CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Universidade Federal
de Campina Grande

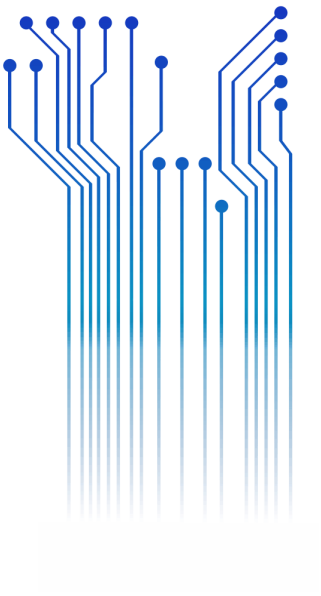
JOSÉ LUCAS MORAES VIEIRA

Centro de Engenharia
Elétrica e Informática

TRABALHO DE CONCLUSÃO DE CURSO

CONSTRUÇÃO DE UM EDITOR DE CENÁRIOS PARA O SIMULIHM

Departamento de
Engenharia Elétrica



Campina Grande
2016

JOSÉ LUCAS MORAES VIEIRA

CONSTRUÇÃO DE UM EDITOR DE CENÁRIOS PARA O SIMULIHM

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientadora:
Professora Maria de Fatima Queiroz Vieira, PhD.

Campina Grande
2016

JOSÉ LUCAS MORAES VIEIRA

CONSTRUÇÃO DE UM EDITOR DE CENÁRIOS PARA O SIMULIHM

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professora Maria de Fatima Queiroz Vieira, PhD.
Universidade Federal de Campina Grande
Orientadora, UFCG

Dedico este trabalho à minha família, que está sempre me apoiando, nos bons e nos maus momentos.

AGRADECIMENTOS

Agradeço a minha família, em primeiro lugar, por me darem apoio e estarem sempre presentes, contribuindo para minha felicidade.

Agradeço aos meus grandes amigos José Victor, Hélcio e Maria, presentes que ganhei com a minha mudança para Campina Grande.

Agradeço aos colegas de curso, principalmente Alex Carlos e Armando, que sempre me acompanharam nessa jornada árdua.

Agradeço aos companheiros de trabalho do LIHM, Alex, André, Renan, Flávio, Ademar e à professora Fátima, que me auxiliaram no desenvolvimento desse trabalho.

Enfim, agradeço a todos meus amigos, que me ajudam a tornar a vida mais agradável e mais feliz.

“Se você encontrar
um caminho
sem obstáculos,
ele provavelmente
não leva a lugar
nenhum.”

Frank Clark.

RESUMO

A simulação em realidade virtual para treinamento de operadores de subestações elétricas é a principal característica do SimuLIHM, simulador desenvolvido no LIHM. No entanto, a geração de cenários de treinamento ainda é uma tarefa difícil, devido à ausência de uma interface gráfica que venha a apoiar o tutor na construção dos cenários, de uma maneira mais ergonômica. Este trabalho tem como objetivo o desenvolvimento de um Editor de cenários, que venha a solucionar a dificuldade atual de construção de cenários. Para isto, foi desenvolvida uma interface gráfica única para apoiar a representação em mundo virtual tridimensional de uma sala de controle e, para apoiar a definição dos eventos a serem disparados durante as simulações. O trabalho foi validado a partir da geração de um cenário, com a definição do ambiente tridimensional e de eventos temporais associados aos objetos integrantes desse ambiente.

Palavras-chave: Simulação, Mundos Virtuais 3D, Cenários de treinamento, Interface Gráfica.

ABSTRACT

The virtual reality simulation for electric substations operators training is the main feature of SimuLIHM, a simulator developed at LIHM, in UFCG. However, generating training scenarios is still a very complex task, which could benefit from a graphical user interface cable of supporting the definition and generation of such scenarios, in a more ergonomic way. The objective of this work was to develop a Scenarios Editor that could solve the current scenario generation issues. During this project, a unified graphical user interface was created for the task of generating a virtual world representation of a substation control room and for generating the sequence of events to be triggered during the simulation. This work was validated by generating a scenario, associated with the definition of a corresponding three-dimensional environment, where the events, only temporal ones, are associated to the objects in this environment.

Keywords: Simulation, Three-dimensional, Training scenarios, Graphical user interface.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

X3D	Extensible 3D
XML	Extensible Markup Language
CPN	Coloured Petri Nets
3D	Tridimensional
LIHM	Laboratório de Interfaces Homem Máquina
GPG	Giro Pressão Giro
IDE	Integrated Development Environment
IHM	Interface Homem-Máquina
TCP/IP	Transmission Control Protocol/Internet Protocol
VRML	Virtual Reality Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Devido ao aumento significativo dos sistemas elétricos de potência, o desenvolvimento de ferramentas computacionais, que possibilitem um treinamento de operadores de subestações elétricas, cada vez mais eficaz e eficiente, se tornou uma prioridade nas empresas do setor. A preparação de um operador para os diversos tipos de ocorrências que podem ser vivenciadas no ambiente de uma sala de controle de subestação ou no campo, ou a reciclagem de um operador já experiente evidenciam situações nas quais um ambiente de treinamento apoiado por simulação seria de grande importância.

Nesse contexto de treinamento de operadores de sistemas elétricos, foi desenvolvido no Laboratório de Interfaces Homem-Máquina (LIHM), um simulador com realidade virtual, o SimuLIHM, para representar uma sala de controle de uma subestação típica - ambiente de trabalho dos operadores. Nesse ambiente, o simulador reproduz os painéis de controle utilizados para atuação remota de equipamentos presentes na subestação.

Além disso, também é representada a estação de trabalho do operador que é constituída por um sistema supervisório. Na operação de sistemas elétricos, os sistemas supervisórios representam graficamente a planta do sistema elétrico em um sinótico e em várias telas auxiliares que constituem a IHM. Nela é possível a visualização da localização e do estado de operação dos diversos equipamentos da instalação. Através do sistema supervisório, também pode ser feito o controle e atuação remota de equipamentos do sistema, visando realizar as tarefas típicas de operação, tais como o isolamento de linhas de transmissão e a abertura e fechamento de chaves e disjuntores.

1.1 MOTIVAÇÃO

O treinamento de operadores é tradicionalmente uma maneira de familiarizar os operadores com os diversos tipos de cenários com os quais terão que lidar. Um cenário, consiste em uma situação do sistema durante a qual o operador deve tomar decisões com relação à atuação sobre o sistema, visando colocá-lo na situação normal de operação.

Por outro lado, a escolha de cenários de treinamento é muito importante, pois dá ao operador a oportunidade de vivenciar ocorrências no sistema que no dia-a-dia do treinamento clássico (acompanhando um operador experiente) talvez jamais tivesse oportunidade de presenciar. Há ainda a vantagem adicional do treinamento em pares, quando o operador experiente pode transferir seu conhecimento para o operador menos experiente.

Assim, o uso de um simulador que represente fielmente uma sala de controle, possibilita a imersão do operador e permite a criação de diversas situações com as quais os operadores precisam praticar, sem causar riscos ao sistema real, o qual deve ser mantido em constante operação.

A geração de cenários de treinamento na versão atual do simulador SimuLIHM, exige que sejam feitas alterações diretamente no código, de modo a representar cenas de ambientes de sala de controle de subestações. Assim, a representação dos objetos tridimensionais presentes no ambiente de simulação, tais como portas, maçanetas, painéis de diferentes tamanhos, display de alarmes, luzes de alerta, tipos de chave e equipamentos, entre outros, é feita através de alterações no código do simulador. Semelhantemente, a definição do estado inicial da planta e a evolução ao longo da simulação, com o disparo de eventos, deve ser realizada diretamente no código do simulador, a partir do uso de bibliotecas já existentes.

Visando simplificar esta tarefa, esse trabalho propõe a construção de um editor de cenários para o SimuLIHM. Este Editor de Cenários deverá facilitar a construção do ambiente de simulação (a cena) e da evolução dos eventos (cenário), possibilitando a seleção e posicionamento dos objetos que estarão presentes no ambiente 3D e, seu comportamento diante de eventos.

1.2 OBJETIVOS

Desenvolvimento de um Editor de Cenários para o simulador 3D – SimuLIHM, que representa, em realidade virtual, a operação de subestações, durante o treinamento de operadores.

O Editor deverá facilitar a construção de cenários a partir de uma interface com o usuário que possibilitará selecionar os objetos que fazem parte do cenário, sua

representação visual e seu comportamento, integrando-os em uma representação executável do cenário elaborado.

1.3 ESTRUTURA DO DOCUMENTO

A estruturação desse documento é a seguinte: no capítulo 2 é apresentado o simulador SimuLIHM, descrevendo suas funcionalidades e apresentando sua arquitetura; no capítulo 3 é abordado o tema da representação de mundos virtuais, por meio do uso de arquivos do tipo X3D e XML, as respectivas bibliotecas de objetos presentes em tais arquivos, representados na linguagem Java, e as ferramentas e plataformas utilizadas para auxiliar para visualização, edição e geração destes arquivos; no capítulo 4 é introduzido o software desenvolvido neste trabalho, o Editor de Cenários para o simulador SimuLIHM, apresentado as especificações, funcionalidades, aspectos de interface gráfica e validação do software; por fim, no capítulo 5, são apresentadas as considerações finais, com discussão dos resultados alcançados e propostas de trabalhos futuros relacionados ao simulador SimuLIHM e ao Editor de Cenários desenvolvido.

2 O SIMULADOR SIMULIHM

O Simulador SimuLIHM foi desenvolvido no Laboratório de Interfaces Homem Máquina (LIHM) com o objetivo de apoiar o treinamento de operadores de subestações elétricas. Utilizando-se de uma representação de uma sala de subestação em mundo virtual 3D, o simulador permite que o operador seja submetido a situações similares àquelas que ele pode vivenciar durante o exercício de sua função.

A representação em mundo virtual desse simulador visa a maior verossimilhança possível com o ambiente de uma sala de controle de subestação distribuidora de energia, possibilitando a atuação do operador sobre os equipamentos em dois níveis: remotamente, através de um sistema supervisório, e localmente através da atuação direta nos painéis de controle. Além disso, é possibilitada a interação entre operador e tutor, no mesmo ambiente.

2.1 FUNCIONALIDADES

A principal característica do SimuLIHM é a representação em mundo virtual tridimensional de uma típica sala de controle de subestação. Como citado em (FILHO, 2011), devido à diversidade tecnológica entre os equipamentos em uma subestação, a interação como os mesmos se dá de diferentes maneiras, podendo ser feita através do comando diretamente sobre o equipamento no campo ou através de painéis nas salas de controle da subestação, ou ainda através de supervisórios que permitem a interação remota sobre os equipamentos existentes nas subestações mais modernas.

No SimuLIHM a interação com os equipamentos da planta da subestação se dá através da representação dos painéis de controle e de um sistema supervisório. Os comandos do supervisório e dos painéis de controle são ambos associados a um modelo da planta, como será mostrado posteriormente nesse capítulo. O operador interage com o ambiente utilizando-se do teclado, para se movimentar, e do mouse, para operar sobre os dispositivos presentes nos painéis, tais como: chaves, botoeiras e sinalizadores, telefone, portas, computadores (utilizados para acionar o sistema supervisório), dentre outros

objetos ali representados. Na Figura 1 é ilustrada a representação de uma sala de controle em ambiente 3D no SimuLIHM.

Figura 1 - Representação de uma Sala de Controle em Ambiente Virtual

Fonte: o próprio autor.

A representação do mundo virtual nesse simulador é feita utilizando-se uma descrição em arquivo X3D, utilizado para representação de objetos 3D. A visualização dos objetos descritos nessa linguagem é feita através do *browser* Xj3D. Mais detalhes sobre esse tipo de arquivo e seu uso na representação do ambiente tridimensional serão expostos no capítulo seguinte.

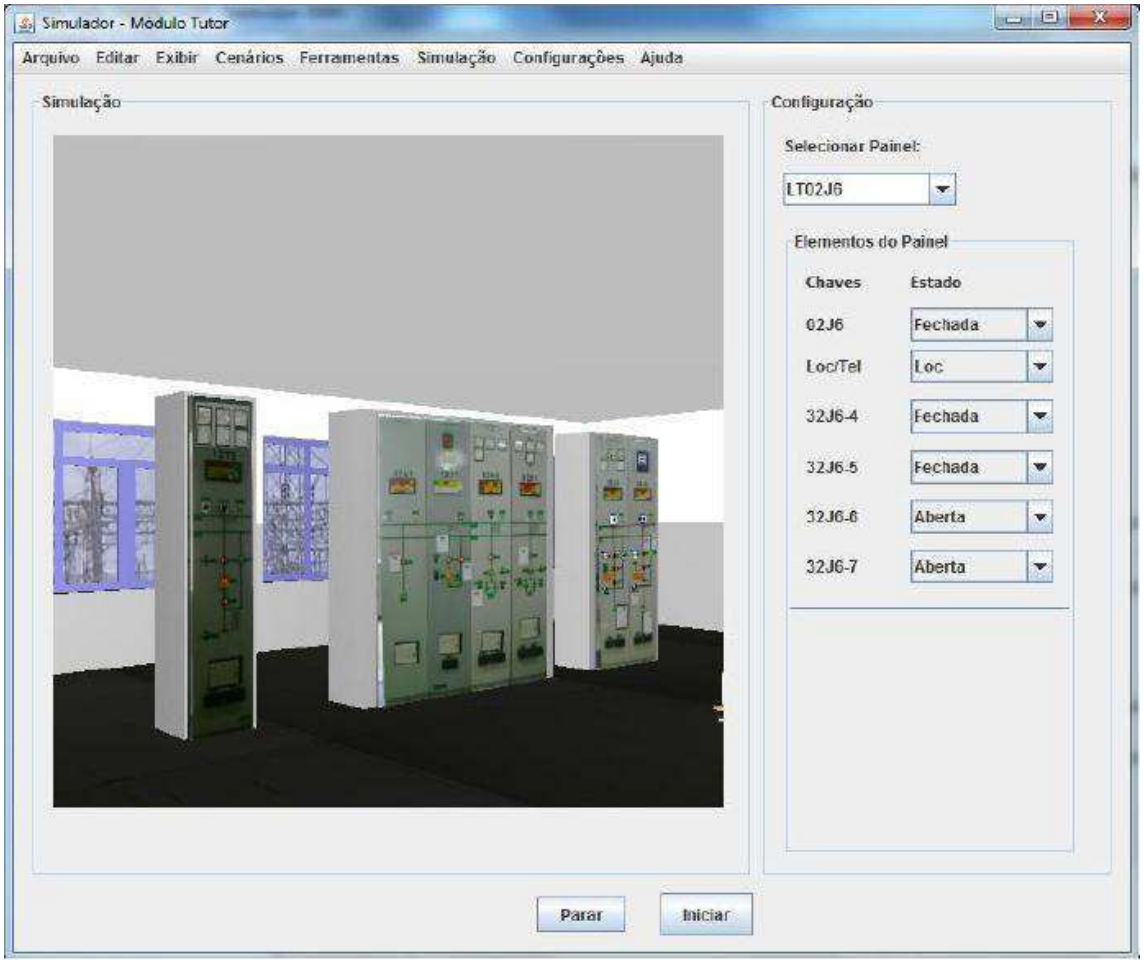
Além da representação da sala em um ambiente virtual e da possibilidade de interação com os painéis de controle, objetos 3D e com o sistema supervisório da planta, o simulador também permite que a simulação seja executada em dois ambientes: o ambiente do tutor e o ambiente do operador. Enquanto no ambiente do operador o usuário interage apenas com o mundo virtual (acionando dispositivos, abrindo portas, atendendo telefone ou utilizando o sistema supervisório), no ambiente do tutor o usuário não só visualiza a simulação 3D e as ações do operador, como também pode cadastrar treinamentos, criar e editar cenários presente no banco de dados e disparar eventos durante a simulação, tais como: abertura de chaves e ativação de alarmes (FILHO, 2011). Além disso, o tutor tem acesso a um arquivo contendo o histórico (*log) das* ações do operador durante a simulação, para auxiliar na análise de seu comportamento diante dos

eventos disparados. Na Figura 2 é ilustrada a tela do simulador referente à área de trabalho do Tutor.

