DOI: 10.5585/ExactaEP.v12n2.4983

HeDcpp: estrutura de dados para aplicação em programas de engenharia voltados ao ensino-aprendizagem

HeDcpp: a data structure for applying in engineering programs geared to teaching and learning

Gilberto Gomes

Engenheiro Civil, Doutor em Estruturas, Dep. Engenharia Civil/ENC/FT – Universidade de Brasília – UnB.
Brasília, DF [Brasil]
ggomes@unb.br

Resumo

O desenvolvimento e/ou adaptação de sistemas computacionais mediante recursos da computação gráfica e das linguagens orientadas a objetos, como C++ e Java, tem-se revelado importante na construção de ferramentas de auxílio ao processo de ensino-aprendizagem, uma vez que possibilita ao aluno e ao professor explorar os elementos essenciais de problemas típicos da engenharia, tais como forma, configuração e comportamento estrutural. O desenvolvimento dessas ferramentas requer, além dos recursos citados, a utilização de estrutura de dados e algoritmos sofisticados que permitam à modelagem, percepção e solução desses problemas. Assim, este trabalho apresenta uma estrutura de dados robusta, denominada HeDcpp, escrita em linguagem C++, que pode ser utilizada como base para programas de computador voltados ao ensino de engenharia, aborda o problema físico da forma mais real possível, bem como auxilia o aprendizado e torna as aulas muito mais produtivas e motivadoras. Alguns exemplos de utilização e aplicação da HeDcpp são apresentados.

Palavras-chave: Ensino-aprendizagem. Estruturas de dados. Modelagem. Programação Orientada a Objetos (POO).

Abstract

The development and/or adaptation of computer systems through computer graphics resources and object-oriented languages, like C++ and Java, have been shown to be substantially important in building tools that aid the process of teaching and learning, as they enable both the student and the teacher to explore the essential elements of typical engineering problems, such as shape, configuration and structural behavior. The development of these tools requires, in addition to the resources already mentioned, the use of sophisticated data structures and algorithms that enable the modeling, perception, and solution of these problems. Thus, this paper presents a robust data structure, called HeDcpp, originally written in the C++ language, which can be used as a basis for computer programs aimed at engineering education, addresses the physical problem of form as realistically as possible, assists in learning lessons, and makes classes much more productive and motivating. Some application examples are presented.

Key words: Data structure. Modeling. Object-oriented programming (OOP). Teaching and learning.

1 Introdução

Segundo Silva (2003), ambientes computacionais têm-se revelado uma alternativa de auxílio ao processo de ensino-aprendizagem, por facilitarem a assimilação pelos alunos dos conteúdos ministrados e tornarem esses processos mais dinâmicos, ágeis e prazerosos. A utilização de ferramentas computacionais no ensino prende mais a atenção do estudante, aproximando a teoria da prática e contribuindo para o aprendizado.

Aliar aos ambientes computacionais a interatividade e a realização de atividades práticas em disciplinas acadêmicas constitui uma importante complementação da parte teórica dessas, além de propiciar aos alunos o entendimento e a fixação dos conceitos aprendidos. Desta forma, os conceitos transmitidos em aulas teóricas de forma interativa entre professor e alunos e associada a situações abstraídas da realidade, torna as aulas muito mais produtivas e motivadoras.

A computação gráfica vem assumindo um papel importante como ferramenta de suporte no desenvolvimento de programas computacionais no campo da engenharia, principalmente na área de modelagem geométrica, na qual se requerem estudos e desenvolvimentos de algoritmos eficientes e de estruturas de dados sofisticadas. Isso demanda inovação, tanto na ferramenta de programação utilizada quanto no modo em que se programa. Sob este aspecto, uma nova forma de programação vem-se destacando: a Programação Orientada a Objetos (POO) (BOOCH, 1994), em que se procura definir os objetos – entidades envolvidas no sistema – e suas classes e relações, ao invés de decompor o programa em procedimentos ou funções.

Desenvolver e adaptar técnicas de computação gráfica, de modelagem geométrica e computacional e de estrutura de dados para confecção de programas orientados a objetos que permitam a simulação computacional dos problemas da engenharia, ou possibilitem o entendimento e aprendizado nas diversas disciplinas dos cursos de Engenharia, tais como Resistência dos Materiais e Isostática, de forma interativa, visual e didática, tem sido objeto de estudo em Gomes (2000), no qual alguns programas de pré-processamento gráfico abordando a POO foram desenvolvidos, a saber: o TRUSS_GI e o BEMOO_GI (pré-processadores gráficos para modelos planos de treliças e de elemento de contorno, respectivamente).

