

CENTRO UNIVERSITÁRIO SENAC

José Fernando B. Caneiro

# Fnk

Um ambiente de programação visual para análise,  
processamento e síntese de áudio e vídeo em tempo real

São Paulo  
2008

JOSÉ FERNANDO B. CANEIRO

# Fnk

Um ambiente de programação visual para análise,  
processamento e síntese de áudio e vídeo em tempo real

Trabalho de conclusão de curso  
apresentado ao Centro Universitário  
Senac – Campus Santo Amaro, como  
exigência parcial para obtenção do grau  
de Bacharel em Design de Interfaces  
Digitais.

Orientador: Prof. Ruggero Ruschioni

São Paulo  
2008

Aluno: José Fernando B. Caneiro

Título: Fnk, um ambiente de programação para análise, processamento e síntese de áudio e vídeo em tempo real

Trabalho de conclusão de curso apresentado ao Centro Universitário Senac – Campus Santo Amaro, como exigência parcial para obtenção do grau de Bacharel em Design de Interfaces Digitais.

Orientador: Prof. Ruggero Ruschioni

A banca examinadora dos Trabalhos de Conclusão em sessão realizada em \_\_\_/\_\_\_/\_\_\_\_\_ considerou o candidato:

1) Examinador(a)

2) Examinador(a)

3) Presidente

## RESUMO

Produções de arte interativa ou experimental necessitam de ferramentas de desenvolvimento que possuam uma curva de aprendizado suave, de modo a estimular a criação e a experimentação de forma adequada; além disso, o processo de descobrimento dos recursos disponíveis na ferramenta deve ser gradual e contínuo, garantindo ao desenvolvedor maneiras de aprimorar seu conhecimento da solução em seu próprio ritmo. Considerando-se essas necessidades, o panorama de soluções já existentes na área e o cenário de desenvolvimento artístico atual, este trabalho propõe um novo ambiente para desenvolvimento de projetos multimídia interativos que realizam análise, processamento e síntese de áudio e vídeo em tempo real. Este projeto é, antes de tudo, fundamentado numa abordagem de desenvolvimento que seja mais amigável ao usuário; não só utilizando-se de um plataforma de programação visual baseado no paradigma de execução de fluxo de dados – baseado no conceito de nós e de ligações entre cada nó – em tempo real, mas também certificando-se de que o ambiente de desenvolvimento esteja à disposição do desenvolvedor a partir de tantas plataformas e em tantas situações quanto possível.

Palavras-chave: programação visual. fluxo de dados. síntese de imagem e som. arte interativa. interface.

## ABSTRACT

Interactive or experimental art creations require development tools with a gentle learning curve, designed to satisfactorily stimulate creativity and experimentation; furthermore, the process of discovering the features available in the tool should be gradual and continuous, granting the developer ways to increase his knowledge of the solution at his own pace. Considering these needs, the group of tools already available on the field and the current artistic development scenario, this paper proposes a new environment for the development of multimedia projects that use analysis, processing and synthesis of audio and video in real-time. This project is, above all, based on a development approach that feels more friendly for the user; not only by using a visual programming platform grounded on dataflow paradigms – built on the concept of nodes and links between each node – in real-time, but also by making sure that the development environment is accessible to the developer from as many platforms and at as many situations as possible.

Keywords: visual programming. dataflow. image and audio synthesis. interactive art. interface.

## SUMÁRIO

<b>1 Introdução</b> .....	<b>6</b>
1.1 Arte contemporânea: o problema do museu	6
1.2 Tecnologia como suporte para a arte	7
1.3 Programação de computadores como instrumento de educação	8
<b>2 Panorama atual</b> .....	<b>12</b>
2.1 Softwares para interação	12
2.2 Cenário sócio-cultural	15
2.3 Ações online	17
<b>3 Definições de interação</b> .....	<b>18</b>
3.1 Controle	18
3.2 Processamento	19
3.3 Mensagem	20
<b>4 Proposta</b> .....	<b>22</b>
4.1 Objetivos principais	22
4.2 Segmentação e posicionamento	22
4.3 Método de execução	23
4.4 Interface gráfica de usuário	26
4.5 Ambiente de execução	28
4.6 Exemplos a serem construídos	30
4.7 Cronograma	34
<b>Referências</b> .....	<b>35</b>