O comportamento dos objetos 3D é regido pelos modelos construídos utilizando Rede de Petri Coloridas – CPN (*Coloured Petri Nets*). Foram construídos modelos para o comportamento dos objetos da sala de controle, presentes nos painéis de controle, e da planta da subestação. Esses modelos são parte do motor de simulação.

O simulador também possui uma conexão com um banco de dados que permite o armazenamento e a recuperação dos cenários utilizados, contendo as informações iniciais do sistema e os eventos disparados, além de informações da simulação e dos treinamentos realizados.

Figura 2 - Área do Tutor do SimuLIHM



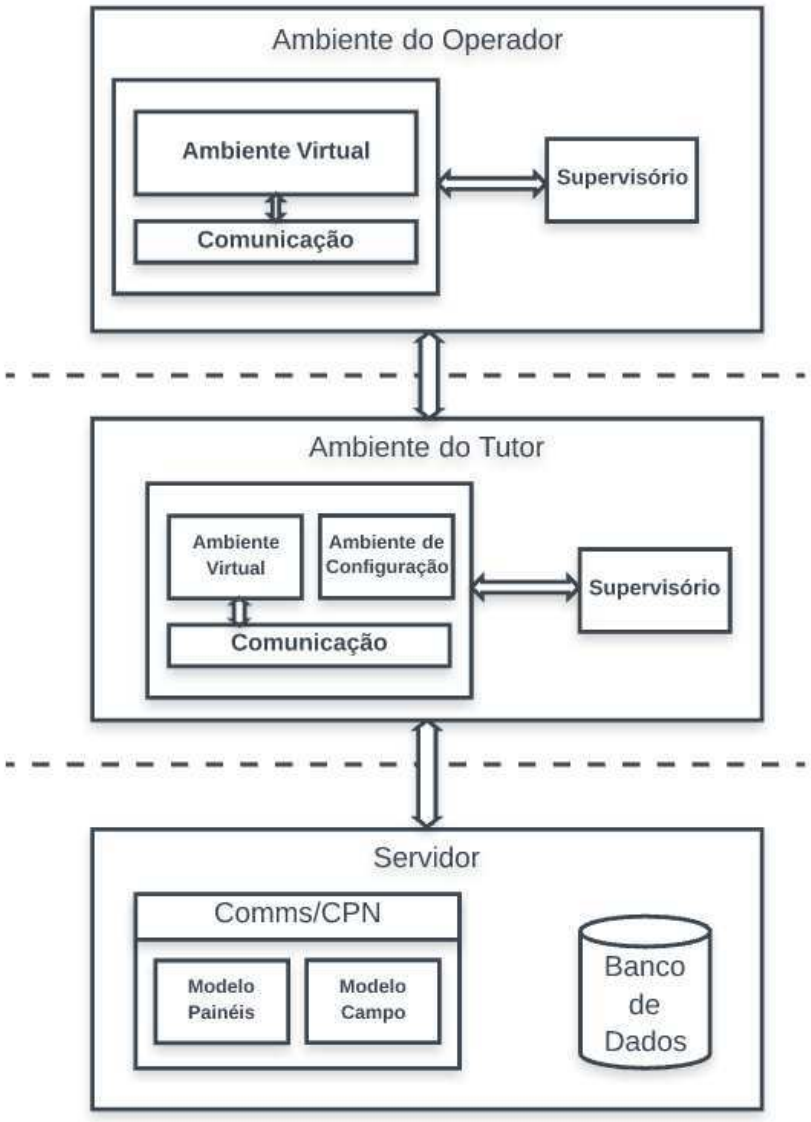
Fonte: (FILHO, 2011)

2.2 ARQUITETURA

A versão atual da arquitetura do simulador é fruto do trabalho de (NETTO, 2010), que resultou na proposição de uma arquitetura, e de (FILHO, 2011), com a integração do módulo de um supervisório. A arquitetura é do tipo cliente-servidor e é ilustrada na Figura 3.

A comunicação entre os módulos do simulador (Ambiente do Tutor, Ambiente do Operador e o Servidor) é feita utilizando o protocolo de comunicação TCP/IP (*Transmission Control Protocol/Internet Protocol*) (FILHO, 2011).

Figura 3 - Arquitetura Cliente-Servidor do Simulihm



Fonte: o próprio autor

A arquitetura também pode ser representada como uma arquitetura clássica modular contendo os módulos: de visualização, de comunicação além de modelos CPN, como apresentado nos trabalhos de (LUCENA, 2010) e de (LOPES, 2011). A Figura 4 ilustra essa representação da arquitetura.

Figura 4 - Representação alternativa da arquitetura do simulador

Fonte: o próprio autor

A comunicação entre as camadas da arquitetura do simulador se dá através do envio de mensagens pelo modo de comunicação. No ambiente do tutor, qualquer interação com o ambiente 3D origina uma mensagem que é enviada ao módulo de comunicação, e em seguida, ao motor de simulação. Se a interação for com o supervisório, a mensagem é enviada diretamente ao modelo do campo. No entanto, se a interação for feita com um objeto 3D da sala de controle, a mensagem é enviada primeiramente ao modelo dos painéis e só então é processada no modelo do campo. Após o processamento da mensagem pelo motor de simulação, é enviada uma mensagem de resposta para o módulo de comunicação, que atualiza o ambiente 3D e a representação apresentada pelo sistema supervisório.

No ambiente do operador a comunicação é feita de maneira semelhante, no entanto, a mensagem é enviada primeiramente ao ambiente do tutor e em seguida enviada ao motor de simulação.

2.3 CONSTRUINDO CENÁRIOS: A SITUAÇÃO ATUAL

A construção de cenários de simulação na versão mais atual do simulador ainda não possui uma interface ergonômica que venha a facilitar a edição do ambiente 3D, com relação aos objetos que serão inseridos no ambiente e ao estado inicial de: chaves, botoeiras, quadro de eventos, sinalizadores e mostradores, presentes nos painéis de controle. Atualmente, para que seja criado um objeto 3D no ambiente virtual, é necessário que sejam feitas modificações no código do arquivo X3D do cenário, manualmente. Ou seja, é necessário que se abra o arquivo X3D com algum editor de código de arquivo XML, e que sejam inseridos trechos de código relativos ao objeto 3D que se de deseja inserir no ambiente, modificando também no código os parâmetros correspondentes à representação visual desses objetos (rotação, translação, rótulo, posição inicial).

Figura 5 - trecho de código de inserção manual de uma chave em um painel de controle

```
113
114             <!-- Os componentes do painel devem ser inseridos aqui-->
115             <Transform DEF = "PainelDeControle3D_ModeloA_Componente">
116
117         <Group DEF = "ChaveGPG_ModeloA_Id">
118           <Transform DEF = "ChaveGPG_ModeloA_Translacao" translation="0 0 0" rotation="0.0 0.0 0.0 0">
119
120             <Transform translation="0 0 0" rotation="1.0 0.0 0.0 0">
121
122                 <Transform DEF = "ChaveGPG_ModeloA_Rotacao"  translation="0 0 0.126" center="0.0 0.0 0.0" rotation="0 0 1 0">
123                 <!-- Chave na posição horizontal -> rotation="0 0 1 0"
124                     Chave na posição vertical -> rotation="0 0 1 1.57" -->
125
126                     <Shape >
127                         <Appearance >
128                             <Material diffuseColor="0 0 0"/>
129                         </Appearance>
130                     <Box  size="0.257089 0.06 0.035"/>
131                     </Shape>
132                 </Transform>
133
134                 <Transform translation="0 0 0" rotation="1.0 0.0 0.0 1.57">
135                 <Shape>
136                     <Appearance >
137                         <Material DEF = "ChaveGPG_ModeloA_Cor" diffuseColor="1 1 0"/>
138                         <!-- VERMELHO -> "1 0 0"
139                              VERDE    -> "0 1 0"
140                              AMARELO -> "1 1 0"   -->
141                     </Appearance>
142                 <Cylinder  height="0.25" radius="0.15"/>
143                 </Shape>
144               </Transform>
145           </Transform>
```

Fonte: o próprio autor

Na situação atual, a construção de cenários, ou seja, dos eventos que serão disparados durante a simulação de treinamento, também é feita sem o uso de uma interface ergonômica. Este trabalho visa a construção de uma interface que possibilite a edição da cena 3D do ambiente de simulação além dos eventos que serão disparados durante a mesma, ou seja, as animações da simulação.

3 REPRESENTAÇÃO DE MUNDOS VIRTUAIS

Uma das características mais importantes do SimuLIHM é oferecer a possibilidade de representar o ambiente de uma sala de controle de subestação em um mundo virtual tridimensional. A representação em mundo virtual tem como objetivo a imersão do operador que irá interagir com objetos e personagens, simulando o comportamento de um ambiente real (LOPES, 2011).

Nas primeiras versões do simulador foi utilizada a linguagem VRLM para construção do mundo virtual. No entanto, nas versões posteriores foi utilizada a linguagem X3D para a descrição e construção dos objetos tridimensionais que fariam parte da representação, em mundo virtual, da sala de controle de uma subestação.

Neste capítulo serão abordados as linguagens e os recursos utilizados, tanto no projeto do SimuLIHM como neste trabalho, para possibilitar a manipulação de objetos tridimensionais e representar o ambiente de uma sala de controle de subestação em um mundo virtual.

3.1 ARQUIVOS XML E X3D

A linguagem XML, sigla para *Extensible Markup Language*, é uma linguagem de marcação criada com o objetivo de simplificar a estrturação de um texto, tornando-o sintaticamente distinguível e se tornando uma alternativa simples para o armazenamento de dados e informações. Um documento XML é fácil de ser entendido pelo programador e é também facilmente interpretado por um programa de computador, devido à sua organização semelhante a uma árvore, com *tags* que definem uma hierarquia de dados, com raiz e nós que contém informações. Na Figura 6 é ilustrado um extrato de código XML, utilizado na descrição de produtos, contendo informações como: descrição (*description*), preço (*price*) e quantidade (*quantity*).

A linguagem X3D, desenvolvida pelo Web3D Consortium, descreve um formato de arquivo (.x3d) utilizado para representar e realizar a comunicação entre cenas e objetos 3D, utilizando a linguagem XML. É considerada a sucessora do VRML (*Virtual Reality Modeling Language*), na representação e descrição de objetos tridimensionais.

O fato do arquivo X3D ser baseado na estrutura de um arquivo XML o torna mais fácil de ser lido e manipulado por programas de computador, possibilitando modificar parâmetros de objetos tridimensionais através de linhas de códigos em qualquer linguagem de programação que possua bibliotecas para manipulação de arquivos em XML. Na Figura 7 é ilustrado um trecho de código X3D concebido para descrever uma chave GPG (Giro Pressão Giro). Algumas partes do código foram ocultadas para uma melhor visualização da figura.

Figura 6 - Exemplo de código XML

```
<?xml version="1.0"?>
<items>
    <item>
        <product>
            <description>Ink Jet Refill Kit</description>
            <price>29.95</price>
        </product>
        <quantity>8</quantity>
    </item>
    <item>
        <product>
            <description>4-port Mini Hub</description>
            <price>19.95</price>
        </product>
        <quantity>4</quantity>
    </item>
</items>
```

Fonte: Livro Big Java (Horstmann, 2004)

Figura 7 - Código X3D de descrição de uma Chave GPG

```
<?xml version="1.0" encoding="iso-8859-1"?>

<X3D profile="Immersive" version="3.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xsd:noNamespaceSchemaLocation=" http
    <head>
    <!-- Chave GPG -->
    <Scene>
        <Group DEF = "ChaveGPG_ModeloA_Id">
            <Transform DEF = "ChaveGPG_ModeloA_Translacao" translation="0 0 0" rotation="0.0 0.0 0.0 0">
                <Transform translation="0 0 0" rotation="1.0 0.0 0.0 0">
                    <Transform DEF = "ChaveGPG_ModeloA_Rotacao"  translation="0 0 0.126" center="0.0 0.0 0.0" rotation="0 0 1 0">
                        <!-- Chave na posição horizontal -> rotation="0 0 1 0"
                        Chave na posição vertical -> rotation="0 0 1 1.57" -->
                        <Shape >
                    </Transform>
                    <Transform translation="0 0 0" rotation="1.0 0.0 0.0 1.57">
                        <Shape>
                            <Appearance >
                                <Material DEF = "ChaveGPG_ModeloA_Cor" diffuseColor="1 1 0"/>
                            <!-- VERMELHO -> "1 0 0";VERDE    -> "0 1 0";AMARELO -> "1 1 0"-->
                            </Appearance>
                            <Cylinder  height="0.25" radius="0.15"/>
                        </Shape>
                    </Transform>
                </Transform>
                <!-- placa com nome -->
                <Transform translation ="0 -0.4 0" rotation="0.0 0 0 0">
            </Transform>
            <TouchSensor DEF = "ChaveGPG_ModeloA_Sensor"/>
        </Group>
    </Scene>
</X3D>
```

Fonte: Documentação do LIHM

3.2 A LINGUAGEM JAVA

Java é uma linguagem de programação, baseada em classes e orientada a objetos que foi desenvolvida com objetivo de alcançar o mínimo de dependências de implementação, permitindo que programas nessa linguagem sejam executáveis em qualquer plataforma. Neste trabalho, a linguagem Java foi utilizada para desenvolvimento da interface gráfica do software desenvolvido, bem como na definição dos modelos básicos e na comunicação entre módulos.