Neste contexto, e dando continuidade aos trabalhos de Gomes (2000) e Gomes e Noronha (2000), este estudo apresenta uma estrutura de dados denominada HeDcpp, baseada na clássica estrutura de semiarestas (MANTYLA, 1988) e fundamentada nos conceitos da POO, capaz de tratar a modelagem, a representação e a manipulação de modelos planos de uma forma didática, visual e interativa, cujo propósito é sua aplicação em programas voltados ao ensino e aprendizagem na engenharia, principalmente naquelas disciplinas que tem como objetos de pesquisa a modelagem físico-geométrica, as propriedades de seções planas, a geração de malhas e a análise de estruturas como vigas, pórticos e treliças, por exemplo. A HeDcpp foi escrita em linguagem C++ por suportar totalmente a POO.

O conteúdo deste artigo está organizado como segue: na seção 2, mostra-se uma breve descrição da topologia, modelagem e estrutura de dados; na seção 3, apresentam-se os principais conceitos da POO e o modelo de classes HeDcpp; na seção 4, as principais características da HeDcpp e suas formas de utilização são mostradas por meio de uma interface gráfica; na seção 5, encontram-se exemplos de aplicações, e na seção 6, as conclusões e discussões finais.

2 Topologia, modelagem e estrutura de dados

2.1 Topologia e modelagem

A descrição topológica de modelos geométricos demanda a representação das relações de adjacências de suas entidades primitivas. Em geral, a topologia de modelos planares é constituída a partir da combinação de três entidades topológicas primitivas: vértices (V), arestas (E) e faces (F) (Figura 1a). Com essas três primitivas, é possível estabelecer nove relações fundamentais (Figura 1b) que descrevem a proximidade e o arranjo de um grupo específico de entidades em relação a uma entidade qualquer.

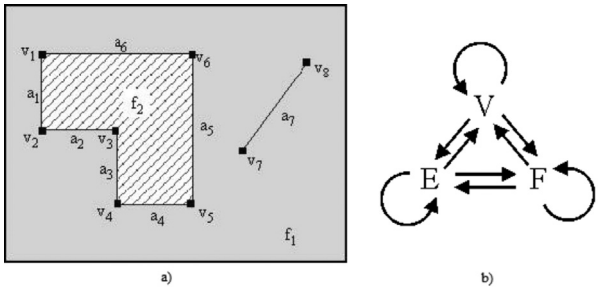

Figura 1: a) um grafo planar e seus elementos topológicos; b) relações de adjacência entre os elementos topológicos

Além das informações geométricas contidas nos vértices, arestas e faces, as estruturas de dados de representação do contorno consideram a representação das informações topológicas do modelo, sendo estas consideradas pelo armazenamento explícito de algumas das nove relações de adjacências. Por sua vez, a manipulação e criação destes modelos podem ser realizadas por meio de operações que consideram as propriedades geométricas e topológicas de modelos planos arbitrários. De acordo com a teoria de manipulação de modelos planos, um conjunto de poucas operações é suficiente para descrever e manipular a maioria dos modelos planos de interesse prático. Na manipulação de modelos de contorno, a realização destas operações é feita por entidades denominadas de operadores de Euler, segundo a nomenclatura introduzida por Baumgart (1975).

2.2 Estrutura de dados

Em Gomes (2000), a representação da topologia do modelo foi tratada por intermédio de uma estrutura de dados, baseada na bem conhecida estrutura de semiarestas (*half-edge*) (MÄNTYLÄ, 1988), capaz de registrar todos os passos intermediários que descrevem o modelo de forma sequencial. No trabalho referente em Gomes e Noronha (2000), os autores introduziram algumas modificações na estrutura original de semiarestas. Estas modificações foram baseadas no conceito de dualidade, objetivando fornecer relações semelhantes para as faces e vértices de um modelo. Uma primeira modificação consiste na desconsideração dos ciclos (*loops*) na hierarquia do modelo. Tem-se assim que a estrutura de dados resultante possui a forma reduzida (Figura 2a), e a analogia entre as relações de faces e vértices pode ser verificada, já que ambas as entidades relacionam-se diretamente com semiarestas. Por outro lado, fez-se necessária a utilização de arestas especiais, denominadas de arestas nulas, para a representação conveniente de faces contendo furos (Figura 2b). Embora estas arestas desempenhem um papel semelhante ao dos ciclos da estrutura de dados de Mäntylä, elas são consideradas como sendo entidades do tipo arestas cujos vértices não são identificados.

