Implementação de um Visualizador Algorítmico de Notação Musical Microtonal para o programa Pure Data.

Unindo computer-assisted composition e digital signal processing
no sistema operacional Linux.

Setembro, 2006

por **Marcus Alessi Bittencourt**

Professor efetivo de Composição e Matérias Teóricas,
curso de Música da Universidade Estadual de Maringá.
Doutor em Música (Composição),
pela Columbia University in the City of New York.

RESUMO:

Desde que foram criados os programas para Computer-Assisted Composition (CAC) e para Digital Signal Processing em tempo real, sempre se tentou reunir as funcionalidades básicas destes em um só aplicativo, unindo seus aspectos composicionais e performáticos (Puckette 02 e 04). Esta comunicação apresenta um software que criei para acrescentar a visualização de dados sonoros em notação musical à funcionalidade básica do programa Pure Data de Miller Puckette.

Palavras-chave: *Composição Musical, Computação Musical, Composição Algorítmica.*

ABSTRACT:

Since the creation of software for Computer-Assisted Composition (CAC) and for real-time Digital Signal Processing, researchers have tried to unite the basic functions of those softwares in one single application, combining their compositional and performative aspects (Puckette 02 e 04). This paper presents a software I created to add the visualization of sound data as musical notation to the basic functionality of the program Pure Data by Miller Puckette.

Keywords: *Music Composition, Computer Music, Algorithmic Composition.*

1. Introdução.

Desde que foram criados os programas para Computer-Assisted Composition (CAC), como os famosos PatchWork (Laurson 89) e OpenMusic (Assayag et al 99) do Institut de Recherche et Coordination Acoustique/Musique (IRCAM) de Paris, responsáveis pelos estudos que geraram por exemplo, entre outros, o repertório da música Espectral de Tristan Murail, Gérard Grisey e Jonathan Harvey, e desde que foram criados os programas para Digital Signal Processing em tempo real como o Max/MSP, o Pure Data e o jMax, sempre se tentou reunir características e funcionalidades básicas destes em um só aplicativo integrado (Puckette 02 e 04).

Fortes do ponto de vista "performático", os softwares para DSP têm uma linhagem que vem de longa data, desde o programa Patcher (Miller Puckette, 1986), passando por uma longa lista de desenvolvimentos posteriores, como o MAX/Opcode (David Zicarelli, 1990), o Max/FTS (Puckette e Zicarelli, 1990), o PureData (Puckette, 1996), o notório Max/MSP (Zicarelli, 1997), até chegar, mais recentemente, ao jMax (François Déchelle et al, 1998), este já um projeto infelizmente defunto. Todos estes softwares têm sido extremamente úteis para o agenciamento de música eletroacústica em tempo-real, servindo de verdadeiros "cavalos-de-batalha". Aqui, o Max/MSP (no campo comercial/proprietário) e o Pure Data (no campo do sofware-livre, gratuito) têm se mantido os mais ativos e utilizados atualmente.

Fortes do ponto de vista "composicional", os softwares para Computer-Assisted Composition, desde os pioneiros trabalhos de Lejaren Hiller (Hiller 58) com o computador ILLIAC, de Iannis Xenakis (Xenakis 71) e de Pierre Barbaud (Barbaud 66), estes têm utilizado o poder de cálculo dos computadores como auxílio à aplicação de conceitos e princípios matemáticos à escrita e estruturação musicais. Nos anos 70, compositores como Grisey e Murail também valeram-se do computador para estudar a constituição de timbres e projetar composições baseadas nos dados levantados. O time de programadores, pesquisadores e compositores do IRCAM de Paris, atendendo a essas necessidades, tem desenvolvido famílias de programas para essa finalidade, como o PatchWork e seu sucessor, o OpenMusic.

Sendo a maioria destes programas muito semelhantes no que tange à interface (todos se agenciam por meio de patchers, constituindo um ambiente visual de programação completo com data structures, funções, métodos e objetos, onde um programa é montado e representado por um *patch)*, é natural pensar no intercâmbio e interconversibilidade de funções entre estes softwares.

Esta comunicação apresenta um software que criei para acrescentar a visualização de dados sonoros em notação musical à funcionalidade básica do programa Pure Data de Miller Puckette, em um modelo inspirado no das caixas *chord* e *chord-seq*

do PatchWork e OpenMusic, com aproximação de frequências ao oitavo de tono e notação rítmica proporcional.