Além das utilidades listadas acima, a linguagem Java foi utilizada para manipulação de arquivos X3D, modificando parâmetros de objetos, adicionando componentes ao cenário 3D, a partir de modelos de objetos X3D utilizados na construção do ambiente virtual. Esses modelos foram definidos em trabalhos anteriores realizados pela equipe do LIHM. Para que seja possível o uso de arquivos XML e a sua manipulação é necessário que sejam importados os seguintes pacotes de bibliotecas: *org.w3c.dom, javax.xml* e *org.xml*. Além dessas bibliotecas, o pacote *org.web3d.x3d.sai* também teve de ser importado para visualização através de um *browser* do ambiente tridimensional definido pelo arquivo X3D gerado.

A Figura 8 mostra um exemplo de um método escrito em Java para edição de um arquivo X3D, como se fosse em XML. Esse método foi criado para a criação de uma Chave GPG. Os métodos *getElementsByTagName()*, e *setAttribute()* são exemplos de métodos da biblioteca para manipulação de documentos em XML.

Figura 8 - Método Java para criação de uma chave GPG

```
public Element criarChaveGPG(Chave chave) {
    //SET o nome da Chave no modelo
    NodeList listaDeGrupos = root.getElementsByTagName("Group");
    ((Element) listaDeGrupos.item(0)).setAttribute("DEF", chave.getRotulo());
    //SET a translacao da Chave no modelo
    NodeList listaDeTransform = root.getElementsByTagName("Transform");
    ((Element) listaDeTransform.item(0)).setAttribute("translation", chave.getTranslacao());
    ((Element) listaDeTransform.item(0)).setAttribute("DEF", chave.getRotulo() + "_Posicao");
    //SET a rotacao da Chave no modelo (Posição)
    ((Element) listaDeTransform.item(2)).setAttribute("rotation", chave.getRotacao());
    ((Element) listaDeTransform.item(2)).setAttribute("DEF", chave.getRotulo() + "_Rotacao");
    //SET a cor da Chave no modelo
    NodeList listaDeMaterial = root.getElementsByTagName("Material");
    ((Element) listaDeMaterial.item(1)).setAttribute("diffuseColor", chave.getCor());
    ((Element) listaDeMaterial.item(1)).setAttribute("DEF", chave.getRotulo() + "_Material");
    //SET o rotulo da Chave no modelo
    NodeList listaDeText = root.getElementsByTagName("Text");
    ((Element) listaDeText.item(0)).setAttribute("DEF", chave.getRotulo() + "_Rotulo");
    ((Element) listaDeText.item(0)).setAttribute("string", chave.getRotulo());
    //SET o Sensor da Chave no modelo
    NodeList listaDeSensor = root.getElementsByTagName("TouchSensor");
    ((Element) listaDeSensor.item(0)).setAttribute("DEF", chave.getRotulo() + "_Sensor");

    return (Element) root.getElementsByTagName("Group").item(0);

}
```

Fonte: o próprio autor.

3.3 FERRAMENTAS E PLATAFORMAS

Para o desenvolvimento deste trabalho foram utilizadas diversas plataformas que auxiliaram: na tarefa de representação dos mundos virtuais, no desenvolvimento de interfaces gráficas e na verificação dos modelos de objetos 3D.

Como citado na seção anterior, é necessária a utilização de um *browser* para a visualização do ambiente virtual, descrito no formato X3D. A utilização de um *browser* também possibilita a interação com o ambiente. Nesse projeto foi utilizado como *browser* o Xj3D, sendo também utilizado durante o estudo dos arquivos X3D. Para a utilização desse *browser* através de um programa Java é necessário o uso de uma API SAI (Scene Access Interface) que possui métodos que permitem acessar a cena 3D e definir o comportamento dos objetos ali representados (FILHO, 2011).

Para desenvolvimento do código em Java e criação da interface gráfica no desenvolvimento deste trabalho, foi utilizada a plataforma de desenvolvimento NetBeans IDE. Essa plataforma oferece recursos que tornam a criação de uma interface gráfica uma tarefa mais simples, com a possibilidade de arrastar painéis, botões, caixas de texto, dentre outros, e editá-los. Além disso, essa IDE apresenta uma aba de configuração que permite que bibliotecas sejam facilmente importadas para o projeto.

Mais um recurso que foi utilizado nesse projeto foi o editor de ontologia Protégé. O Protégé é um *framework* que permite a criação e edição de modelos ontológicos e visualização dos mesmos. Resumidamente, ontologia, na ciência da computação, é um modelo de dados utilizado para a representação de conceitos. No projeto do simulador foi desenvolvida uma ontologia para representar os objetos 3D, as animações e as interações com os mesmos em um modelo de dados, permitindo que informações sobre os cenários de simulações fossem salvos localmente. Neste trabalho, o Protégé foi utilizado apenas para a visualização dos modelos de dados de objetos e cenários 3D, com o objetivo de auxiliar na concepção da interface gráfica.

4 EDITOR DE CENÁRIOS PARA O SIMULIHM

Neste capítulo serão apresentadas as etapas de concepção e desenvolvimento de um Editor de Cenários de treinamento para uso no SimuLIHM. O objetivo é facilitar a geração do ambiente virtual tridimensional de uma sala de controle e de eventos que constituem um cenário de treinamento no simulador. Serão apresentadas as funcionalidades desse Editor, a especificação UML (diagrama de casos de uso e diagrama de classes) e a interface que foi concebida para este trabalho. Por fim, será apresentado processo de validação desse trabalho, ilustrando o passo a passo adotado na geração de um cenário.

4.1 FUNCIONALIDADES DO EDITOR

O Editor de Cenários apresenta um conjunto de funcionalidades que permitem a criação de um cenário de treinamento para o SimuLIHM. Para a construção de um cenário são realizadas duas tarefas: a construção da cena 3D representativa do ambiente da sala de controle e a construção do cenário de treinamento. Um cenário se refere aos eventos que serão disparados durante a simulação.

Para a primeira tarefa, a construção da cena 3D do ambiente, o software possibilita ao usuário criar objetos 3D e definir seus parâmetros através do uso de elementos da interface, como botões, caixas de texto e caixas de combinação. Na versão atual desse software foram criadas telas apenas para os objetos que possuíam um modelo padrão na Biblioteca de Modelos 3D do LIHM. São esses objetos: Amperímetro, Botoeira, Cadeira, Chave (43T, GPG e CLT), Mesa, Painel de Controle, Quadro de Eventos, Sinalizador, Telefone e Sala de Controle.

Após a criação dos objetos, é possível visualizá-los em um menu com estrutura de árvore (elemento *jTree*) e selecioná-los para edição ou simplesmente para verificar os parâmetros do objeto. Ao término da criação dos objetos, é possível gerar e salvar o arquivo X3D correspondente ao ambiente 3D, o qual estará povoado com os abjetos criados pelo usuário.

Outra funcionalidade relativa à criação da cena 3D é a visualização do ambiente 3D gerado. Isso é feito com a utilização do *browser* Xj3D para abrir o arquivo X3D gerado. Essa visualização pode ser feita durante a criação do objeto e após a geração do arquivo X3D do cenário.

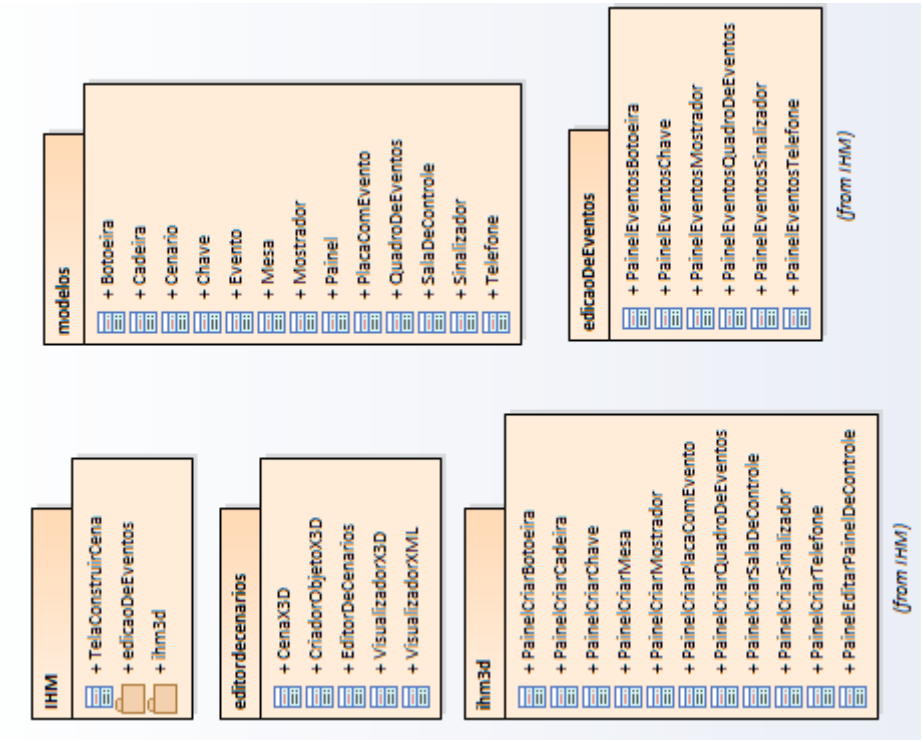
A segunda tarefa, relativa à geração dos eventos para cada objeto é feita após a criação dos objetos. O usuário pode definir eventos para cada objeto, tais como o toque de um telefone, a abertura de uma chave ou o acionamento de um sinalizador. Essas informações são guardadas no objeto durante a execução do programa. No entanto, não foi desenvolvido um módulo para salvamento das informações dos eventos em uma ontologia ou no banco de dados, pois estava fora do escopo proposto.

As funcionalidades do Editor de Cenários ficarão mais claras na seção 4.3, na qual serão mostradas as telas relativas às funcionalidades que foram apresentadas nesta seção.

4.2 ESPECIFICAÇÃO UML DO EDITOR DE CENÁRIOS

Do ponto de vista estrutural, a organização do projeto consistiu da separação de pacotes de interface e modelos, e de um pacote principal, com classes gerenciadoras de arquivo XML e X3D e classe principal. Foi construído um pacote principal de interface "IHM", onde se localiza a classe da Tela Inicial, com dois subpacotes: um para os painéis relativos à criação de objetos e, outro para os painéis relativos à criação dos eventos para cada objeto. Além disso, foi criado um pacote para os modelos (chaves, painéis, botoeiras, etc.) e um pacote para os gerenciadores de arquivos e a classe principal. O diagrama de pacotes do projeto é mostrado na Figura 9.

Figura 9 - Diagrama de Pacotes



Fonte: o próprio autor.

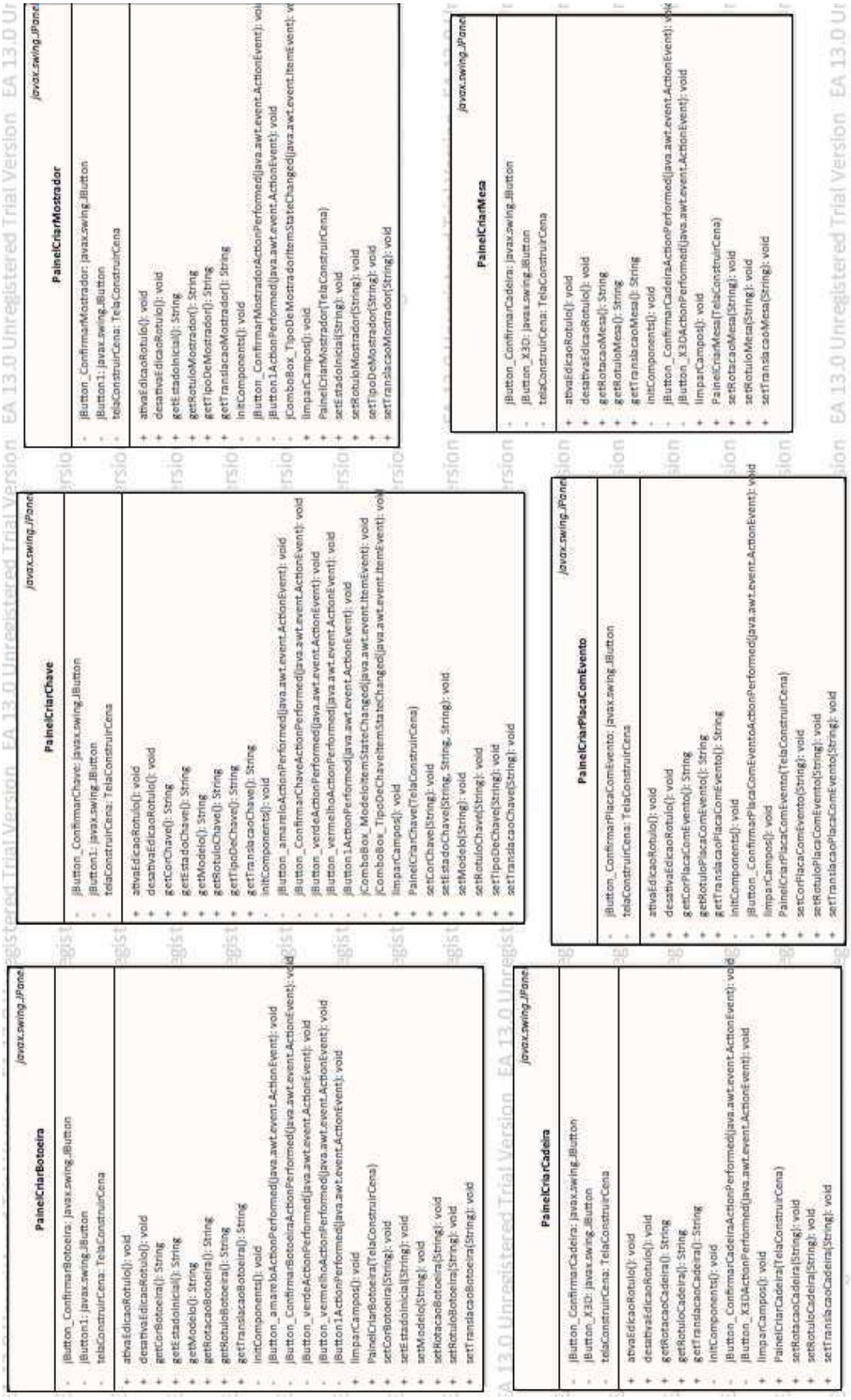
Devido à grande quantidade de classes, tanto de interface como de modelos, o diagrama de classes não pode ser colocado em uma só imagem. Dessa forma, decidiu-se separar o diagrama de classes de cada pacote separadamente.

