FLUXOVENTO – UM SIMULADOR GRÁFICO INTERATIVO PARA O ESTUDO DE VENTILAÇÃO EM AMBIENTES CONSTRUÍDOS

Carlos Vitor de A. Carvalho (1); Luiz Fernando Martha (1) (2); Walter Teixeira (2)
(1) Grupo de Tecnologia em Computação Gráfica – Tecgraf/PUC-Rio,
Rua Marques de São Vicente 225, (21) 2512-5984,
(2) Curso de Arquitetura e Urbanismo – Departamento de Engenharia Civil – PUC-Rio,
Rua Marques de São Vicente 225, (21) 3114-1190
e-mail: {alencar,lfm}@tecgraf.puc-rio.br, walteixeira@terra.com.br

RESUMO

Este trabalho apresenta o desenvolvimento de um simulador numérico bidimensional gráfico-interativo chamado FLUXOVENTO. Trata-se de um sistema voltado para a área de conforto ambiental cujo objetivo é simular uma ventilação cruzada. É uma ferramenta de grande utilidade para arquitetos e estudantes de Arquitetura, pois proporciona uma visualização clara e convincente do comportamento do ar dentro de edificações. O sistema pode simular diversas condições geométricas, visualizar linhas de correntes de vento e regiões de concentração e rarefação de vento dentro de ambientes construídos.

ABSTRACT

This work describes the development of an interactive graphics two-dimensional numerical simulator called FLUXOVENTO. It is a system related to the area of environmental comfort, and its purpose is to simulate crossed ventilation airflow. It is a very useful tool for architects and Architecture students, providing an easy and convincing visualization of airflow paths inside a building. The system is able to simulate various geometrics conditions and to display wind streamlines and regions with concentration and rarefaction of wind inside building environments.

1. INTRODUÇÃO

No clima tropical úmido, os benefícios da ventilação cruzada não podem ser desprezados. A sua importância está na ampliação da zona de conforto térmico com respeito a temperaturas e umidades mais elevadas, tão freqüentes em nosso clima. O bom aproveitamento desta solução natural, além do conforto proporcionado, leva à redução dos custos energéticos para a refrigeração das edificações.

Para se atingir este objetivo, as decisões tomadas durante a elaboração do projeto de arquitetura são fundamentais. Portanto, é de grande auxilio dispor de uma metodologia que permita analisar um compartimento ou mesmo um pavimento inteiro, bastando se conhecer a direção dos ventos dominantes do local.

Uma das metodologias utilizadas para fazer tal análise se baseia nos conceitos de Dinâmica dos Fluidos Computacional (Fortuna, 2000). A utilização dessa técnica já vem sendo pesquisada há alguns anos no exterior, porém no Brasil ainda é incipiente na área de Conforto Ambiental.

O programa computacional FLUXOVENTO, desenvolvido pelos autores do presente artigo e aqui apresentado, utiliza esses conceitos em conjunto com técnicas de Computação Gráfica Interativa

(Foley,1990). Com isso, a construção do modelo de análise se torna fácil para o usuário. É possível inserir, de forma interativa, anteparos externos tanto em plantas baixas como em cortes. Ou seja, a análise pode ser feita no plano horizontal ou em elevações. A representação gráfica dos resultados da análise, mostrando a construção das linhas de correntes, gera uma visualização adequada do fenômeno físico, estimulando a intuição e a percepção do caminhamento do vento através dos vãos dos compartimentos de uma edificação.

2. DESENVOLVIMENTO NUMÉRICO

O programa FLUXOVENTO se baseia nos conceitos de Dinâmica dos Fluidos Computacional. Ele emprega as equações de Navier-Stokes (Fox, 2000) com algumas simplificações que visam diminuir o esforço computacional, sem entretanto comprometer o processo real do escoamento.