2. Implementação do Software

O software foi implementado em três sub-partes básicas, a saber:

2.1. NotatorCore

Este corresponde ao manufaturador de um arquivo eps (encapsulated postscript) contendo a notação calculada em linguagem postscript para impressão gráfica. Este software foi construído em linguagem C e programado a partir de uma tradução que fiz para C dos comandos em linguagem Lisp utilizados para desenhar em postscript os caracteres musicais no software CMN (Common Music Notator), escrito por Bill Schottstaedt na Stanford University. Este programa, uma extensão para o CM (Common Music) de Heinrich Taube (Taube 91), por sua vez utiliza uma tradução para o Lisp realizada por Matti Koskinen da fonte musical para o MusicTex de Daniel Taupin (Taupin 96). Este meu programa, que não possui interface gráfica, recebe uma linha simples de comandos contendo uma matriz de quadras de números, cada quadra correspondendo a uma nota-evento. Os números da quadra correspondem aos parâmetros tempo-de-início, duração (ambos medidos em segundos), frequência (medida em Hertz) e amplitude (linear, medida entre 0 e 1, preferencialmente) do evento.

O NotatorCore notará os eventos da matriz em um longo sistema único com quatro pentagramas, estes representando a maior parte do espectro sonoro humanamente audível. Uma régua temporal é desenhada embaixo do sistema, iniciando no tempo do primeiro evento e terminando no tempo do último evento (ou do primeiro evento mais dez segundos, o que for mais longo). A notação do ponto de ataque dos eventos é notada segundo um sistema proporcional, colocando-se uma cabeça preta de nota no local de tempo correspondente sobre a régua. A duração da nota-evento é representada por um travessão grosso em cinza claro (para não obscurecer o pentagrama e as notações), no mesmo princípio proporcional descrito acima. A amplitude é representada pelo tamanho da cabeça da nota, sendo que a maior amplitude dada é representada pelo tamanho natural usual e a menor, por um tamanho de um décimo do tamanho natural. A frequência é marcada segundo uma aproximação ao oitavo de tono mais próximo (em um sistema de temperamento igual de 48 subdivisões por oitava) com o A4 fixado a 440 Hz. Considerei esta subdivisão como bastante razoável, não só por esta encaixar-se bem dentro do 12-equal-temperament da tradição musical ocidental, mas também devido a sua proximidade ao coma sintônico (aproximadamente 21.51 cents, contra os 25 cents do oitavo de tono, sendo 1200 cents = 1 oitava). Conhecendo a literatura a respeito da notação microtonal,

analisando os modelos fornecidos pelos PatchWork e OpenMusic do IRCAM e tendo também experimentado nas minhas próprias obras musicais com o uso de sinais de acidentes para microtons, resolvi por adotar a seguinte simbologia, dada a sua adequação às minhas necessidades e à disponibilidade dos sinais nos programas de notação CMN (fonte dos glifos utilizados) e Finale (http://www.codamusic.com):

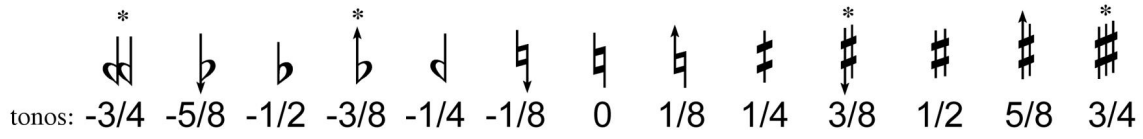

Figura 1. Tabela dos símbolos de acidentes utilizados, em oitavos de tono temperados.
(os acidentes marcados com [*] não são utilizados no software)

Ao contrário dos modelos de notação do PatchWork e do OpenMusic, que só usam acidentes de sustenidos, pensei em criar uma série de princípios fixos de regras para enharmonização que repartissem ao máximo possível o uso de bemóis e sustenidos. Assim, foram eliminados o uso dos acidentes de alteração de 3/4 de tono (trocados pelos correspondentes sinais de 1/4 de tono vindos do nome de nota vizinho), e dos bemol-mais-oitavo e sustenido-menos-oitavo, que vão de encontro ao "ideal histórico" dos acidentes tradicionais (uma nota bemolizada sendo idealmente mais baixa que a sustenida e vice-versa). O processo de conversão de uma frequência para uma notação começa localizando-se a nota natural ou nota com sustenido mais próxima da frequência a ser convertida e calculando-se um valor de desvio desta para aquela, que será, obviamente, de no máximo 1/4 de tono. Deste ponto, a escolha enharmônica procede de acordo com o seguinte algoritmo:

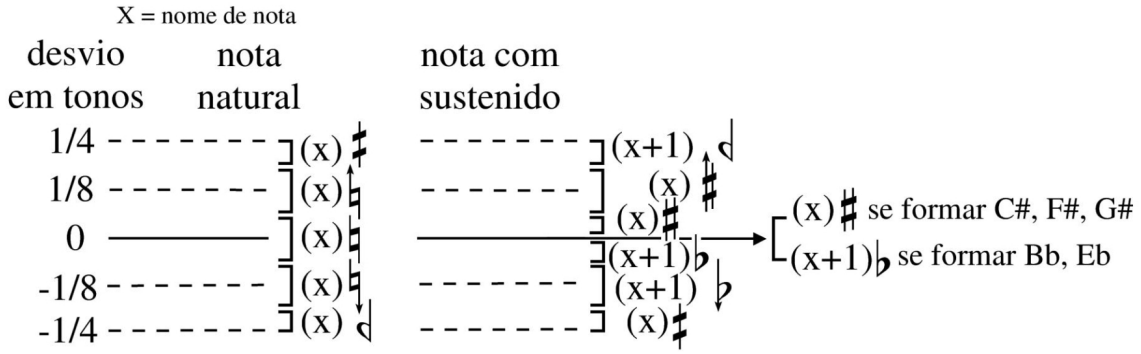

Figura 2. Tabela de aproximação e conversão enharmônica.

Desta maneira, o NotatorCore prepara, a partir da matriz de quadras

fornecida, um arquivo simples de texto com as instruções em postscript necessárias para a impressão gráfica da notação calculada, segundo o formato encapsulated (eps) e os princípios descritos acima.

2.2. NoteViewer

Este corresponde ao GUI (graphical user interface), a ser invocado de dentro do Pure Data. Construído em linguagem C com as bibliotecas gtk+ 1.2 e Cairo graphics, o GUI é a parte do software que realiza a tarefa de desenhar numa scrolled window a notação elaborada pelo NotatorCore. São também incluídos no GUI alguns controles clicáveis para navegação e ajuste da imagem, mais um botão para salvar o arquivo eps fabricado pelo NotatorCore. O programa mantém uma matriz de quadras na memória (cada quadra como antes: tempo, duração, frequência e amplitude de uma nota-evento) e, cada vez que esta matriz é alterada, ele aciona o NotatorCore, que então prepara a notação do novo estado atual desta matriz de eventos e lê o eps formado desenhando-o na scrolled window. O NoteViewer comunica-se com o Pure Data via um UNIX socket, recebendo uma nota-evento (quadra) de cada vez, junto com um comando básico, dentre os seguintes possíveis: "*add*" que adiciona o evento à matriz existente, e "*resetadd*", que zera a matriz primeiro e depois adiciona a esta o evento. O programa também recebe internamente mais dois comandos: "*quit*", para desconexão do socket e fechamento de sua janela, e "*empty*", para zerar a matriz.

2.3. Z_visualizer

Este consiste em um objeto/elemento externo de biblioteca confeccionado para o Pure Data, a ser instanciado dentro de um pd patch. Este objeto recebe mensagens (message boxes) de dentro do Pure Data e as repassa interpretadas via UNIX socket para o GUI NoteViewer. As mensagens interpretáveis pelo Z_visualizer são:

[connect] : abre uma janela com o NoteViewer e estabelece um socket de comunicação com este.

[disconnect] : fecha o socket de comunicação com o GUI e fecha a janela deste.

[empty] : esvazia a notação de eventos.

[send COMMAND time dur freq amp], sendo este *command* ou "add" ou "resetadd", segundo explicação acima.

A figura seguinte exemplifica o processo. A notação corresponde ao resultado quando do acionamento do bang button do pd patch.