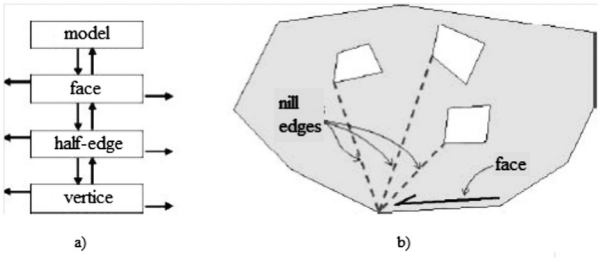

Figura 2: a) armazenamento da hierarquia das entidades na estrutura de dados utilizada; b) utilização de arestas nulas para representar faces contendo furos

A técnica de representação usada nos trabalhos de Gomes e Noronha (2000) é a de representação pelo contorno (Boundary Representation ou B-Rep) (LIENHARDT, 1990), em que as superfícies do modelo (com suas arestas e vértices) são representadas explicitamente. Neste tipo de modelagem, o contorno de um sólido tridimensional é uma superfície bidimensional que é usualmente representada como uma coleção de faces. Por sua vez, as faces do modelo são representadas em termos de curvas unidimensionais que definem os seus contornos. Os modelos de contorno, de modo geral, representam faces em termos de nós explícitos de uma estrutura de dados de contorno, possibilitando, assim, muitas alternativas para representar a geometria e a topologia de um modelo de contorno.

Neste trabalho, a estrutura de dados, inicialmente desenvolvida para representar modelos planos de elementos de contorno, será utilizada para representar modelos planos quaisquer em programas de engenharia voltados ao ensino e aprendizagem dos seguintes conteúdos: Equilíbrio Estático e Análise de Estruturas, em nível de graduação, e Análise via Método dos Elementos de Contorno, em pós-graduação.

3 A POO e o modelo de classes da HeDcpp

3.1 Programação Orientada a Objeto (POO)

A tecnologia orientada a objeto é fundamentada nos modelos de objetos, o que engloba os princípios da abstração, hierarquização, encapsulamento, classificação, modularização, relacionamento, simultaneidade e persistência. Segundo Booch (1994), o paradigma da POO constitui o mundo real dos sistemas computacionais, sendo formados por vários objetos que interagem entre si e têm sua ênfase em reutilização de código. Neste sentido, a orientação a objeto é um paradigma de análise, projeto e programação de sistemas que se baseia na interação de objetos – entidades abstraídas do mundo real. A Tabela 1 apresenta os principais conceitos da POO.

Tabela 1: Principais conceitos da POO

Termos	Definição
Objeto	Uma instância de uma classe.
Classe	Um modelo, ou molde, de objeto para encapsulamento de dados, funcionalidade e comportamento.
Membro	Uma variável ou método (operação, serviço) dentro de um objeto.
Instância	Uma classe inicializada e alocada em memória.
Encapsulação	A blindagem de dados e métodos dentro de uma classe de objetos.
Herança	Classes derivadas herdam variáveis e métodos de sua classe-base.
Polimorfismo	Refere-se ao mesmo nome de um método ou de uma classe assumindo muitas formas.

3.2 Modelo de classe HeDcpp

Com base nos conceitos da POO, o modelo de classe da Figura 3, descrito a seguir, foi projetado para representar todas as entidades envolvidas no processo de modelagem.

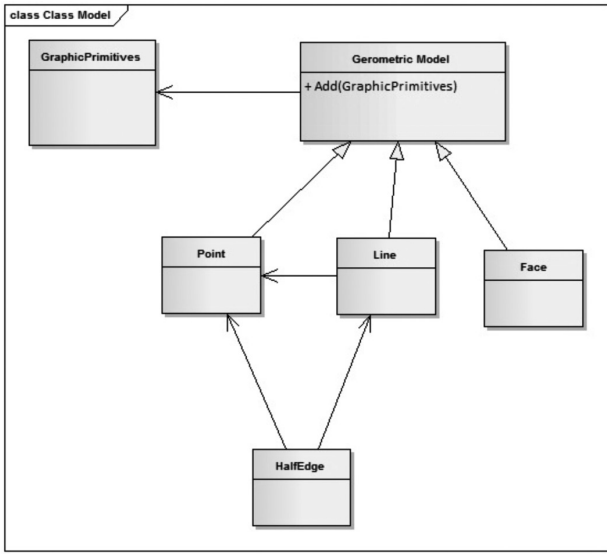

Figura 3: Diagrama de classe para o processo de modelagem

Class GraphicPrimitives: esta é uma classe-base que representa todas as primitivas geométricas do modelo. Os objetos derivados desta herdam seus dois únicos métodos: *Draw( )* e *Select( )*, possibilitando-os de se desenharem e de se selecionarem, respectivamente. Os objetos derivados desta classe são do tipo *Point*, *Line* e *Face*.

Class GeometricModel: esta classe possui métodos e atributos próprios que a determinam. Sua funcionalidade está diretamente relacionada à definição da geometria do modelo. Um objeto desta classe tem por objetivo executar as ações necessárias ao processo de construção da geometria. Estas ações, que são realizadas sobre as primitivas gráficas, são representadas pelos vários métodos desta classe que se relacionam com o propósito de definir a geometria, dentro de um processo de associação e armazenamento das primitivas através de seus atributos. A estrutura de dados proposta também é implementada nesta classe, permitindo que seu objeto possa se atualizar a cada modificação durante o processo de construção. Isto é, a estrutura de dados possibilita a atualização da geometria e da topologia do modelo em tempo de execução. Os objetos desta classe possuem referências para todos os objetos derivados de GraphicPrimitives com a finalidade única de armazenamento destes.

