COPPE/UFRJ

3DBYSTEP – FERRAMENTA PARA AUTORIA DE APRESENTAÇÕES 3D

Elisabete Thomaselli Nogueira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas.

Orientador: Claudio Esperança

Rio de Janeiro

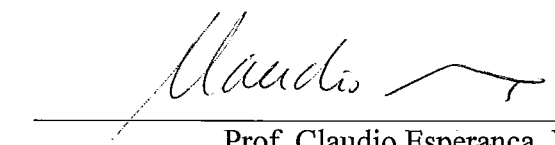
Outubro de 2008

3DBYSTEP – FERRAMENTA PARA AUTORIA DE APRESENTAÇÕES 3D

Elisabete Thomaselli Nogueira

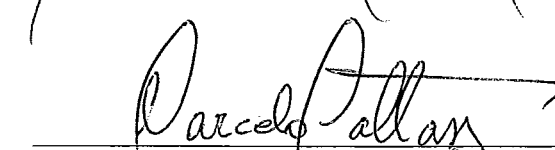
DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



Prof. Claudio Esperança, Ph.D.

Prof. Paulo Roma Cavalcanti, D.Sc.



Prof. Marcelo Gattass, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 2008

Nogueira, Elisabete Thomaselli

3DbyStep – Ferramenta para autoria de apresentações 3D. / Elisabete Thomaselli Nogueira - Rio de Janeiro: UFRJ/COPPE, 2008.

XIII, 77 p.: il.; 29,7 cm.

Orientador: Claudio Esperança

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2008.

Referencias Bibliográficas: p. 58-62.

1. Computação gráfica. 2. Autoria. 3. Modelagem 3D. I. Esperança, Claudio. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

Dedicatória

Ao meu amado pai José Luiz Nogueira, às minhas duas filhas Camila e Daniele e ao meu esposo Marcelo.

Cada pessoa vê no mundo a própria imagem.

Francisco Cândido Xavier

Agradecimentos

Gostaria de agradecer em primeiro lugar a todos que, direta ou indiretamente, contribuíram para a conclusão deste trabalho.

Agradeço aos professores da UFRJ, Claudio Esperança, Paulo Roma, Ricardo Farias, Antônio Oliveira e Gerson Cunha, pelo aprendizado recebido. Muito obrigada ao meu colega de pesquisa Victor Bursztyn, por ter participado com vontade e qualidade no desenvolvimento do protótipo, ao meu irmão José Luiz Thomaselli Nogueira, por ter sempre me indicado o caminho certo a ser seguido, ao colega de trabalho e amigo Luiz Eduardo Borges por ter me ajudado com muita presteza nas dúvidas sobre o ambiente de *software* proposto. Obrigada, principalmente, ao meu orientador Claudio Esperança por sempre ter me apoiado e esclarecido minhas dúvidas.

Aos que acompanharam comigo este desafio: Alexandre, Álvaro, Disney, Flávio, Jonas, Kaique, Marroquim, Okamoto, Pilato, Saulo, Wagner, Yalmar, Diogo, Djeisson, Guina, Luís, e aos demais colegas inscritos do LCG no período de 2005 - 2008, obrigada e parabéns pelos resultados obtidos. Agradeço em especial ao meu amigo Djeisson Rober Vieira pela grande ajuda.

A toda a minha família, por ter me dado condições e apoio para chegar a concluir este trabalho. Sem eles nada disso teria sido possível e tampouco haveria uma felicidade completa.

E é claro, agradeço a meu esposo Marcelo por estar sempre ao meu lado, me ajudando e apoiando, e às minhas pequenas filhas Camila e Daniele, pela compreensão da importância dos estudos da mamãe.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

3DBYSTEP - FERRAMENTA PARA AUTORIA DE APRESENTAÇÕES 3D

Elisabete Thomaselli Nogueira

Outubro/2008

Orientador: Claudio Esperança

Programa: Engenharia de Sistemas e Computação

Neste trabalho investiga-se a construção de ferramentas para autoria de conteúdo 3D. Em particular, define-se um conjunto de características que permitam sua utilização por usuários não especialistas. Este perfil compreende aspectos relacionados, tanto à interface humano-computador quanto às funcionalidades oferecidas por tais ferramentas. Uma ferramenta, denominada 3DbyStep, foi construída com base nesse perfil e usada em um experimento piloto para autoria de uma apresentação 3D acerca de equipamentos de geração de energia eólica.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

3DBYSTEP - A TOOL FOR AUTHORING 3D PRESENTATIONS

Elisabete Thomaselli Nogueira

October/2008

Advisor: Claudio Esperança

Department: Systems Engineering and Computer Science

This work investigates the construction of tools for authoring 3D content. In particular, it is defined a set of issues that make such tools amenable to use by non-specialists. This profile includes aspects related to both HCI and functionality. A sample application called 3DbyStep was implemented and used in an experiment for creating a 3D presentation about wind power equipments.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A evolução dos computadores pessoais e a queda nos preços de componentes de *hardware* fizeram com que muitas pessoas, especialistas ou não, pudessem utilizar computadores para realizar suas tarefas. Com isso, a preocupação com a qualidade da interface de interação do usuário cresceu, apoiada pela evolução do ambiente gráfico.

Uma variedade de recursos de computação gráfica, como *software* de modelagem 3D, linguagens de programação com bibliotecas apropriadas e placas gráficas de alta performance, é usada na produção de soluções.

Atualmente, uma grande quantidade de objetos 3D está disponível no mercado, trazendo considerável riqueza de conteúdo visual. Ilustrações e animações em 3D possibilitam entendimento rápido e amplo do contexto visual apresentado. Observa-se, contudo que, nem sempre existe conteúdo didático associado a esses objetos, em razão da especialização necessária ao desenvolvimento deste conteúdo e a existência de custo de aquisição de ferramentas comerciais para este fim.

1.1 Objetivos e motivação

O objetivo deste trabalho é investigar requisitos de ferramentas interativas para autoria de conteúdo educacional, adequadas para usuários não especialistas em computação. Seu foco é habilitar esses usuários a construir apresentações multimídia, compostas de elementos textuais, imagens, modelos tridimensionais, som e animações. Em particular, o uso de objetos tridimensionais permite o emprego de

operações como translação, rotação, *zoom* dos objetos, uso de sombras, texturas, perspectiva ou, até mesmo, visão estereoscópica [1], minimizando problemas de percepção. A educação e, especialmente o treinamento, podem ter valor agregado com a utilização desta tecnologia [2] [3] [4], proporcionando aos sistemas de ensino novos subsídios para o aprendizado.

Existem diversas ferramentas convencionais que possibilitam a editoração de apresentações sob a forma de uma seqüência de quadros (*slides*), sendo aplicáveis à autoria de conteúdos de treinamento. Como exemplos, podemos citar os produtos comerciais PowerPoint, Corel Draw, Adobe Flash e Adobe Director, e outras de código aberto (*open source*) como OpenOffice [5], Slithy [6], Pyntor [7] e Kpresenter [8]. Todavia, estas ferramentas restringem-se à construção de apresentações bidimensionais.

Existem ferramentas para modelagem e animação de conteúdo tridimensional, como os produtos comerciais 3DStudio [9], Shockwave, Maya [10], e de código aberto como Blender [11], POV-Ray [12] e Wings3D [13]. Tais aplicações são caracterizadas pela complexidade de uso, interface de difícil interação e necessidade de *hardware* de alto desempenho (espaço em disco, memória e velocidade de processamento). Estas ferramentas, todavia, não são propícias para usuários eventuais, sendo mais adequadas aos especialistas em computação gráfica. Soma-se a isso, o custo de aquisição e atualização, de ferramentas comerciais.

Existem ainda, *software* com finalidade educacional tais como o TelEduc [14], projetado pela Unicamp, que permite disponibilizar cursos à distância através da internet e o WBTExpress [15], ambiente para criar cursos básicos, como, por exemplo, para uso de processadores de texto ou de técnicas de redação. Estes ambientes, contudo, não tratam da autoria de conteúdo 3D.

O conhecimento intrínseco disponível em modelos tridimensionais é passível de ser minuciosamente explorado, caso se disponha de ferramenta de manipulação de objetos, navegável, que disponibilize conteúdos para utilização de forma interativa. A qualidade da interação do usuário com o aplicativo, entretanto, depende da facilidade de utilização deste.

Este trabalho assume que usuários não especialistas são capazes de criar conteúdo 3D, desde que disponham de uma ferramenta com interface e funcionalidades ade-

quadas.

O perfil de ferramenta proposto, neste trabalho, integra em um mesmo ambiente modelos tridimensionais, informação textual e em áudio, além de imagens 2D. As configurações para apresentação e disposição destes elementos são programadas pelo autor do conteúdo, com a vantagem adicional de permitir seu controle pelo usuário alvo. Os conteúdos produzidos são fáceis de manipular, não só como apresentações seqüênciais, mas também de forma livre, permitindo navegação interativa, segundo necessidades de aprendizado. Por ser de fácil utilização e específico para autoria de conteúdo em 3D, o perfil de aplicação proposto possibilita o aprendizado mais ágil e o entendimento mais amplo do conteúdo didático ao público em geral. Além disso, propõe que o conteúdo de apresentação gerado possa ser exibido usando *software* multi-plataforma, desenvolvido, em sua grande parte ou totalidade, com ferramentas *open source.*

O perfil proposto é especialmente importante para áreas onde a habilidade de visualização de modelos 3D seja crucial ao pleno entendimento dos assuntos abordados.

1.2 Metodologia

Propôs-se um perfil de aplicativo, utilizado para especificação e implementação de um protótipo, denominada *3DbyStep.* Este foi testado e usado na confecção de conteúdo didático, na área de energia eólica, com a finalidade de validar as especificações e funcionalidades descritas nesta dissertação.

Este trabalho empregou também conceitos de usabilidade [16] [17] [18], aplicados à manipulação de conteúdo multidimensional, coleta de conteúdo pedagógico (onde se identifica o que deve ser priorizado e destacado do conteúdo e de que forma fazê-lo), e em estudo de casos dentro de um escopo de treinamento e demonstração para indústria.

1.3 Estrutura da dissertação

O restante deste trabalho tem a seguinte organização: O capítulo 2 apresenta conceitos de interface homem-máquina e de computação gráfica pertinentes ao tra-

balho. O capítulo 3 trata da especificação das funcionalidades do modelo e da interface. O capítulo 4 aborda os detalhes de implementação e apresenta um estudo comparativo entre ferramentas de autoria de conteúdo. O capítulo 5 apresenta um estudo de caso feito com uso do *3DbyStep*.

Por fim, o capítulo 6 apresenta as contribuições deste trabalho, as limitações do protótipo (*3DbyStep*) e orientações para trabalhos futuros.

Capítulo 2

Fundamentos Teóricos

Neste capítulo são apresentados conceitos de interface humano-computador (item 2.1) e de instanciação de objetos 3D utilizados na dissertação (item 2.2).

2.1 Conceitos de interface humano-computador

Algumas máximas, que norteiam o desenvolvimento de interfaces, descrevem o sistema ideal como aquele que esconde a tecnologia, para que o usuário não se preocupe, ou mesmo tenha consciência dela. Isso é feito para que as pessoas fiquem focadas em executar a tarefa.

O ideal é que se empregue a ferramenta na tarefa, e não como usualmente ainda ocorre, de adequar a tarefa à ferramenta. Para tanto, é útil seguir os princípios de simplicidade, versatilidade e satisfação, recomendados nos axiomas do *design* [19]. Para adequação da interface, outro fator importante, é que se saiba o perfil de usuário que se pretende atender.

Nesta seção é apresentada uma classificação de usuários em categorias, motivação para estudo de interface e conceitos relacionados. São descritas também as heurísticas utilizadas e aspectos gerais relacionados às interfaces de aplicações 3D.

2.1.1 Categorias de usuários

Determinar o público alvo e conhecer as características dos usuários são tarefas fundamentais e determinantes nas decisões do projeto de interface.