O desenvolvimento numérico do sistema foi feito considerando um fluxo em regime permanente, não-viscoso e incompressível. Os efeitos de turbulência e variações de temperatura também não foram levados em conta na análise numérica. Desse modo, a equação simplificada que governa a simulação é:

$$\frac{\partial \phi^2}{\partial x^2} + \frac{\partial \phi^2}{\partial y^2} = 0 \qquad \text{[Eq. 01]}$$

onde:

ϕ é o potencial de velocidade do fluxo;

x e y são as coordenadas cartesianas.

Essa equação é conhecida na literatura como *Equação de Laplace*. Trata-se de uma equação diferencial parcial elíptica escrita em coordenadas cartesianas. Em geral, equações desse tipo representam problemas de equilíbrio, em que a propriedade de interesse não se altera com o passar do tempo.

O potencial de velocidade se relaciona com as componentes de velocidade v_x e v_y através das equações:

$$v_x = \frac{\partial \phi}{\partial x} \qquad \text{[Eq. 02]}$$

$$v_y = \frac{\partial \phi}{\partial y} \qquad \text{[Eq. 03]}$$

A solução para esse tipo de problema é obtida especificando-se condições para a variável dependente na fronteira δR da região R, em que se quer obter a solução. Os problemas que exigem condições ao longo da fronteira (contorno) δR de toda a região são denominados *Problemas de Valor de Contorno* (PVC).

Neste trabalho foi utilizado o Método das Diferenças Finitas (MDF) (Cunha, 2003) para discretizar a equação acima. O MDF foi um dos primeiros métodos numéricos desenvolvidos e é aplicado, até a atualidade, a uma extensa gama de problemas. Nesse método, utiliza-se uma malha, *grid*, sobre todo o domínio físico do problema, a qual contém determinados pontos em que são efetuadas as aproximações envolvidas. Representações das derivadas em diferenças finitas são baseadas na expansão em série de Taylor. Discretizando a Eq. 01 por diferenças finitas e utilizando o esquema de pontos da Figura 1, chega-se à seguinte equação:

$$\frac{\left(\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}\right)}{\left(\Delta x\right)^2} + \frac{\left(\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}\right)}{\left(\Delta y\right)^2} = 0 \qquad \text{[Eq. 04]}$$

Considerando $\Delta x = \Delta y = 1.0$ a Eq. 04 é simplificada para:

$$\phi_{i,j+1} + \phi_{i,j-1} + \phi_{i-1,j} + \phi_{i+1,j} - 4\phi_{i,j} = 0 \qquad \text{[Eq. 05]}$$

Reorganizando a Eq. 05 tem-se:

$$\phi_{i,j} = \frac{\phi_{i,j+1} + \phi_{i,j-1} + \phi_{i-1,j} + \phi_{i+1,j}}{4} \qquad \text{[Eq. 06]}$$

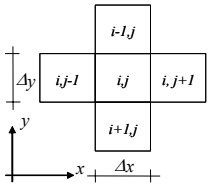

Figura 1: Esquema de pontos utilizado para a discretização das equações.

A Eq. 06 é a equação discretizada que foi utilizada para determinar o potencial de velocidade em todas as células internas do *grid*. A partir do potencial de velocidade, as componentes v_x e v_y do vetor velocidade em cada célula do *grid* são determinadas utilizando as Eqs. 02 e 03.

É importante observar que, no caso deste trabalho, as condições de contorno não são definidas na forma de potenciais de velocidade, mas sim na forma de um vetor velocidade, como mostra a Figura 2. Logo, é necessário modificar a Eq. 06 para simular essa consideração. Esses vetores de velocidade simulam a ventilação cruzada dentro das construções.

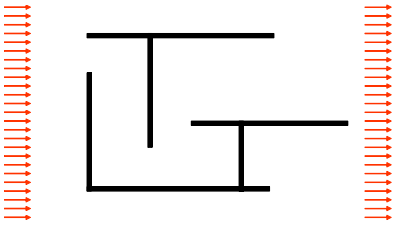

Figura 2: Velocidades de contorno utilizadas para simular a ventilação cruzada.