Class HalfEdge: esta é uma classe-base que não possui métodos, apenas dados membros que referenciam as demais entidades topológicas do modelo. Esta é a classe que define os objetos semiarestas e cujos objetos possuem referências para arestas e faces. Logo, nenhuma implementação de métodos é feita nesta.

4 Características da HeDcpp

A fim de visualizar e entender o funcionamento da estrutura de dados, uma interface gráfica foi desenvolvida, unicamente para este propósito, com o objetivo de permitir a geração de modelo a partir de uma sequência de ações acionadas pelo usuário. A interface consiste de um programa gráfico baseado em sistemas de janelas, em que as ações são controladas por menus, botões de ferramentas, teclado e *mouse*. Nesta interface, o usuário fornece como elemento básico um ponto por meio do *mouse*, sendo todo o modelo atualizado, a cada nova inserção, de forma a garantir sua consistência pela estrutura de dados. Deste modo, as principais características da interface são:

- Geração de modelos planos quaisquer.
- Identificação automática da topologia do modelo (reconhecimento de vértices, arestas e faces) através da estrutura de semiarestas implantada.
- Manipulação do modelo em tempo de execução por intermédio de comandos comuns de edição (adicionar, selecionar, remover, etc.).

O armazenamento das primitivas do modelo (vértices, arestas e faces) foi realizado mediante listas duplamente encadeadas. Esse tipo de armazenamento é vantajoso, uma vez que essas listas não requerem um tamanho definido, sendo limitadas apenas pela memória disponível do sistema computacional usado. Resumidamente, o programa de interface é organizado em cinco níveis hierárquicos, como ilustrado na Figura 4, no qual o primeiro nível possibilita ao usuário fornecer primitivas do tipo vértices e arestas, que são possíveis candidatos a serem introduzidos no modelo. Antes de incluir os vértices e arestas candidatos, o nível de verificação de consistência realiza as seguintes tarefas:

- Verifica se a primitiva candidata já existe no modelo.
- Verifica possíveis interseções com as primitivas existentes no modelo.

Após este procedimento, as primitivas podem ser armazenadas em suas listas específicas. O nível seguinte trata as descrições topológicas por meio dos operadores de Euler, e algumas funções que auxiliam na chamada adequada do operador. Por último, o nível mais baixo considera a estrutura de dados topológica e os atributos de cada elemento. O acesso a estas informações é realizado pelos operadores de Euler, no nível imediatamente superior.

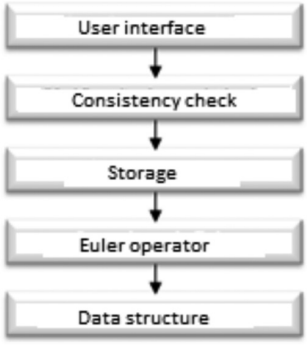

Figura 4: Níveis hierárquicos do programa

A Figura 5a ilustra as principais características da HeDcpp na geração do modelo: desenho, atualização e criação de novos modelos em tempo de execução, isto é, se há interseção entre segmentos, estes originarão novos segmentos (a aresta se divide em duas novas arestas). O reconhecimento de faces (regiões selecionadas em preto), ou furos, também é identificado automaticamente a partir do relacionamento entre semiaresta e vértice ou aresta nula. Na Figura 5b, tem-se um modelo de engenharia gerado a partir de seus elementos principais V, E e F, representando furos e fissuras.

5 Exemplos de aplicação da HeDcpp

Os exemplos, a seguir, ilustram a aplicação da estrutura de dados HeDcpp em três diferentes programas:

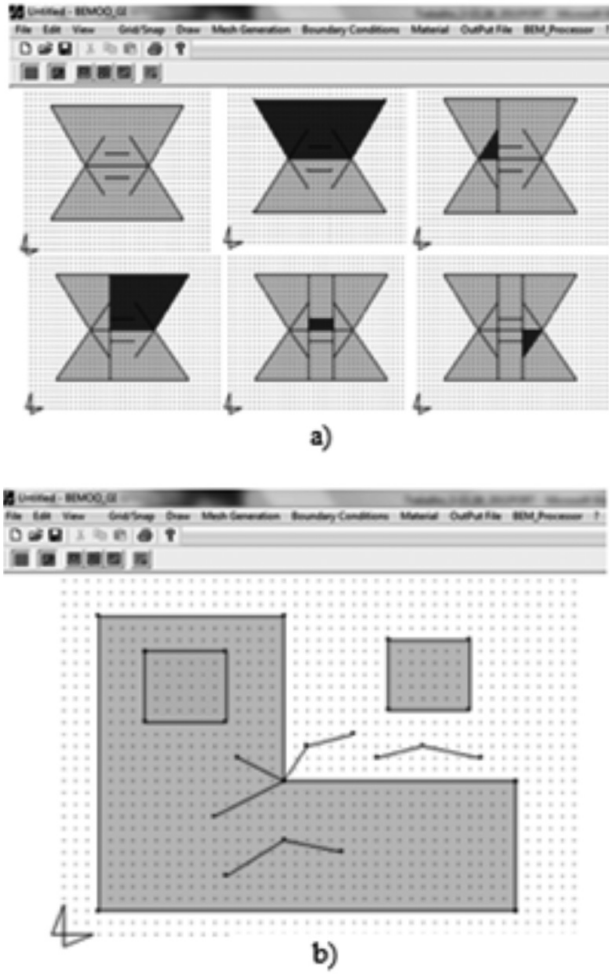

Figura 5: a) interseção, atualização e criação de novos modelos; b) modelo de engenharia

- TRUSS_GI: pré-processador gráfico orientado a objeto para construção e entendimento da modelagem geométrico-física de treliças planas. Este programa foi escrito em C++, mantendo as características da HeDcpp em sua totalidade.
- BEMOO_GI: pré-processador gráfico orientado a objeto para construção e entendimento da modelagem via método dos elementos de contorno. Este programa foi escrito em C++, mantendo as características da HeDcpp em sua totalidade.
- SENG2D: plataforma computacional de ensino-aprendizagem da engenharia. Este programa traz a versão da HeDcpp em linguagem Java, devido à necessidade de trabalhar

em multiplataforma e de estendê-la em aplicações do tipo *web*, preservando, em partes, as características da HeDcpp.

5.1 TRUSS_GI

A Figura 6 mostra o modelo geométrico para uma treliça plana, usando apenas o objeto linha. Inicialmente (Figura 6a), a treliça é constituída de três barras e, ao desenhar as outras três barras, o algoritmo de interseção e a atualização da topologia do modelo produziram a treliça da Figura 6b com nove barras. Na Figura 6c, tem-se o objeto seleção interagindo com os objetos apoio e carregamento para modelagem física da treliça. Essa interação é conseguida por meio de diálogos amigáveis entre usuário e programa.

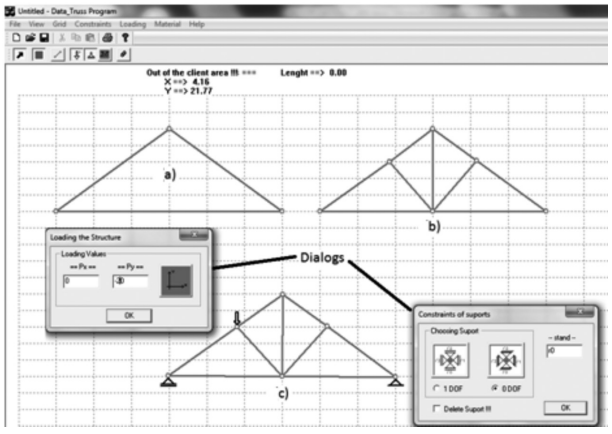

Figura 6: Programa TRUSS_GI para modelagem de treliças planas

5.2 BEMOO_GI

A Figura 7 mostra uma placa retangular modelada por meio do método dos elementos de contorno. Nesta, a geometria foi desenhada utilizando-se apenas a primitiva Linha e, automaticamente, a HeDcpp reconhece a primitiva Face (região sombreada, Figura 7a). A modelagem física (Figura 7a) foi desenhada a partir do relacionamento entre o objeto Select e as primitivas com seus respectivos diálogos (Dirichler e Newman, Figura 7c). Na Figura 7b, temos a malha de elementos de contorno que foi gerada a partir do relacionamento entre o objeto Select e as primitivas com o respectivo diálogo de geração de malha, que se dá a partir da escolha do elemento e divisão das arestas ou face selecionada.

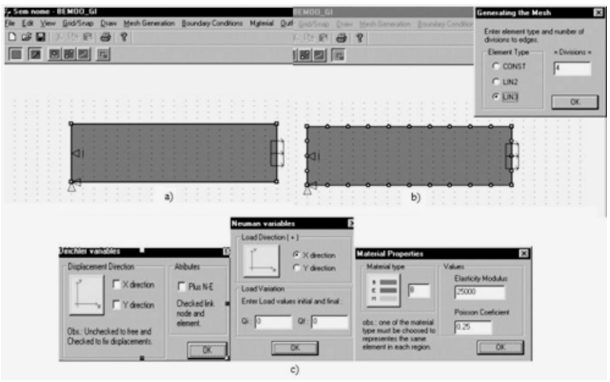

Figura 7: Programa BEMOO_GI para modelagem do MEC

5.3 SENG2D

A Figura 8 apresenta o modelo visual da plataforma SENG2D com seus botões de atalho (tarefas/funções) para auxiliar o usuário, a qual é constituída de:

- Área Cliente – para construção do modelo de engenharia (vigas, pórticos e treliças) por meio das primitivas e dos objetos Select, Apoios e Cargas.
- Área de Diagramas – para montagem do Diagrama de Corpo Livre e Diagramas de Esforços Solicitantes.
- Equações de Equilíbrio – para escrita das equações de equilíbrio da estática.
- Relatório – para impressão dos desenhos, cálculos e resultados em formatos PDF/JPEG.