Os usuários podem ser classificados, por exemplo, quanto à sua **categoria de especialização** como usuário casual, operador treinado para rotina, operador treinado em atividades complexas, profissional de computação e usuário especialista em ferramentas poderosas [20], **quanto ao grau de especialização** como iniciante, executor, intermediário, avançado e especialista [21] e **quanto ao grau de conhecimento do sistema e da tecnologia**, como novato em computadores, novato no domínio, novato no sistema, especialista no sistema, especialista em computadores e especialista no domínio [16] [19]. Um especialista, *expert*, é alguém considerado perito em lembrar onde encontrar informação no mundo externo.

Essa classificação, **quanto ao grau de conhecimento do sistema e da tecnologia**, trata a experiência como um fator relevante, que pode ser analisada em três dimensões. Nielsen apresenta essa diferenciação em um gráfico que ele denomina de cubo do usuário (ver Figura 2.1).

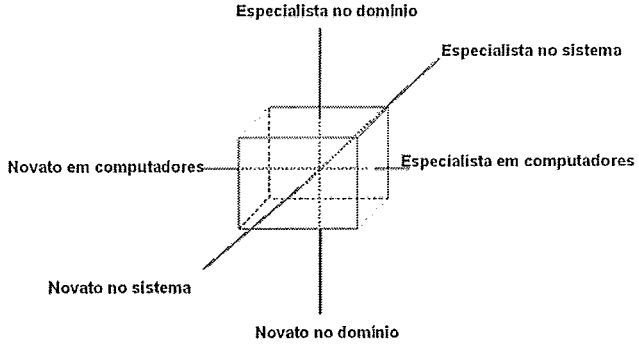


Figura 2.1: *Classificação quanto ao grau de conhecimento do sistema e da tecnologia.*

É importante ressaltar que o uso do sistema altera a categoria do usuário e isso tem importantes implicações no *design*. Algumas interfaces são e devem ser projetadas com ênfase apenas em usuários novatos, onde a facilidade de aprendizagem é o requisito principal. Mas a maioria das interfaces é projetada tendo em vista tanto especialistas, quanto novatos e portanto precisa prover múltiplos estilos de interação, de tal forma que os usuários iniciem aprendendo um estilo mais fácil e depois migrem para outro mais eficiente.

2.1.2 Motivações para o estudo de interfaces com usuários (IUs)

Os procedimentos e estratégias que os seres humanos adotam para se comunicar devem servir como base ao projeto de uma boa interface com o usuário [22]. O conhecimento das habilidades psicológicas e fisiológicas do público a qual se destina o sistema é fundamental. A Figura 2.2 detalha a relação entre o uso de interfaces de *software* e o aspecto humano desse conhecimento. A idéia de interface em IHC está ligada à comunicação. Nesse contexto, o homem interage através de seus aspectos cognitivos, emocionais e sentidos próprios.

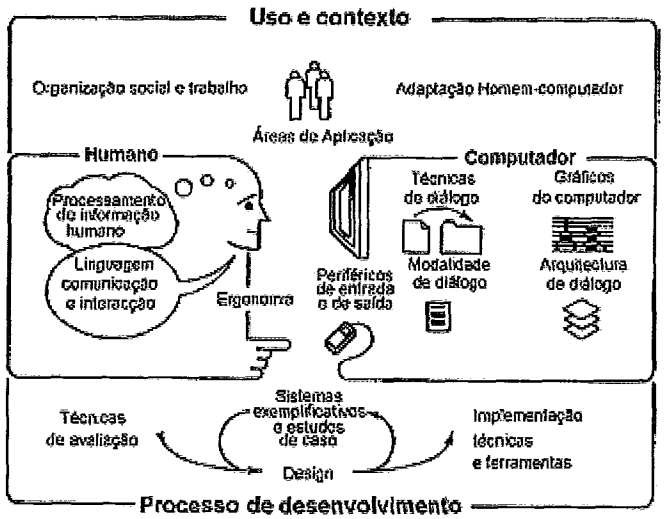


Figura 2.2: *Relacionamento entre homem e computador.*
(Fonte: PREECE, 1994)

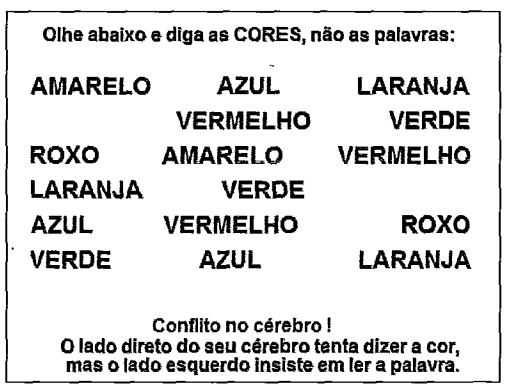


Figura 2.3: *Observação das cores e das palavras*
(Fonte: http://nemesvv.googlepages.com/ilusõesdeÓptica)

Existem exemplos clássicos de fatores humanos em IHC que podem ilustrar o quão sutil é a percepção humana de um determinado evento ou contexto. Na Figura 2.3, apesar das instruções claras sobre a relevância da observação das cores das palavras, o cérebro se retém, por vezes, na palavra escrita.

Os principais fatores que podem ser contabilizados no *design* da interface se relacionam, primariamente, com o ambiente de trabalho, com características próprias dos usuários, com o trabalho realizado e também com a tecnologia que está sendo utilizada (ver Figura 2.4). Por exemplo, se mudanças são feitas para aumentar a produtividade, isto pode ter um efeito não desejado na motivação dos usuários e no nível de satisfação, porque fatores relacionados com tratamento do *design* e organização do trabalho foram ignorados.

<table>
<tr><td colspan="2">FATORES ORGANIZACIONAIS
TREINAMENTO, POLÍTICAS,
ORGANIZAÇÃO DO TRABALHO,
ETC.</td><td colspan="2">FATORES AMBIENTAIS
BARULHO, AQUECIMENTO,
VENTILAÇÃO, LUMINOSIDADE, ETC.</td></tr>
<tr><td>SAÚDE E SEGURANÇA
estresse, dores de cabeça, perturbações musculares, etc.</td><td colspan="2">capacidades e processos cognitivos
O USUÁRIO
motivação, satisfação,
personalidade, experiência, etc.</td><td>CONFORTO
posição física, layout do equipamento, etc.</td></tr>
<tr><td colspan="4">INTERFACE DO USUÁRIO
dispositivos de entrada e saída, estrutura do diálogo, uso de cores, ícones, comandos, gráficos, linguagem natural, 3-D, materiais de suporte ao usuário, multimídia, etc.</td></tr>
<tr><td colspan="4">TAREFA
fácil, complexa, nova,
alocação de tarefas, repetitiva,
monitoramento, habilidades, componentes, etc.</td></tr>
<tr><td colspan="4">RESTRIÇÕES
custos, orçamentos, equipe,
equipamento, estrutura do local de trabalho, etc.</td></tr>
<tr><td colspan="4">FUNCIONALIDADE DO SISTEMA
hardware, software, aplicação</td></tr>
<tr><td colspan="4">PRODUTIVIDADE
aumento da qualidade, diminuição de custos, diminuição de erros,
diminuição de trabalho, diminuição do tempo de produção,
aumento da criatividade , oportunidades para idéias criativas em direção a novos produtos, etc.</td></tr>
</table>

Figura 2.4: *Fatores em IHC.*
(Fonte: Preece, 1994)

A qualidade da interface afeta diretamente o custo do sistema a longo prazo. Isto porque interfaces de alta qualidade são projetadas para reduzir o tempo de treinamento não-produtivo, fornecer os meios para que o usuário utilize melhor os recursos do sistema e fazer com que o usuário aprenda, por si só, como atingir seus

objetivos, melhorando a qualidade do serviço [23].

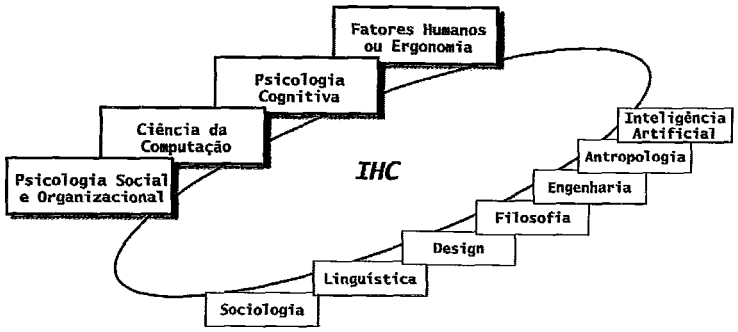


Figura 2.5: *As disciplinas que contribuem para IHC.*
(Fonte: Preece, 1994)

Diferentes disciplinas contribuem com suas áreas de interesse para o atendimento destes fatores (ver Figura 2.5). As áreas que contribuem mais diretamente para IHC são ciência da computação, psicologia cognitiva, psicologia social e organização psicológica, ergonomia e suas disciplinas irmãs de fatores humanos [24]. Outras áreas de interesse incluídas são inteligência artificial, lingüística, filosofia, engenharia e *design*.

Para cumprir com seus objetivos as interfaces devem ser fáceis de aprender. Reconhecer é muito mais efetivo que lembrar [25], por isso é fortemente aconselhável reduzir na interface o número de componentes que têm que ser lembrados. Um exemplo disso é que as interfaces baseadas em menu são mais fáceis de usar quando comparadas à sistemas baseados em comandos.

Em computação, chama-se interação a relação entre o homem e o sistema computacional. Uma aplicação tem interação com *design* bem definido sempre que contém qualidade (afirmações verdadeiras), quantidade (volume de informação adequado), relevância (tópico de diálogo bem fechado) e clareza (entendimento claro dos assuntos) adequados.

2.1.3 Princípios de interface

Os princípios de expectativa, implicação, experimentação e motivação compõem a base dos procedimentos e estratégias que os serem humanos adotam para se comunicar e estes quatro princípios devem ser observados ao projetar qualquer interface [26].

Abaixo, detalhamento de cada um destes quatro princípios, que foram observados para compor o perfil de aplicativo proposto neste trabalho:

- **Expectativa** - A expectativa do usuário é de que sistemas possuam, ao máximo, aspectos familiares. Em contraste, um grande número de aplicações de uso comum não privilegia esse aspecto, pressupondo a necessidade de manuais e treinamento.

- **Implicação** - Existe um contexto ativo o tempo todo e este é dinâmico. O conjunto de experiências anteriores relacionadas às entradas que são selecionadas é utilizado para gerar nossas expectativas sobre as próximas interações.

- **Experimentação** - A interação humana se faz de forma experimental e a aplicação deve se propor a repetir este modo de funcionamento.

- **Motivação** - Para que o usuário não abandone a aplicação, é fundamental que ele pressuponha que seus comandos serão obedecidos, que o desenvolvimento de seu trabalho está sendo bem executado e, principalmente, que a máquina realmente irá ajudá-lo. É fundamental que haja confiabilidade na interação.

Interface separada da aplicação

Seja para objetos do mundo real, ou para programas de computador, o primeiro princípio do *design* de interfaces é focar o usuário em primeiro lugar, sendo a tecnologia subordinada a este objetivo maior [27].

Por causa de procedimentos e estratégias inerentes à comunicação, novas IUs separam a lógica da aplicação da parte diretamente relacionada à interface com o usuário.

O objetivo principal da utilização de um sistema de gerência de interface com os usuários, SGIU, é produzir uma especificação da interface, com a vantagem de se ter um novo conjunto de possibilidades de interação disponível e maior economia de tempo e esforço, quando a interface está sendo modificada.

Os SGIUs tem como meta fornecer ao projetista da interface com o usuário um conjunto de ferramentas apropriadas, com um mínimo de habilidades de programação. Liberando os programadores dos detalhes de nível mais baixo, eles se detêm aos aspectos de mais alto nível da IU da aplicação.

O projeto de uma interface é um processo repetitivo, onde cada iteração consiste de 3 fases: projeto, implementação e avaliação [28] e um SGIU facilita este processo iterativo. Sendo a interface separada da aplicação, além de ambas poderem ser desenvolvidas separadamente, há melhoria substancial na qualidade da interface.