# 1 INTRODUÇÃO

Este projeto consiste numa proposta de construção de um ambiente para criação de peças multimídia interativas. Dentro do projeto proposto, no entanto, existem várias linhas de pesquisa e desenvolvimento aparentemente díspares que devem ser levadas em consideração; estas linhas transitam entre núcleos de arte, tecnologia, educação e experimentação, e são parte vital para que o contexto no qual o projeto se insere fique claro.

## 1.1 Arte contemporânea: o problema do museu

Em meados da década de 90, a produção artística encarava um problema de posicionamento. A produção artística – criada para o espaço de museus ou galerias – distanciava-se do público, tornando-se algo intangível, quase que sagrada; as produções deviam ser *vistas*, mas nunca *tocadas* (Coulter-Smith, 2006). Assim, qualquer tentativa de interação da obra com o público, quando presente, era meramente visual, e sempre percorrendo uma única direção: da obra para o público, nunca o contrário.

É nesse contexto, e a partir de idéias trazidas dos movimentos Dadaísta e Surrealista, que a *installation art* acabou encontrando seu espaço. Embora não fosse um fenômeno exatamente novo, as idéias *desconstrutivistas* abraçadas pela *installation art* acabaram funcionando como crítica adequada à arte institucionalizada, presente em museus, criando assim um movimento onde as obras buscavam comunicar-se não só através de suas características físicas, mas também através dos valores que as inspiravam. Nesse novo movimento, o público não só é convidado a interagir com a obra, como torna-se peça necessária de todo seu aparato; é uma transgressão que, embora também institucionalizada, bate de frente com o conceito de arte aceito até então.

Além do propósito de desconstrução da obra, a imersão criada pela *installation art* busca a exploração do corpo e da mente, e o reconhecimento do corpo como algo novo. Em suas narrativas do espelho, Lacan entende que a criança, ao se ver diante de seu reflexo pela primeira vez, atinge uma nova *ordem simbólica* de reconhecimento: ela está se vendo como os outros a vêem. Esta teoria, embora pouco calcada em fatos científicos comprovados, funciona como um modelo estético eficiente: ao se ver através dos olhos de outras pessoas, há uma transferência de sentidos, e a representação física distancia-se da representação social do indivíduo.

Esse modelo de distanciamento é parte do arsenal usado pela installation art em sua tentativa de reconciliar a arte e o público.

## 1.2 Tecnologia como suporte para a arte

Produções de installation art estão, no entanto, limitadas por sua própria natureza material. Em certo sentido, essas obras são pouco mais do que *esculturas expandidas* – e os recursos imersivos, embora presentes, são mínimos.

É a partir desse problema que a tecnologia começa a se aliar à produção artística. Os recursos de imersão permitidos por essas interfaces digitais em muito ultrapassam o que era possível através de meios convencionais; então, é natural que esse suporte torne-se parte quase que vital das soluções a serem empregadas em instalações interativas.

Mais do que uma *evolução* resultante de uma instalação que passa a utilizar a tecnologia, no entanto, há uma *convergência* entre a arte que busca novos meios de interação através da tecnologia e da tecnologia que busca novos meios de expressão através da arte; pesquisas sobre o processo criativo e interativo – e o impacto da tecnologia neste processo – já eram comuns na década de 80, e é natural que experimentos realizados dentro da área acadêmica tivessem ramificações externas, servindo não só como solução para problemas encontrados em outras áreas, mas também apresentando novos modos de comunicação e interação.

O surgimento da arte interativa apoiada pela tecnologia dá-se pelo desejo de transformar "espectadores em participantes" (Candy, 2002). É a partir de premissas como essa que, para a arte interativa, o conteúdo da obra começa a ser expressada pelo processo de *transformação* que ela busca empregar, mais do que seus componentes estáticos.