2.1 Tratamento das condições de contorno do *grid*.

Considerando a velocidade v_x como sendo a componente *x* do vetor velocidade, como mostra a Figura 3, pode-se discretizar a Eq. 02 por diferenças finitas, chegando a:

$$v_x = \frac{\partial \phi}{\partial x} \cong \frac{\phi_{i,j} - \phi_{i,j-1}}{\Delta x} \qquad \text{[Eq. 07]}$$

Da expressão acima, chega-se a:

$$\phi_{i,j-1} = \phi_{i,j} - v_x \Delta x \qquad \text{[Eq. 08]}$$

Substituindo a Eq. 08 na Eq. 06 tem-se:

$$4\phi_{(i,j)} = \phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j} - v_x \Delta x \qquad \text{[Eq. 09]}$$

Reorganizado a Eq. 09 chega-se a:

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} - v_x \Delta x}{3} \qquad \text{[Eq. 10]}$$

A Eq. 10 é a equação que considera as condições de contorno para as células que estão na borda esquerda (exceto as do canto). Para as células do lado direito, na borda de cima e na borda de baixo, o tratamento das condições de contorno é feito da mesma forma.

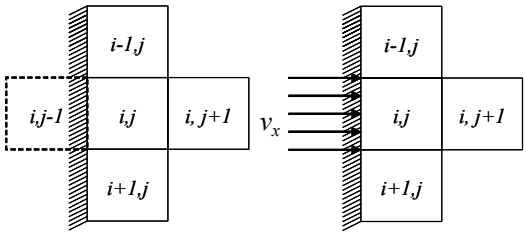

Figura 3: Tratamento das condições de contorno das células que estão na borda do modelo discretizado.

Para as células do contorno que estão nos cantos do modelo (Figura 4), deve-se considerar as componentes de velocidades v_x e v_y, ou seja, assim como foi feito para a velocidade v_x, a Eq. 03 pode ser discretizada chegando à forma:

$$v_y = \frac{\partial \phi}{\partial y} \cong \frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta y} \qquad \text{[Eq. 11]}$$

Com isso tem-se:

$$\phi_{i-1,j} = \phi_{i,j} - v_y \Delta y \qquad \text{[Eq. 12]}$$

Substituindo as Eq. 08 e 12 na Eq. 06 e considerando $\Delta x = \Delta y$, chega-se a:

$$\phi_{i,j} = \frac{\phi_{i+1,j} + \phi_{i,j+1} - \Delta x(v_x + v_y)}{2} \qquad \text{[Eq. 13]}$$

A Eq. 13 é a equação que considera as condições de contorno para a célula do canto superior esquerdo. Para as células do canto superior direto, inferior direto e inferior esquerdo, o tratamento das condições de contorno é feito da mesma forma.

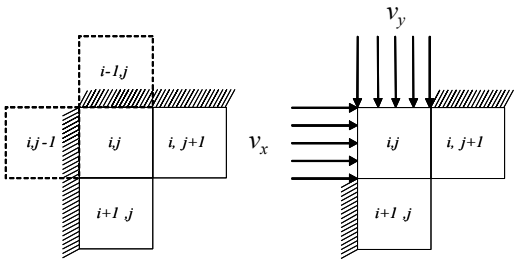

Figura 4: Tratamento das condições de contorno das células que estão nos cantos do modelo discretizado.

2.2 Tratamento de obstáculos

As células do tipo obstáculo são utilizadas para indicar a condição de uma célula que não possui fluxo dentro do *grid*. Isso irá acontecer quando o usuário desenhar uma linha na área de visualização e edição do modelo. Desse modo, as células adjacentes a uma célula obstáculo serão consideradas células de borda ou de canto, como acontece no contorno do *grid*.