2.1.4 Heurísticas utilizadas

Uma avaliação heurística é baseada na verificação de uma lista de regras, ou seja, heurísticas. Esta forma de avaliação pode ser aplicada em qualquer fase do ciclo de desenvolvimento do *software*, permitindo apoiar o desenvolvimento de projetos e sendo aconselhável nas fases iniciais, onde a interface, às vezes, se restringe a um esboço em papel. Neste trabalho, optou-se por utilizar as heurísticas de usabilidade propostas por Nielsen [29] como parte do processo de *design* interativo e aplicá-las à iconografia.

A seguir é apresentada uma descrição suscinta de cada uma destas heurísticas:

1. **Visibilidade do sistema** - O usuário deve ser informado pelo sistema em tempo razoável sobre o que está acontecendo.

2. **Compatibilidade do sistema com o mundo real** - O modelo lógico do sistema deve ser compatível com o modelo lógico do usuário. O sistema precisa conter a linguagem do usuário, com palavras, frases e conceitos familiares a quem vai utilizá-lo. Seguir convenções do mundo real, com a informação disponibilizada numa ordem natural e lógica.

3. **Controle do usuário e liberdade** - O sistema deve tornar disponíveis funções que possibilitem anular os efeitos de ações indesejadas. Uma boa solução para isso é prover funções para desfazer (*undo*) ou refazer (*redo*) uma operação.

4. **Consistência e padrões** - O sistema deve ser consistente quanto à utilização de sua simbologia e à sua plataforma de *hardware* e *software*. Usuários não precisam inferir que palavras, situações ou ações diferentes significam a mesma coisa.

5. **Prevenção de erros** - O sistema deve ter um *design* que se preocupe com as possibilidades de erro. Melhor que uma boa mensagem de erro é um *design* que previne o erro antes dele acontecer.

6. **Reconhecimento ao invés de lembrança** - As instruções (objetos, ações e opções), para o bom funcionamento do sistema, devem estar visíveis no contexto em que o usuário se encontra.

7. **Flexibilidade e eficiência de uso** - O sistema deve prever o nível de proficiência do usuário em relação ao próprio sistema. Permitir que os usuários experientes cortem caminho em ações freqüentes, com o uso de aceleradores.

8. **Estética e *design* minimalista** - Os diálogos do sistema devem conter somente informações relevantes ao funcionamento. Unidade de informação extra no diálogo deve ser evitada, pois compete com unidades relevantes de informação e diminui sua visibilidade relativa.

9. **Ajuda aos usuários no reconhecimento, diagnóstico e correção de erros** - As mensagens devem ser expressas em linguagem clara, indicando precisamente o problema e sugerindo possíveis soluções.

10. ***Help* e documentação** - A informação desejada deve ser facilmente encontrada, de preferência deve ser contextualizada e não muito extensa. Embora seja melhor um sistema que possa ser usado sem documentação, é necessário prover *help* e documentação.

Com a observância a esses preceitos se espera propor uma interface que seja fácil de usar, agilize o aprendizado, satisfaça o usuário e proporcione produtividade.

2.1.5 Aspectos relacionados à interface de aplicações 3D

Ferramentas de computação gráfica nem sempre são atrativas para usuários leigos, desestimulando seu uso.

Técnicas de IHC aplicadas à área de computação gráfica (CG) devem ser cuidadosamente avaliadas e há um potencial razoável de comportamentos ainda não explorados [24]. Alguns desafios para projetar boas interfaces são [30]:

- Prover controles que tenham operações e efeitos relativamente óbvios e que denotam um retorno imediato e útil.
- Propiciar aos *designers* de *software* IHCs que também explorem o potencial e funcionalidade de CG.
- Criar cultura de utilização dos conceitos de IHC no desenvolvimento de *software* em 3D.
- Atender a rápida evolução tecnológica.

O grau de complexidade em 3D é alto e o desenvolvimento de aplicativos 3D requer o equacionamento de questões relativas à interação entre os objetos como: forma mais adequada de estruturar a cena, problemas de visibilidade, oclusão, iluminação adequada, controle dos atributos e de colisão entre objetos, suporte a diversos dispositivos gráficos, programação independente dos sistemas operacionais, seleção com várias visões e suporte a agrupamento de objetos.

Por exemplo, técnicas que usem grau de opacidade eliminam ou minimizam ao extremo a oclusão e são bastante exploradas em jogos [31]. Esta e outras técnicas são passíveis de adaptação para plataformas de *software* aplicativo e foram estudadas para composição do protótipo do *3DbyStep*.

Interação com objetos 3D

Em regra geral, os graus de liberdade da tarefa devem combinar com os graus de liberdade do dispositivo da interação. Um objeto 3D instanciado tem 6 graus de liberdade e o dispositivo de interface usual, *mouse*, somente 2. A posição relativa em 2D de um *mouse*, por exemplo, não é adequada para especificar o posicionamento em 3D. Uma vez que o número de funções excede o número de controles, a interação torna-se complicada e, até mesmo, não natural ou arbitrária.

2.2 Conceitos de instanciação de objetos 3D

Aplicações de manipulação de objetos 3D, comumente, permitem a instanciação, seleção, rotação, alteração do tamanho e deslocamento de objetos para outra posição na cena. Para implementar estas funcionalidades foram utilizados grafos de cena,

técnicas para especificar rotação de objetos, descritos nos itens 2.2.1 e 2.2.2, respectivamente, e conceitos associados a animação, interpolação e interação de objetos, descritos nos itens 2.2.3, 2.2.4 e 2.2.5.

2.2.1 Criação de grafo de cena

Um grafo de cena é uma estrutura de dados que representa a disposição de objetos em uma cena virtual. Ele tem a forma de uma coleção dos nós em um grafo ou estrutura de árvore, onde cada nó pode ter vários filhos, mas geralmente um único pai. Assim, uma operação aplicada a um grupo de nós propaga, automaticamente, seu efeito a todos os seus descendentes.

Um grafo de cena é um recurso muito útil na construção de aplicativos gráficos 3D. Uma funcionalidade comum é a habilidade de agrupar objetos relacionados em um objeto composto e aplicar transformações a esse grupo (por exemplo, mover, transformar e selecionar), como se fosse um único objeto.

É muito importante observar os requerimentos de memória ao projetar um grafo de cena, especialmente em aplicações grandes, o que foi feito neste trabalho. Para tal, os grafos mantêm na memória somente uma cópia de cada tipo de dado; todos os nós similares, no grafo de cena, fazem referência a estas cópias, através do processo de instanciação de objetos [32].

A instanciação de um objeto é um processo básico de aplicativos de manipulação de objetos em 3D e consiste em associar uma série de propriedades (como, por exemplo, material, textura e dimensões) a um objeto canônico (forma previamente definida). Para inclusão de um objeto em uma cena é preciso especificar, minimamente, uma transformação linear afim, que proverá sua posição, orientação e dimensões, no sistema de referência adequado. A forma usual de representar, matematicamente, uma transformação linear afim é com uma matriz 4 x 4, que aplicada a um objeto 3D, isto é, a um conjunto de pontos, gera o deslocamento destes pontos no espaço.

O grafo de cena deve ser concebido de forma que os objetos instanciados previamente não tenham uma segunda instanciação. A cada chamada ao objeto uma nova cópia deste deve ser armazenada, junto com suas propriedades de rotação, translação, escala e cor [1], por exemplo. A implementação desta estrutura de dados deve utilizar mecanismo para cópia das referências de memória.

As tecnologias VRML [33] e OpenSceneGraph [34], por exemplo, utilizam os processos de criação de grafo de cena na composição de suas funcionalidades.

Grafos de cena foram utilizados neste trabalho, não só para construção da estrutura dos objetos da cena, mas também na construção da estrutura de animações, descrita na seção 2.2.4.

2.2.2 Rotação de objetos com arcball

Para rotação de objetos 3D pode-se citar a técnica dos ângulos de Euler [1] e *ArcBall* [35], sendo esta última a mais comum. Simples e intuitiva, ela se propõe a girar objetos, acompanhando o movimento de seleção e arraste da esfera imaginária que o envolve, como ilustrado na Figura 2.6.

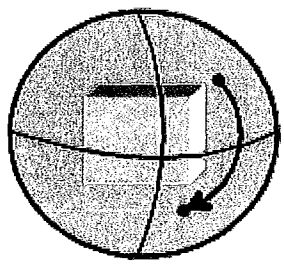


Figura 2.6: *Esquema do ArcBall.*

Dada a esfera da Figura 2.7, seja **C** o centro da esfera imaginária e A e B dois pontos sobre a mesma. Então, o eixo de rotação passa por **C** e tem direção dada por $\vec{v} = (A - C) \times (B - C)$. Por sua vez, o ângulo de rotação α é dado por $\sin^{-1} \hat{v}$, onde $\hat{v}$ é o $\vec{v}$ normalizado, isto é, $\hat{v} = \frac{\vec{v}}{|\vec{v}|}$.

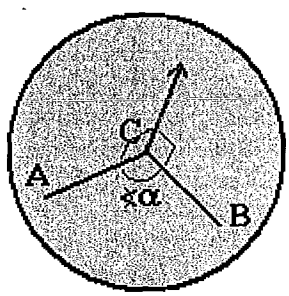


Figura 2.7: *Base das rotações.*
(Fonte: http://rainwarrior.thenoos.net/dragon/arcball.html)

2.2.3 Criação de animações com quadros-chave

Animação é uma funcionalidade desejável em uma apresentação e *Keyframing* é o método mais usado para sua criação.

Não é obrigatório que haja sempre movimento para existir animação. A ilusão de movimento pode ser obtida, por exemplo, através de mudanças na cor ou na forma dos objetos e no movimento ou orientação do ponto de vista de um observador.

Estruturas de dados para animação são meios de armazenar dados sobre os quadros-chave, *keyframes*, bem como formas de interpolá-los.

Um quadro chave, *keyframe*, é uma instância de tempo para o qual propriedades dos objetos da cena, por exemplo, cor, textura, posição e orientação são conhecidas.

A especificação de uma animação com *keyframes* pode ser mais ou menos complexa, de acordo com a propriedade que se quer animar. Por exemplo, a animação da posição e orientação de objetos rígidos (Figura 2.8 (a)) requer a determinação dos quadros-chave, da propriedade e da função de interpolação a esta propriedade, enquanto a animação de objetos articulados (Figura 2.8 (b)) requer a animação dos *bones* [11].

Keyframing consiste em um processo pelo qual objetos são posicionados em quadros-chave e um algoritmo de animação interpola os quadros intermediários (*in-between frames*), para inferir o valor da propriedade entre esses quadros-chave. No contexto deste trabalho, é o movimento de objetos na cena virtual, em trajetórias pré-estipuladas em um intervalo de tempo.

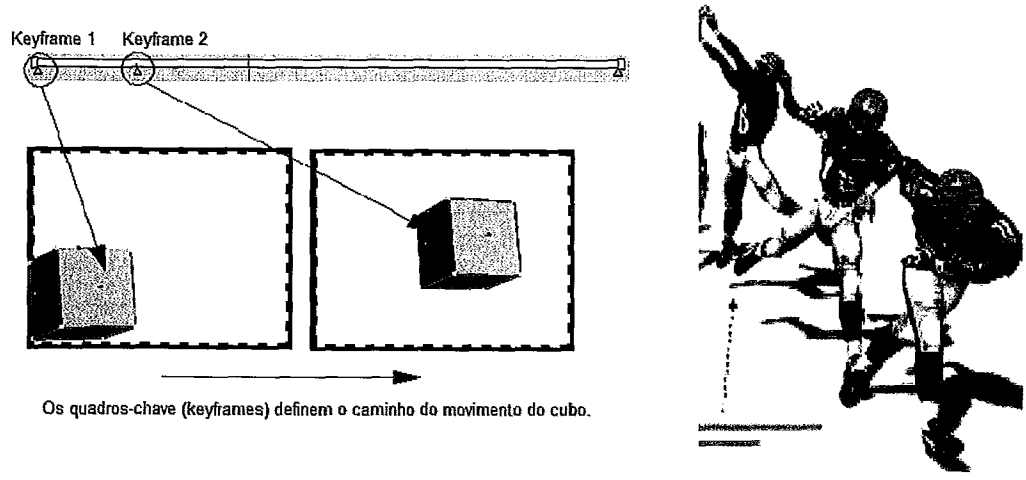


Figura 2.8: *Exemplo de movimento rígido (a) e de movimento articulado (b).*

Um objeto rígido é um objeto que não permite alterações que deformem sua estrutura, mesmo quando submetido a eventos externos.