Neste trabalho, restringiu-se a apresentação do módulo I do SENG2D, cujo objetivo é o entendimento do equilíbrio externo de vigas, pórti-

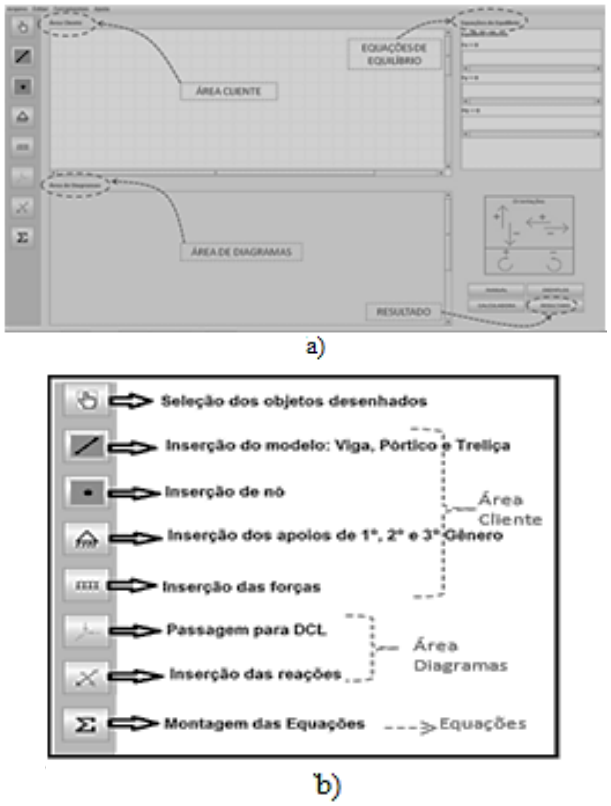

Figura 8: Modelo visual do SENG2D e seus botões de atalho

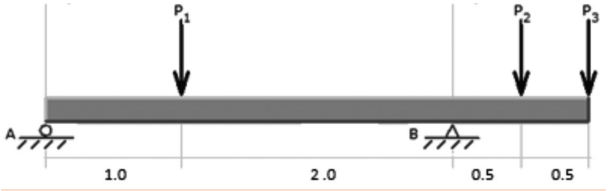

Figura 9: Modelo de viga

cos e treliças. A fim de ilustrar as funcionalidades do SENG2D, considere-se o modelo de cálculo da Figura 9, o qual tem uma viga biapoiada com um trecho em balanço e três cargas concentradas.

As Figuras 10 a 13, a seguir, ilustram as tarefas do módulo I para compreensão do equilíbrio externo, quais sejam:

- Tarefa 1: Modelagem físico-geométrica (Figura 10);
- Tarefa 2: Montagem do diagrama de corpo livre (Figura 11);

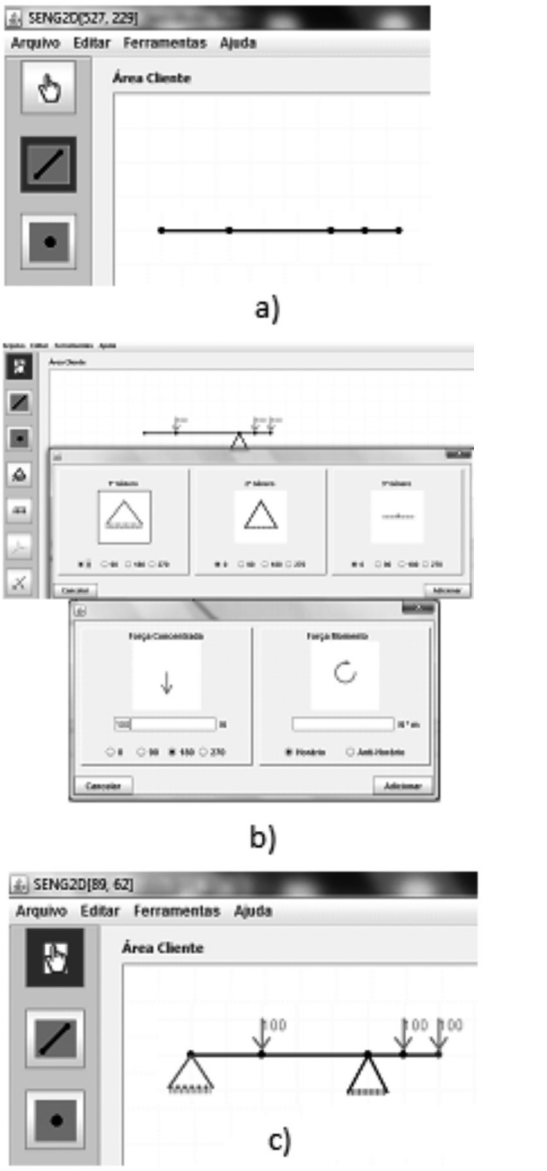

Figura 10: Tarefa 1: a) definindo a geometria; b) interagindo com diálogos para modelagem física; e c) desenho final da viga