A identificação das células que são do tipo obstáculo é feita utilizando-se um algoritmo de rasterização chamado *algoritmo de Bresenham* (Foley, 1990), também conhecido como *algoritmo do ponto médio*. A rasterização é um processo de amostragem dentro de um domínio discretizado. A rapidez nesse tipo de processamento é essencial em aplicativos gráficos. Esse algoritmo, que é clássico em Computação Gráfica, determina a partir de dois pontos P_1 e P_2 quais células serão obstáculos, utilizando para isso

somente soma e subtração de inteiros. Esse algoritmo é resumido a seguir, e pode ser visto nas Figuras 5a e 5b.

Considere que a célula que acabou de ser selecionada foi X. A próxima célula deve ser escolhida dentre as células da direita (A ou B). Para cada vertical, os passos são:

1) Cálculo do Ponto M (médio);

2) Cálculo da Interseção I (ponto de interseção do segmento de reta com a vertical)

3) Análise da posição de M em relação a I:

Se M estiver abaixo de I, deve-se marcar a célula A.

Se M estiver acima de I, deve-se marcar a célula B.

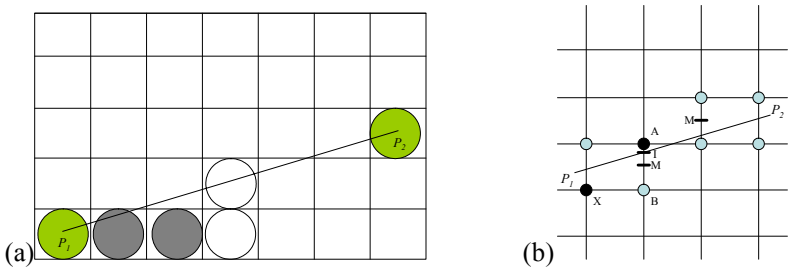

Figura 5: (a) Esquema utilizado pelo algoritmo de Bresenham e (b) Detalhe do algoritmo.

As Figuras 6 e 7 mostram o resultado da implementação do algoritmo de Bresenham para uma reta traçada pelo usuário.

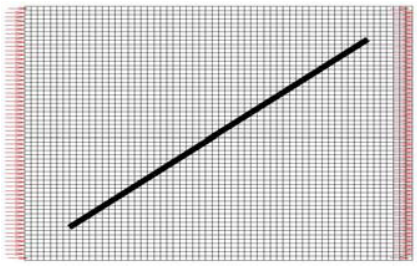

Figura 6: Linha traçada pelo usuário.

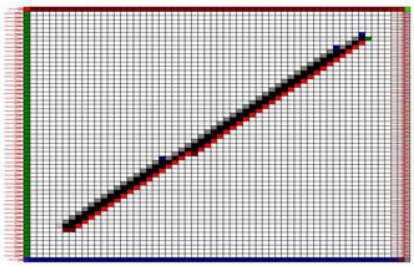

Figura 7: Resultado do algoritmo de rasterização.

Para considerar as condições de contorno e obstáculos internos, cada célula recebe um código que designa a sua posição dentro do *grid*, como mostra a Tabela 1. A Figura 8 mostra um *grid* com as células pintadas de acordo com a Tabela 1.

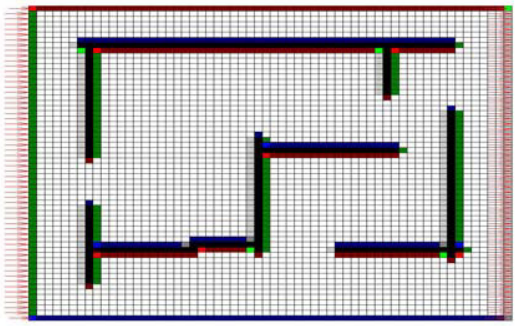

Figura 8: *Grid* com as células pintadas conforme sua posição.

Tabela 1 – Código para cada célula do *grid*.