Na Figura 10 é apresentado o diagrama de classes do pacote IHM. Nas imagens Figura 11 e Figura 12 é mostrado o diagrama de classes do subpacote IHM.ihm3d. Na Figura 13 é apresentado o diagrama de classes do subpacote IHM.edicaoDeEventos. Nas imagens Figura 14 e Figura 15 é mostrado o diagrama de classes do pacote modelos. Por fim, na Figura 16 é mostrado o diagrama de classes do pacote editorDeCenarios. Os diagramas de classe foram gerados utilizando-se o software *Enterprise Architect*.

Figura 10 - Diagrama de classes para o pacote IHM

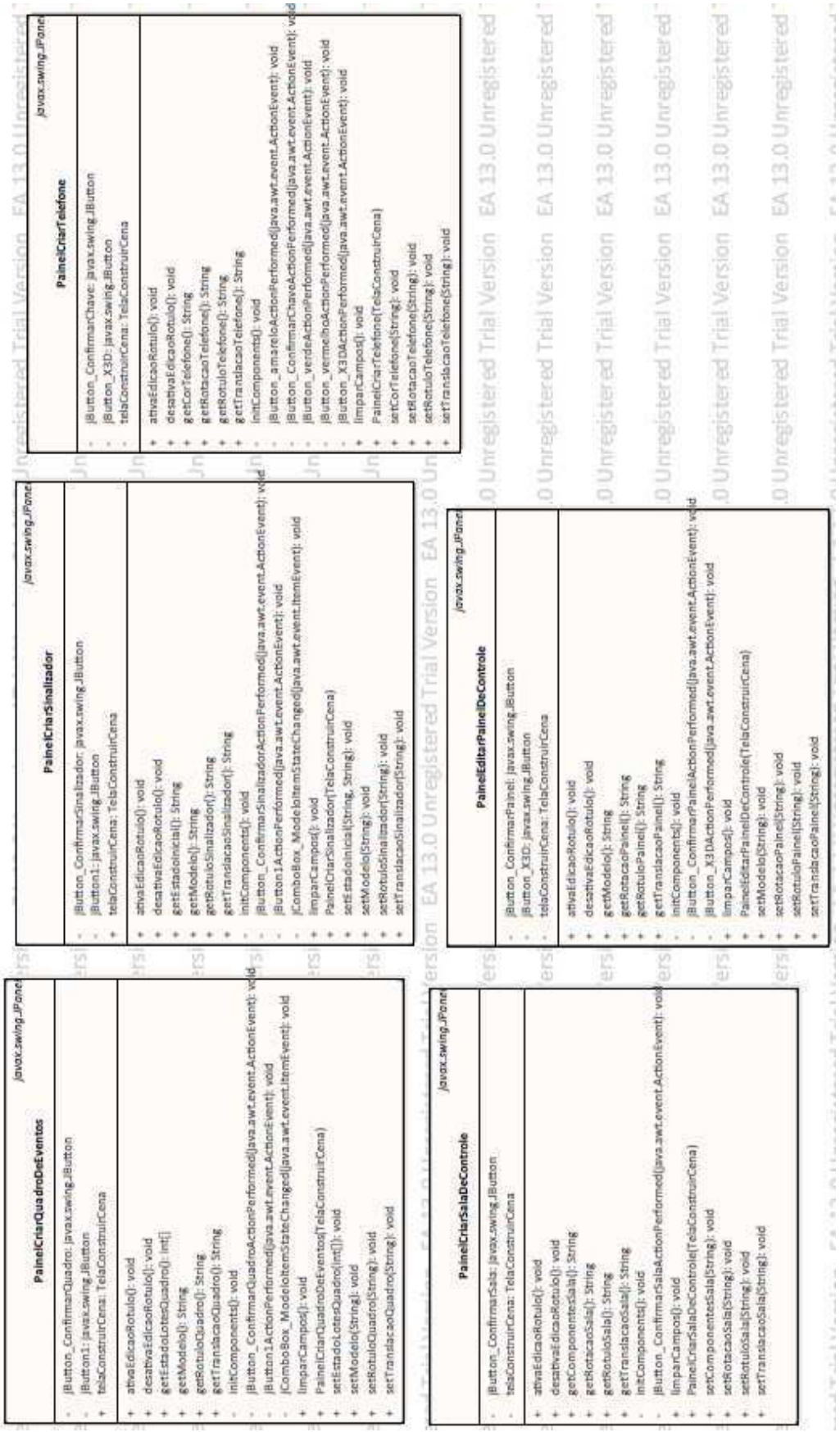
Fonte: o próprio autor.

Figura 11 - Diagrama de Classes do pacote IHM.ihm3d parte 1



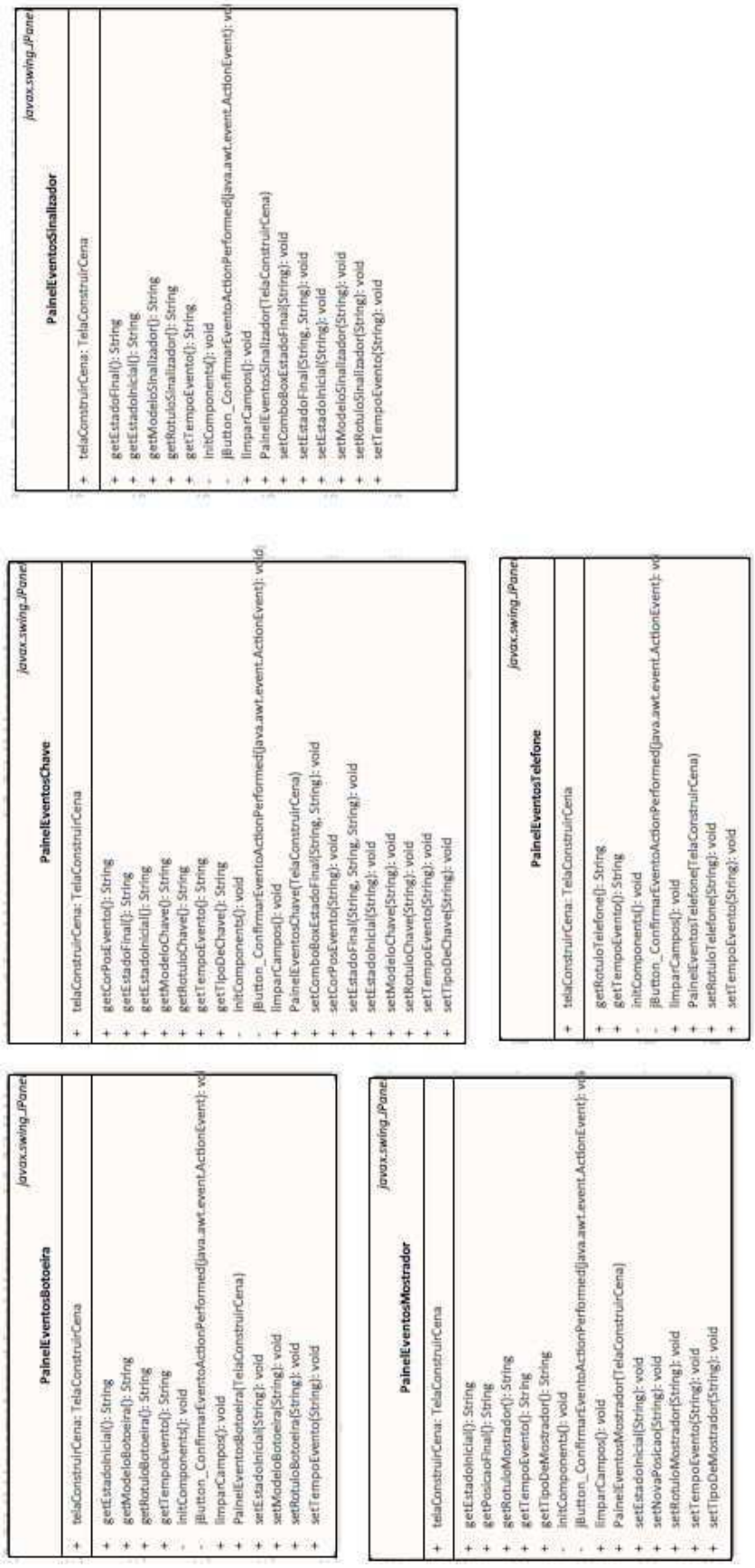
Fonte: o próprio autor.

Figura 12 - Diagrama de Classes do pacote IHM.ihm3d parte 2



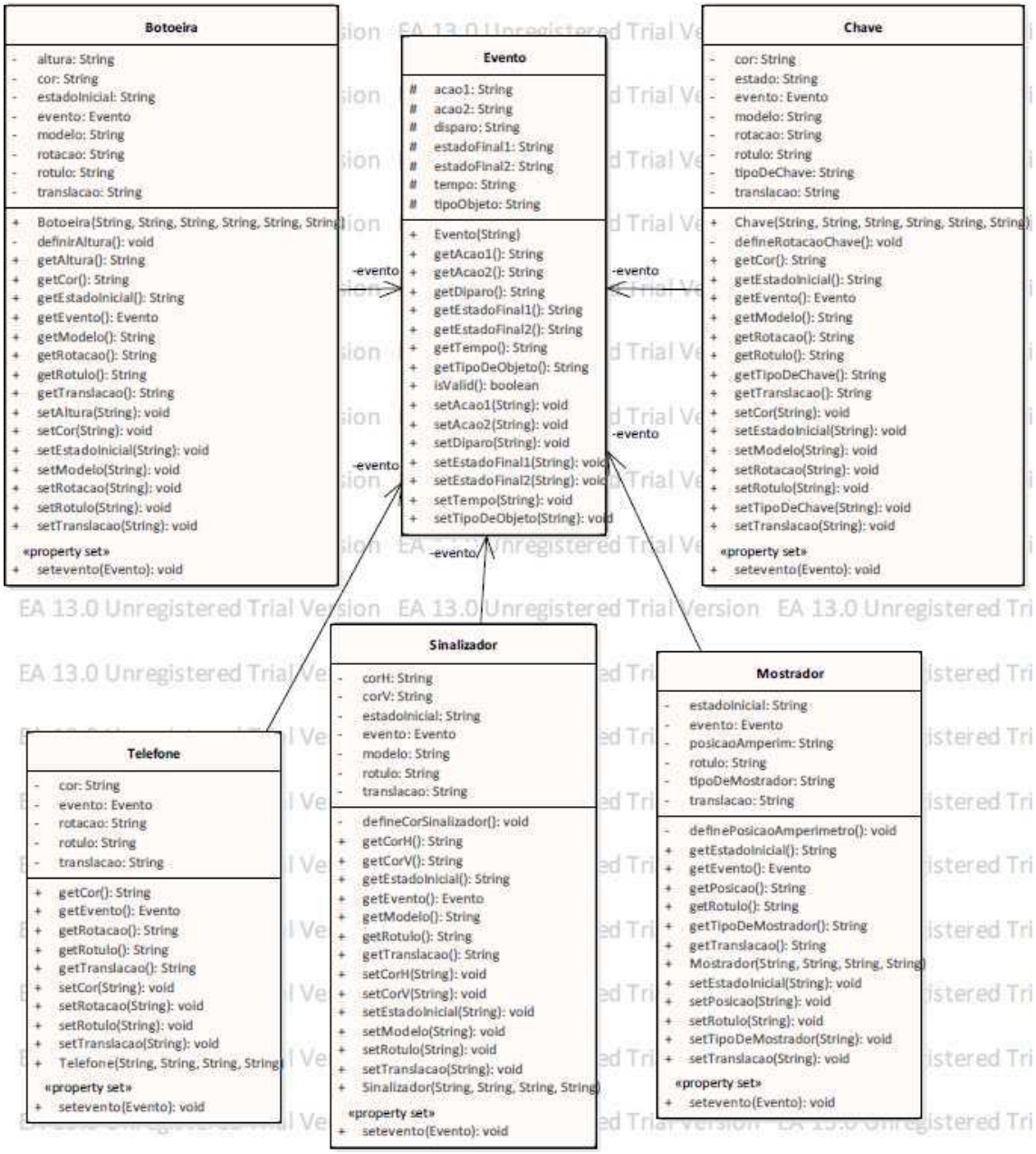
Fonte: o próprio autor.

Figura 13 - Diagrama de Classes do Pacote IHM.edicaoDeEventos



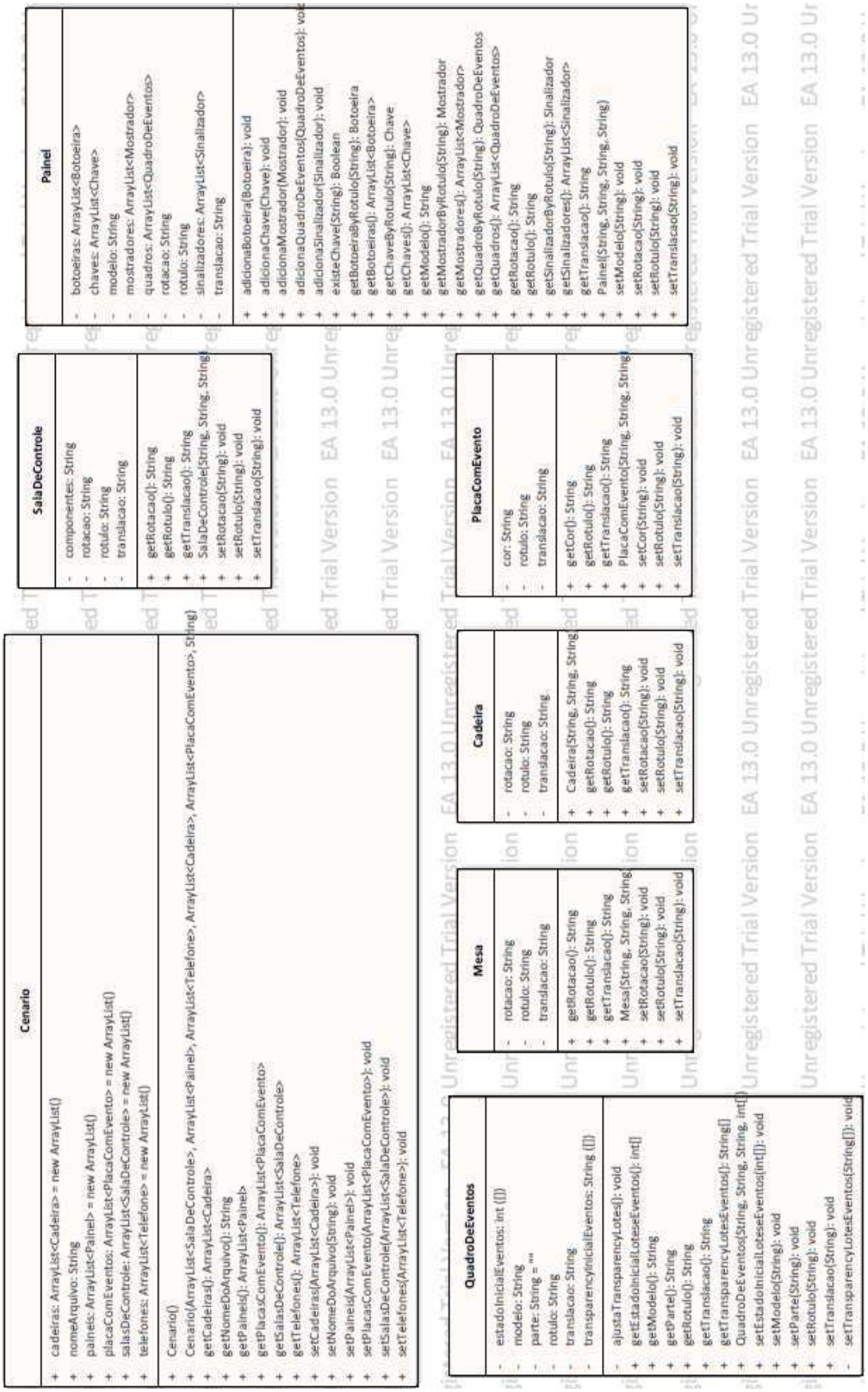
Fonte: o próprio autor.

