface correspondente ao carregamento do jogo.

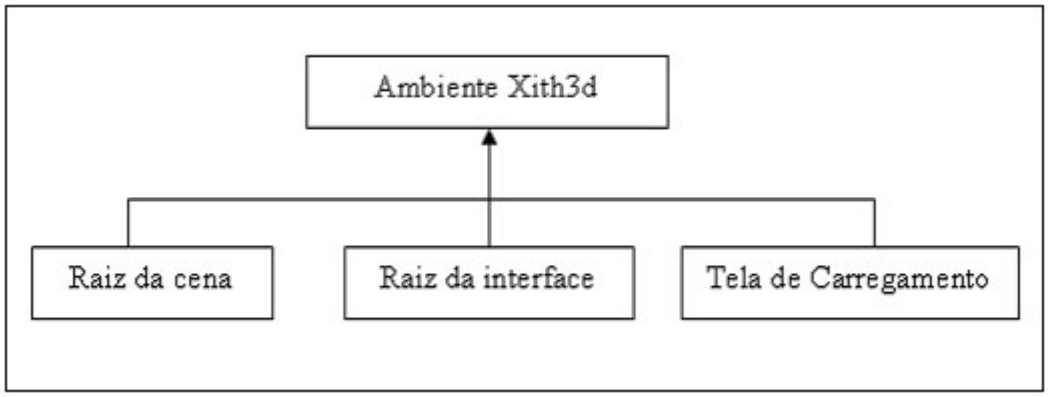


**Figura 30 –** Grafo dos nós primários

O nó Raiz de Interface, Figura 31, contém todos os nós correspondentes à interface do jogo após este ser iniciado.

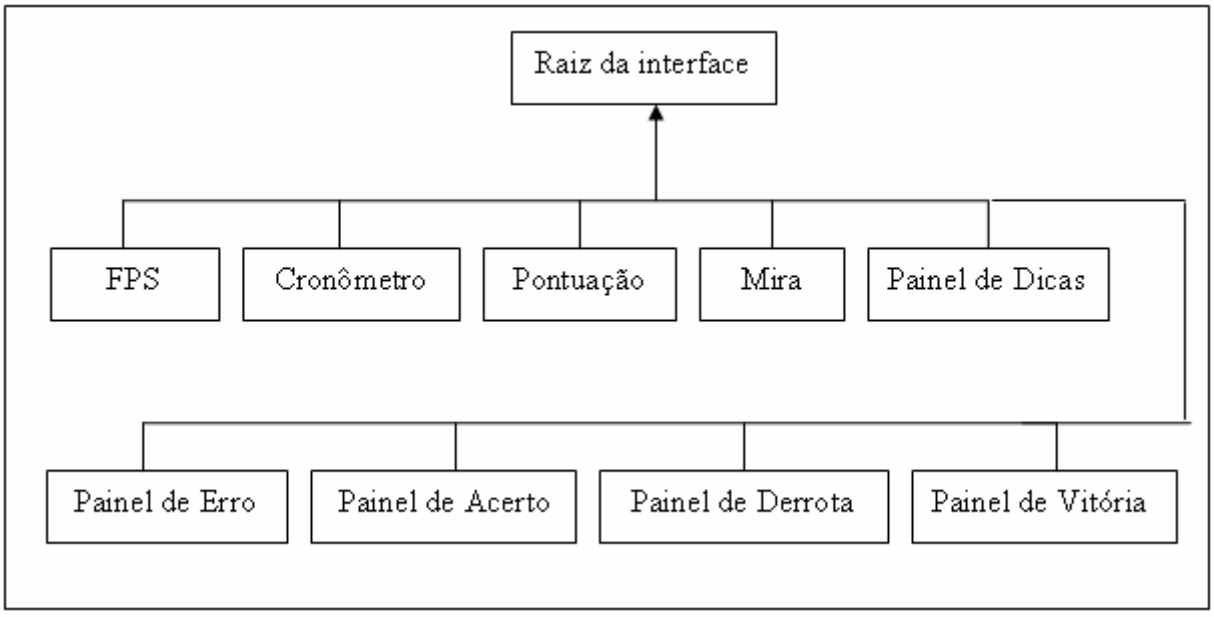


**Figura 31 –** Grafo da Interface

Os cinco primeiros nós são de interfaces exibidos constantemente na tela. O nó FPS indica o número de quadros por segundo do jogo. O nó cronômetro indica quanto tempo o jogador tem para entregar o elemento requerido. O nó Pontuação mostra a pontuação obtida pelo jogador no jogo. O nó Mira mostra na tela a uma imagem que serve como “mão” para o

jogador, para que este saiba em que ponto está clicando. O nó Painel de Dicas mostra as dicas do elemento corrente no jogo, que o jogador deve localizar e trazer ao professor.

Os nós Painel de Erro e Painel de Acerto são inseridos e, após alguns segundos, são removidos da cena. Esses nós indicam quando o jogador trouxe o frasco incorreto e correto respectivamente.

Por fim, os nós Painel de Derrota e Painel de Vitória são inseridos na cena quando o jogador conclui o jogo. O primeiro nó é inserido quando o cronômetro zera, e o segundo quando o jogador completa o jogo no tempo estabelecido.

Ao nó Raiz de Cena (Figura 32) foram adicionados os nós correspondentes à cena.

