

ViMus: Sistemas Interativos de Tempo-real para Processamento Audiovisual Integrado

Jarbas Jácome, Márcio Dahia, Geber Ramalho e Sílvio Meira

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 – 50.670-901 – Recife – PE – Brasil

{jjoj, mlmd, glr, srlm}@cin.ufpe.br

***Abstract.** This paper presents an approach for usability and expressivity trade-off for real-time multimedia systems. Through a new graphic user interface paradigm and using visual programming languages architecture, ViMus is an expressive tool, but still accessible for beginners.*

***Resumo.** Este artigo apresenta uma alternativa para um problema de compromisso entre expressividade e usabilidade em sistemas de tempo-real para multimídia. Através de um novo paradigma de interface gráfica e utilizando a arquitetura de sistemas de linguagens de programação visual, o ViMus é uma opção de ferramenta expressiva porém acessível para usuários iniciantes.*

1. Introdução

A popularização crescente das artes digitais (novas mídias) e, em particular, da atividade do visual-jóquei (Makela 2006), estimulam as pesquisas em sistemas multimídia de tempo-real. Esta demanda vem sendo atendida com o surgimento de programas específicos para o controle de exibição de vídeos e efeitos como Resolume; e com a criação de extensões para processamento gráfico como a GEM para linguagens visuais de computação musical como o Pd (Puckette 1997).

Este artigo apresenta uma pesquisa em andamento para dissertação de mestrado que propõe uma alternativa para o problema de balanço entre expressividade e usabilidade nestes sistemas. A sessão 2 apresenta os sistemas existentes, a sessão 3 explica o problema “usabilidade x expressividade” e a sessão 4 descreve o programa ViMus que está sendo desenvolvido como parte desta pesquisa.

2. Trabalhos relacionados

Entre as pesquisas acadêmicas relacionadas ao tema podemos citar, além do Pd/GEM, o Max/MSP/Jitter (Jones e Nevile 2005), Aura (Dannenberg 2005), Sonnet+Imager (Collopy 1999). Contudo, existe uma grande quantidade de programas comerciais não-acadêmicos relevantes como Resolume, Arkaos, Isadora, Visual Jockey, Modul8 e vvvv.

Identificamos dois grandes grupos aos quais pertencem a maioria dos sistemas interativos de tempo-real: os Sistemas Orientados a Fluxograma (SOFs) e os Sistemas Orientados a Amostras de Vídeo e Efeitos (SOAVEs).

2.1. Sistemas Orientados a Fluxograma (SOFs)

Uma das principais características desse tipo de programa é a grande quantidade de possibilidades de aplicações devido à sua natureza modular, extremamente flexível e dinâmica. A maioria dos SOFs oferece uma linguagem de programação visual e seu conceito está historicamente associado à computação musical (Figura 1).

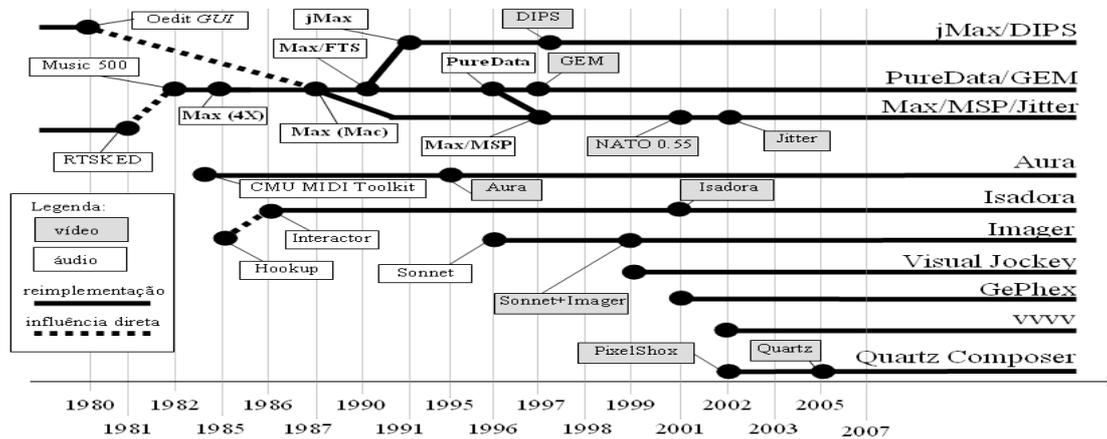


Figura 1. Linha do tempo dos principais SOFs

De um modo geral, um “fluxograma” (sugestão de tradução do termo “*patch*”) é constituído por objetos e conexões entre os mesmos. Um objeto é um módulo responsável por uma função específica e pode possuir entradas e saídas. O usuário pode interagir com o fluxograma modificando-o em tempo-real (adicionando e removendo objetos e conexões) e/ou através de objetos de interface (como botões e *sliders*) ou objetos que processam MIDI.