Um objeto articulado é um objeto composto de segmentos conectados ou ligações, cujo movimento de um segmento é restringido de alguma forma pelos outros.

Somente a animação de objetos rígidos é objeto de estudo nesta pesquisa.

2.2.4 Interpolação

A simulação do movimento de um objeto rígido requer, pelo menos, dois *keyframes*, na linha de tempo daquele objeto, para os quais se especifica sua posição e orientação. Assim, uma série de *keyframes* pode ser entendida como a trajetória do objeto na cena.

A Figura 2.9 apresenta os pontos relativos a interpolação de um objeto 3D no espaço, em uma direção até a metade do número de *frames* e em outra até o último *frame*. Este exemplo, extraído do *3DbyStep*, utiliza curva B-Spline cúbica interpoladora [36].

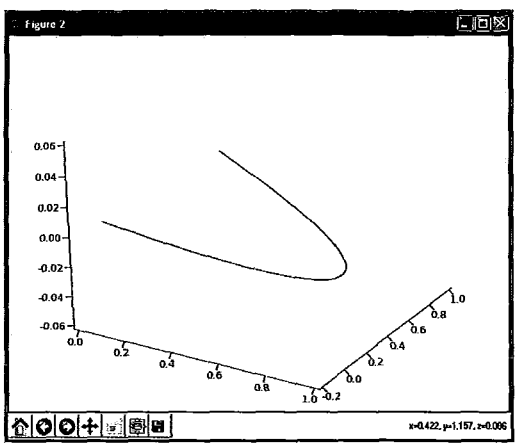


Figura 2.9: *Exemplo de curva de interpolação no espaço 3D.*

Splines

A base de Bézier [37] [1] não é propria para modelagem de curvas longas. Neste caso utiliza-se como base alternativa, curvas *Spline* (ver Figura 2.10). Uma *Spline* é composta da junção de curvas mais simples, obedecendo a algum critério de suavidade.

Para *Splines* polinomiais, essas curvas são polinomiais, de um dado grau **d**, sendo que, normalmente, o critério de suavidade consiste em requerer continuidade algébrica C^{d-1} [38].

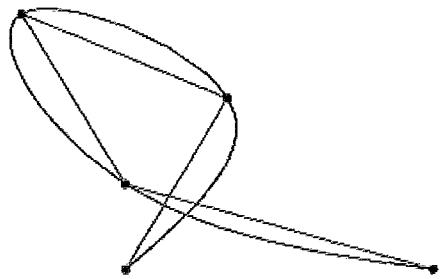


Figura 2.10: *Exemplo de curva de Spline.*

Neste trabalho, utilizou-se curvas *splines* cúbicas interpoladoras, ou seja, curvas paramétricas de grau 3, especificadas a partir de pontos de controle, que devem ser interpolados [36].

Em um ponto x em condição natural, a expressão genérica para i-ésima spline cúbica interpoladora em C^2 pode ser encontrada utilizando a fórmula abaixo:

Seja $S : [a, b] \rightarrow \mathbb{R}$ uma Spline polinomial, composta de trechos polinomiais
$P_i : [t_i, t_{i+1}) \rightarrow \mathbb{R}$, onde
$a = t_0 < t_1 < ... < t_{k-2} < t_{k-1} = b.$
Ou seja,
$S(t) = P_0(t), t_0 \leqslant t < t_1,$
$S(t) = P_1(t), t_1 \leqslant t < t_2,$
.
.
.

$S(t) = P_{k-2}(t), t_{k-2} \leqslant t \leqslant t_{k-1}.$
Os k pontos t_i são chamados nós. O vetor $t = (t_0, ..., t_{k-1})$ é chamado de vetor nó da Spline.

$$S_i(x) = \frac{z_i(x - t_{i-1})^3}{6h_i} + \frac{z_{i-1}(t_i - x)^3}{6h_i} + [\frac{f(t_i)}{h_i} - \frac{z_i h_i}{6}](x - t_{i-1}) + [\frac{f(t_{i-1})}{h_i} - \frac{z_{i-1}h_i}{6}](t_i - x) \tag{2.1}$$

onde:

$z_i = f''(t_i)$ são os valores das 2^a derivada no i-ésimo nó;

$h_i = t_i - t_{i-1}$;

$f(t_i)$ são os valores da função no i-ésimo nó.

Capítulo 3

Arquitetura Geral

Neste capítulo são apresentados os requisitos e estrutura do perfil de ferramenta para autoria de apresentações 3D.

Uma apresentação é um meio de mostrar uma idéia, um produto, um serviço, entre outros, sendo, costumeiramente, organizada em tópicos que facilitem o entendimento. Neste trabalho, dá-se o nome de apresentação 3D ao resultado da organização de conteúdo multimídia, tal como imagem, modelos 3D e som, de forma a ser utilizada como auxílio em conteúdos educacionais e de treinamento.

3.1 Atores

Contempla-se a utilização da ferramenta proposta por dois grupos distintos de usuários, segundo os papéis de:

- **Instrutor**: tem acesso completo e é detentor do conhecimento de negócio que se quer sistematizar. Ele organiza os modelos e textos, utilizando uma ferramenta de autoria e disponibiliza este material para uso posterior pelos treinandos;

- **Treinando**: tem acesso a um tocador de apresentações, capaz de visualizar o material organizado pelo instrutor, podendo acessá-lo segundo necessidade e preferência pessoal.

  Um tocador de apresentações (*player stand-alone*) é um componente semelhante ao editor de animações, na medida em que implementa parte de suas

funcionalidades. Normalmente é dotado de uma interface mais simples com botões para tocar, parar, avançar e retroceder.

3.2 Estrutura das apresentações 3D

Dentro do contexto de apresentações 3D, busca-se combinar hipertexto e hipermídia [39], para montar treinamentos (ver Figura 3.1 (a) e (b)).

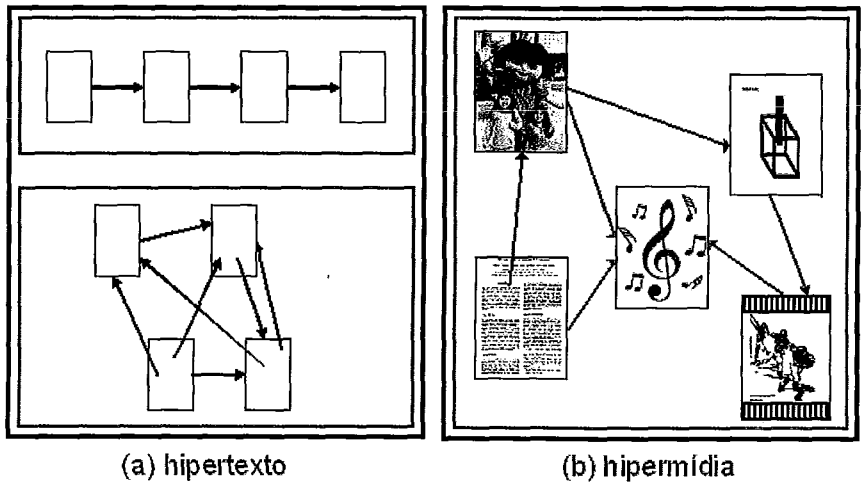


Figura 3.1: *Representação visual para hipertexto e para hipermídia.*

Em uma forma geral, considera-se que apresentações 3D são compostas de uma seqüência de quadros, *slides*, onde cada quadro pode ser constituído de uma combinação de elementos visuais dispostos no espaço - eventualmente movimentados por meio de animações. Sendo o funcionamento interativo, existe certo grau de liberdade na ordem cronológica na qual os elementos visuais serão apresentados e documentados. O relacionamento entre as entidades básicas do modelo, para compor uma apresentação, é ilustrado na Figura 3.2.

3.2.1 Modelo conceitual

Os principais elementos que compõem uma apresentação de autoria em 3D são *elementos visuais* e *animações.*

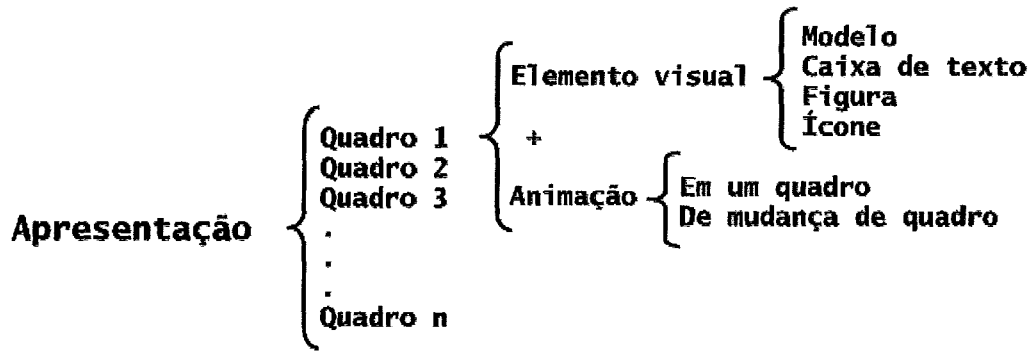


Figura 3.2: *Elementos e funcionalidades que compõem uma apresentação 3D.*

Elementos visuais

A ferramenta deve possibilitar a inclusão dos seguintes elementos em qualquer ordem e quantidade:

- **Objetos 3D**: objetos tridimensionais modelados em outro *software*. Opcionalmente divididos em partes, cada qual com propriedades visuais, como cores e materiais.
- **Referências textuais**: referências de documentação, associadas a outras instâncias visuais, com atributos de estilo editáveis.
- **Figuras**: imagens raster (*jpeg*, *png*, por exemplo) ou gráficos vetoriais (formato *svg*, por exemplo).
- **Ativador de ação**: representado por ícones que, quando ativados, efetuam uma animação programada, tal como: exibir uma animação, tocar um som ou comandar a troca de quadro.

Animações

As animações podem ser iniciadas de forma manual, onde o próprio usuário comanda o início da animação, ou automática, sem necessidade de intervenção do usuário. Podem ser divididas em dois tipos básicos:

- **Animações de elementos visuais em um determinado quadro**, podendo ser:

Animações simples - Coloca um elemento visual em evidência, move o mesmo em volta da tela ou faz com que ele apareça e desapareça após um determinado tempo.

Animações compostas - Várias animações executadas em seqüência, juntas, ou seguindo uma ordem arbitrária.

Ao fim de cada animação de elementos visuais em um determinado quadro, o mesmo pode ficar em dois estados. Ou é mantido no estado final ou retorna ao inicial.

- **Animações de passagem de quadros**, onde se carrega outro quadro da apresentação, podendo ser:

  Seqüência trivial, onde os quadros são apresentados na ordem em que foram incluídos na apresentação.

  Transição entre quadros, onde a navegação livre entre quadros permite ao usuário determinar a ordem exata em que os quadros serão apresentados.

  Para cada passagem de quadro, pode-se ter:

  Transição simples - Não há animação na transição entre quadros.

  Transição animada - Associar um efeito de animação na transição entre quadros.

3.3 Recursos de interface

Aplicações que sigam o perfil de ferramenta proposto neste trabalho devem observar os recursos abaixo relacionados para composição do projeto de interface. Aspectos relacionados a atratividade do sistema, estilos de interação e modos de funcionamento, são tratados, a seguir:

3.3.1 Atratividade do sistema

Algumas regras gerais que devem observadas para melhorar a atratividade do sistema são listadas abaixo:

- Bloquear ou aumentar o grau de opacidade de menus e itens que não são compatíveis com a função que está sendo executada.

- Habilitar e desabilitar as barras de tarefas, segundo preferências do usuário ou quando uma funcionalidade não faz sentido no contexto atual.

- Definir teclas de atalho para as funções mais executadas (sugestão no Anexo A).

- Manter uma coerência no padrão de ícone adotado (por exemplo, tamanho, formato, estilo, escala de cor).