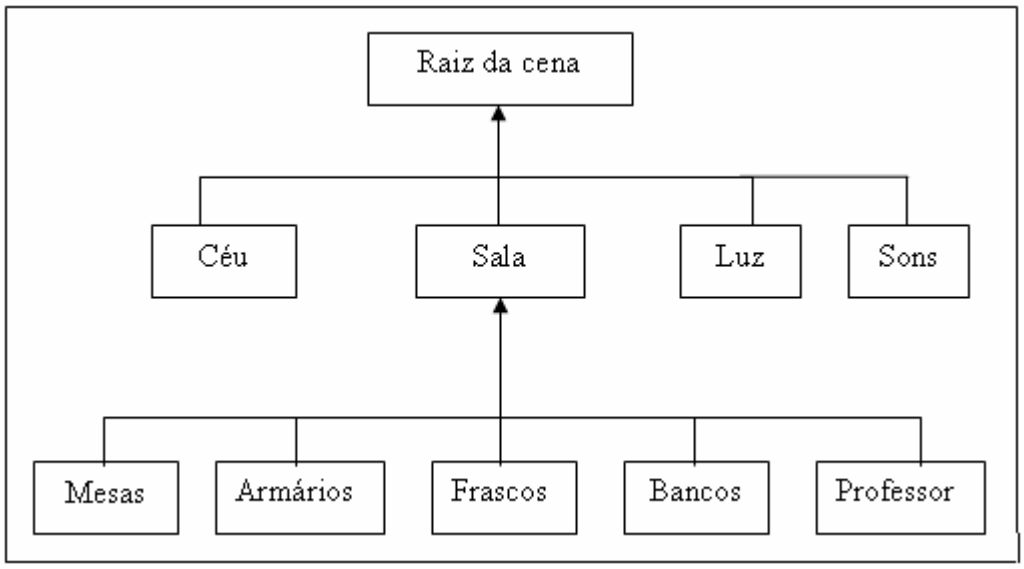


**Figura 32 -** Grafo de Cena

Os nós Céu e Luz são adicionados por meio do *Xith3D*. Ao nó Sons foram adicionados os sons utilizados no jogo, como som de passo e relógio. O nó Sala e seus filhos são lidos através do arquivo de modelo e são posteriormente adicionados à cena. O modelo escolhido para ser utilizado no jogo foi o modelo ASE (*3DS Max ASCII*), pois o *Xith3D* constrói o grafo de cena a partir da leitura do arquivo deste tipo de modelo, permitindo que

objetos contidos no modelo possam ser retirados e adicionados à cena, o que ocorre com os nós filhos do nó Frascos. Além disso, permite que objetos possam ser animados. No jogo isto é utilizado para construir a animação das portas dos armários, contidos no nó Armários.

## 5.7 Interface do Jogo

Ao executar o jogo, primeiramente será apresentada a tela inicial do jogo, no qual o jogador pode escolher a resolução do jogo e se deseja executá-lo em tela cheia ou não. Após escolher entre as opções o jogador deve selecionar a opção OK, como pode ser visto na Figura 33.

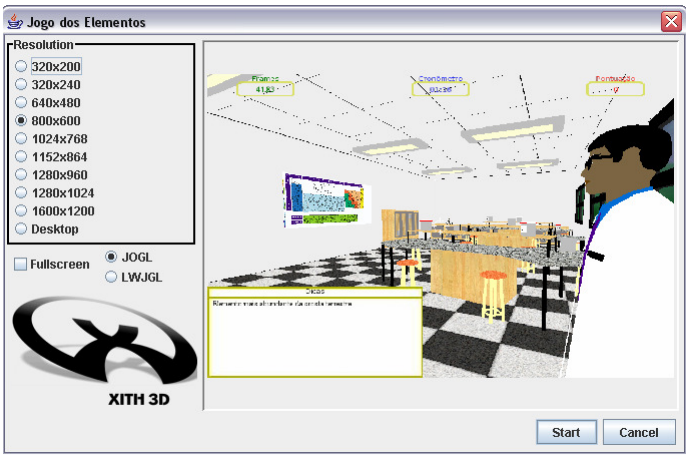


**Figura 33** - Janela de configuração inicial

A Figura 34 exibe a tela de carregamento do jogo que será mostrada até que o jogo seja totalmente carregado. A seguir o usuário deve clicar no botão “Iniciar o jogo”.

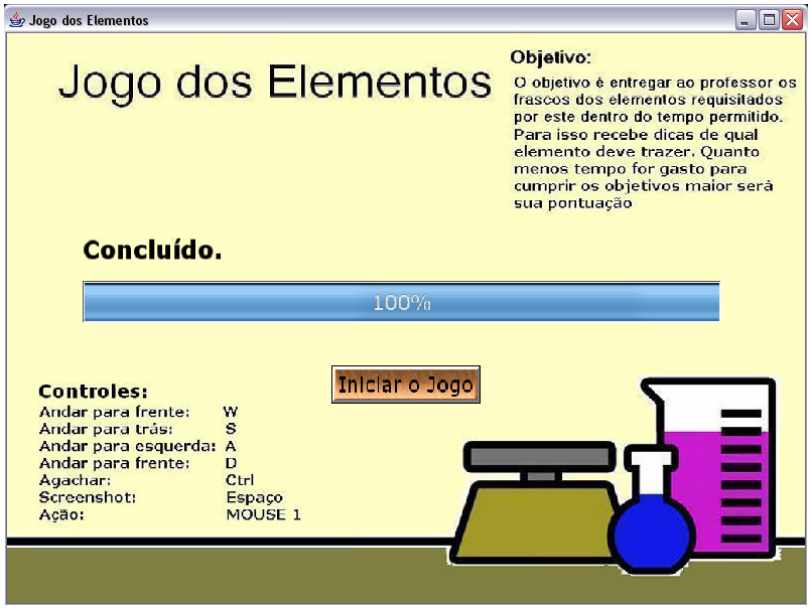


**Figura 34** – Tela de carregamento do jogo

Quando o usuário clicar no botão iniciar o tempo começa a ser decrementado. No painel de dicas serão exibidas as dicas do elemento corrente. Neste ponto o usuário deve buscar o frasco do elemento correto, baseado nas dicas recebidas. Quando o usuário clica em um frasco este é removido da cena. Caso queira trocar de frasco basta ao jogador clicar em outro elemento e o anterior será reposto na cena.

Após pegar um frasco, o jogador deve se dirigir ao professor e clicar nele para saber se trouxe o frasco correto. Caso tenha acertado o desafio um painel será apresentado indicando o acerto, conforme Figura 35. O painel de pontuação será atualizado com um valor calculado a partir do tempo que o jogador levou para entregar o frasco e o tempo para o próximo teste será incrementado.


**Figura 35** - Entrega de frasco correto

Caso o frasco não corresponda ao requerido o painel de erro será exibido (Figura 36).


**Figura 36** – Entrega de frasco incorreto

Após entregar uma série de 10 elementos o jogo termina, é exibida a pontuação total do jogador e é adicionado um botão para fechar o jogo à cena, Figura 37. Após clicar no botão fechar o jogo é finalizado.