2.2. Sistemas Orientados a Amostras de Vídeo e Efeitos (SOAVES)

A maioria dos SOAVES são programas destinados à atividade do visual-jóquei que consiste basicamente em controlar a execução de vídeos e efeitos de forma improvisada em tempo-real. As amostras de vídeo e efeitos podem ser disparadas a partir de comandos do usuário utilizando o teclado do computador ou um controlador MIDI.

A interface gráfica do usuário dos SOAVES geralmente apresenta metáforas de equipamentos de controle de vídeo como *mixers* de vídeo do tipo “A e B” e formas de controle de vídeo já consagradas entre os VJs como o teclado de computador e teclado de piano (controlador MIDI). Geralmente é possível associar uma tecla ao comando de disparo de um efeito ou a uma amostra de vídeo. Também é possível fazer um número limitado de associações entre áudio e efeitos de vídeo em tempo-real.

3. Problema Usabilidade x Expressividade

Ao analisarmos estes sistemas, observamos um problema que na verdade é comum em tecnologias operadas diretamente por um ser humano: quanto mais funcionalidades e pluralidade de uso, pior é a curva de aprendizado e usabilidade em geral (Nielsen 1994). Os programas mais simples e específicos como Resolume têm uma excelente curva de aprendizado, porém oferecem pouca flexibilidade para criação de novos efeitos e

interações entre as mídias. Já programas como Pd/GEM são extremamente expressivos, porém assustam artistas iniciantes por sua complexidade.

Este problema será chamado nesta pesquisa de “usabilidade x expressividade”. Definiremos expressividade aqui como uma medida para o número de associações de funções e efeitos permitidas. No extremo dessa expressividade estão os programas como Pd que oferecem para o usuário uma linguagem de programação visual. Para facilitar a visão do problema, o gráfico da Figura 2 ilustra a configuração em um plano “usabilidade x expressividade” dos principais *softwares* atuais.

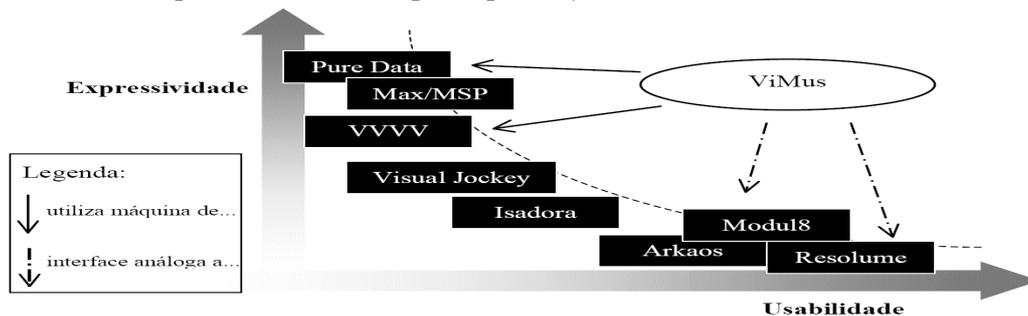


Figura 2. Principais sistemas em termos de usabilidade x expressividade e posicionamento do ViMus e suas relações.

4. ViMus

ViMus (junção das palavras “visual” e “música”) é uma alternativa para o problema “usabilidade x expressividade”. O ViMus é uma camada de interface amigável para sistemas expressivos, isto é, um metassistema que reúne características boas dos SOFs e dos SOAVES.

O poder de expressão do ViMus é oferecido ao usuário de forma gradativa e natural. Para isso a arquitetura e interface gráfica do usuário evidenciam dois conceitos já existentes nos SOFs: abstração de fluxogramas e flexibilidade de interface. Uma abstração de fluxograma consiste em um objeto especial que contém um fluxograma. Flexibilidade de interface consiste na possibilidade de composição e modificação de interfaces gráficas para os fluxogramas. Além disso, o ViMus utiliza o conceito de *graph-on-parent* (Puckette 2002), uma opção que determina que objetos de interface (*sliders*, botões, etc) de uma abstração de fluxograma possam ser visualizados dentro do objeto que representa esta abstração no fluxograma “pai”.