Figura 14 - Diagrama de classes do pacote modelos parte 1



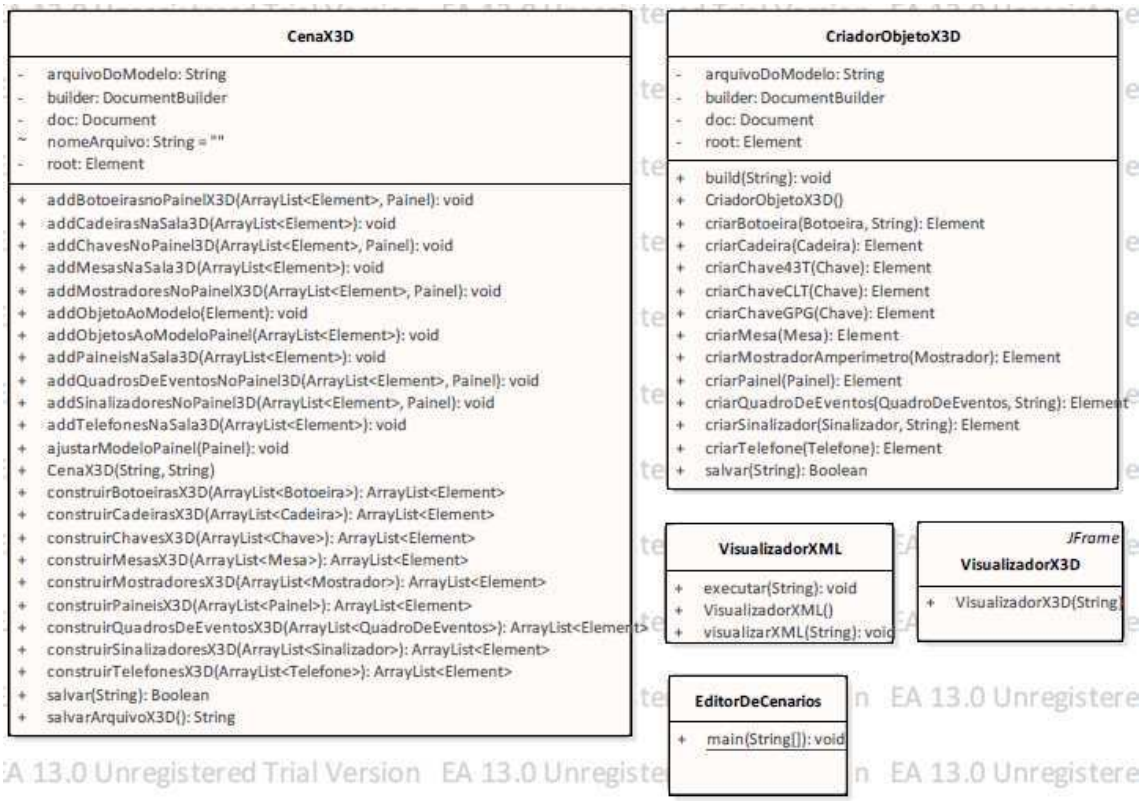
Fonte: o próprio autor.

Figura 15 - Diagrama de classes do pacote modelos parte 2



Fonte: o próprio autor.

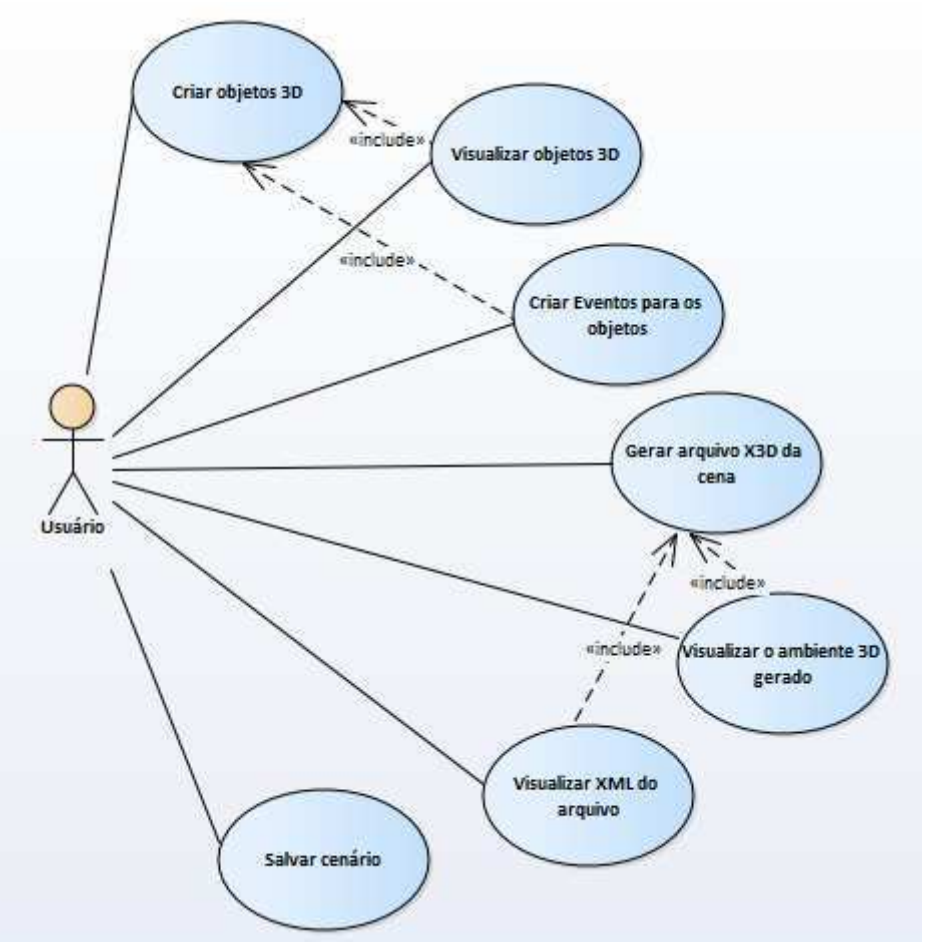
Figura 16 - Diagrama de Classes do pacote editorDeCenarios



Fonte: o próprio autor.

Em seguida, para complementar a especificação UML do Editor de Cenários, agora no ponto de vista comportamental, foi elaborado o Diagrama de Casos de Uso, de acordo com as funcionalidades, como mostrado na Figura 17.

Figura 17 - Diagrama de Casos de Uso



Fonte: o próprio autor.

4.3 INTERFACE COM O USUÁRIO DO EDITOR

Nessa seção são apresentados os objetivos que levaram à concepção da interface gráfica do Editor de Cenários. Em seguida serão apresentadas as principais telas que compõem a interface com o usuário do editor e as funcionalidades associadas a cada uma.

4.3.1 OBJETIVOS DA INTERFACE

Numa discussão inicial sobre como deveria ser construída a interface do Editor de Cenários, foram discutidos alguns requisitos que deveriam ser cumpridos pela interface. Esses objetivos eram:

- Criação de telas para criação e edição de objetos. Essas telas deveriam conter elementos de interface que coletassem informações de cada objeto

a ser criado. Cada objeto possui parâmetros diferentes, daí a necessidade de telas diferentes para cada um. Por exemplo, uma botoeira tem como parâmetros: rótulo, rotação, translação, cor, modelo 3D e estado inicial (pressionado ou não pressionado).

- Possibilidade de visualização dos objetos criados numa lista e seleção dos mesmos para edição.
- Criação de telas para criação e edição de eventos relacionados aos objetos criados. Essas telas deveriam conter elementos de interface que coletassem informações sobre eventos temporais para cada um os objetos. Por exemplo, uma chave GPG tem como possíveis eventos a mudança de cor (verde, vermelho ou verde) e a mudança de posição (aberta ou fechada).
- Salvar o arquivo X3D relativo à cena 3D do cenário.
- Visualização de objetos 3D durante sua criação e edição, para facilitar a escolha dos parâmetros e visualização de resultados da edição.
- Visualização do cenário 3D após a geração do arquivo X3D.
- Salvar as informações da cena 3D e dos Eventos criados em uma ontologia ou banco de dados.

Com exceção do salvamento das informações em uma ontologia ou banco de dados, todas os objetivos foram implementados em termos de funcionalidades do Editor.

4.3.2 Apresentação das Telas do Editor

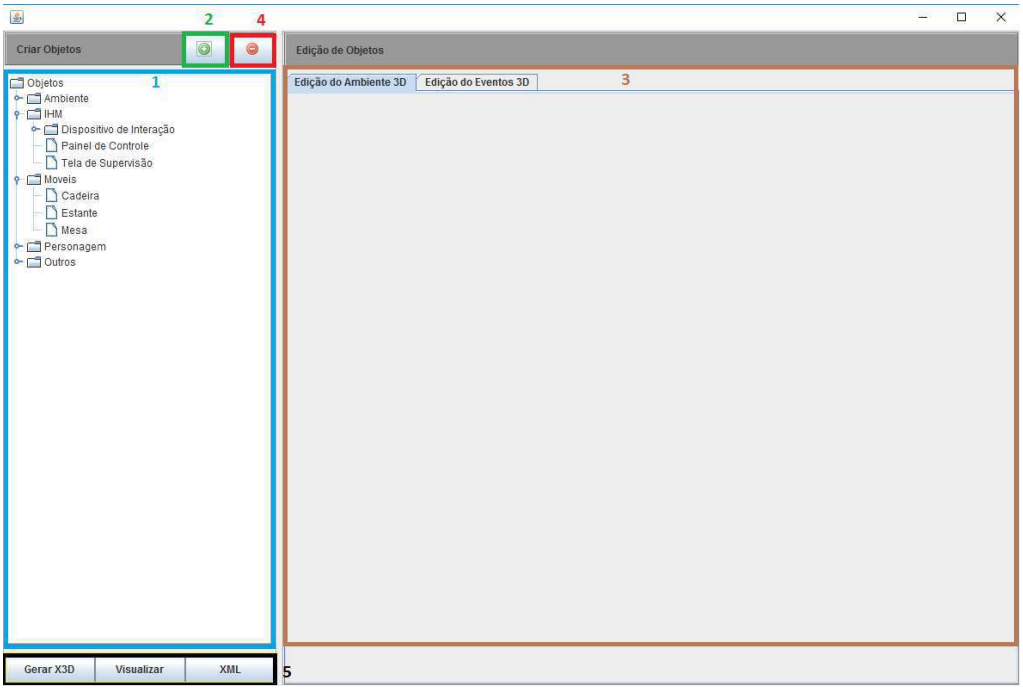
Nessa seção do relatório, serão apresentadas as telas do Editor de Cenários que cumprem as funcionalidades apresentadas na seção 4.2 e os objetivos expostos na seção 4.3.1. Serão mostradas apenas algumas das telas mais utilizadas do Editor, com o objetivo de ilustrar o funcionamento da interface.

Primeiramente será apresentada a tela inicial do Editor. Essa tela é a tela principal do software e será mostrada durante toda a execução do Editor. Na Figura 18 é apresentada a tela inicial com algumas indicações de partes importantes dessa interface.

Como pode-se ver na figura, foram desenhados retângulos enumerados de diferentes cores para indicar partes dessa interface. O significado de cada uma dessas indicações está enumerado abaixo:

1. **Lista Hierárquica de Objetos:** Essa lista, implementada com o uso do elemento *jTree*, disponibilizado pelo Java, possibilita listagem hierárquica dos tipos de objetos que podem ser criados. Após a criação de um objeto, o mesmo é listado nessa área, abaixo do nó da árvore referente ao seu tipo.
2. **Botão para adicionar um novo objeto:** para que se crie um novo objeto é necessário que o usuário selecione o tipo de objeto na Lista de Objetos e clique no botão (*jButton*) indicado. Dessa forma, será criado um painel com as informações para criação desse novo objeto.
3. **Área de Edição de Objetos:** área nas quais são inseridos os painéis referentes a cada objeto a ser criado. Implementada utilizando-se um painel com abas (*jTabbedPane*). Ao selecionar a aba **Edição do Ambiente 3D** o usuário terá acesso ao painel de edição do objeto selecionado na lista. Selecionando a aba **Edição do Eventos 3D** o usuário terá acesso aos eventos relacionados ao objeto selecionado.
4. **Botão para remover um novo objeto:** ao clicar nesse botão, o objeto selecionado na lista de objetos é removido, apagando suas informações sobre os parâmetros 3D e sobre os eventos relacionados.
5. **Menu do cenário:** Nesse menu, formado por 3 botões, estão presentes as funcionalidades de geração e salvamento o arquivo X3D do cenário, com os objetos criados e listados, visualização do ambiente 3D do cenário criado e visualização do código X3D através do uso de uma aplicação externa de edição de documento XML.

Figura 18 - Tela Inicial do Editor de Cenários



Fonte: o próprio autor.