**Figura 37** – Conclusão do jogo com sucesso

O jogador perde quando não entrega o frasco requisitado dentro do tempo disponível. Neste ponto é adicionada a cena o painel indicando que o tempo acabou além do botão para finalizar o jogo (Figura 38).


**Figura 38** – Conclusão do jogo sem sucesso

## 5.8 Implementação

As principais ações que jogador deve executar para cumprir os objetivos do jogo são pegar um frasco e entregá-lo ao professor. Essas ações são controladas por um ouvidor de *pick* (*PickListener*), que marca o ponto que o jogador clicou e cria uma "reta" (*PickRay*) tridimensional passando pela cena. Os objetos que fizerem intersecção com essa reta são armazenados em uma lista. No jogo a classe que controla o *picking* é a classe Game, que herda a classe *ExtRenderLoop* e implementa a classe *PickListener*. A classe *ExtRenderLoop* controla o laço de renderização e a classe *PickListener* "ouve" os eventos de *picking*.

O método abstrato *onMouseButtonReleased*, Figura 39, é herdado da classe ExtRenderLoop e é executado quando um botão do mouse é liberado, após ter sido pressionado. A esse método são passadas como parâmetro as coordenadas do ponto clicado (x, y) e a identificação do botão liberado (*button*). Após isto, na linha 99, é verificado se o botão do mouse liberado é o esquerdo. O programador de *pick* (*PickScheduler*) é uma interface implementada na classe *ExtXith3dEnviroment* e controla os eventos de *pick*. O método *pick*, na linha 101, programa uma nova operação de *pick* no laço de renderização. A este método são passados cinco parâmetros, o nó raiz da cena (*scene*), as coordenadas do ponto em que ocorreu o *pick* (x e y), o *PickListener* (*this*) e a variável booleana *pickAll*, que faz com que o método retorne todos os objetos que fizeram intersecção com a reta de *pick*.

```
95      public void onMouseButtonReleased(int button, int x, int y)
96      {
97          switch (button)
98          {
99              case MouseCode.BUTTON1:
100                 boolean pickAll = true;
101                 pickScheduler.pick( scene, x, y, this, pickAll );
102                 break;
103         }
104     }
```

**Figura 39** – Código de checagem do botão esquerdo do mouse

Após um evento de *pick* ser executado, o *loop* de renderização executa o método abstrato *onObjectsPicked*, Figura 40. Neste método é passado como parâmetro a lista de objetos que fazem intersecção com a reta (*pickResults*).

```
108 public void onObjectsPicked(List<PickResult> pickResults, Object userObject, long pickTime)
109 {
110  int i = 0;
111
112  for (PickResult pr: pickResults)
113  {
114      // Verifica a distancia do objeto clicado
115      if ( (pr.getMinimumDistance() < 3000f) && (i < 3) )
116      {
117       GroupNode parent = pr.getShape().getParent().getParent().getParent();
118
119       if (doorsMgr.checkDoorPicked( parent.getName(), this.getGameTime() ))
120       {
121           i = 3;
122       }
123       else if (!inventoryMgr.checkBottlePicked( pr.getShape().getName(), parent ))
124       {
125           ProfessorManager.Message message = professorMgr.checkProfessorPicked( parent.getName(), inventoryMgr, ui );
126
127           if (message != null)
128           {
129               if (message != ProfessorManager.Message.WRONG)
130               {
131                   // Aumenta pontuação do jogador
132                   ui.incrementPontuation( cronometer.getTimeCounter() + 100 );
133
134                   // Aumenta o tempo para o prximo teste
135                   cronometer.pushTimeCounter();
136               }
137
138               switch (message)
139               {
140                   case YOU_WON:
141                       ui.setCrosshairHidden( true );
142                       ui.showWinnerDialog( "Parabens! \nVocê venceu com " + ui.getPontuation() + " pontos!" );
143                       this.removeAllIntervals();
144                       avatarMgr.setSuspended( true );
145                       mouse.setExclusive( false );
146
147                       break;
148               }
149              }
150             }
151         }
152
153         i++;
154     }
155 }
```

**Figura 40** – Checagem dos resultados de um *pick* no *loop* principal do jogo

Na linha 115 é checado a distância mínima entre o *PickRay* e o objeto, para que o jogador não pegue um frasco do outro lado da sala, por exemplo. Na linha 123 é chamado a função *checkBottlePicked()* da classe *InventoryManager*. Esta classe gerencia o inventário do jogador, ou seja, guarda o frasco que o jogador pegou no tempo corrente do jogo. Isto permite que o frasco seja reposto na cena caso o jogador queira trocar de frasco, ou, caso entregue o frasco incorreto ao professor.

A função *checkBottlePicked()*, Figura 41, tem como parâmetros o nome do objeto obtido e o nó pai deste objeto. Esta função verifica na linha 61 se o objeto é um frasco. Caso seja verdadeiro é chamado então o método *removeBottle()*, da mesma classe, que verifica se já um frasco no inventário e, caso tenha um, remove esse frasco do inventário, recolocando-o na cena. A partir deste ponto é necessário colocar o frasco no inventário e removê-lo da cena. Isto é executado no método *addBottle()*, linha 67, que retira-o da cena e guarda o nó deste frasco para que possa ser reposto na cena se necessário. O método *checkBottlePicked()* finalmente retorna verdadeiro, indicando que o objeto obtido era um frasco.

```
57      /** Checa se clicou em um frasco */
58      public boolean checkBottlePicked(String testName, Node testNode)
59      {
60          // Se clicou em um frasco
61          if ( testName.contains("frasco_") )
62          {
63              // Remove o anterior do inventario (se houver um)
64              removeBottle();
65
66              // Adiciona o frasco ao inventario
67              addBottle( testNode );
68
69              return( true );
70          }
71          else
72          {
73              return( false );
74          }
75      }
```

**Figura 41 –** Checagem de *picking* para um frasco

Caso o valor booleano retornado pela função seja falso, é necessário checar o objeto pertence ao grupo do professor. Isto é executado pela função *checkProfessorPicked()*, linha 125 da Figura 40, da classe *ProfessorManager*.