Posição da Célula no *Grid*	Código
Interna	0
Canto Superior Esquerdo	1
Canto Superior Direito	2
Canto Inferior Esquerdo	3
Canto Inferior Direito	4
Borda Superior	5
Borda Esquerda	6
Borda Inferior	7
Borda Direita	8
Obstáculo	9

2.3 Solução do campo de velocidades

Com as equações discretizadas e as condições de contorno tratadas, tem-se um sistema de equações para resolver o problema do campo de velocidades. Esse sistema forma uma matriz conhecida como *pentadiagonal*, devido à concentração dos elementos ao longo de cinco diagonais.

A solução desse sistema pode ser obtida através de métodos diretos ou iterativos, sendo estes normalmente preferidos devido ao fato de que a matriz de coeficientes é esparsa, isto é, possui uma proporção elevada de elementos nulos. Em geral, quanto maior a ordem da matriz, maior é essa proporção (Fortuna, 2000).

Assim, o sistema de equações foi montado e resolvido utilizando-se o método iterativo de Gauss-Seidel. Durante o processo iterativo, o código de cada célula (Tabela 1) é consultado para que sejam consideradas as equações de cada condição de contorno. Como resultado, é obtido o potencial de velocidade em todas as células do *grid*. As Eqs. 02 e 03 são utilizadas para calcular as componentes de velocidades. A Figura 9 mostra os vetores de velocidades para uma planta baixa definida pelo usuário.

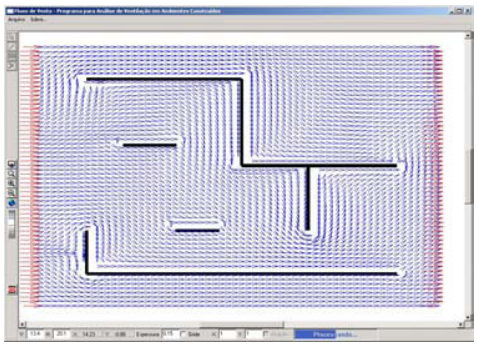

Figura 9: Exemplo de campo de velocidades para uma planta baixa.

2.4 Construção das linhas de correntes

Com o campo de velocidades calculado, deve-se agora avaliar a trajetória do fluxo dentro do *grid*. Na visualização e análise do escoamento de fluxos tornou-se comum, para avaliar essas trajetórias, a utilização de linhas de correntes ou *streamlines* (Martinez, 1995).

Linhas de correntes são curvas integrais do campo vetorial de velocidade instantânea em um dado ponto do espaço. Em outras palavras, são linhas tangentes em todos seus pontos ao campo velocidades, num dado instante de tempo. Cada linha de corrente possui o mesmo potencial de velocidade.

Para determinar as linhas de correntes é necessário resolver um Problema de Valor Inicial (PVI), no qual o valor inicial de cada linha de corrente corresponde às coordenadas (x, y) na borda da região a ser modelada, como mostra a Figura 10. Esse ponto inicial (x, y) está associado a um vetor velocidade, que será a velocidade de entrada do fluxo utilizada na análise de escoamento.

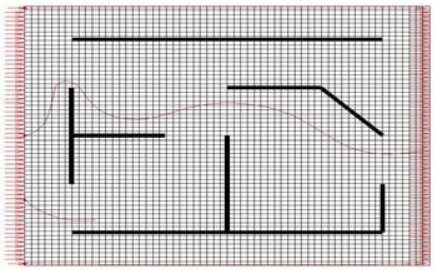

Figura 10: Linhas de correntes iniciadas na borda esquerda da região modelada.