Assim, esse novo movimento, quando apoiado pela tecnologia, especializa-se na análise de parâmetros externos, através de sensores ou câmeras, no processamento desses parâmetros através de algoritmos, e na síntese de novo conteúdo – como imagens ou som – de modo a interagir com o espectador, que passa a ser chamado de *iterator*.

A instalação então encontra-se com a arte interativa durante a tarefa de dar a essa

novas obras um contexto físico no qual elas possam ser inseridas – um *ambiente* onde a interação possa acontecer de modo eficiente. Mais do que mero coadjuvante, no entanto, esse ambiente também deve ser parte integrante da obra, e é a partir dessa união – do cruzamento dessas linhas de desenvolvimento inicialmente distintas da arte e da tecnologia – que as instalações interativas ganham vida, possibilitando mais do que esculturas expandidas: através da interação, cada participante cria seu próprio resultado, encontrando então seus próprios significados na obra.

### **1.3 Programação de computadores como instrumento de educação**

Além de ser usada como suporte para a arte, no entanto, a tecnologia também tem traçado outro caminho em paralelo, desta vez no círculo acadêmico: como ferramenta no auxílio da educação.

Linguagens de programação criadas para servir um papel puramente educacional não são algo novo. Já na década de 60, quando o uso de computadores começou a se tornar mais corriqueiro em laboratórios de computação acadêmicos, diversas linguagens criadas especialmente para fins educacionais começaram a se popularizar entre grupos de pesquisa em universidades ao redor do mundo, em especial nos Estados Unidos. Mais do que servir como ferramenta para a criação de aplicativos reais, tais linguagens buscavam encontrar formas mais práticas de ensino dos conceitos abstratos envolvidos nas ciências da computação.

Um bom exemplo de esforços deste tipo é a linguagem *Logo*. Criada em 1967 por Wally Feurzeig e Seymour Papert no Laboratório de Inteligência Artificial do Massachusetts Institute of Technology (MIT), Logo é uma linguagem de programação *funcional*, isto é, uma linguagem que trabalha com a avaliação de uma sequência de funções matemáticas, ao invés do controle ditado por estados e ações usado por linguagens de programação *imperativas*. O resultado foi uma curva de aprendizado mais suave, seguindo uma filosofia de que não deveria haver *limites* impostos à linguagem; o usuário deveria ver o resultado do que estava tentando realizar de forma imediata. Mais do que servir de introdução à programação de computadores numa linguagem específica, no entanto, o propósito da linguagem era funcionar como um auxílio ao ensino de conceitos envolvidos na matemática computacional.



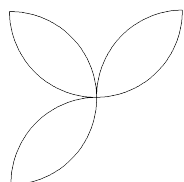
```
REPEAT 90 [FORWARD 5 RIGHT 1]
```



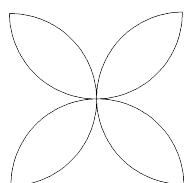
```
RIGHT 90  
REPEAT 90 [FORWARD 5 RIGHT 1]
```



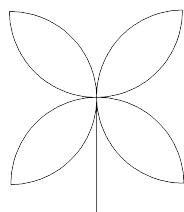
```
REPEAT 90 [FORWARD 5 RIGHT 1]  
RIGHT 90  
REPEAT 90 [FORWARD 5 RIGHT 1]
```



```
REPEAT 90 [FORWARD 5 RIGHT 1]  
RIGHT 90  
REPEAT 90 [FORWARD 5 RIGHT 1]
```



```
REPEAT 90 [FORWARD 5 RIGHT 1]  
RIGHT 90  
REPEAT 90 [FORWARD 5 RIGHT 1]
```



```
RIGHT 180  
FORWARD 400
```

*Desenhando uma flor passo-a-passo em Logo: do lado esquerdo, o resultado do desenho.*

*Do lado direito, os comandos utilizados em cada passo.*

Nesse sentido, a linguagem não deixou de obter sucesso. Porém, o que tornou o Logo especialmente popular era a possibilidade de se interagir com dispositivos gráficos com facilidade: baseado em comandos interpretados em tempo real que controlavam um cursor virtual, era possível criar desenhos que eram mostrados na tela do computador, ou ainda utilizar uma *tartaruga gráfica* – um dispositivo robótico, bastante semelhante a uma tartaruga, ligado a computadores através da porta serial – que, com o auxílio de canetas de diversas cores acopladas a seu corpo, percorria uma superfície de papel, reproduzindo os mesmos desenhos, mas de uma forma mais concreta.