A função *checkProfessorPicked()*, Figura 42, tem como parâmetros o nome do nó pai, o gerenciador do inventário e a interface com o usuário (*UserInterface*). Primeiramente, na linha 32, a função verifica se o nó pai do objeto é o professor, e, em seguida verifica se existe um frasco no inventário. Na linha 35 é comparado se o frasco trazido é o requisitado pelo professor (função *getCurrentRequestElement*).

```
29      /** Checa se clicou no professor */
30      public Message checkProfessorPicked(String testName, InventoryManager inventoryMgr, MessageManager messageMgr)
31      {
32          if ((testName.equals("Professor")) && (inventoryMgr.containsBottle()))
33          {
34              // Se troxue o frasco correto
35              if (inventoryMgr.containsBottle( getCurrentRequestedElement() ))
36              {
37
38                  if (elCounter < 9)
39                  {
40                      // Incrementa o indice do vetor de Elementos
41                      elCounter++;
42
43                      inventoryMgr.removeDeliveredBottle();
44
45                      // Mostra a mensagem de Acerto
46                      messageMgr.showRightMessage();
47                      return( Message.RIGHT );
48                  }
49                  else
50                  {
51                      inventoryMgr.removeDeliveredBottle();
52                      return( Message.YOU_WON );
53                  }
54              }
55              // Se no trouxe o frasco correto
56              else
57              {
58                  // Recoloca o frasco na cena
59                  inventoryMgr.removeBottle();
60
61                  // Mostra mensagem de erro
62                  messageMgr.showWrongMessage();
63
64                  return( Message.WRONG );
65              }
66          }
67          else
68          {
69              return( null );
70          }
71      }
```

**Figura 42** – Checagem de *picking* para o professor

A variável *elCounter* marca o índice do vetor de elementos, já que o jogador deve entregar uma série de dez elementos (elementos 0 a 9). Na linha 43 o frasco entregue é removido da cena através do método *removeDeliveredBottle()* da Classe *InventoryManager*.

Em seguida, na linha 46 é chamado o método *showRightMessage()* da interface *MessageManager*. Este método cria um intervalo de um segundo que adiciona o nó da mensagem ao nó raiz de interface quando criado, e, em seguida, retira o nó de mensagem do grafo e finaliza o intervalo, mostrando a mensagem durante um segundo na tela. Por fim, a função retorna a mensagem de acerto (*Message.RIGHT*).

Se for o ultimo elemento a ser entregue (*elCounter* igual a 9), e este for o requisitado, retorna-se a mensagem de vitória (*Message.YOU_WON*), linha 52.

Caso o elemento entregue não seja o requisitado, recoloca o frasco anterior na cena, pela execução do método *removeBottle()* da interface *InventoryManager*. Após isto, é mostrada a mensagem de erro na tela, pela chamada do método *showWrongMessage()*. Finalmente retorna a mensagem de erro, (*Message.WRONG*).

Após a mensagem ter sido retornada ao método *onObjectsPicked()*, na linha 129 da Figura 40, verifica se a mensagem retornada não é uma mensagem de erro. Caso isto ocorra, na linha 132, é incrementada a pontuação do jogador pela chamada do método *incrementPontuation()* da classe *UserInterface*. Após isso, na linha 135, é chamado o método *pushTimeCounter()* da classe *Cronometer* para incrementar o tempo para o jogador cumprir a próxima tarefa.

Na linha 140, é comparado se a mensagem retornada foi de vitória (*Message.YOU_WON*). Ocorrendo isto é retirada a mira do jogador da interface, pela chamada do método *setCrosshairHidden()*, na linha 141.

O método *showWinnerDialog()*, na linha 142, adiciona ao grafo de interface um painel com a pontuação do jogador e um botão para finalizar o jogo.

Em seguida todos o intervalos correntes são finalizados pela execução do método *removeAllIntervals()*.

Na linha 144, executa-se o método *setSuspended()*. Este método é herdado da Classe *EgoInputAdapter* e suspende o avatar pela finalização do *listener* de mouse e teclado.

Por fim, na linha 145 chama-se o método *setExclusive()*, da classe *MouseDevice*, com valor o booleano *false*. Este método libera o mouse do centro do *canvas* para que o jogador possa clicar no botão de fechar.

## 5.9 Desempenho

O *Xith3D* atualmente possui uma taxa de FPS mais baixa do que a *engine JME*.

Em testes efetuados no computador um composto por processador Pentium 4 de 2.4 GHz, Placa de vídeo de 128 MB, 512 MB de memória RAM, sistema operacional Microsoft Windows XP, a média de FPS foi de 57.3 com uma variação entre 48 FPS e 93 FPS.

# CONCLUSÕES

O avanço tecnológico constante permite o desenvolvimento de *softwares* educacionais mais elaborados e complexos, tornando-os uma excelente ferramenta de auxílio ao ensino. Os jogos educacionais que utilizam RV são capazes de prender a atenção do aluno com seus gráficos elaborados e alta capacidade de exploração, e, com a supervisão do professor treinado permitem uma assimilação de conhecimento maior do que a compreendida em um ambiente normal de aprendizado escolar.

Os jogos educacionais tridimensionais podem ser desenvolvidos com facilidade e rapidez apresentando um custo baixo. A implementação de um jogo educacional utilizando uma linguagem de alto nível como o Java com o apoio das *engines Xith3D* e *jME* atualmente é desejável, mas apresenta algumas vantagens e desvantagens.

As principais vantagens são: rapidez na implementação do projeto; não é necessário um conhecimento aprofundado na criação de mundos virtuais devido à linguagem apresentar classes e métodos específicos para essa função; a linguagem é gratuita. E as desvantagens são: deve haver um cuidado na implementação para não permitir que o *Xith3D* e também o *jME* aloque muita memória, tornando o desempenho do *software* baixo; algumas bibliotecas do *Xith3D* são de difícil implementação e atualmente não funcionam de forma correta, como por exemplo, bibliotecas para tratamento de fatores físicos nos mundos virtuais, colisões, gravidade, etc, sendo que o *jME* já possui estas bibliotecas funcionando corretamente. Durante o desenvolvimento utilizando a *engine jME,* não foram encontrados *bugs* em nehuma das classes utilizadas, apenas a falta de alguns recursos, como a criação de uma tela de carregamento (*loading*), porém possui uma programação mais difícil e complicada.