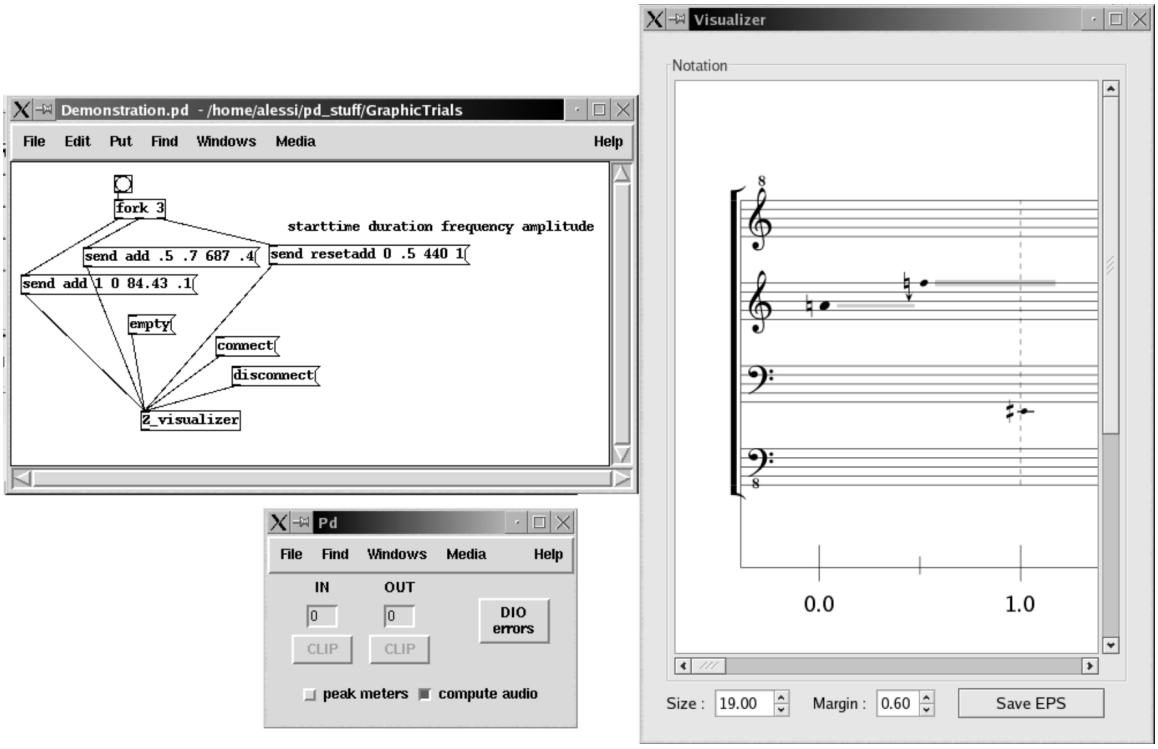

Figura 3. Exemplo de uso do objeto visualizador [Z_visualizer] no PureData, com a janela do GUI NoteViewer.

3. Possibilidades de Utilização

No melhor da tradição da Computer-Assisted Composition (CAC) (ver Laurson 89, Barrière 90, Assayag et al. 93 e 99, Xenakis 71), este software é especialmente útil para a visualização de dados musicais gerados por algoritmos e análise FFT. A visualização pode ser impressa com qualidade superior (o eps é também facilmente conversível em formato pdf Acrobat Reader por meio do programa eps2pdf de Sebastian Rahtz), podendo assim ser juntada aos materiais de estudo e esboços do compositor. À quantização rítmica tradicional dos programas de CAC preferiu-se optar por uma notação de tipo proporcional, baseada na progressão natural do tempo. Uma "barra-de-compasso" pontilhada em cinza é desenhada a cada segundo para auxílio à leitura.

Sendo que o Pure Data integra eximiamente as funções de cálculo matemático e de Digital Signal Processing (e em tempo real), não há quebra entre os

processos de cálculo, análise, ressíntese e visualização de dados musicais. Por exemplo, o notório objeto para análise FFT [fiddle~], de Miller Puckette, pode ser facilmente utilizado para agenciar uma notação musical dinâmica, em tempo real, da evolução do espectro de frequências de um som digitalizado:

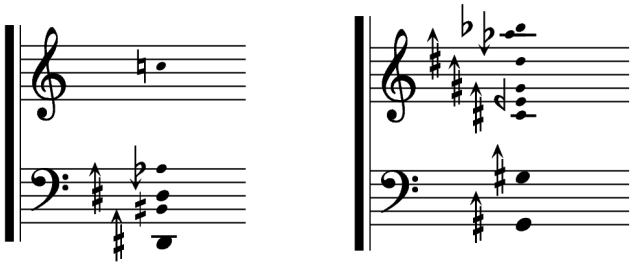

Figura 4. Visualização de instantes da análise do espectro de frequências de dois gongos javaneses.