Esta novidade tornou o ensino de computação mais atrativo, fazendo com que deixasse de ser exclusividade de laboratórios de matemática universitários e fosse trazido como ferramenta para o ensino fundamental. Tal experiência pode ser especialmente observada nos grupos de crianças coordenados por Seymour Papert ao longo das décadas de 60, 70 e 80. Assim, no final da década de 60, o Logo tornara-se uma ferramenta madura de educação, não devido à sua capacidade de aplicação prática – em aplicativos reais – mas sim em sua facilidade de engajar o usuário, em especial crianças, e ao modo como estimulava a *experimentação*: um "ambiente rico que estimula a reflexão sobre a matemática e sobre as rotinas de solução de problemas de um indivíduo", como observado por Clements (2002, p. 163).

Outro sinal da popularidade do Logo são os diversos projetos paralelos ou com objetivo semelhante à qual a linguagem acabou dando origem; centenas de diferentes implementações do Logo ou de seu ambiente foram produzidas, e muitas ainda estão em constante desenvolvimento.

O melhor exemplo desta popularidade é, provavelmente, a linguagem *StarLogo*. Criada por Mitchel Resnick e Eric Klopfer durante a década de 90 no Media Lab (também do MIT), *StarLogo* é uma extensão do Logo criada também para ambientes educacionais, mas com foco especial na simulação do comportamento de sistemas descentralizados: ou seja, embora ainda utilize o recurso de *tartarugas gráficas*, a linguagem o faz de forma mais distribuída, permitindo que várias tartarugas estejam em ação ao mesmo tempo, e até mesmo interagindo umas com as outras.

Outro exemplo mais moderno – inspirado tanto pela linguagem *Logo* quanto pelo *StarLogo* – é o ambiente *Etoys*, criado por uma equipe dirigida por Alan Kay em

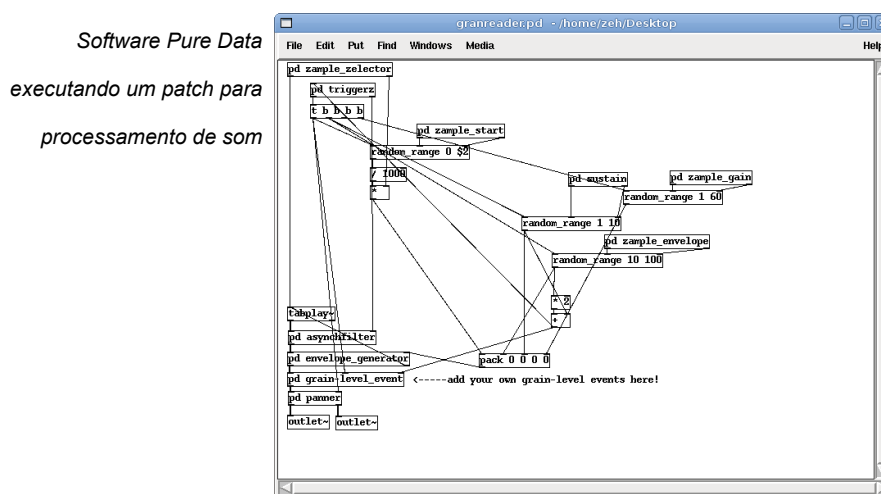
1996. Criada especialmente para ser utilizada por crianças, a plataforma oferece uma interface gráfica mais atrativa na tentativa de facilitar o seu aprendizado.