Por fim, é verificada a validade do desenvolvimento de um ambiente virtual para um jogo educacional 3D utilizando Java e as *engines Xith3D* e *jME*, pois é possível apresentar um bom nível de desempenho e uma ótima qualidade gráfica.

## 6.1 Trabalhos Futuros

Uma continuidade para o desenvolvimento do jogo E.C.A.S. seria o desenvolvimento de novos desafios, incluindo desafios que estariam registrados em um banco de dados, e que fossem exibidos aleatoriamente durante os desafios do jogo.

Outras continuidades seriam a criação de um menu durante o jogo, contendo duas novas opções e a exibição de uma pontuação do usuário após a finalização do jogo. As duas novas opções para o usuário no menu durante o jogo seriam:

- Uma opção para salvar o estado atual do jogo para que, caso o usuário saia do jogo, ele possa reiniciar a ação do jogo a partir do ponto salvo.
- E a outra opção para reiniciar o jogo.

Uma continuidade para o "Jogo dos Elementos" seria o desenvolvimento e a implementação de um sistema de colisão, já que o atual sistema de colisão da ferramenta *Xith3D* não está funcionando corretamente. Outra continuidade seria o desenvolvimento de novos desafios para o jogo.

# REFERÊNCIAS

APACHE Ant. Version 1.6.5. [S.l.]: Apache, 2005 – Disponível em <http://ant.apache.org> Acesso em: 12 ago. 2006.

AUTODESK 3D Studio Max. Version 6. [S.l.]: Autodesk, 2005. 1 CD-ROM.

BARROS, E.; ARAÚJO, F. **Uma visão histórico-taxonômica dos ambientes de aprendizagem assistidos por computador**. In: SEMANA DE INFORMÁTICA DA UFBA, 7., 1998, Salvador. *Anais*... Salvador: UFBA, 1998.

COREL Winzip. Version 11.0 [S.l]: A Corel Company, 2006. – Disponível em <http://www.winzip.com>.

ENJINE Enjine – Disponível em: < http://enjine.incubadora.fapesp.br/portal> Acesso em: 12 jun. 2006.

JAVANET Java.net - Disponível em: <http://java.net> Acesso em: 28 ago. 2006.

JME jMonkey Engine. 2006. Disponível em: <http://www.jmonkeyengine.com> Acesso em: 3 maio 2006.

JME-FORUM jMonkey Engine Discussion Forums – Disponível em: <http://www.jmonkeyengine.com/jmeforum/> Acesso em: 28 jul. 2006.

JME-TUTORIALS jMonkey Engine Tutorials – Disponível em: <http://www.jmonkeyengine.com/wiki/doku.php?id=the_tutorials> Acesso em: 28 jul. 2006.

JOGL Java Bindings for OpenGL**.** Disponível em: <http:// https://jogl.dev.java.net> Acesso em: 10 abr. 2006.

LEHMANN, J. **Getting started.** In: XITH3D PROJECT GROUP, 2004. Disponível em: < http://xith.org/tutes/GettingStarted/getting_started_guide.pdf>. Acesso em: 15 mar. 2006.

LINDAMOOD, J. **Learning jME,** 2005. Disponível em: <http://www.jmonkeyengine.com/wiki/doku.php?id=learning_jme>. Acesso em: 15 maio 2006.

LWJGL. Lightweight Java Game Library. Disponível em <http://www.lwjgl.org> Acesso em: 4 maio 2006.

MARTINS, J.G. et al. **Realidade virtual através de jogos na educação.** In: *INTERCOM*: Sociedade Brasileira de Estudos Interdisciplinares da Comunicação, 2002. Disponível em: <http://www.intercom.org.br/papers/xxii-ci/gt13/13m02.PDF>. Acesso em: 20 fev. 2006.

NETBEANS Netbeans. Version 5.5. [S.l.]. 2006. Disponível em: <http://www.netbeans.org>. Acesso em: 21 out. 2006.

NOBELPRIZE NobelPrize Disponível em: <http://nobelprize.org/index.html>. Acesso em: 28 abril 2006.

OPENAL Cross-Platform 3D Audio, 2006. Disponível em: <http://www.openal.org/>. Acesso em: 29 maio 2006.

SCHANK, R. VIRTUAL LEARNING: **A revolutionary approach to building a highlyskilled workforce.** Ed McGraw-Hill. New York, NY, USA. 1997.

SUN Java se development kit. Version 1.5. [S.l.]: Sun Microsystems, 2006 – Disponível em: <https://sdlc4a.sun.com/ECom/EComActionServlet;jsessionid=406A0CAC851A81F0E1130054FC91E0FF> Acesso em: 19 jul. 2006.

SUPERCHARGED! In: The Education Arcade: An MIT-University of Wisconsin Partnership. Disponível em: <http://educationarcade.org/supercharged>. Acesso em: 23 fev. 2006.

TAROUCO, L. M. R. et al. **Novas Tecnologias na Educação:** Jogos educacionais, v. 2, n. 1, mar. 2004. Disponível em: < http://www.cinted.ufrgs.br/renote/mar2004/artigos/30-jogoseducacioanis.pdf >. Acesso em: 20 fev. 2006

TIGRES TortoiseSVN. Version 1.4.1. [S.l.]. 2006. Disponível em: < http://tortoisesvn.tigris.org/>. Acesso em: 16 nov. 2006.

VALENTE, J.A. **Diferentes usos do computador na educação**. Campinas: UNICAMP, 1995. Disponível em: <http://www.nied.unicamp.br/publicacoes/separatas/Sep1.pdf>. Acesso em: 20 fev. 2006.

VALENTE, J.A. Uso inteligente do computador na educação. **Pátio – Revista Pedagógica**, ano 1, n. 1, p. 19-21, set. 2002. Disponível em: <http://www.unidavi.edu.br/~afischer/content/2002-Sep-27_19-57-37.pdf>. Acesso em: 20 fev. 2006.