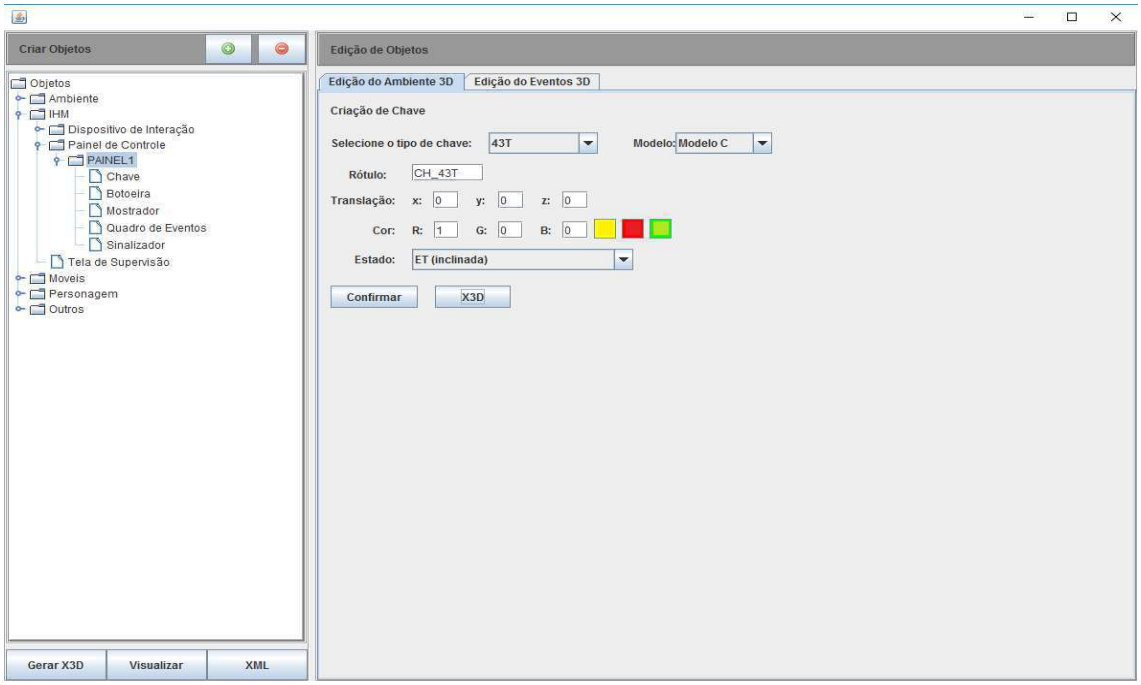
Para que sejam criados os objetos selecionados na Lista de objetos é necessário que sejam criados painéis específicos para cada tipo de objeto, de modo a obter as informações necessárias para essa tarefa. Para a concepção dessa interface, foram analisados os parâmetros 3D de cada objeto e escolhidos elementos de interface para representar esses parâmetros. Os parâmetros 3D e seus respectivos elementos de interface são listados abaixo:

- Rótulo (ou nome) do objeto: representado por uma caixa de texto *jTextField*.
- Rotação: representado por 4 caixas de texto, referentes aos quatro parâmetros da rotação 3D do objeto (x, y, z e ângulo em radianos).
- Translação: representado por 3 caixas de texto, referentes aos três parâmetros da translação (posição) 3D do objeto (x, y, z).
- Cor: representado por 3 caixas de texto, referentes aos três parâmetros da cor 3D do objeto (RGB). São também utilizados 3 botões para auto preenchimento do campo relativo à cor do objeto, com cores padrão pré-definidas (vermelho, amarelo e verde).
- Modelo: representado por uma caixa de seleção (*jComboBox*), no caso de objetos que possuem mais de um modelo de representação 3D.

- Tipo: No caso de chaves e mostradores, podem existir mais de um tipo desses objetos. No caso de chaves, temos chaves GPG, CLT, 43T, Punho e GG, e no caso de mostradores, temos amperímetros, voltímetros, mostradores analógicos e mostradores digitais. Esse parâmetro é representado por uma caixa de seleção na interface.
- Estado Inicial: representado por uma caixa de seleção, para listar os possíveis estados do objeto. No caso de uma chave GPG, por exemplo, tem-se dois estados possíveis: aberta e fechada.

Nesses painéis, ainda temos mais dois botões que constituem a interface: um botão de confirmar e um botão de visualizar o objeto 3D. O botão de confirmar vai efetuar a finalização da criação ou edição do objeto selecionado. Já o botão de visualizar (X3D) é utilizado para a visualização do objeto 3D que está sendo criado ou editado, utilizando o *browser* Xj3D. Para uma melhor visualização desses painéis de criação e edição de objetos 3D, pode-se observar a Figura 19.

Figura 19 - Painel de Criação e Edição de uma Chave

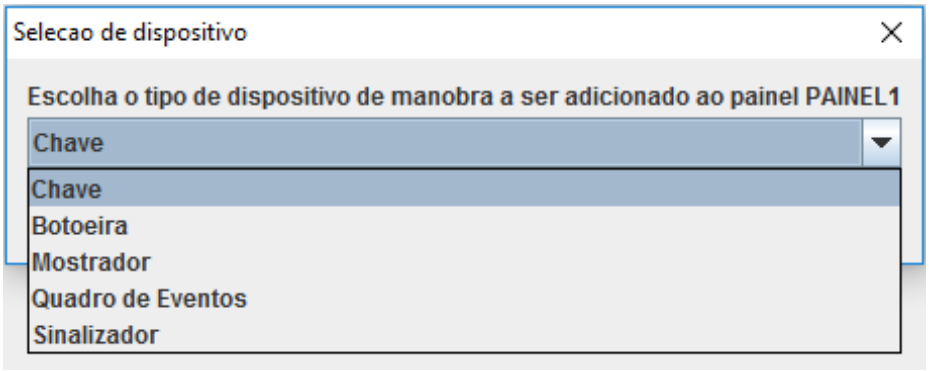


Fonte: o próprio autor.

Para criação de uma chave, ou qualquer elemento componente de um painel (chaves, mostradores, quadro de eventos, sinalizadores e botoeiras) é necessário que se selecione um painel já criado e clique no botão de criar um novo objeto, mencionado

anteriormente. Após clicar no botão, aparecerá uma janela *pop-up* para escolha do objeto componente de painel a ser criado, como mostrado na Figura 20.

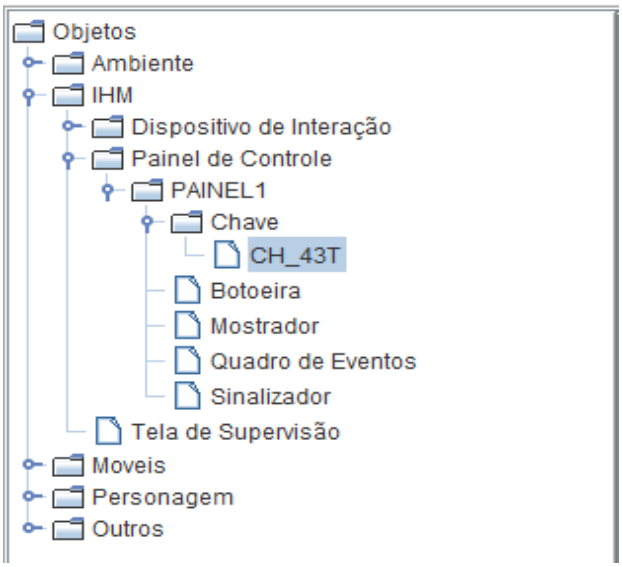
Figura 20 - Seleção de dispositivo de manobra adicionado ao Painel



Fonte: o próprio autor

Ao clicar no botão de confirmar, no painel de criação de uma chave, o objeto criado será listado abaixo do nó “Chave” abaixo do painel criado, na Lista de Objetos. Se o objeto criado (neste caso o objeto de nome “CH_43T”) for selecionado, o painel de edição da chave é mostrado da mesma forma da Figura 19, com as informações relativas a esse objeto. Na Figura 21 é mostrada a Lista de Objetos atualizada apóes a confirmação da criação do objeto

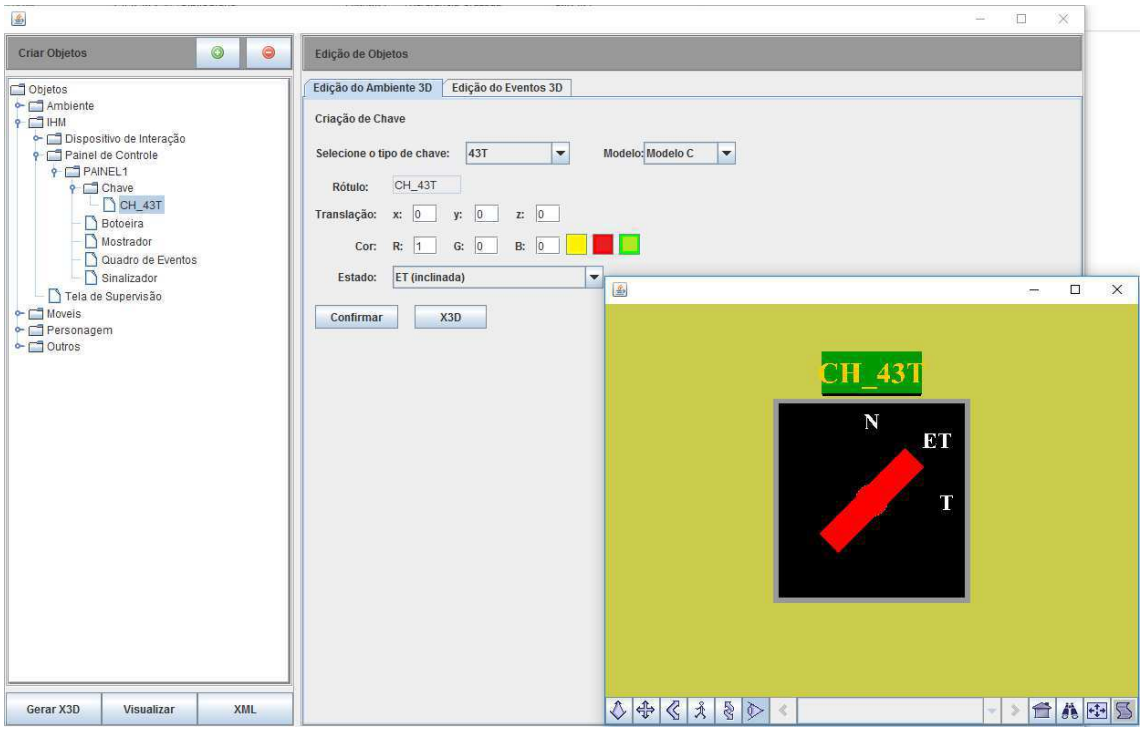
Figura 21 - Lista atualizada após criação da chave



Fonte: o próprio autor.

Ao clicar no botão referente à visualização do objeto que está sendo criado ou editado, o Editor aciona o *browser* Xj3D para a visualização do objeto. Para o caso da criação da chave ilustrada pela Figura 19, será obtido o resultado mostrado na Figura 22.

Figura 22 - Visualização 3D da chave criada



Fonte: o próprio autor.

O próximo elemento de interface a ser analisado são os painéis referentes à criação dos eventos relacionados aos objetos 3D que foram criados. Para criação de um evento para um objeto basta selecionar o objeto desejado e selecionar a aba de “Edição de Eventos 3D”. Neste trabalho foram implementados apenas eventos temporais.

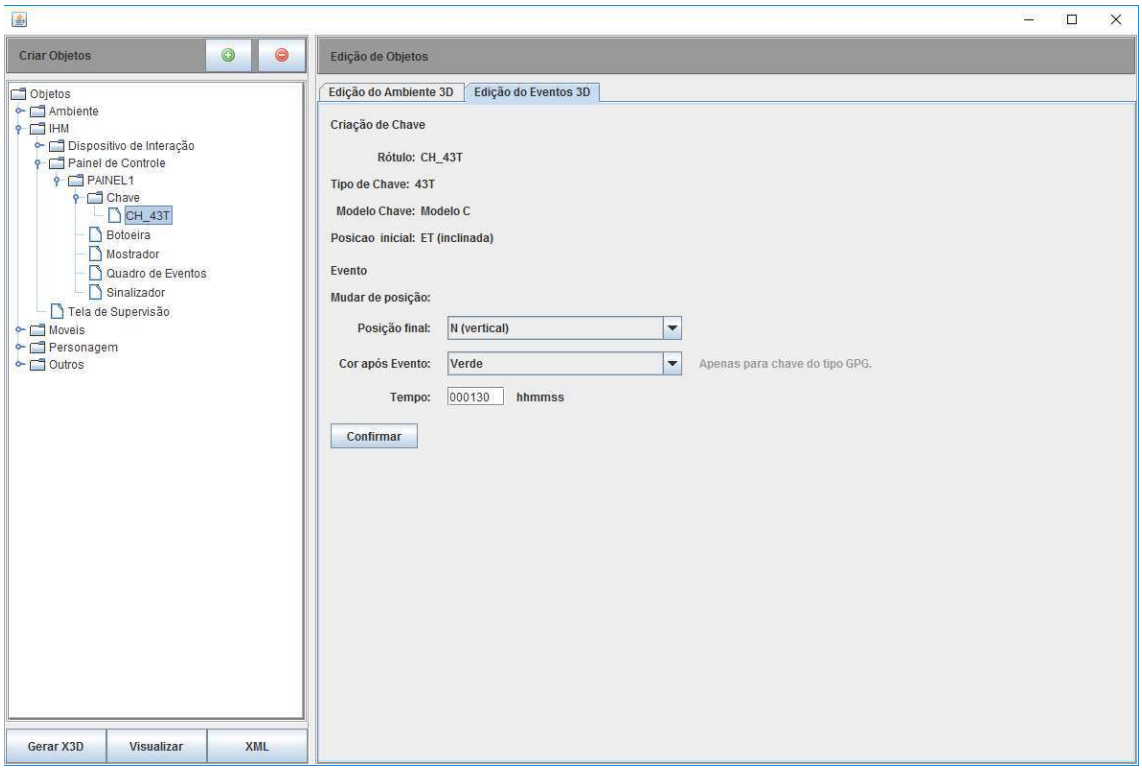
Na interface para edição 3D foram selecionados os seguintes elementos de interface:

- Rótulos de texto: utilizados para a exibição de informações dos objetos relevantes para construção do evento, como o rótulo, modelo, tipo e posição inicial.
- Caixas de seleção: utilizadas para escolher o estado final dos objetos após o disparo do evento.

- Caixa de texto: utilizado para receber o parâmetro "**tempo**" relativo ao disparo temporal do evento, ou seja, após quanto tempo o evento irá acontecer após o disparo do mesmo.
- Botão: utilizado para confirmar a criação do evento relacionado ao objeto.

Na Figura 23 pode-se visualizar a criação de um evento para a chave do tipo 43T criada anteriormente. Ao clicar no botão "Confirmar" o evento será criado e poderá ser visualizado ou editado posteriormente, pelo mesmo procedimento utilizado para criação do evento.

Figura 23 - Criação de evento para uma chave



Fonte: o próprio autor.