- Tarefa 3: Escrita das equações de equilíbrio (Figura 12);
- Tarefa 4: Cálculo das reações (Figura 13).

A Figura 10a mostra o desenho geométrico da viga modelo feito a partir dos objetos "reta" e "ponto". Finalizado o modelo geométrico, inicia-se a modelagem física da viga a partir dos objetos "apoios" e "cargas", conforme visto em 10b. A Figura 10c apresenta a modelagem final da viga.

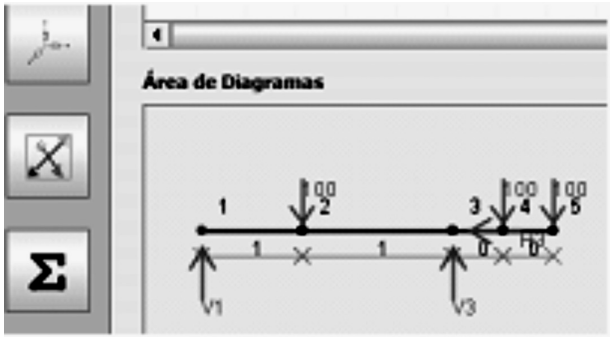

Figura 11: Tarefa 2: construção do diagrama de corpo livre

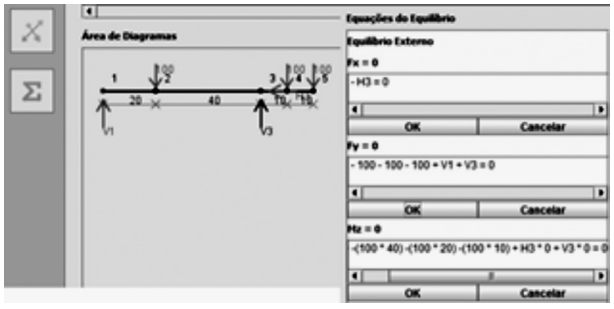

Figura 12: Tarefa 3: escrita das três equações de equilíbrio

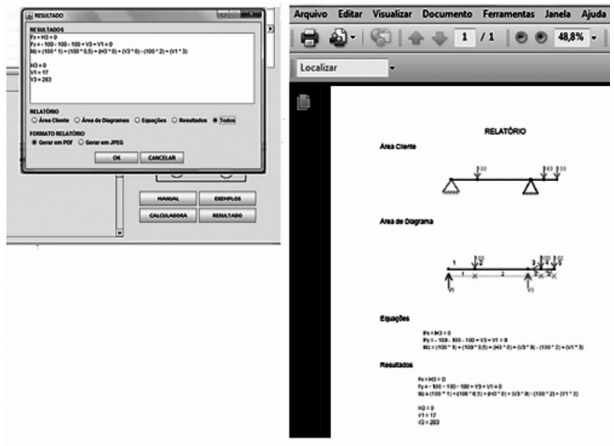

Figura 13: Tarefa 4: cálculo das reações e relatório em formato PDF

Finalizada a Tarefa 1, a área de diagrama é ativada para construção do Diagrama de Corpo Livre (DCL), conforme ilustrada na Figura 11. Nesta, o modelo geométrico e o carregamento são copiados e replicados, ficando para o usuário a inserção das reações conforme vinculação (apoios) pré-definidos. A inserção é feita selecionando-se o ponto de vínculo e interagindo com o diálogo de cargas, e, em vez de um valor numérico, letras H ou V, seguida do número de ordem, identificam a reação horizontal ou vertical.

Finalizada a Tarefa 2, ou o DCL, as equações de equilíbrio são escritas sequencialmente a partir da seleção de suas respectivas forças. Por exemplo, para a soma em X, depois de o usuário ativar o módulo de escritas (clicar no objeto "soma"), ele deve selecionar, sequencialmente e em qualquer ordem, todas as forças na direção X e, quando não existir mais forças, deve pressionar o botão "OK". Assim, a equação é finalizada e a próxima equação é solicitada, agora para a soma em Y. Na equação de momento, antes de selecionar as forças, deve primeiro escolher o ponto de aplicação deste e, em seguida, deve proceder da mesma forma. As três equações são ilustradas na Figura 12.