VBL The Virtual Biochemistry Laboratory, disponível em: < http://nobelprize.org/chemistry/educational/vbl/index.html>. Acesso em: 28 abril 2006.

XITH3D. Xith3D engine. 2006.Disponível em <http://www.xith.org> Acesso em: 8 abril 2006

WINCVS WinCVS. Version 2.0.2. [S.l.] 2006 – Disponível em < http://www.wincvs.org/> Acesso em: 17 jul. 2006.

# ANEXOS

## **ANEXO A –** Instalação Do *JME*

# INSTALAÇÃO DO *JME*

## Introdução

Este guia irá detalhar todos os passos necessários para a obtenção e instalação do *jME* em seu ambiente de desenvolvimento.

Antes de prosseguir, acesse o *site* do Java.net (JAVANET, 2006) e cadastre-se.

Para que se obtenha o *jME* e para que ele execute corretamente, são necessários alguns programas e a execução de algumas etapas. A seção "Etapas e Componentes" cita os nomes destes programas e quais os passos que devem ser seguidos. Este guia irá explicar como obter cada um desses programas e como configurá-los corretamente.

## Etapas e Componentes

Os programas necessários para a compilação e execução do *jME* são:

- Compilador e Ambiente de Execução da linguagem de programação Java;
- CVS – Utilitário de Controle de Código Fonte (WINCVS, 2006);
- Ant (APACHE, 2005);

- Biblioteca de Jogo LWJGL (*LightWeight Java Game Library)*
- Código Fonte do *jME*

Alguns passos são necessários para obter e instalar o *jME* corretamente. Siga os passos na ordem descrita para que não haja nenhum problema durante a execução de qualquer um dos programas acima. Os passos são:

- *Download* e Instalação do Java SDK
- *Download* e Instalação da Ferramenta CVS
- *Download* e Instalação do Ant
- *Download* e Instalação do *jME*
- Compilar o *jME*
- Testar o *jME*

**Primeiro Passo:** *Download* e Instalação do Java SDK

O *jME* é uma *engine* baseada na Linguagem de Programação Java. Para que se possa desenvolver algo utilizando o *jME* é necessário possuir o pacote para desenvolvimento Java. Caso já possua este pacote, este passo poderá ser pulado, caso contrário, é necessário o *download* e a instalação do pacote *Java Development Kit* (JDK) da *Sun*. É recomendável a utilização da versão 5.0 ou mais nova.

Depois que a instalação estiver completa, é interessante verificar se a instalação foi concluída com sucesso. Abra uma janela de *prompt* de comando e digite o seguinte comando:

*java –version*

Se a instalação for bem sucedida, será exibido algo como na Figura 43.

```
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed mode, sharing)
```

**Figura 43 -** Checagem de versão do Java

**Segundo Passo:** *Download* e Instalação da Ferramenta CVS

O Sistema de Versões Concorrentes (CVS) é um sistema de controle de versão que permite que se trabalhe com diversas versões de arquivos organizados em um diretório e que estejam localizados localmente ou remotamente, mantendo-se suas versões antigas e os *logs* de quem e quando manipulou os arquivos. É especialmente útil para se controlar versões de um *software* durante seu desenvolvimento, ou para composição colaborativa de um documento.

Para obter o código fonte do *jME*, será necessário acesso ao repositório de código fonte em um servidor remoto. Para isso, é utilizado o CVS.

Vários clientes CVS são disponibilizados para *download* na Internet, muitos deles gratuitamente. Caso já possua um cliente CVS, este passo poderá ser pulado. Neste passo será utilizado o cliente WinCVS. Outros clientes possuem execuções similares.

Faça o *download* do WinCVS e instale-o. O WinCVS possui versões somente para *Windows*.

**Terceiro Passo:** *Download* e Instalação do Ant

O Ant, ou Apache Ant é uma ferramenta utilizada para automatizar a construção de *software*. Ela é similar ao *make* do Linux mas é escrita na linguagem Java e foi desenvolvida

inicialmente para ser utilizada em projetos desta linguagem. Para utilizá-lo, é necessário efetuar seu *download* e instalá-lo.

Simplesmente acesse o *site* e siga as instruções para fazer o *download* do arquivo zip.

Depois que o *download* estiver concluído, é necessário instalá-lo para utilizá-lo. O Ant não possui nenhuma interface de instalação como outros programas, mas não é difícil configurá-lo.

Primeiro, selecione um caminho para a instalação. Seguindo o paradigma *Windows*, "C:\Arquivos de Programas\Apache" é uma boa opção. Entretanto, qualquer caminho pode ser utilizado (outros preferem simplesmente "C:\ant"). Em seguida, usando Winzip (COREL, 2006) (ou qualquer outra ferramenta de extração de arquivos zip), extraia o Ant do arquivo zip no diretório escolhido (por exemplo: "C:\Arquivos de Programas\Apache").

Depois de concluído, renomeie a pasta onde foi extraído o ant, ao invés de utilizar a pasta extraída pelo zip com nome extenso.

A seguir, será necessária a declaração de duas novas Variáveis de Ambiente: ANT_HOME declarando o caminho da instalação do ant (como no exemplo: C:\Arquivos de Programas\Apache\ant), e JAVA_HOME com o local de onde se encontra o pacote de desenvolvimento Java (normalmente em: C:\Arquivos de programas\Java\jdk1.5.0_06, podendo variar de acordo com a versão instalada).

Também na Variável de Ambiente *Path*, adicione no final de seu campo o valor: ";%JAVA_HOME%\bin;%ANT_HOME%\bin", sem as aspas duplas.

Agora, o Ant deve estar instalado e pronto para sua utilização. Para testá-lo, abra uma janela de *prompt* de comando e digite: *ant –version*.

Deverá ser exibida uma mensagem do ant informando a versão instalada do ant, como na Figura 44.

```
Apache Ant version 1.6.5 compiled on June 2 2005
```

**Figura 44 -** Verificação da versão instalada do Ant

**Quarto Passo:** *Download* e Instalação do *JME*

Neste passo, serão obtidos os códigos fontes do *jME*. Para isso, será utilizado o cliente CVS.