Optamos por uma metáfora de caixas tridimensionais abertas em vez de janelas. Uma caixa, assim como uma janela em Pd, contém um fluxograma cujos objetos podem ser outras caixas (abstrações de fluxogramas). Uma caixa possui apenas três lados opacos, os lados restantes são transparentes, permitindo a visualização de seu interior, ou seja, do fluxograma. O lado frontal da caixa é chamado “painel de controle” e exhibe elementos de interface gráfica para controle do fluxograma incluindo o painel de controle de caixas “filhas”. O lado superior exhibe as entradas e o lado inferior, as saídas.

O ViMus sempre apresenta pelo menos uma caixa contendo um fluxograma. O programa será distribuído com algumas caixas iniciais extremamente simples inspiradas em SOAVES já consagrados como Resolume e novas caixas poderão ser criadas. O usuário escolhe a caixa que lhe parecer mais apropriada. Uma caixa pode ser girada

permitindo a visualização de seu fluxograma (Figura 3). Da mesma forma, qualquer outra caixa de abstração de fluxograma pode ser aberta e girada. Ativando o modo de edição, as conexões podem ser modificadas, objetos removidos e uma paleta de opções de objetos é exibida para que o usuário possa adicioná-los, modificando a aplicação de acordo com suas necessidades.

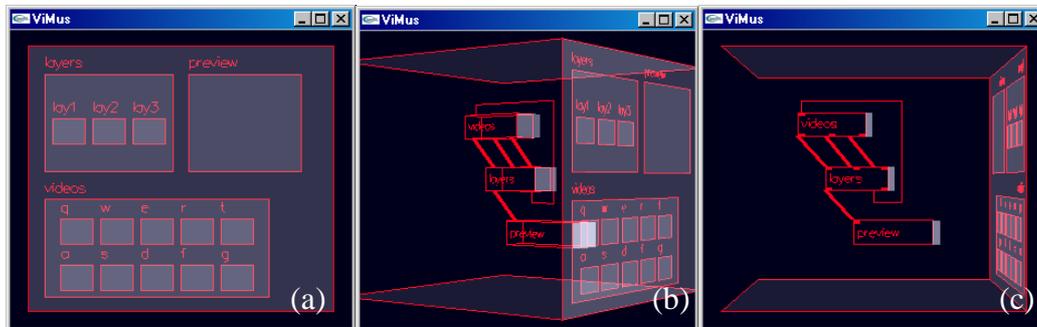


Figure 3. ViMus: a) ex. de painel de controle, b) girando e c) ex. de fluxograma

5. Considerações finais

Neste artigo apresentamos o ViMus, uma camada de interface amigável para sistemas interativos de tempo-real. A máquina de processamento de vídeo e análise de áudio para o ViMus foi implementada em C++ e OpenGL. Esta máquina será integrada ao Pd/GEM e outros sistemas de código aberto. Um protótipo da interface está sendo avaliado por visual-jóqueis com repostas positivas.

Finalmente, esta pesquisa tem duas contribuições fundamentais: inclusão de mais artistas na cultura de criação de suas próprias ferramentas multimídia para tempo-real e formalização de conhecimento na área.

References

- Collopy, F. (1999). "Visual Music in a Visual Programming Language." IEEE Symposium on Visual Music, Tokyo, Japan, IEEE.
- Dannenberg, R. B. (2005). "Interactive Visual Music: A Personal Perspective." *Computer Music Journal* 29(4): 25-35.
- Jones, R. and B. Nevile (2005). "Creating Visual Music in Jitter: Approaches and Techniques." *Computer Music Journal* 29(4): 55-70.
- Makela, M. (2006). *Live Cinema: Language and Elements*. Media Lab. Helsinki, Finlândia, Helsinki University of Art and Design. MA in New Midia: 70.
- Nielsen, J. (1994). "Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier." Acessado em nov., 2006, http://www.useit.com/papers/guerrilla_hci.html.
- Puckette, M. S. (1997). *Pure Data: Recent Progress*. Third Intercollege Computer Music Festival, Tokyo, Japan.
- Puckette, M. S. (2002). "Max at Seventeen." *Computer Music Journal* 26(4): 31-43.