Em outra experiência, requisitei a simples visualização de uma série harmônica:

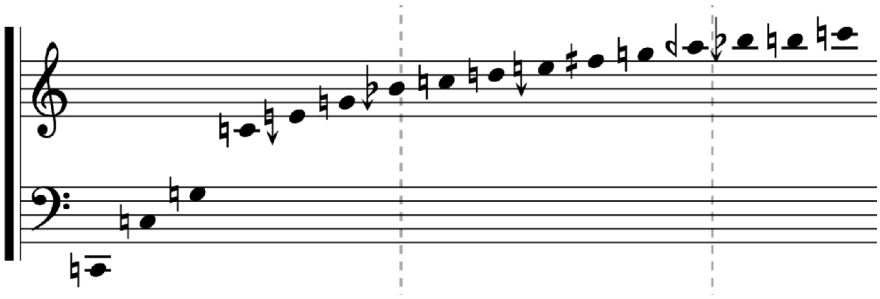

Figura 5. Visualização de uma série harmônica de fundamental C2.

4. Considerações finais.

Este programa foi criado inicialmente como um *proof-of-concept* primeiro de um desejo que eu ainda tenho de tornar o programa OpenMusic do IRCAM obsoleto pela absorção das funções deste ao Pure Data, um programa gratuito, peça de software-livre e amplamente utilizado e apreciado em todo o mundo. Tendo adquirido em diversas outras ocasiões e projetos de programação muitos conhecimentos sobre a linguagem postscript, a programação de sockets, de GUIs em gtk+ e de externals para Pure Data, o uso deste arsenal técnico na confecção de um aplicativo que serviria efetivamente como auxílio à pesquisa para criação sonora foi uma tentação à qual não resisti. No entanto, apenas o uso em projetos reais de criação e pesquisa musicais confirmará a utilidade do aplicativo.

Quanto a possibilidades futuras de desenvolvimento deste projeto, a biblioteca Cairo dá amplas possibilidades de extensão da visualização à plotagem de ondas sonoras, e ao desenho e uso de maquetes, no espírito mesmo original do OpenMusic. Interações via clicagem do usuário por sobre o GUI e por sobre os elementos desenhados podem ser facilmente capturáveis e transferidas de volta ao Pure Data para que sirvam de controle para alguma tarefa qualquer. Adicionar capacidades de processamento de mensagens MIDI também não é difícil. Enfim, há muitas possibilidades. Que venham as idéias.

5. Referências

5.1. Bibliografia

Adobe Systems Incorporated. PostScript Language Tutorial & Cookbook. Massachusetts: Addison-Wesley Publishing Company, 1985.

Assayag, Gérard; Rueda, Camilo. The Music Representation Project at IRCAM. In: International Computer Music Conference-ICMC 93, Tokyo, anais do evento, Tokyo, ICMC, 1993, pg. 206-209.

Assayag, Gérard et al. Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic. Computer Music Journal, Cambridge-Mass, volume 23, issue 3, pg. 59-72, fall 1999.

Barbaud Pierre. Initiation à la composition musicale automatique. Paris: Dunod, 1966.

Barrière J.B. Devenir de l'écriture musicale assistée par ordinateur: formalisme, forme, aide à la composition. L'analyse musicale, Paris, No. 20, pg. 52-68, 1990.

Boehm, Laszlo. Modern Music Notation. New York: G. Schirmer, 1961.

Cowell, Henry. Our Inadequate Notation, Modern Music, New York: The League of Composers, Vol. 4, No. 3, 1927.

Déchelle, François et al. jMax: An Environment for Real-Time Musical Applications. Computer Music Journal, Cambridge-Mass, volume 23, issue 3, pg. 50-58, fall 1999.