- Minimizar a utilização de manuais, *help online* e cursos de treinamento, para atender ao "príncipio da expectativa" (citado no Capítulo 2).

- Implementar o mecanismo de menu de contexto (sugestão no Anexo A), para atender ao "princípio de implicação" (citado no Capítulo 2). Neste mecanismo, somente as funcionalidades adequadas a cada modo de funcionamento (por exemplo, funções básicas, documentação, desenho e apresentação) devem estar habilitadas.

- Atualizar simultâneamente várias janelas em um único terminal (mecanismo de saídas concorrentes [26]). Um exemplo disso é que ao abrir um arquivo com modelo 3D, pode-se atualizar simultaneamente a janela de visualização e a janela hierárquica da aplicação. Outro exemplo é a utilização simultânea de efeitos acústico e visual, em processos de controle de erro ou na apresentação da documentação do modelo.

- Compor diálogos estreitamente relacionados, por exemplo, uso de uma palheta de cores para troca de cor da pintura de um modelo.

- Preocupar-se com a consistência da interface, com repetição de fatores de tipografia, simbolismo, cor, layout e seqüênciamento, aplicando o conceito de "Identidade conjunta da interface" [23].

- Prover soluções para apoio à memória [29]. Por exemplo, a utilização da lista de <<Últimos arquivos abertos>> e a disponibilização do diretório corrente do sistema operacional na função <<Abrir Arquivo>>.

- Prover representação contínua do objeto de interesse [40].
- Privilegiar ações físicas ao invés de comandos que envolvam sintaxe complexa (movimento e seleção por *mouse* ou *button*) [40].
- Prover operações incrementais, rápidas e reversíveis, cujo resultado reflete-se imediatamente na representação do objeto para o usuário [40].
- Permitir a operação do sistema com um mínimo de conhecimento, tratando a complexidade de forma estruturada por camadas [40].
- Implementar a manipulação direta de conteúdo com o paradigma de interação mais usual (seqüência objeto-verbo).

3.3.2 Estilos de interação

Interfaces atuais tendem a combinar diferentes estilos de interação. Várias funcionalidades podem ser implementadas em mais de um estilo de interação. Por exemplo, paradigmas de manipulação direta (WIMP), interfaces tridimensionais e a técnica WYSIWYG podem ser combinados em uma mesma interface.

Este trabalho utilizou diferentes formas de interação para atender às funções definidas nos requisitos do perfil proposto. Na Tabela 3.1, foram listadas funcionalidades do perfil de aplicação, com seus pré-requisitos e algumas formas de interação mapeadas.

Funcionalidade	**Pré-requisito**	**Menu**	**Botão**	**Drag and drop**	**Manipulação direta**
Instanciação	-	X	X	X	X
Seleção	Carga	X	X		X
Modificação	Seleção	X	X	X	X
Visualização	Carga		X	X	
Navegação	Visualização	X	X		X
Cópia	Seleção	X	X		

Tabela 3.1: Relação entre funcionalidades e interfaces.

Devido à especificação de ser direcionado para público não especialista, não foram mapeadas atividades com utilização de linguagem de comandos.

3.3.3 Grupos de funcionalidades e modos de operação

Para composição das barras de tarefas e do menu de atividades, foram feitos agrupamento de funções correlatas.

A Tabela 3.2 apresenta uma classificação para composição de grupos de funcionalidades, descrevendo o tipo de função de cada um deles.

Grupo de funcionalidades	**Descrição**
1.Básico	Reune funções básicas de manipulação de arquivos.
2.Documentação	Permite a descrição do conteúdo visual dos modelos 3D importados pelo *software*.
3.Animação	Disponibiliza os efeitos a serem aplicados na cena, nos modelos 3D ou em parte dos modelos 3D.
4.Desenho	Permite a transformação dos modelos segundo os atributos de cor, textura e dimensão. Além destes, este modulo disponibiliza os recursos de rotação, translação e escala.
5.Modo de apresentação	Permite a execução da cena construída dentro do aplicativo.

Tabela 3.2: Grupos de funcionalidades identificados para criação de uma apresentação.

Há ainda, dois modos de trabalho disponíveis. São eles:

- **Modo de apresentação** - Apresenta cena construída dentro do aplicativo.
- **Modo de edição** - Disponibiliza editor para inclusão de textos.

Capítulo 4

Implementação

Com o objetivo de avaliar a viabilidade e validade dos conceitos propostos para um perfil de aplicativo de autoria em 3D, foi desenvolvido um protótipo da ferramenta *3DbyStep*. Neste capítulo, são apresentados detalhes de implementação (divididos em ambiente de implementação e funcionalidades do protótipo) e um estudo comparativo entre ferramentas para autoria de conteúdo 3D.

4.1 Ambiente de implementação

A plataforma de *hardware* e *software* básico foi selecionada segundo critérios de expansibilidade, independência de plataforma e mercadológicos. O ambiente é composto de tecnologias bastante conhecidas e em expansão, possibilita o uso multiplataforma e é obtido a um custo bastante acessível.

A escolha do *software* básico empregado na plataforma foi guiada pelos seguintes requisitos:

- Sem custo para aquisição e/ou manutenção;
- Aderência a padrões internacionais, para possibilitar fácil implementação;
- Versões estáveis;
- Comunidade de usuários bem estabelecida, para garantir sua longevidade.

O protótipo foi implementado e testado nos sistemas operacionais Windows XP e Linux. O *hardware* mínimo previsto está descrito no Anexo B.

A plataforma escolhida para o desenvolvimento do protótipo *3DbyStep* foi composta da linguagem Python [41], das bibliotecas Qt (*toolkit* de desenvolvimento) [42], OpenGL (para manipulação de objetos 3D) [43] e *Pygame* (para utilização de som [44]).

O desenvolvimento foi apoiado pelo aplicativo de documentação Doxygen [45] e uso repositório de código no Google Code.

A relação entre as ferramentas da plataforma de desenvolvimento do protótipo, dentro das tecnologias as quais pertencem é mostrada na Figura 4.1.

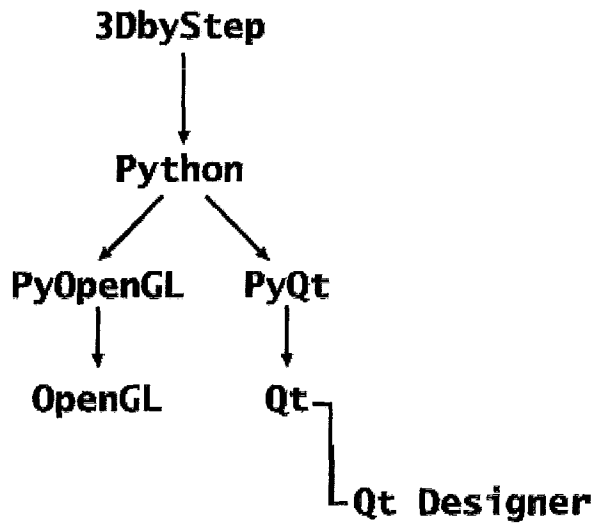


Figura 4.1: *Relacionamento entre as ferramentas.*

4.1.1 Python

Python é uma linguagem de programação com uma comunidade de usuários estabelecida, fácil de aprender e que vem se firmando como uma linguagem interpretada bem aceita. Uma das grandes vantagens do Python é ser facilmente estendida com funções em C ou C++.

Python é frequentemente comparado com Tcl, Perl, Ruby, Scheme, Lua ou Java. Suas principais características são:

- Possui sintaxe muito clara e de fácil entendimento.
- Propicia orientação a objeto de forma natural.
- Possibilita grande modularidade.

- Permite módulos e extensões escritos em C, C++ (ou Java para Jython, ou linguagens .NET para IronPython).
- Tem características avançadas de linguagem O.O. como meta-classes, decoradores, tipagem forte e dinâmica (*duck typing*).
- Possui vasta biblioteca de módulos inclusa na distribuição padrão (*batteries included*).

4.1.2 Qt

Qt é um produto desenvolvido pela Trolltech [42] e consiste de um *toolkit* de desenvolvimento de aplicações em C++, multiplataforma que, além das funcionalidades relacionadas à montagem de interfaces gráficas, oferece funções de acesso a bancos de dados, rede e gerenciamento de arquivos.

Qt provê recursos que habilitam a integração das aplicações com o ambiente de *desktop* preferido do usuário. Permite a visualização no aspecto nativo de cada plataforma suportada (Windows, Linux, MAC) e este estilo *default* pode ser sobreposto.

Qt permite o tratamento modularizado de eventos, através do mecanismo de sinais ("signals") e encaixes ("slots"). Em particular, um objeto (*"signal"*) do Qt pode ser associado a um número grande de funcionalidades (*"slots"*).

PyQt

PyQt é um módulo que permite o acesso à biblioteca Qt (*Q toolkit*) em Python. A versão 4.2.2 do *software* PyQt [42] foi utilizada nesse trabalho.

Qt Designer

A ferramenta *Qt Designer*, integrante do *toolkit* Qt, permite o desenho e carga de GUIs, a partir de componentes do QT. Segue conceitos de SGIU e foi utilizada no projeto de interface, para dar dinamismo e flexibilidade ao desenvolvimento da aplicação.

O *Qt Designer* possibilita o desenho da GUI visualmente e salva o arquivo em um formato XML, que é convertido por outro programa em código fonte Python,

criando uma classe que implementa o *layout* da janela, já com os métodos e conexões disponíveis.

O *Qt Designer* inclui funções bastante interessantes como, por exemplo, modo *preview*, *layout* automático, suporte para *widgets* customizados, biblioteca de ícones e editor avançado de propriedades.

O recurso de biblioteca de ícones do *Qt Designer* permite a carga dos formatos como png e jpeg, que foram utilizados no protótipo. A imagem pode residir em arquivos no formato .py, mas o conteúdo original precisa gerar um arquivo .qrc para listá-lo. O utilitário <<pyrcc4>> gera um módulo Python para carga dos ícones a partir de arquivo com extensão .qrc. Este módulo deve ser importado na aplicação, antes de ser utilizado, para que os ícones possam ser reconhecidos.

Após a criação da interface com o *Qt Designer*, o passo seguinte é a geração do código correspondente ao conteúdo visual destas telas. Utiliza-se o utilitário <<pyuic4>>, para criar interfaces para PyQt.

4.1.3 OpenGL

OpenGL (*Open Graphics Library*) é uma API procedural, multiplataforma e multilinguagem, para a escrita de aplicações capazes de produzir e manipular gráficos 2D e 3D. A principal vantagem na sua utilização é o aproveitamento, de forma transparente, dos recursos de aceleração disponíveis em placas gráficas modernas. Entretanto, por se tratar de uma API voltada para desenho, obriga o desenvolvedor da aplicação a informar todos os passos necessários a renderização de uma cena. Isso traz certo grau de liberdade na implementação das rotinas de renderização, mas obriga o desenvolvedor a ter um bom conhecimento de *pipelines* gráficos [46].

O protótipo foi desenvolvido e testado com OpenGL versão 1.1 (no Windows) e OpenGL versão 2.1.2 (Linux).

PyOpenGL

PyOpenGL é um módulo que permite o acesso à biblioteca OpenGL e APIs relacionadas em Python. PyOpenGL interage com um largo número de bibliotecas externas para Python.

4.1.4 Documentação de código

Documentar o código-fonte facilita a manutenção do sistema, disponibiliza documentação profissional de bibliotecas para usuários do *software* e contribui com a gestão do conhecimento. A documentação foi feita em **Doxygen**, um sistema flexível de documentação de código-fonte, multilinguagem (suporta as linguagens C++, C, Java, Objective-C, Python, IDL, PHP, C# e D), multi-plataforma (suporta as plataformas GNU/Linux (Unix), Mac OS X, Windows) e com múltiplas saídas (suporta saídas HTML, Latex, RTF, PostScript, PDF, XML, compressed HTML (CHM), Unix man pages) [45].

Suas principais características são:

- Busca por *tags* de documentação no código-fonte.
- Extrai informação de códigos-fonte não documentados.
- Gera automaticamente diversos gráficos UML de acordo com o relacionamento das entidades envolvidas.