Basicamente, por meio da linha de comando do CVS, é necessário apenas executar conforme a Figura 45 (tenha certeza de repor [java.net.username] com o seu nome de usuário do java.net).

```
cvs -d :pserver:[java.net username]@cvs.dev.java.net:/cvs checkout -P jme
cvs -d :pserver:[java.net username]@cvs.dev.java.net:/cvs login
```

**Figura 45 -** Comandos CVS para obter o *jME* através de um Cliente de Linha de Comandos

Estes são os passos necessários para a obtenção do código fonte *jME* através de um cliente de linha de comando. Caso deseje utilizar o WinCVS, continue a leitura deste passo.

Primeiro, caso já não o tenha feito, execute o WinCVS. A obtenção do *jME* ficou muito simples com o WinCVS a partir da versão 2.0.

Ao abrir o programa, clique em *Admin* (na barra de menu), e em seguida clique em *Preferences*. Irá abrir uma janela chamada "*WinCVS Preferences*". Nesta janela, clique na aba CVS. Na parte CVS *Setup*, no campo *HOME*, digite o caminho aonde deseja que o código fonte seja instalado (por exemplo: C:\ProjetoJME). Após feito isso, clique em OK.

Feita a configuração do diretório de destino, clique na opção de menu *Admin* novamente, e em seguida clique em *Login*. Na nova janela que abrir, no campo de texto a frente de CVSROOT, basta digitar:

```
:pserver;username=seu_usuario;password=sua_senha:@cvs.dev.java.net:/cvs
```

Não esquecendo de substituir seu_usuario e sua_senha com seus respectivos nome de usuário e senha cadastrados no Java.net. Clique em OK. Depois de feito isso, o WinCVS deverá se conectar com o *host*.

Em seguida, clique na opção *Remote*, na barra de menu, e depois em *Checkout module*. Em "*Module name and path on the server*" digite 'jme' (sem as aspas). Após especificar o nome do módulo, clique em OK. O código do *jME* deverá começar a ser baixado.

Depois que o *download* estiver terminado, clique na opção *Admin* e em seguida *Logout*, para que sua conexão seja encerrada com o servidor.

O benefício ganho por ter efetuado o *download* do código fonte ao invés dos arquivos ".jar" é que a API LWJGL é obtida junto com o código. O LWJGL é uma API que comunica com sua placa de vídeo através do OpenGL, necessária para o funcionamento do *jME*. O LWJGL estará localizado em (como no exemplo): "C:\ProjetoJME\jme\libs".

A próxima etapa é a compilação do código fonte. Para isso será utilizada a instalação do Ant, já efetuada anteriormente.

**Quinto Passo:** Compilar o *JME*

Para a compilação do *jME*, abra uma janela de *prompt* de comando (ou use a mesma que foi utilizada para a verificação da versão do ant) e altere a sua localização para o diretório onde se encontra o código fonte do *jME*.

Agora, são necessários apenas alguns comandos para compilá-lo. O primeiro comando compila o código fonte principal do *jME*:

*ant dist-all*

É possível visualizar o ant compilando os vários diretórios do código do *jME*. Nesta etapa não devem haver erros do "javac".

O próximo comando é para compilar todos os códigos testes e demos:

*ant dist-test*

O ant deverá compilar rapidamente os arquivos testes. Novamente, não devem haver erros do "javac".

O último passo a ser seguido é a de verificar se a instalação do *jME* foi bem sucedida.

**Sexto Passo:** Testar o *jME*

Para testar a instalação do *jME* é necessário executar um dos testes compilados no passo anterior.

Para isso abra inicialmente uma nova janela do *prompt* de comando ou utilize a mesma em que foi compilado o *jME*. Então, acesse o diretório principal do *jME* (como por exemplo, C:\ProjetoJME\jme).

O *jME* já vem com um aplicativo que indica quais demos podem ser executados. Para isso digite o comando da Figura 46.

```
C:\>java -Djava.library.path=./lib -cp ./lib/lwjgl.jar;./lib/jogg-0.0.5.jar;./li
b/jorbis-0.0.12.jar;./target/jme.jar;./target/jmetest.jar;./target/jmetest-data.
jar jmetest\TestChooser_
```

**Figura 46 -** Comando para execução da Classe de Testes do *jME*

Este aplicativo abrirá uma janela com vários demos que podem ser executados. Selecione uma das opções e clique em OK. A seguir será exibida uma janela com as preferências de execução. Selecione a resolução, qualidade, freqüência do monitor e se deseja ou não executar em Tela Cheia. Depois, clique em OK.

Caso haja erros em alguns dos passos, refaça o passo referente ao erro.

Outros tutoriais podem ser obtidos na página de documentações do *jME* (JME-TUTORIAIS, 2006) e também no Fórum do *jME* (JME-FORUM, 2006).

# **ANEXO B** – Instalação do *Xith3D*

## Instalação do *Xith3D*

A seguir serão detalhados todos os passos necessários para a obtenção e instalação do *Xith3D* em seu ambiente de desenvolvimento.

Para que se obtenha o *Xith3D* e para que ele execute corretamente, são necessários alguns programas e a execução de algumas etapas. A seção "Etapas e Componentes" cita os nomes destes programas e quais os passos que devem ser seguidos. Este guia irá explicar como obter cada um desses programas e os configurá-los corretamente.

## Etapas e Componentes

Os programas necessários para a compilação e execução do *Xith3D* são:

- Compilador e Ambiente de Execução da linguagem de programação Java
- SVN – Ferramenta de controle de versão para código fonte
- Ant
- Código Fonte do *Xith3D* e *Xith-tk*

Alguns passos são necessários para obter e instalar o *Xith3D* corretamente. Siga os passos na ordem descrita para que não haja nenhum problema durante a execução de qualquer um dos programas acima. Os passos são:

- Download e Instalação do Java SDK

- Download e Instalação da Ferramenta SVN
- Download e Instalação do Ant
- Download e Instalação do *Xith3D*
- Compilar o *Xith3D*
- Download e Instalação do Xith-tk
- Compilar o Xith-tk

**Primeiro Passo:** Download e Instalação do Java SDK

O *Xith3D* é uma *engine* baseada na Linguagem de Programação Java. Para que se possa desenvolver algo utilizando o *Xith3D* é necessário possuir o pacote para desenvolvimento Java. Caso já possua este pacote, este passo poderá ser pulado, caso contrário, é necessário o *download* e a instalação do pacote *Java Development Kit* (JDK) da *Sun*. É recomendável a utilização da última versão.