Por fim, a Tarefa 4 pode ser realizada, uma vez que o sistema de equações foi gerado e a solução para as incógnitas do problema (reações) foram encontradas. Para visualizar as reações, bem como o modelo físico-geométrico e o DCL, o usuário pode interagir com o objeto "relatório" e escolher o que visualizar/imprimir e em que formato (PDF ou JPEG). A Figura 13 ilustra o relatório em PDF com a opção de visualização/impressão definida como "Todos".

6 Conclusões e discussões finais

Este trabalho mostrou o uso de uma estrutura de dados alternativa baseada no conceito de dualidade de modelos planos. A estrutura desenvolvida revelou-se consistente quanto ao caráter dual, fornecendo informações topológicas análogas para vértices e faces do modelo.

O programa desenvolvido para verificação das funcionalidades da HeDcpp permite que a geometria do modelo seja introduzida de manei-

ra amigável, sequencial e interativa. Isto se deve à versatilidade da estrutura de dados proposta e aos algoritmos clássicos da computação gráfica implementados. Além disto, o programa permite também a visualização imediata do modelo gerado, evitando, assim, a tediosa tarefa de definir a geometria na forma de texto (arquivo de dados). Uma vez gerado o modelo geométrico, o programa permite introduzir as condições de contorno (apoios e carregamentos) bem como atribuir o material específico (coeficiente de Poisson e Módulo de Young, por exemplo) a este, por meio de diálogos bastante amigáveis.

Assim, mostrou-se a aplicabilidade de editores gráficos com a estrutura de dados HeDcpp tanto na área de modelagem de análises numéricas de engenharia, no caso dos programas TRUSS_GI e BEMOO_GI, quanto na área de programas educacionais, no caso do programa SENG2D, em que a manipulação de modelos geométricos facilita e motiva o aprendizado.

A plataforma SENG2D aqui apresentada encontra-se, inicialmente, voltada ao entendimento do equilíbrio externo das estruturas do tipo viga, treliça e pórtico – objeto de estudo de disciplinas de engenharia, como Estática das Estruturas e Resistência dos Materiais, e visa a auxiliar o ensino e a aprendizagem de forma amigável, sequencial e interativa, permitindo tanto ao professor quanto ao aluno a visualização imediata do modelo gerado, das etapas de análise e dos resultados. Sua versatilidade no ensino e aprendizagem do equilíbrio externo é alcançada devido à forma elegante no tratamento didático, visual e interativo de seus objetos, quando estes interagem entre si e com outros, buscando, recuperando e trocando informações necessárias, simulando e reproduzindo fielmente o conteúdo.

Uma característica vantajosa da plataforma desenvolvida é o processo de reutilização, isto é, quando se deseja utilizar uma classe já pronta e aumentar suas funcionalidades, por exemplo, a consideração do equilíbrio interno e diagramas de esforços solicitantes, seriam de fácil implementação, uma vez que a POO possibilita a inserção de código com versatilidade, bastando apenas implementar o método na classe adequada e aproveitar os resultados para as reações.

Referências

BAUMGART, B. G. A polyhedron representation for computer vision. In: NATIONAL COMPUTER CONFERENCE, AFIPS, 1975. *Proceding...*, 1975.

BOOCH, G. *Object-oriented analysis and design with applications*. Redwood City, CA, USA: Benjamin/ Cumming Publishing Company, Inc., 1994.

GOMES, G. *Estrutura de dados para representação de modelos Bi-dimensionais de elementos de contorno/A data structure for representing two dimensional boundary element models*. Dissertação (Mestrado em Engenharia Civil e Ambiental)– Universidade de Brasília, DF, 2000.

GOMES, G.; NORONHA, M. A. M. Estrutura de dados para geração de malhas bidimensionais de elementos de contorno. In: CILAMCE, 21., 2000, Rio de Janeiro *Anais...* Rio de Janeiro: PUC-RIO, 2000.

LIENHARDT, P. Topological models for boundary representation: a comparison with n-dimensional generalized maps, research Report R90-10. Strasbourg, France: Department d'Informatique, Université Louis Pasteur, 1990.

MÄNTYLÄ, M. An introduction to solid modeling. Rockville, Maryland: Computer Science Press, 1988.

SILVA, M. A. da. *Protótipo de uma ferramenta para auxiliar no ensino de técnicas de programação*. 42 f. 2003. Trabalho de conclusão de curso (bacharelado em Informática)– Departamento de Ciências Exatas e Tecnológicas, Universidade do Planalto Catarinense, Lages, 2003.

Recebido em 3 jun. 2014 / aprovado em 16 set. 2014

Para referenciar este texto
GOMES, G. HeDcpp: estrutura de dados para aplicação em programas de engenharia voltados ao ensino-aprendizagem. *Exacta – EP*, São Paulo, v. 12, n. 2, p. 209-218, 2014.