O serviço SVN do Google Code foi utilizado para versionamento do código-fonte. As funcionalidades implementadas em cada versão publicada foram descritas no arquivo *functions aaa-mm-dd.txt*, importado junto da versão.

4.2 Funcionalidades do protótipo

Esta seção descreve as principais funcionalidades do protótipo, incluindo aspectos relacionados a sua interface e alguns detalhes de implementação.

4.2.1 Estrutura da janela principal

O sistema utiliza uma interface que contêm, uma janela principal, uma janela com editor de documentação e algumas janelas de diálogo.

O aspecto geral da janela principal é ilustrado na Figura 4.2. Ela possui 2 componentes (paineis) em sua área central, 5 barras de ferramentas e 1 barra de *menus*, na sua parte superior. Os componentes são:

- Painel de visualização do quadro corrente.

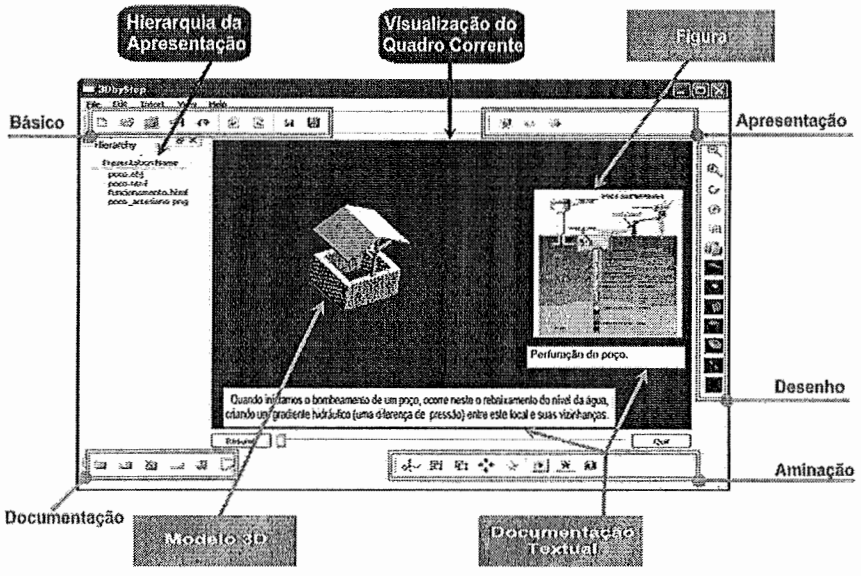


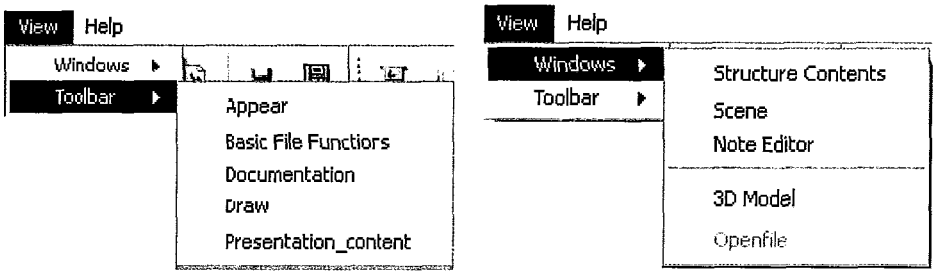
Figura 4.2: *Aspecto geral da janela principal.*
(Fonte: Nogueira, 2007a)

- Painel com lista hierárquica dos elementos da apresentação.

O painel de visualização do quadro corrente é a área mais importante da janela principal. É usado para exibir e permitir o posicionamento dos objetos da cena, através de manipulação direta.

À esquerda, existe um painel contendo uma lista hierárquica dos elementos que constituem a apresentação. Esta lista permite acesso à estrutura lógica do conteúdo da apresentação e a seleção de objetos ocultos no painel de visualização. Observe que este painel, assim como as barras de ferramentas, pode ser destacado da janela principal, tornando-se uma janela independente.

Das cinco barras de ferramentas móveis, 2 encontram-se posicionadas do lado superior (manipulação básica e apresentação de conteúdo) e, as 3 demais, no lado inferior (documentação de modelo, transformações e desenho) da janela principal, podendo ser ativadas ou não, através da opção <<View>> do *menu*, na janela principal (Ver Figuras 4.3 (a) e (b)). Opcionalmente, pode-se ativar ou não as barras de ferramentas (*toolbars*) pelo menu de contexto (clicar com o botão direito do mouse na área das barras).



Barras de ferramentas (a) Janelas móveis (b)

Figura 4.3: *Controle de visibilidade.*

4.2.2 Customização de cursores e iconografia

Cursores

Foi utilizado o recurso de customização de cursores, nas movimentações e operações executadas na interface, para maior compreensão da tarefa e suas nuances (por exemplo, tempo para execução, possíveis controles adicionais). Foram usados apenas ícones de cursores nativos do Qt [47].

A Tabela 4.1 contém os cursores utilizados na aplicação e o modo de ação associado a cada um deles.

Iconografia

Para que haja uma interação satisfatória, a imagem do sistema criada pelo *designer* deve refletir as intenções e objetivos do usuário em termos de comandos e funções do sistema. Com esse propósito, foi feita uma avaliação heurística (ver Anexo C) aplicada à iconografia, seguindo as recomendações de Nielsen, apresentadas no item 2.1.4.

Houve uma preocupação com o fácil reconhecimento visual dos ícones. Os desenhos foram escolhidos de forma a lembrar ícones de aplicativos muito populares.

Os ícones utilizados na interface foram editados para preservar a cor de fundo geral do aplicativo, margeando as figuras com um contorno branco, para melhor destaque.

Além disso, sempre que o ícone proposto é composto de outros pré-existentes, ele foi cuidadosamente montado.

Cursor	Descrição	Modo de operação
	Cursor seta padrão.	Modo geral de operação.
	Cursor em cruz, tipicamente usado para auxiliar o usuário a selecionar um ponto da tela. Usado para seleção de modelo na janela de visualização de objetos.	Modo de seleção.
	Uma ampulheta ou cursor de relógio, normalmente mostra a duração das operações que impeça o usuário de interagir com a aplicação. Usado na carga dos modelos.	Modo de espera.
	Cursor representando uma mão fechada, tipicamente utilizado para indicar que a operação de arraste está em processo que envolve rolagem. Usado para arraste do *mouse* para rotacionar objeto.	Modo de arraste.

Tabela 4.1: Cursores nativos do Qt utilizados no protótipo.

4.2.3 Edição da cena

Manipulação direta

Para implementar a manipulação direta de conteúdo na cena, no painel de visualização do quadro corrente, foram necessários mecanismos para seleção, rotação, translação e escala de objetos. Além disso, um objeto selecionado pode ser recortado (*cut*), colado (*paste*), agrupado ou desagrupado.

As funcionalidades das teclas de atalho e do menu de contexto estão disponíveis no Anexo A.

A seleção dos objetos da cena é feita com a técnica de *picking* do OpenGL [46]. No *3DbyStep*, para prover um *feedback* visual da seleção de objetos na cena, os elementos selecionados são apresentados com suas respectivas caixas delimitadoras (*bounding boxes*). Uma caixa delimitadora, exibida na Figura 4.4, é o menor pa-

ralelepípedo que envolve uma determinada geometria. Para construção da caixa delimitadora, calcula-se o centro do modelo e o tamanho do segmento de reta da caixa, e esses são passados como parâmetro para rotina que calcula os vértices da caixa delimitadora e a desenha.

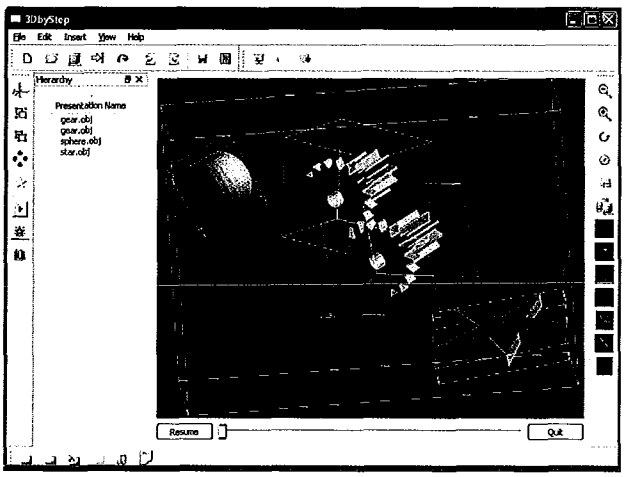


Figura 4.4: *Exemplo de caixas limitantes extraído do* 3DbyStep.

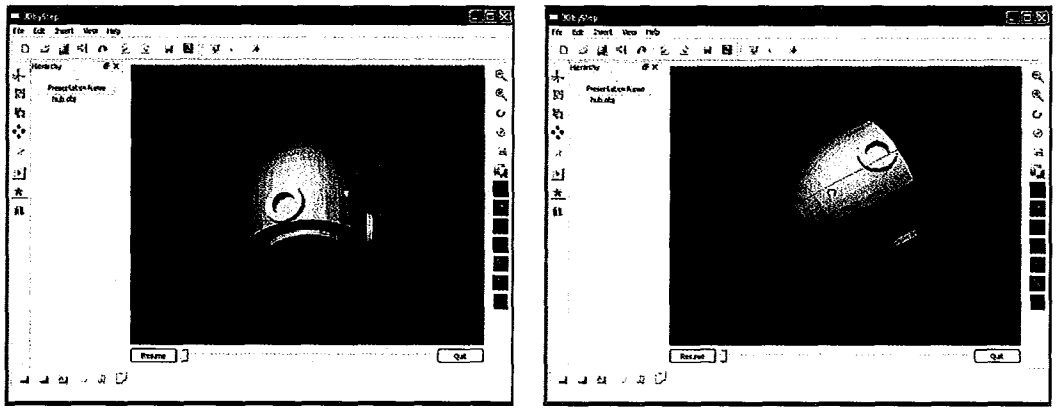
Para implementar a rotação de modelos (ver Figura 4.5) foi utiliza a técnica de ArcBall [35]. O *Arcball* tem que ser aproximadamente do tamanho do objeto. O centro da esfera do *Arcball* pode ser o centro da caixa limitante e o diâmetro o tamanho da maior diagonal da caixa. Com estas coordenadas definidas, o raio do *ArcBall* é estimado como metade da maior distância entre dois vértices do *Bounding Box* do objeto 3D. Esse tamanho de esfera para o *Arcball* não gera oclusões consideráveis do objeto, além de proporcionar bom acompanhamento visual das rotações, através da esfera que os envolve.

Undo e redo

O protótipo provê a funcionalidade de *undo* (desfazer) e *redo* (refazer) para todas as operações de manipulação direta.

Carga

Elementos visuais são carregados na janela de visualização, tendo suas partes descritas na janela de hierarquia da apresentação (à esquerda). Partes podem ser obtidas e



(a) (b)

Figura 4.5: *Exemplo de rotação com Arcball, extraído do* 3DbyStep.

selecionadas na própria estrutura hierárquica dos elementos e cada peça selecionada é destacada da janela de visualização.

O *3DbyStep* importa imagens (nos formatos png e jpeg), documentação textual (textos explicativos) em html e modelos 3D, oriundos de modelagem externa (nos formatos OBJ e OFF). Atualmente, para documentar modelos 3D complexos, com partes separadas, deve-se salvar os arquivos de cada parte individualmente.

Animação

O *3DbyStep* prevê animações simples para modelos 3D, textos ou figuras.

A solução implementada no *3DbyStep* provê rotinas, parametrizáveis, permitindo seu uso em diversos outros contextos. Novas animações podem ser implementadas facilmente, ou seja, o *3DbyStep* foi desenvolvido com uma arquitetura escalável.

Atualmente, o protótipo permite a utilização de alguns tipos de animações, que podem ser terminadas no estado do *frame* inicial ou final e 3 velocidades de execução da apresentação.