Donahoo, Michael; Calvert, Kenneth. TCP/IP Sockets in C: Practical Guide for Programmers. San Francisco: Morgan Kaufmann, 2000.

Gale, Tony; Main, Ian. GTK+ Tutorial. Documento disponível online: http://www.gtk.org/tutorial/

Hába, Alois. Nuevo Tratado de Armonia. Trad. Ramón Barce. Madrid: Real Musical, 1984.

Hiller Lejaren.A.; Isaacson L.M. Experimental Music. New York: McGraw-Hill, 1958.

Karkoschka, Erhard. Notation in new music; a critical guide to interpretation and realisation. New York: Praeger, 1972.

Laurson, Michael; Duthen, J. PatchWork, a graphical language in PreForm. In: International Computer Music Conference-ICMC 89, San Francisco, anais do evento, San Francisco, ICMC, 1989, pg. 172-175.

Möllendorf, Willi. Musik mit Vierteltönen. Erfahrungen am bichromatischen Harmonium. Verlag von F.E.C. Leuckart, Leipzig, 1917.

Puckette, M.; Zicarelli, David. MAX - An interactive graphic programming environment. Technical Manual. Menlo Park, CA, USA: Opcode sytems Inc., 1990.

Puckette, Miller. Pure Data: Another Integrated Computer Music Environment. In: Second Intercollege Computer Music Concerts, Tachikawa, anais do evento, Tachikawa, 1996, pg. 37-41.

Puckette, Miller; Apel, T.; Zicarelli, David. Real-time Audio Analysis Tools for Pd and MSP. In: International Computer Music Conference-ICMC 98, Ann Arbor, Michigan, anais do evento, Ann Arbor, Michigan, ICMC, 1998, pg. 109-112.

Puckette, Miller. Using Pd as a score language. In: International Computer Music Conference-ICMC 02, Göteborg, Sweden, anais do evento, Göteborg, Sweden, ICMC, 2002, pg. 184-187.

Puckette, Miller. Max at seventeen. Computer Music Journal, Cambridge-Mass, volume 26, issue 4, pg. 31-43, fall 2002.

Puckette, Miller. A divide between "compositional" and "performative" aspects of Pd. In: First Internation Pd Convention, Graz, Austria, anais do evento, electronic distribution, 2004. Disponível em: http://crca.ucsd.edu/~msp/Publications/graz-reprint.pdf/

Read, Gardner. Music Notation: A Manual of Modern Practice, New York: Taplinger Publishing, 1979.

Roads, Curtis. The Computer Music Tutorial. Cambridge, Mass.: The MIT Press, 1996.

Stevens, Richard. UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI. New Jersey: Prentice Hall, 1998.

Taube, Heinrich. Common Music: A Music Composition Language in Common Lisp and CLOS. Computer Music Journal, Cambridge-Mass, volume 15, issue 2, pg. 21-32, summer 1991.

Taupin, Daniel. MusicTeX: Using TeX to Write Polyphonic or Instrumental Music (Version 5.17). Orsay: Laboratoire de Physique des Solides, Centre Universitaire, 1996.

Xenakis, Iannis. Formalized Music. BIoomington: Indiana University Press, 1971.

Zicarelli, David. How I Learned to Love a Program that Does Nothing. Computer Music Journal, Cambridge-Mass, volume 26, issue 4, pg. 44-51, winter 2002.

5.2. Links importantes.

PureData: http://www.puredata.info/

PureData documentation: http://www.crca.ucsd.edu/~msp/Pd_documentation/index.htm

Miller Puckette: http://crca.ucsd.edu/~msp/

Max/MSP: http://www.cycling74.com/

IRCAM, Institut de Recherche et Coordination Acoustique/Musique: http://www.ircam.fr/

jMax: http://freesoftware.ircam.fr/rubrique.php3?id_rubrique=14

OpenMusic: http://freesoftware.ircam.fr/rubrique.php3?id_rubrique=15

Cairo Graphics, API reference manual: http://www.cairographics.org/manual/

GTK+ Tutorial: http://www.gtk.org/tutorial/

UNIX Sockets Tutorial: http://www.cs.rpi.edu/courses/sysprog/sockets/sock.html/

A First Guide to PostScript: http://www.cs.indiana.edu/docproject/programming/postscript/postscript.html/