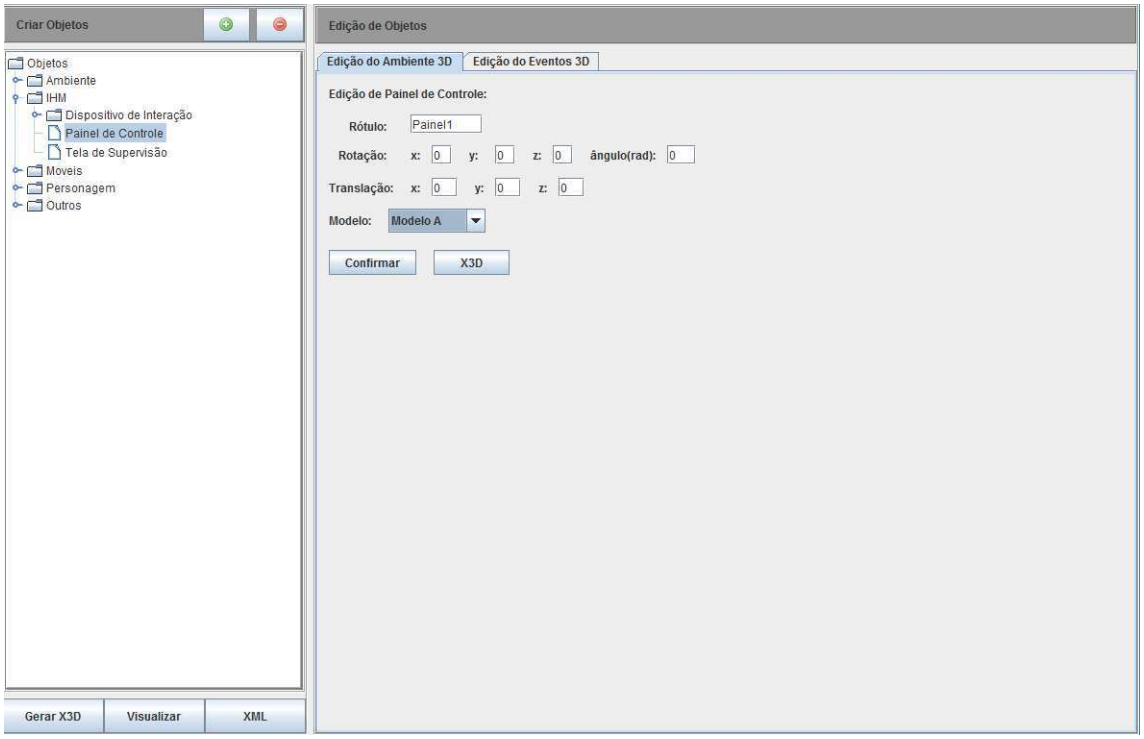
4.4 VALIDAÇÃO DO EDITOR

Para a validação do trabalho desenvolvido, nesta seção será criado um cenário completo, com a criação de todos os objetos que foram implementados nessa proposta de interface, ou seja, todos aqueles com modelos 3D pré-definidos.

Primeiramente, foi criado um painel de controle. Selecionou-se o item "Painel de Controle" na Lista de Objetos e em seguida foi dado um clique no botão de criar objetos.

Dessa forma, foi adicionado o painel de interface para criação de um painel de controle na área de Edição de Objetos 3D. Esse painel é mostrado na Figura 24.

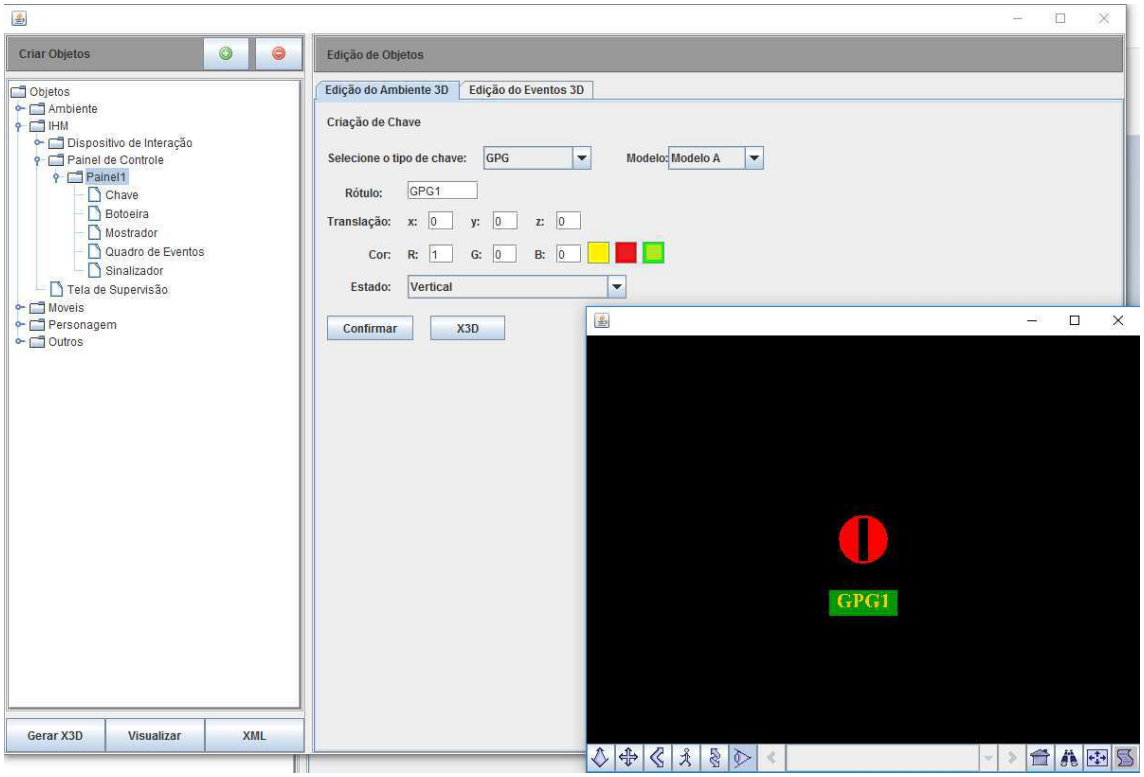
Figura 24 - Criação de um Painel de Controle



Fonte: o próprio autor.

O próximo passo foi a criação de dispositivos de interação que iriam fazer parte do painel de controle criado. Um dos dispositivos criados foi a chave GPG de rótulo “GPG1”. Em seguida foi dado um clique no botão “X3D” para obter a visualização 3D dessa chave. O painel de criação de chave e o resultado da visualização são mostrados na Figura 25.

Figura 25 - Criação e visualização de uma Chave GPG

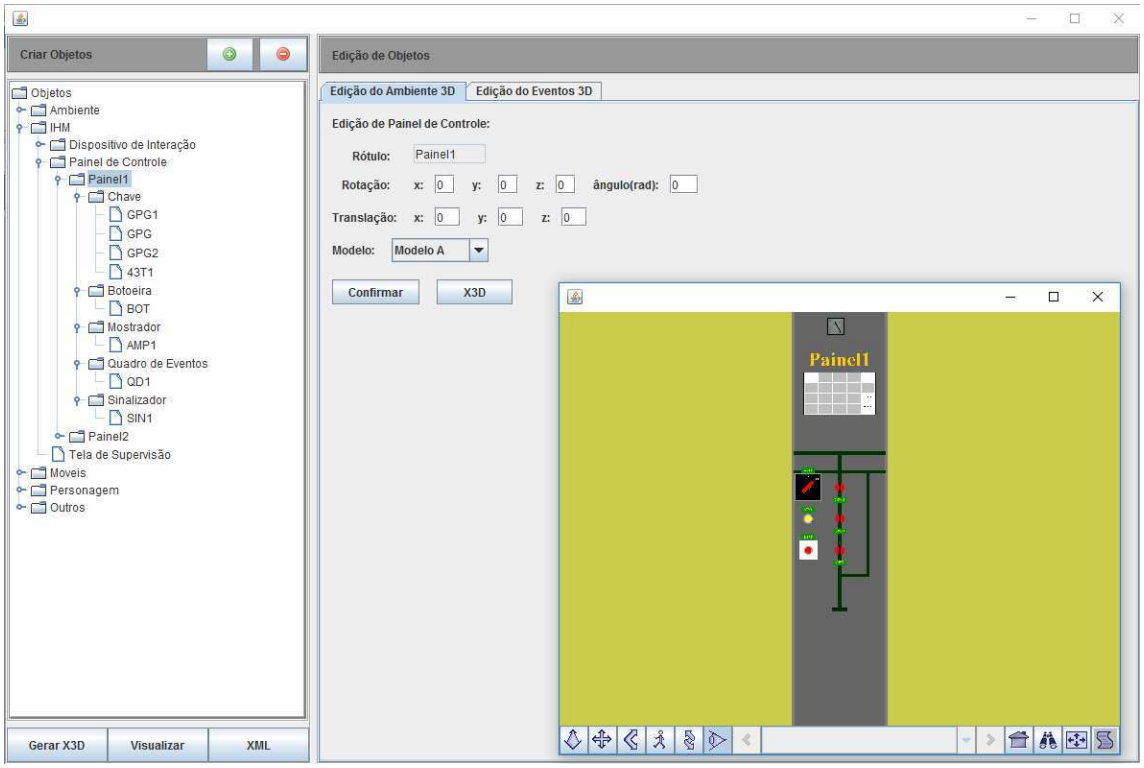


Fonte: o próprio autor.

O mesmo procedimento que foi feito para criação da chave GPG foi seguido para que outros dispositivos de interação fossem adicionados ao painel de controle “Painel1” que foi criado. Foram criadas mais 2 chaves GPG, uma chave 43T, um mostrador, uma botoeira, um alarme e um quadro de avisos.

Após a criação desses dispositivos, clicou-se no item “Painel” da lista de objetos, o que ativou a edição do painel. Na área de edição do painel de controle foi dado um clique no botão “X3D” de modo a obter a visualização do painel de controle criado, com os dispositivos de interação já adicionados. O resultado desse procedimento pode ser visto na Figura 26.

Figura 26 - Visualização do Painel de Controle criado e seus componentes de interação

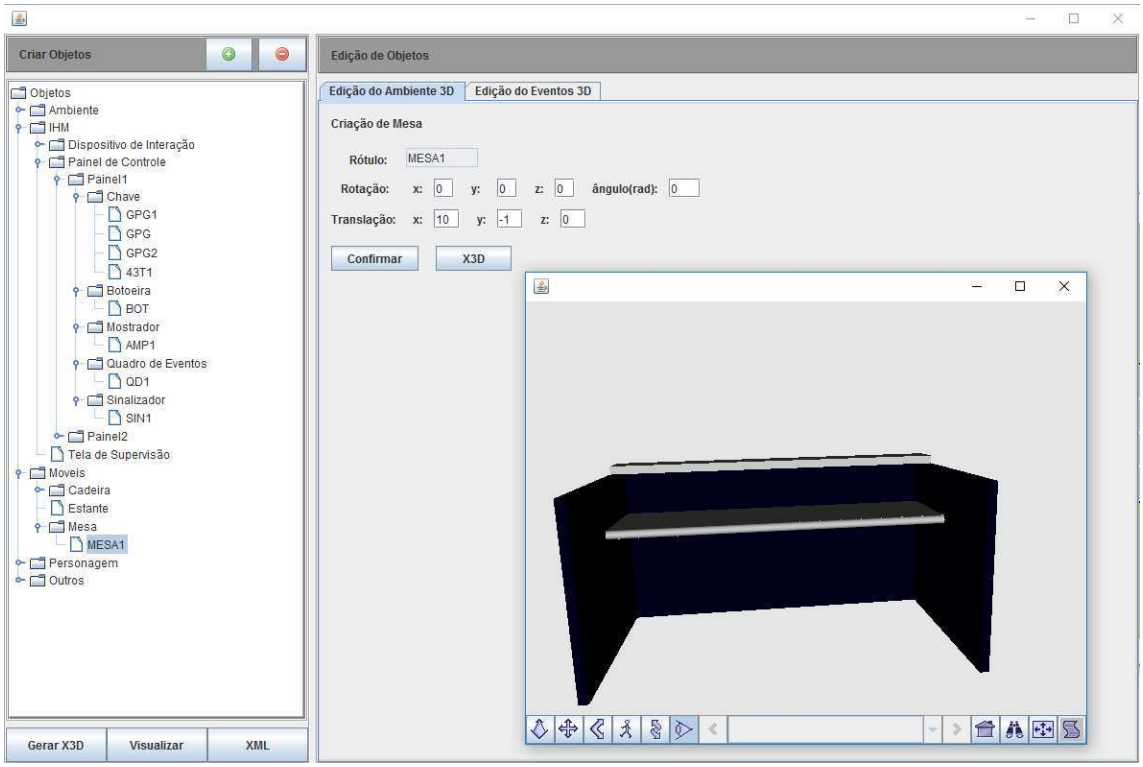


Fonte: o próprio autor.

Observa-se que, até o momento, as funcionalidades de criação dos objetos e visualização dos mesmos foram executadas com sucesso.

De forma semelhante à criação do objeto “Painel1”, foram criados mais 3 objetos: um painel de controle “Painel2”, uma mesa e uma cadeira. Para ilustração dessa etapa, apresentou-se na Figura 27 a tela de criação de uma mesa 3D e sua visualização.

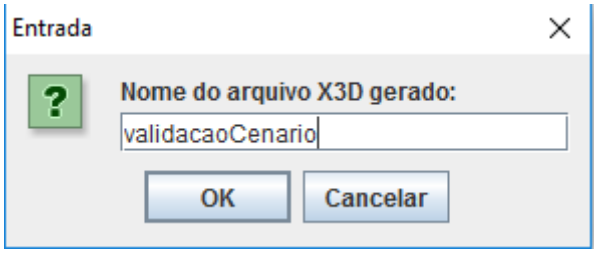
Figura 27 - Criação e visualização de uma Mesa 3D



Fonte: o próprio autor.

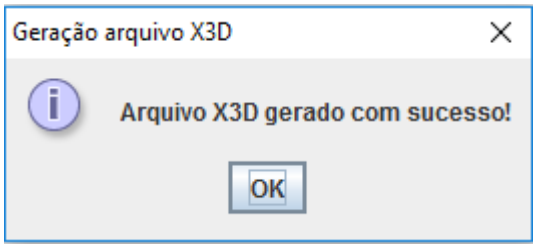
Para que fosse gerado e salvo o arquivo X3D relativo à cena 3D descrita pela criação dos objetos listados até o momento, foi dado um clique no botão “Gerar X3D”. Ao clicar no botão, é aberta uma janela com um espaço de texto para que seja definido o nome do arquivo, como mostrado na Figura 28. Nesse caso, foi escolhido o nome “validacaoCenario”, que irá gerar o arquivo “validacaoCenario.x3d” na pasta “GeradoX3D” do projeto. Após a confirmar o nome do arquivo desejado, outra janela *pop-up* é mostrada, com a confirmação do sucesso ou da falha do salvamento. Na Figura 29 é mostrada a janela de confirmação do sucesso do salvamento.