Os tipos de animação (em um objeto selecionado) são:

1. Translada e rotaciona (*script* anim_TransRot);
2. Traz para perto (*script* anim_bringObj);
3. Move para o canto superior direito (*script* anim_topRight);
4. Move para o canto inferior direito (*script* anim_botRight);

5. Move para o canto superior esquerdo (*script* anim_topLeft);

6. Move para o canto inferior esquerdo (*script* anim_botLeft).

É possível incorporar novas animações, sendo necessário para isso incluí-las em uma janela de diálogo (ver 4.6) e criar um *script* para o seu funcionamento.

O botão <<Definir animação>> abre a janela de diálogo de animação, Figura 4.6, onde o tipo, evento de ocorrência final, *frames* inicial e final e velocidade de execução da animação são parametrizáveis.

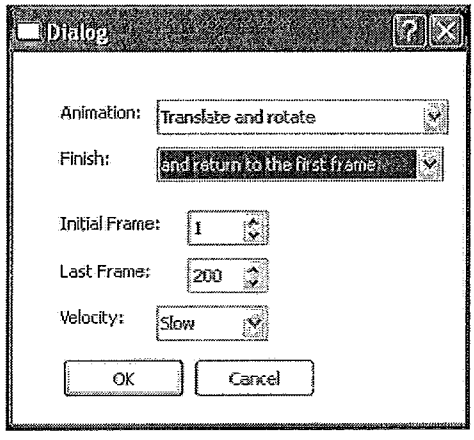


Figura 4.6: *Janela de diálogo de animação.*

Todos os *scripts* de animação são importados a partir do método `init` da classe `animations`, arquivo `__init__.py` do mesmo diretório.

Para criação das animações, o *3dByStep* define uma estrutura de dados onde cada *keyframe* contém as seguintes informações:

- Número dos *frames* inicial e final;
- Posição do objeto (translação) em relação à origem do sistema de coordenadas;
- A orientação (rotação) do objeto;
- O tamanho (escala) do objeto.

  Segue exemplo de *script* de parametrização de animação de objeto (Rotina `anim_TransRot.py`):

```
1: import animation
2: from keyframe import PosKeyFrame, RotKeyFrame
3: def anim_TransRot(firstFrame, lastFrame, nFrames):
4:     anim = animation.Animation(
5:                 [PosKeyFrame(firstFrame, [0, 0, 0]),
6:                  PosKeyFrame(nFrames/2, [1, 0, 0]),
7:                  PosKeyFrame(lastFrame, [0, 1, 0]),
8:                  RotKeyFrame(firstFrame, [1, 1, 1, 0]),
9:                  RotKeyFrame(nFrames/2, [1, 1, 1, 180])])
10:    interptrans = anim.interpolate(nFrames, firstFrame)
11:    return anim, interptrans
```

A classe `PosKeyFrame` define um *keyframe* posicional com a operação de alteração da localização do objeto, através da passagem dos parâmetros número do *frame* e uma tupla com coordenadas (x, y, z) de uma translação, e a classe `RotKeyFrame` define um *keyframe* que representa a rotação de um objeto, utilizando para isso os parâmetros número do *frame* e uma tupla com as três coordenadas do eixo de rotação e o ângulo. Ambas fazem parte do módulo `keyframe`.

A função exemplo `anim_TransRot` recebe como parâmetros: o número do primeiro, do último *frame* e o número de *frames*. Para composição da trajetória de um objeto são determinados alguns *keyframes*, que indicam valores para sua posição e orientação no espaço.

Em seguida são calculados os *frames* intermediários(*in-between frames*) com a interpolação destes pontos de controle (*keyframes*). Internamente, `interptrans` é o resultado da interpolação é feita com uso da técnica de interpolação B-Splines, que está encapsulado no método `interpolate`, da classe `Animation`.

4.2.4 Documentação de conteúdo em 3D

A documentação do conteúdo de treinamento é composta de imagem, texto e som.

A funcionalidade para importação de texto foi construída com uso da classe `QtextEdit`, que é capaz de manipular texto em formato html. É adequada, por permitir flexibilidade na escolha e construção dos conteúdos, pela manipulação desses

conteúdos como objetos 3D.

O texto é renderizado como textura em um retângulo que gera um objeto 3D da cena. Desta forma é possível aplicar transformações lineares afim sobre estes (rotação, translação, escala e animação) 4.7.

O texto é incluído na janela de visualização do modelo como uma textura e não é limitado a área de visualização desta.

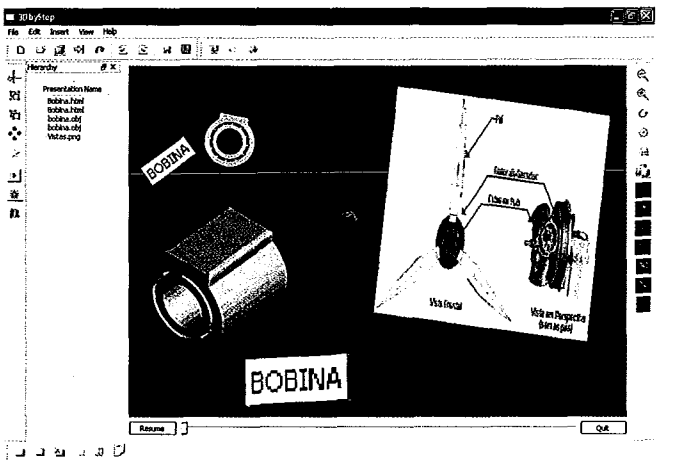


Figura 4.7: *Exemplo de documentação de conteúdo, extraído do* 3DbyStep.

A documentação por som está disponível no protótipo de forma experimental. Esta foi implementada no botão de som, com recursos do módulo *mixer* do *pygame* [44].

Editor de documentação

Para criação de documentação textual, foi incorporado ao protótipo um editor de textos, capaz de suportar customizações para estilo, tamanho e cor de fontes. O editor também permite a indentação de conteúdo, alinhamento de texto (esquerda, centralizado, direita e customizado), *cut*, *paste* de blocos de texto, além de *undo* e *redo* das últimas operações executadas.

A Figura 4.8 mostra as funcionalidades implementadas no editor de documentação do protótipo, que salva o texto em formato html.

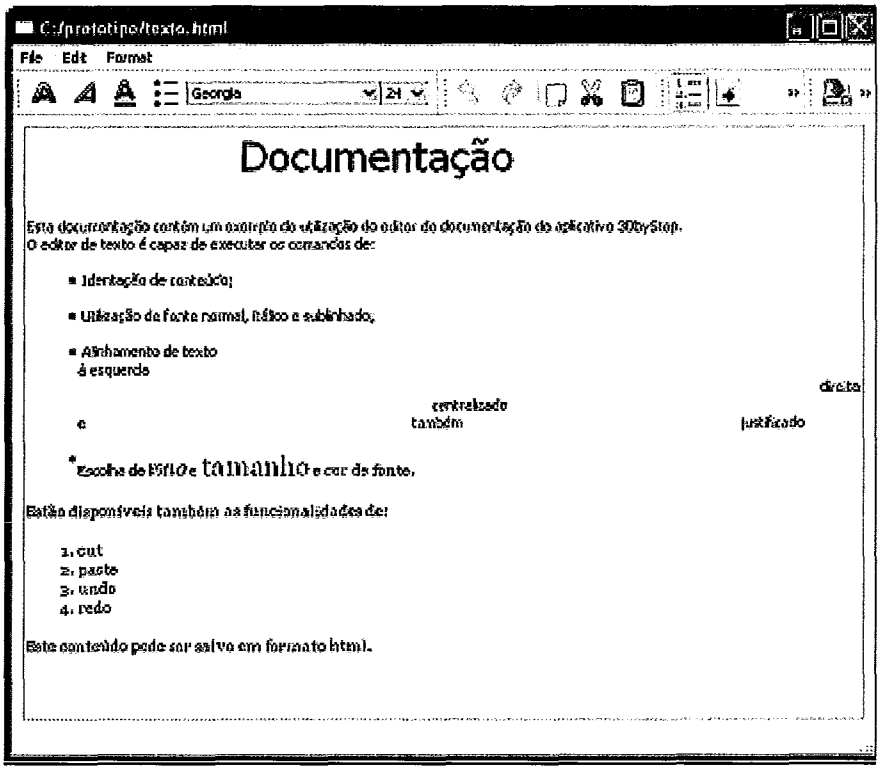


Figura 4.8: *Janela de diálogo de documentação textual.*

4.3 Estudo comparativo

Uma comparação sumária entre o `3DbyStep` e outros 2 aplicativos populares, com suporte a hipermídia, que podem ser utilizados para autoria de conteúdo 3D, está disponível na Tabela 4.2.

(*1) Formatos 3DS, AVI, BMP, Iris, JPG, OBJ, PNG, SGI Movie, TGA, e Quicktime, GIF, TIFF e MOV (Windows and Mac OS X).

(*2) Formatos AVI, BMP, Iris, JPG, PNG, SGI Movie, IFF, TGA, Quicktime, GIF, TIFF, PSD e MOV(Windows and Mac OS X), blend.

É importante observar que os 2 produtos analisados são ferramentas com anos de uso. São produtos bastante complexos e com um grande número de funcionalidades.

A característica de código não proprietário das ferramentas Blender e *3DbyStep* permitem que suas respectivas comunidades de usuários participem do seu desenvolvimento e, consequentemente, que sua aplicabilidade seja moldada às necessidades dos mesmos.

A arquitetura proposta na ferramenta Blender é a que mais se assemelha com o modelo do *3DbyStep*, contando com 2 características em comum; a forma de cons-

trução das cenas e apresentação das janelas, e também a linguagem de desenvolvimento.

Macromedia Director MX 2004	**Blender(2.36)**	**3DbyStep 1.0**
	Interface	
Complexa	Pouco amigável (Não trivial)	Bastante simplificada e amigável
	Modelagem 3D	
Mínima. Não se propõe a ser um modelador, possuindo apenas algumas poucas funcionalidades de modelagem.	Restrita Processo de modelagem trabalhoso, por não ter muitas ferramentas que facilitem esta tarefa.	Mínima. Não se propõe a ser um modelador, possuindo apenas algumas poucas funcionalidades de modelagem.
	Formatos de entrada	
AVI, JPEG, QuickTime, BMP e PDF.	Diversos (*1)	OFF, OBJ, JPEG e pickle
	Formatos de saída	
Formato proprietário DIR	Diversos (*2)	OFF, OBJ e pickle
	Outras características	
Programa proprietário; Manuais inclusos na aquisição; Previsão de atualização; Garantia de manutenção	Programa gratuito; Possui uma vasta literatura disponível na internet.	Programa gratuito.
Permite complementação com *scripts* em linguagem própria, Lingo.	Permite complementação com *scripts* em Python.	Desenvolvido em Python.

Tabela 4.2: Tabela comparativa do *3DbyStep* com os produtos Director e Blender.

Capítulo 5

Estudo de Caso

Este capítulo relata o estudo de caso realizado para validar o perfil de aplicativo proposto. Apresenta-se motivação para escolha do tema, seguida de um exemplo de autoria de conteúdo e, ao final, resultados da pesquisa de satisfação com os usuários.

5.1 Motivação para escolha do caso de uso

O perfil de aplicativo proposto permite a confecção de material para estudo de uma grande variedade de assuntos. Neste estudo de caso são utilizados equipamentos industriais da área de energia e desenvolvimento sustentável.

Compreender processos que envolvam objetos tanto em escala macro como microscópica não é trivial. Entende-se que uma ótima forma de facilitar o entendimento deste tipo de informação é o uso de ilustrações. Baseado nestas premissas, a área de energia apresenta um grande desafio, tendo sido selecionada como área tema para o estudo de caso realizado.

Através da visualização prévia consegue-se obter mais assertividade pelo melhor entendimento do assunto que está sendo tratado e uma economia real. Estudos e projeções com o recurso de visualização têm se mostrado bastante eficientes, evitando a geração de protótipos físicos ineficientes.

Algumas indústrias utilizam equipamentos de energia em suas linhas de pesquisa, existindo exemplares em produção como, por exemplo, aerogeradores, unidades GTL [48] e ônibus híbridos. Esse conjunto de equipamentos é composto de máquinas de diversos portes.

Uma das soluções da indústria de energia com equipamentos em escala macro é a geração de energia eólica, assunto-tema deste estudo de caso. Esta energia é gerada através da utilização de uma turbina eólica, comumente montada em uma torre. A turbina coleta a energia do vento, convertendo-a em eletricidade.