Existem diversos métodos numéricos para determinar essas trajetórias (Cunha, 2003). Neste trabalho foi utilizado o método de Runge-Kutta de 4ª ordem, por se tratar de um método de passo único e possuir erros da ordem de h^5, sendo h o tamanho do passo, combinando muito bem exatidão e simplicidade. Como neste trabalho a análise é feita em um plano, é necessário calcular a linha de fluxo considerando as duas componentes de velocidades (v_k, v_k) em cada ponto do *grid* (x_k, y_k) (Carvalho, 2003). As equações para esse cálculo são:

$$x_{k+1} = x_k + \frac{h}{6}\left[k_0(x_k, y_k) + 2k_1(x_k, y_k) + 2k_2(x_k, y_k) + k_3(x_k, y_k)\right]$$

$$y_{k+1} = y_k + \frac{h}{6}\left[m_0(x_k, y_k) + 2m_1(x_k, y_k) + 2m_2(x_k, y_k) + m_3(x_k, y_k)\right] \qquad \text{[Eq. 14]}$$

onde:

$$k_0(x_k, y_k) = v_x(x_k, y_k)$$
$$m_0(x_k, y_k) = v_y(x_k, y_k)$$

$$k_1(x_k, y_k) = v_x[x_k + \frac{h}{2}k_0(x_k, y_k), y_k + \frac{h}{2}m_0(x_k, y_k)]$$
$$m_1(x_k, y_k) = v_y[x_k + \frac{h}{2}k_0(x_k, y_k), y_k + \frac{h}{2}m_0(x_k, y_k)]$$
$$k_2(x_k, y_k) = v_x[x_k + \frac{h}{2}k_1(x_k, y_k), y_k + \frac{h}{2}m_1(x_k, y_k)]$$
$$m_2(x_k, y_k) = v_y[x_k + \frac{h}{2}k_1(x_k, y_k), y_k + \frac{h}{2}m_1(x_k, y_k)]$$
$$k_3(x_k, y_k) = v_x[x_k + k_2(x_k, y_k), y_k + m_2(x_k, y_k)]$$
$$m_3(x_k, y_k) = v_y[x_k + k_2(x_k, y_k), y_k + m_2(x_k, y_k)] \qquad \text{[Eq. 15]}$$

Neste trabalho, as linhas de correntes são iniciadas na borda esquerda do *grid*. Depois é feita a análise do campo de velocidades na região a ser modelada e o cálculo das linhas de correntes nas extremidades inferior e superior (Figura 11a). Em seguida é feito um processamento em todos os pontos das duas linhas de corrente e é calculada a maior distância entre elas. Se essa distância (*d*) for maior que uma dada tolerância, uma nova linha de corrente é criada no ponto médio entre as duas linhas (Figura 11b). Com essa nova linha, duas novas distâncias são calculadas (*d1* e *d2*). Novamente é feito o teste com a tolerância e, se as distâncias forem maiores, novas linhas são criadas no ponto médio das distâncias *d1* e *d2*. Este processamento é repetido até que a distância máxima calculada seja menor que a tolerância determinada.

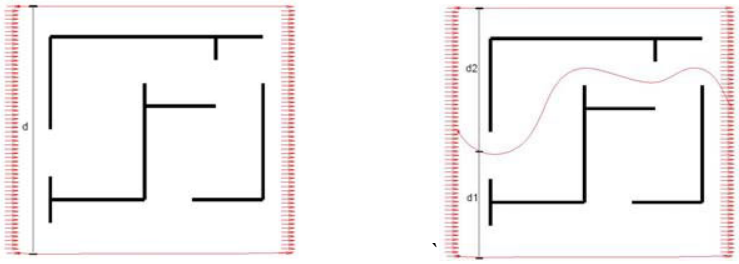

Figura 11: (a) Modelo com duas linhas de correntes e (b) Modelo com três linhas de correntes.

É importante frisar que o *grid* é utilizado apenas para fazer a análise numérica, a rasterização e o cálculo das linhas de correntes, isto é, ele é transparente para o usuário. Na área de visualização, como será mostrado na próxima seção, o usuário tem liberdade para construir o modelo e somente quando a análise é inicializada o *grid* é montado (mas não visualizado) e toda a análise é processada.