Figura 28 - Salvamento do arquivo X3D do cenário



Fonte: o próprio autor.

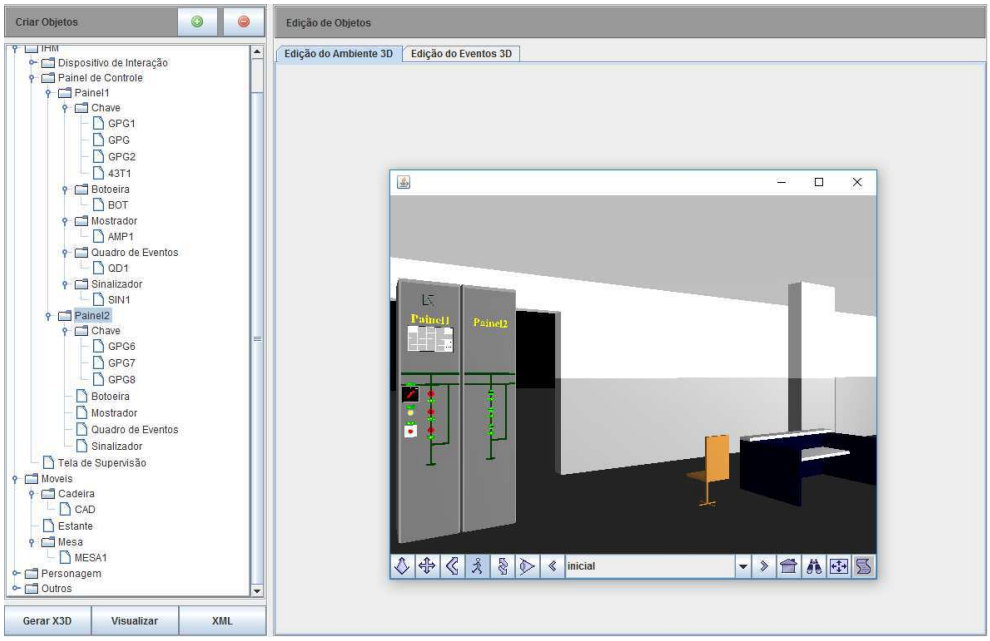
Figura 29 - Janela pop-up de informação de sucesso na geração do arquivo X3D



Fonte: o próprio autor.

Após a geração do arquivo X3D, pode-se visualizar a cena 3D relativa ao cenário que está sendo criado. Ao clicar no botão “Visualizar” o usuário tem acesso à representação3D do ambiente da sala de controle criada. A visualização da cena 3D é mostrada na Figura 30.

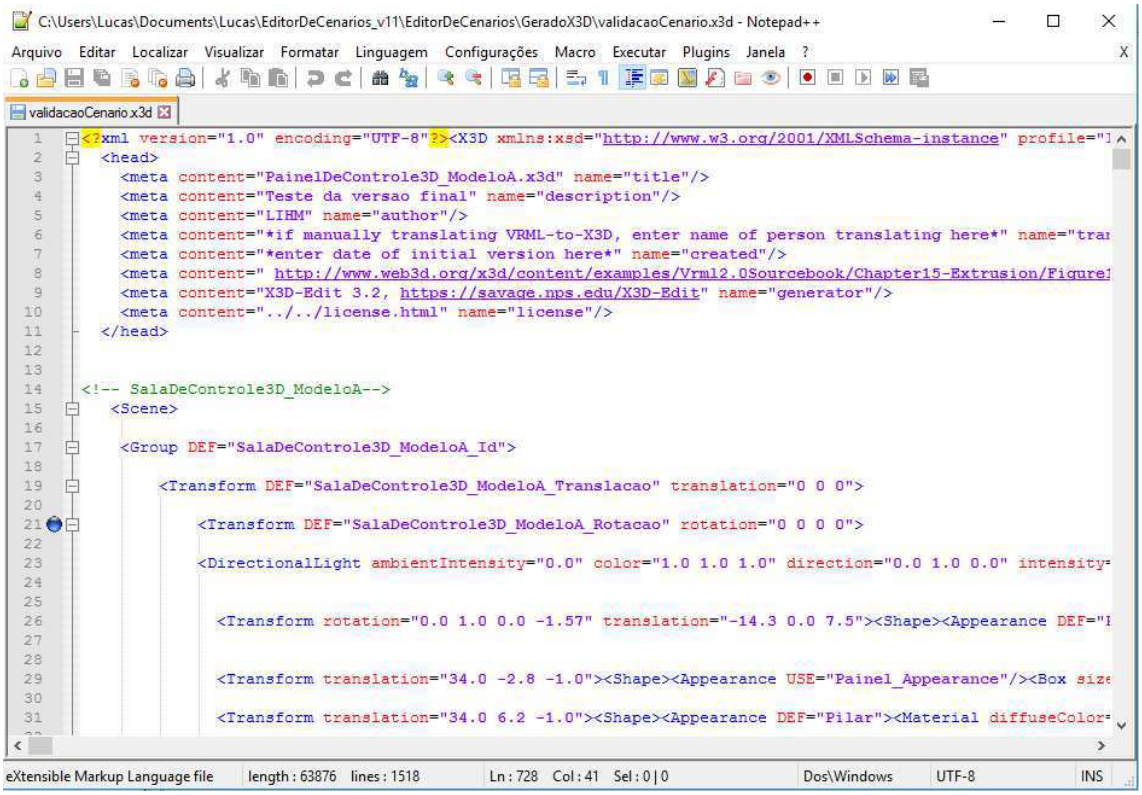
Figura 30 - Visualização da Cena 3D gerada



Fonte: o próprio autor.

Para obter a visualização do código do arquivo X3D no formato XML, pode-se clicar no botão “XML”. Será aberta uma aplicação externa que irá abrir o editor de arquivos XML padrão em uso no computador. No caso desse teste, foi selecionado o programa Notepad++ como editor padrão. O resultado dessa ação pode ser visto na Figura 31.

Figura 31 - Visualização do editor de arquivo XML do Cenário



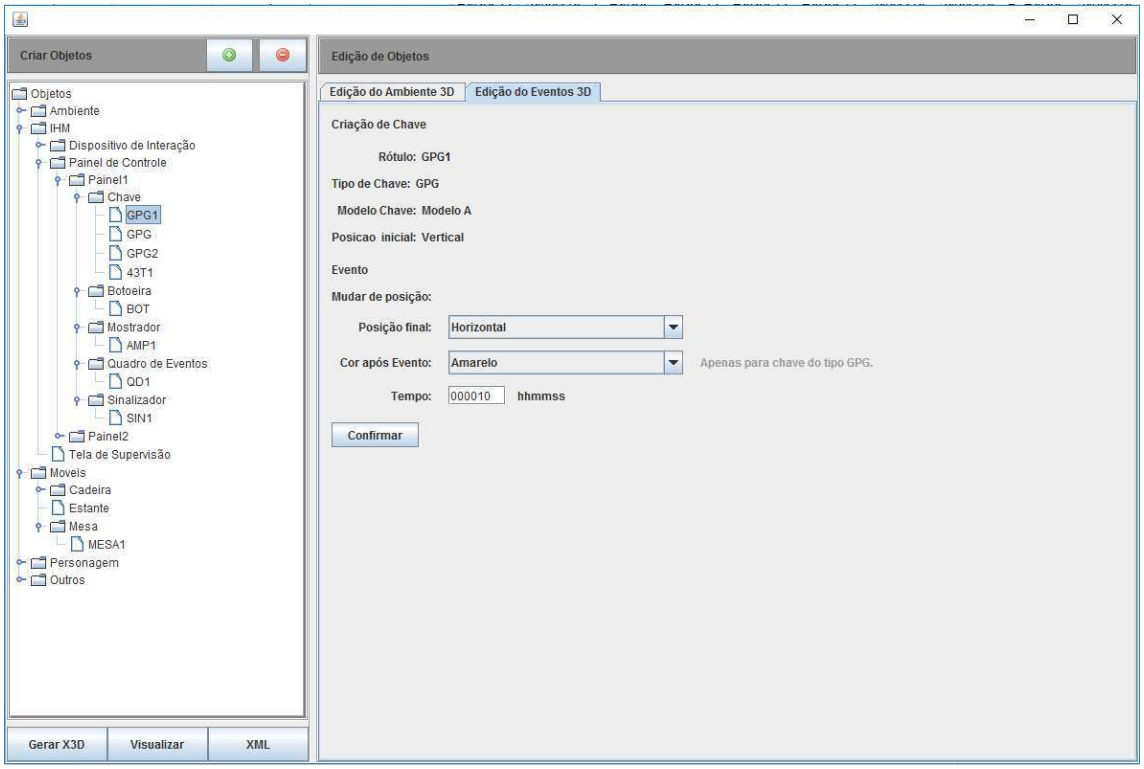
Fonte: o próprio autor.

Com isso, podem-se considerar válidas as funcionalidades relativas à geração do arquivo X3D da cena da sala de controle, da visualização do ambiente 3D e do editor do arquivo X3D no formato XML.

A criação de eventos relacionados aos objetos criados é feita selecionando-se o objeto desejado e a aba “Edição de Eventos 3D”. Para cada um dos dispositivos de interação dos painéis foi definido um evento específico. A edição de um evento da chave GPG de rótulo “GPG1” é mostrada na Figura 32. Apesar de criados, os eventos relacionados a cada objeto não é salvo em nenhum tipo de arquivo ou banco de dados, sendo armazenados apenas como objetos durante a execução do Editor de Cenários.

Com isso, podemos considerar que o desenvolvimento do editor de Cenários foi validado com sucesso, pois atingiu com sucesso a implementação das funcionalidades que foram pré-determinadas para a concepção da interface gráfica.

Figura 32 - Criação de um evento temporal para uma chave GPG



Fonte: o próprio autor.

5 CONSIDERAÇÕES

Neste trabalho foi desenvolvido uma interface gráfica com o objetivo de dar suporte à geração de cenas 3D e cenários de treinamento, para utilização em simulações de operação de uma sala de controle de subestação no SimuLIHM.

O uso do software desenvolvido permite que sejam criados objetos 3D com o uso da interface gráfica para inserção das informações (parâmetros 3D) dos objetos presentes em uma sala de controle como cadeiras, mesas, telefones, painéis de controle e os dispositivos de interação presentes nesses painéis. Além disso, a partir das informações coletadas pelo uso da interface, é gerado um arquivo X3D do ambiente tridimensional com os objetos criados. Pode-se também visualizar os objetos criados e a cena 3D criada através do uso do browser Xj3D, ativado na interface durante a criação de objetos ou após a geração do arquivo X3D.

Com relação à criação de eventos, a interface contém painéis para cada tipo de objeto e os possíveis eventos relacionados. Logo, é possível gerar informações sobre os eventos (cenários) associados à cena 3D que foi gerada. No entanto, ao contrário das informações relativas à cena 3D, que é salva em um objeto 3D, as informações dos eventos não são armazenadas. Uma proposta de trabalho futuro seria implementar uma funcionalidade de salvamento dessas informações dos eventos, numa ontologia local ou em um banco de dados, por exemplo.

Dessa forma, considera-se que este trabalho atingiu seus principais objetivos, com a implementação de um Editor de Cenários para o SimuLIHM. Além disso, este trabalho contribuiu para o aperfeiçoamento dos conhecimentos do autor sobre a linguagem Java, a concepção e criação de interfaces gráficas ergonômicas e de desenvolvimento de software.

5.1 TRABALHOS FUTUROS

Para dar continuidade a esse trabalho e complementar funcionalidades que ainda precisam ser desenvolvidas para uma maior integridade do software desenvolvido, são propostas as seguintes atividades:

- Implementação de um módulo que salve em uma ontologia (localmente) ou em um banco de dados os objetos da cena 3D e dos cenários (eventos) criados com o uso deste software.
- Definição de modelos de objetos X3D para mais objetos presentes numa sala de subestação elétrica como chaves punho, mostradores digitais, voltímetros, computadores, placas com eventos, dentre outros. Com a inclusão de tais modelos X3D poderão ser desenvolvidas novos painéis na interface, de modo a gerenciar a criação de tais objetos.
- Criação de um módulo mais robusto para visualização do código XML relativo aos objetos e cenas 3D criadas.
- Inclusão de eventos condicionais, na seção de criação e eventos do Editor de Cenários.

REFERÊNCIAS

BARBOSA, J. G. **Projeto do simulador para treinamento de operadores: Módulo do banco de dados** [Relatório] : Trabalho de Conclusão de Curso / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2010. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/cgee/projeto-de-engenharia/relatorios>.

FERREIRA, F. F. V. M**. Geração de Relatórios de Avaliação de Treinamentos** [Relatório] : Trabalho de Conclusão de Curso / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2014. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/cgee/projeto-de-engenharia/relatorios>.

FILHO F. T. **Desenvolvimento de um módulo supervisório para um ambiente de treinamento de operadores** [Relatório] : Dissertação de Mestrado / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - Campina Grande : [s.n.], 2011. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/pos-graduacao/banco-de-dissertacoes>.

HORSTMANN, C. **Big Java** [Livro] / trad. Furmankiewicz Edson. - Porto Alegre : Bookman, 2004.

LOPES, D. S. **Construção de uma biblioteca de objetos para edição de ambientes virtuais de treinamento em subestações de sistemas elétricos** [Relatório] : Trabalho de Conclusão de Curso / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2011. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/cgee/projeto-de-engenharia/relatorios>.

LUCENA, E. M. **Extensão da Representação em Realidade Virtual do Simulador do Ambiente de Operação de Subestações de um Sistema Elétrico** [Relatório] : Trabalho de Conclusão de Curso / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2010. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/cgee/projeto-de-engenharia/relatorios>.

NETO, J. A. N. **Modelagem da Interface Homem-Maquina de uma Subestação Elétrica** [Relatório] : Dissertação de Mestrado / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2004. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/pos-graduacao/banco-de-dissertacoes>.

NETTO, A. V. S. **Arquitetura para um ambiente de treinamento representado em Realidade Virtual** [Relatório] : Dissertação de Mestrado / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2010. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/pos-graduacao/banco-de-dissertacoes>.

SANTOS, H. R. **Refinamento do Mundo Virtual de uma Subestação Elétrica** [Relatório] : Trabalho de Conclusão de Curso / Departamento de Engenharia Elétrica ; Universidade Federal de Campina Grande. - 2009. Disponível em: < https://sites.google.com/a/dee.ufcg.edu.br/cgee/projeto-de-engenharia/relatorios>.