Sistemas de energia eólica são compostos de dúzias de aerogeradores e têm aplicabilidade tanto em solo firme quanto em regiões marítimas.

Pode-se afirmar que o uso do vento é uma das fontes de energia renovável que mais cresce no mundo, uma vez que se caracteriza por uma captação com pequeno impacto, contribuindo para preservação do ambiente ao seu redor.

A utilização da energia eólica tem tido espaço em projetos de âmbito nacional, com destaque para a Petrobras [49].

5.1.1 Equipamentos

Há pouco material disponível sobre modelagem de equipamentos da área de energia [49], sendo, muitas vezes, de uso restrito do seu fabricante. É rara a existência de conteúdo não proprietário relacionado ao tema.

Dentre os equipamentos industriais citados, os escolhidos para esse estudo são os aerogeradores, tendo sido modelados e animados neste trabalho. Cada equipamento é composto por inúmeras peças e, muitas vezes, por outros equipamentos menores.

- **Aerogeradores** [50]: Equipamentos responsáveis pela conversão da energia cinética, contida nos ventos, em energia elétrica. De modo geral, um aerogerador é constituído pela turbina eólica, multiplicador mecânico, gerador elétrico e pelos sistemas de conexão elétrica e controle. Algumas das peças que compõem um aerogerador são:

  Caixa de engrenagens: Converte baixa rotação em alta velocidade.

  Cone do rotor: Liga as pás ao eixo de baixa velocidade da turbina elétrica.

  Eixo de alta velocidade: Aciona o gerador elétrico.

  Eixo de baixa velocidade: Conecta o cone do rotor à caixa de engrenagens.

Gerador elétrico: Transforma o conjugado mecânico do rotor em eletricidade.

Nacele: Contém a caixa de engrenagens e o gerador elétrico.

Pás do rotor: Capturam a energia do vento e transfere-a para o cone do rotor.

Rotor: Retira energia cinética do vento e a converte num conjugado mecânico.

Torre: Sustenta a nacele e o rotor. Nos aerogeradores modernos atingem 40 a 60m e podem ser tubulares ou reticulares.

Turbina: É um equipamento construído para converter energia mecânica e térmica, contida em um fluido, em trabalho de eixo.

Usinas ou parques eólicos: São compostos basicamente por aerogeradores e são distribuídos de forma a se obter a potência prevista.

Os equipamentos e peças descritos têm características e tamanhos diferentes. Informações importantes para apoiar o processo de modelagem vão desde o formato de peças até a montagem do equipamento.

5.1.2 Obtenção de conteúdo

Organizar a informação é uma tarefa subjetiva, ou seja, baseada na experiência, contexto e conhecimento de quem está catalogando o conteúdo. Para melhor organização, deve-se inferir as relações mais importantes, pois o conhecimento não é constituído de simples relacionamentos lineares.

Caso haja uma categoria de usuários específica, a organização do conteúdo deve ser direcionada para este público alvo.

Para cada conjunto de equipamento, a forma de aquisição de informações é ligeiramente diferenciada. Por vezes, a descrição de equipamentos e desenhos técnicos pode ser obtida de bancos de conhecimento das empresas fornecedoras do maquinário. Livros, catálogos e manuais são fontes mais comuns, mas podem ser insuficientes ou desatualizadas e não apóiam completamente a visualização do funcionamento desses maquinários. Para preencher essa lacuna e dar maior confiabilidade

ao trabalho foram feitas entrevistas com consultores experientes. O material obtido foi filtrado e organizado.

Em uma divisão básica [2] obtém-se três conjuntos de equipamentos com ligeiras variações na forma de aquisição e análise de dados. São eles:

- **Equipamentos pequenos** ($< 1m^3$): para os quais informações e ilustrações com o formato do equipamento são facilmente encontradas em documentos como catálogos, manuais e livros e costumam conter poucas peças.
- **Equipamentos de médio porte** ($\approx 1m^3$): para os quais as informações são encontradas em documentações, porém, para um entendimento mais detalhado são necessárias informações adicionais. Costumam ter muitas peças.
- **Equipamentos de grande porte** ($> 1m^3$): contêm centenas de peças sendo impossível a visualização global do equipamento.

Uma sugestão para forma de levantamento [2] foi consolidada na Tabela 5.1

	Porte dos equipamentos		
	Pequeno	**Médio**	**Grande**
Catálogos	X	X	
Manuais	X	X	
Entrevistas estruturadas	X	X	X
Visitas		X	X
Aquisição de imagens		X	X

Tabela 5.1: Divisão de equipamentos para aquisição e análise de dados.

As entrevistas devem ser estruturadas abordando, pelo menos, a maior parte das questões referentes a:

- Processo (papel que determinado equipamento realiza).
- Operação (referente ao manuseio).
- Dados do equipamento (características e peças que o compõem).
- Sistemas internos (referentes à dinâmica interna).

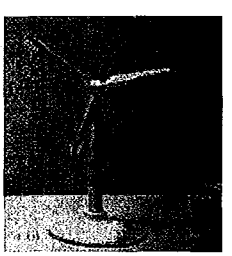
- Funcionamento (informações gerais).

Outras técnicas como, por exemplo, utilização de vídeo e gravação de voz podem ser usadas. Assim, conversas e entrevistas podem ser armazenadas para eventuais consultas.

Desse material, informações relevantes devem ser separadas e armazenadas. A nomenclatura dos arquivos deve respeitar o nome das peças e a forma pela qual a informação foi obtida, sempre com data e profissionais envolvidos na aquisição do dado, para que haja consistência entre a base de dados construída para eventuais consultas e o dado na sua forma original.

As informações de aparência da cena também podem ser inseridas no processo de aquisição. Informações referentes ao ambiente no qual os equipamentos estão inseridos são úteis para a contextualização de suas informações.

O estudo de caso realizado basea-se na geração de energia eólica e teve como apoio livros e manuais [50] [51], uma dissertação de mestrado [52], além de *feedback* de especialistas em fontes de energia alternativa.



Réplica em miniatura (a) Instalado em Mucuripe(CE) (b)

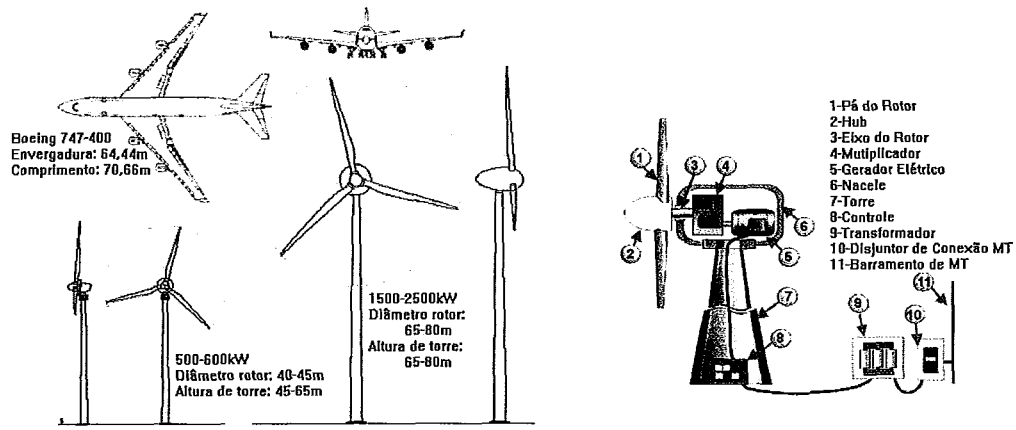
Figura 5.1: *Imagens de um tipo de aerogerador.*

As Figuras 5.1 (a) e 5.1 (b) contêm fotos da réplica e de aerogerador instalado, respectivamente.

Também foram obtidos vários desenhos 2D de equipamentos e algumas dessas ilustrações podem ser visualizadas nas Figuras 5.2 (a) e 5.2 (b), ambas relacionadas à energia do vento.

A Figura 5.2 (a) apresenta um estudo comparativo de dimensões de diferentes aerogeradores comercializados com um avião Boeing 747-400.

A Figura 5.2 (b) ilustra os principais componentes de um aerogerador com transmissão mecânica e multiplicador de velocidade.



Comparação de dimensões (a) Principais componentes (b)

Figura 5.2: *Ilustrações obtidas.*

Para compreensão geral do processo de aquisição, as informações referentes aos equipamentos do estudo de caso foram obtidas pela literatura disponível, o que facilitou análises mais apuradas na pesquisa proposta [53]. Sempre que as informações analisadas foram consideradas suficientes, foi feito o respectivo modelo 3D.

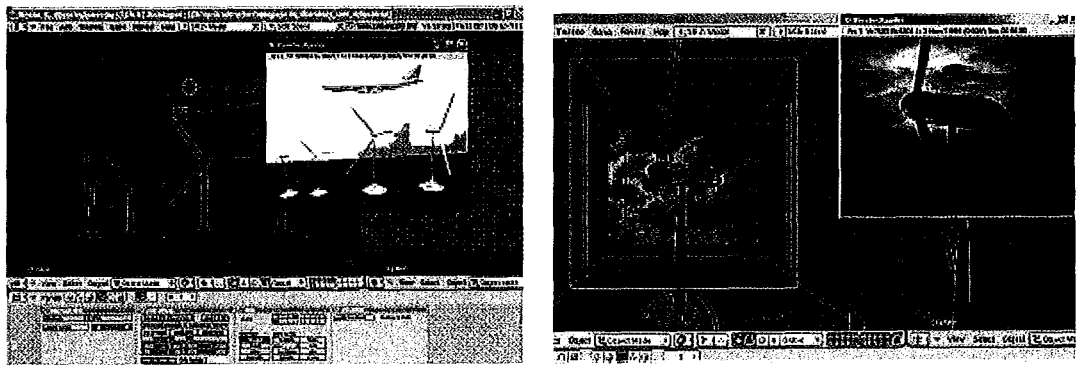
5.2 Montagem do experimento

5.2.1 Modelagem dos objetos

Os modelos foram feitos em escala, seguindo as dimensões reais, conforme as especificações da Figura 5.2:

- Menor aerogerador - potência de 500-600kW: 40-45m de diâmetro do rotor e 45-65m de altura da torre.
- Maior aerogerador - potência de 2,5 MW: 45-65m de diâmetro do rotor e 65-80m de altura da torre.
- Boeing 747-700 - envergadura de 64,40m e comprimento 70,66m.

Alguns modelos do estudo de caso foram feitos com a utilização do *software* de modelagem Blender [11] e exportados para o formato OBJ (Wavefront) e são parte do acervo de modelos do protótipo. Os modelos 3D correspondentes a tais materiais foram criados, conforme pode ser visualizado nas Figuras 5.3 (a) e 5.3 (b).



Modelagem do estudo comparativo (a) e interior do modelo e seus componentes (b)

Figura 5.3: *Modelos renderizados.*
(Fonte: Nogueira, 2007b)

A visualização interna foi obtido por meio de um corte vertical no modelo original e alguns componentes do seu funcionamento (gerador elétrico, eixo do rotor, hub e controle) foram feitos para composição do seu interior, a fim de especificar componentes da mecânica de funcionamento.

5.2.2 Autoria de conteúdo de estudo de caso com aerogeradores

Como parte da validação do *software 3DbyStep*, foi construída uma apresentação 3D com conteúdo na área de energia eólica, explicando conceitos sobre algumas partes do aerogerador, tamanhos possíveis e plantas instaladas. Essa apresentação foi mais tarde exibida e manipulada por 15 usuários em um experimento direcionado, onde eles puderam adicionar elementos visuais e animações, adequando-a as suas necessidades.

As Figuras 5.4, 5.5 e 5.6 mostram visões da apresentação de autoria.

Com o estudo de caso pode-se:

- Disponibilizar as medidas dos aerogeradores e nome de algumas peças, além de confrontar suas dimensões com um avião comercial.
- Fazer o avião sobrevoar o parque eólico, seguindo trajetória pré-definida (*script* anim_goRight).
- Destacar figura 2D que fornece informações da rede elétrica do parque eólico.