3. RESULTADOS

3.1 O SISTEMA FLUXOVENTO

O desenvolvimento descrito acima originou o programa FLUXOVENTO. Esse *software* foi desenvolvido em linguagem C++, utilizando o sistema de interface IUP (Levy, 1996) e o sistema gráfico OpenGL (Woo, 1999). A interface do programa pode ser vista na Figura 12.

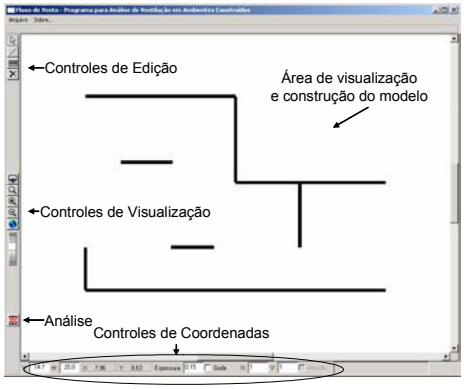

Figura 12: Diálogo principal do sistema.

Utilizando os controles de edição o usuário pode selecionar, inserir e remover uma linha para construir a geometria do modelo (Figura 13). Durante o processo de edição o usuário pode ampliar ou afastar a área do modelo, redefinindo a área de visualização, utilizando os controles de visualização (Figura 14).

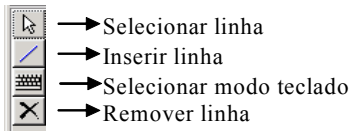

Figura 13: Controles de edição.

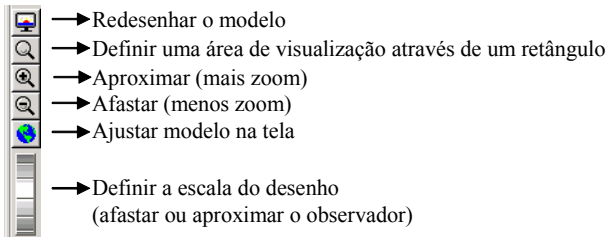

Figura 14: Controles de visualização.

Através dos controles de coordenadas, pode-se alterar o tamanho da área de trabalho, ver a posição do cursor em qualquer instante da construção do modelo, visualizar o *grid* de referência da área de visualização, ligar a atração das linhas para o *grid* de referência além de ver o indicador de processamento da análise (Figura 15).

Figura 15: Controles de coordenadas.

3.2 EXEMPLOS

A Figura 16 mostra imagens de alguns resultados da estratégia adotada para geometrias construídas no sistema FLUXOVENTO. Outras imagens e o próprio programa podem ser vistos e adquiridos no site: www.tecgraf.puc-rio.br/etools/fluxovento.