Depois que a instalação estiver completa, é interessante verificar se a instalação foi concluída com sucesso. Abra uma janela de *prompt* de comando e digite o seguinte:

*java –version*

Se a instalação for bem sucedida, será exibido algo como na Figura 47.

```
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed mode, sharing)
```

**Figura 47 -** Checagem de versão do Java

**Segundo Passo:** *Download* e Instalação da Ferramenta SVN

SVN *(Subversion*) é uma ferramenta de controle de versão para código fonte que fornece não somente os meios de armazenamento da versão atual de parte do código fonte, mas também um registro de todas as mudanças (e quem fez tais mudanças) que ocorreram no código fonte. O uso da SVN é comum em projetos com múltiplos colaboradores, já que a SVN assegura que as mudanças feitas por um colaborador não serão removidas acidentalmente por outro colaborador.

Para obter o código fonte do *Xith3D* e Xith-tk, será necessário acesso ao repositório de código fonte em um servidor remoto. Para isso, é utilizado o SVN.

Vários clientes SVN são disponibilizados para *download* na Internet. Caso já possua um cliente SVN, este passo poderá ser pulado. Neste passo será utilizado o cliente TortoiseSVN (TIGRES, 2006). Outros clientes possuem execuções similares.

Faça o *download* do TortoiseSVN e instale-o. O TortoiseSVN possui versões somente para *Windows* 32 bits e *Windows* 64 bits.

**Terceiro Passo:** *Download* e Instalação do Ant

O Ant é uma ferramenta utilizada para automatizar a construção de *software*. Para utilizá-lo, é necessário efetuar seu *download* e instalá-lo.

Primeiramente acesse o site e siga as instruções para fazer o download do arquivo zip.

Depois que o *download* estiver concluído, é necessário instalá-lo para utilizá-lo. O Ant não possui nenhuma interface de instalação como outros programas, mas não é difícil configurá-lo.

Para instalar selecione um caminho destino para a instalação. Seguindo o paradigma Windows, “C:\Arquivos de Programas\Apache”. Em seguida, usando qualquer ferramenta de

extração de arquivos zip, extraia o Ant do arquivo zip no diretório escolhido (por exemplo: "C:\Arquivos de Programas\Apache").

Depois de concluído, renomeie a pasta onde foi extraído o ant, ao invés de utilizar a pasta extraída pelo zip com nome extenso.

A seguir, será necessária a declaração de duas novas Variáveis de Ambiente: ANT_HOME declarando o caminho da instalação do ant (como no exemplo: "C:\Arquivos de Programas\Apache\ant"), e JAVA_HOME com o local de onde se encontra o pacote de desenvolvimento Java (normalmente em: "C:\Arquivos de programas\Java\jdk1.5.0_06", podendo variar de acordo com a versão instalada).

Também na Variável de Ambiente *Path*, adicione no final de seu campo o valor: ";%JAVA_HOME%\bin;%ANT_HOME%\bin".

Agora, o Ant deve estar instalado e pronto para sua utilização. Para testá-lo, abra uma janela de *prompt* de comande digite: *ant –version*.

Deverá ser exibida uma mensagem do ant informando a versão instalada do ant, como na Figura 48.

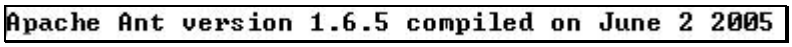


**Figura 48 -** Verificação da versão instalada do Ant

**Quarto Passo:** *Download* e Instalação do *Xith3D*

Neste passo, serão obtidos os códigos fontes do *Xith3D*. Para isso, será utilizado o cliente SVN.

Primeiro, crie uma pasta com nome "*Xith3D*" no diretório em que deseja salvar o código fonte. Em seguida clique com o botão direito do *mouse* e selecione a opção "*SVN Checkout*".

Após o programa abrir, digite em "*URL of repository*":

*"https://svn.sourceforge.net/svnroot/xith3d/trunk"*

Certifique se o caminho para a pasta destino está correto no campo "*Checkout Directory*" e clique em "*OK*" para fazer o *download*. Em seguida aguarde enquanto os arquivos serão baixados e pressione "*OK*" quando o *download* for concluído

A próxima etapa é a compilação do código fonte. Para isso será utilizada a instalação do Ant, já efetuada anteriormente.

**Quinto Passo:** Compilar o *Xith3D*

Para a compilação do *Xith3D*, abra uma janela de prompt de comando e altere a sua localização para o diretório "*Xith3D*" e excute o comando "*ant*". Isto irá compilar o código fonte e criar o arquivo "xith3d.jar". Para criar o javadoc digite "ant javadoc" no prompt de comando.

**Sexto Passo:** *Download* e Instalação do *Xith-tk*

Neste passo, serão obtidos os códigos fontes do *Xith-tk*. Para isso, será utilizado o cliente SVN.

Primeiro, crie uma pasta com nome "Xith-tk" no mesmo diretório em que foi criada a pasta "Xith3d". Em seguida clique com o botão direito do *mouse* e selecione a opção "*SVN Checkout*".

Após o programa abrir, digite em "*URL of repository*":

*"https://svn.sourceforge.net/svnroot/xith3d/trunk"*

Certifique se o caminho para a pasta destino está correto no campo "*Checkout Directory*" e clique em "*OK*" para fazer o *download*. Em seguida aguarde enquanto os arquivos serão baixados e pressione "*OK*" quando o *download* for concluído

A próxima etapa é a compilação do código fonte. Para isso será utilizada a instalação do Ant, já efetuada anteriormente.

**Sétino Passo:** Compilar o *Xith-tk*

Para a compilação do *Xith-tk*, abra uma janela de prompt de comando e altere a sua localização para o diretório "Xith tk" e excute o comando "*ant*". Isto irá compilar o código fonte e criar os arquivos "xith-tk.jar" e "xith-demos.jar". Para criar o javadoc digite "ant javadoc" no prompt de comando.

Caso haja erros em alguns dos passos, refaça o passo referente ao erro.