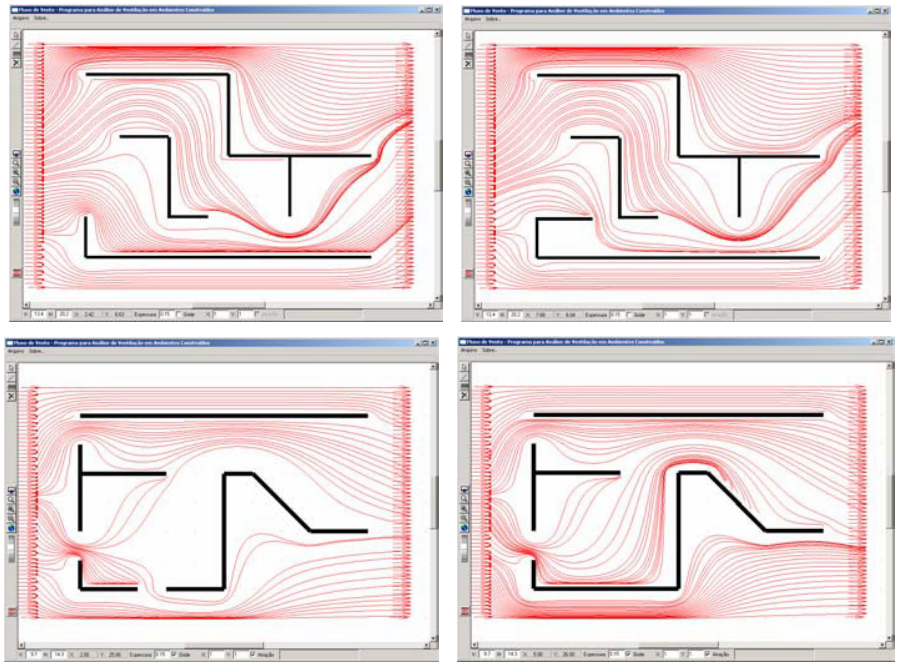

Figura 16: Resultados de algumas simulações feitas no sistema FLUXOVENTO.

4. CONCLUSÕES

Utilizar os conceitos da Dinâmica dos Fluidos Computacional como premissa para a construção de um simulador gráfico-interativo do comportamento do vento foi uma decisão acertada. Os resultados apresentados nas análises desenvolvidas, tanto em planta como em cortes, comprovam que o programa gera uma visualização adequada do fenômeno físico do caminhamento do vento através dos compartimentos de uma edificação.

O programa FLUXOVENTO pode se tornar uma importante ferramenta para os arquitetos na busca de uma Arquitetura sustentável para o clima tropical úmido.

Novas implementações estão sendo desenvolvidas com o objetivo de simular outros comportamentos, tais como:

a) a consideração de um modelo de turbulência;

b) a possibilidade, por parte do usuário, de definir ângulos de entrada da ventilação cruzada;

c) a definição de se o modelo será em planta ou corte; se for corte, não considerar as linhas abaixo da construção;

d) a possibilidade de importar plantas eletrônicas para definição dos compartimentos;

e) a extensão da metodologia numérica para ambientes tridimensionais (3D).

5. REFERÊNCIAS BIBLIOGRÁFICAS

Carvalho, C. V. A. Martha, L. F, Vargas, E. (2003) “Transport of sediments in numerical simulation of sedimentary basins”, Proceedings of 4TH International WorkShop to ACMGE, Ouro Preto, Brasil, pp. 17-20.

Corbella, O e Yannas, S. (2003) “Em Busca de uma Arquitetura Sustentável para os trópicos”, Editora Revam, Rio de Janeiro.

Cunha, Cristina (2003) “Métodos Numéricos”, Editora Unicamp.

Foley, J. D. (1990) “Computer Graphics – Principles And Practice” - Addison-Wesley Publishing Company.

Fortuna, A. O. (2000) “Técnicas Computacionais para Dinâmica dos Fluidos”, Editora EdUsp.

Fox, R. (2000) “Introdução a Mecânica dos Fluidos”, Editora McGraw-hill.

Frota, A. De B., Schiffer, S. R. (1988), “Manual de Conforto Térmico”, Editora Nobel, São Paulo.

Levy, C. H.; Figueiredo, L. H.; Gattass, M.; Lucena, C.; and Cowan, D.(1996) "IUP/LED: A Portable User Interface Development Tool". *Software: Practice & Experience*, 26 #7, pp. 737-762.

Martinez, M. L., (1995) “Uso de linhas de corrente de corrente, Linhas de Trajetória e Linhas de Emissão na Visualização de Fluxos”, Anais do VIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI'95). São Carlos-SP-Brasil, pg. 303-304.

Woo M., Jackie Neider, Tom Davis and Dave Shreiner, (1999) “OpenGL Programming Guide*”*, Third Edition (OpenGL, Version 1.2), Addison-Welsey.

Wright, Richard S. Jr.; Sweet, Michael. (2000) “OpenGL SuperBible”. 2nd ed. Indianapolis, Indiana: Waite Group Press, 2000. 696 p.