Inúmeros outros exemplos de projetos semelhantes existem. Exemplos à parte, no entanto, fica claro que linguagens e plataformas sem propósito comercial específico – como produção de aplicativos – podem ser usadas tanto como auxílio à educação, como ferramenta para experimentação ou simulação com diversos fins. Um bom exemplo pode ser encontrado em *Emergência* (Johnson, 2001): embora não possua um caráter técnico, um capítulo inteiro do livro é dedicado à discussão da simulação de sistemas complexos através de software, sendo a linguagem StarLogo usada como protagonista principal do capítulo.



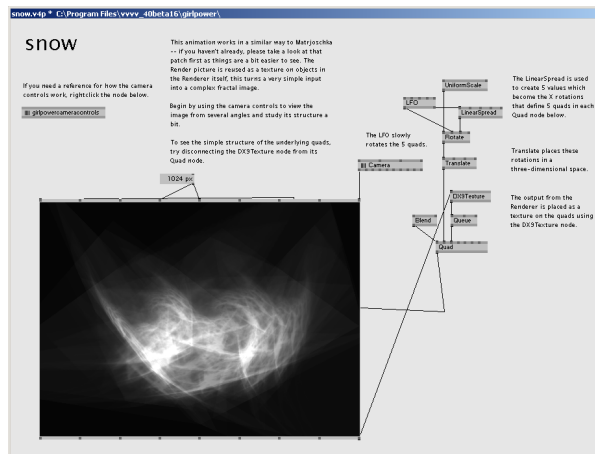
Nos últimos anos, diversas extensões foram lançadas para este software, como o *MSP* – para síntese e processamento de som digital em tempo real – e o *Jitter*, para processamento de vídeo. Além disso, inúmeros periféricos de entrada e saída de dados podem interagir com o software. Tudo isso contribuiu para que o Max se transformasse na solução ideal para processamento de áudio e vídeo em peças multimídia interativas, e devido a esse impacto, inaugurasse um novo segmento de desenvolvimento de softwares: softwares para processamento de áudio e vídeo em tempo real, com foco mais na experimentação e rápido desenvolvimento visual do que na abstração algorítmica de baixo nível geralmente almejada por linguagens de programação convencionais.

Assim, além do próprio Max, vários outros softwares foram criados neste segmento. Um deles é o *Pure Data* (ou *Pd*), criado também por Miller Puckette em 1996, após sua saída do grupo de desenvolvimento do Max; com foco semelhante, Pure Data utiliza também um ambiente de programação visual para síntese e processamento de áudio em tempo real.



Outro exemplo é o software *vvvv*, criado em 2002 pelo estúdio de design e tecnologia alemão MESO. Também inspirado no Max, e utilizando os mesmos paradigmas de programação visual, *vvvv* é um software com maior foco em processamento e síntese de vídeo em tempo real. É interessante notar que, por se tratar de um software mais recente, ao contrário de seus predecessores, este ambiente foi criado *especificamente* para o desenvolvimento de projetos multimídia interativos.

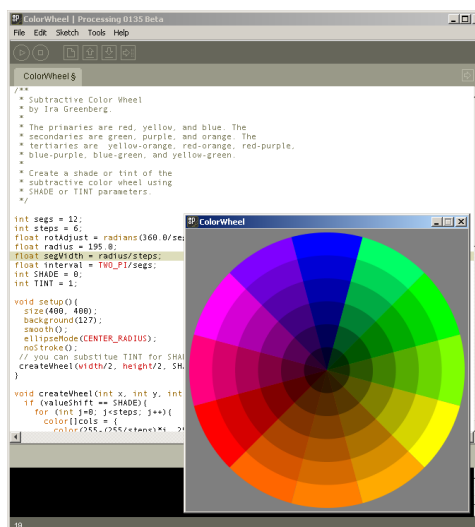
Software vvvv  
executando um patch  
para síntese de imagem



Esses três softwares – Max, Pure Data e vvvv – são provavelmente os mais populares exemplos de ambientes de programação visual para processamento de som e imagem. A estratégia para desenvolvimento empregada em cada um destes ambientes consiste em criar módulos de processamento de dados baseados em nós e ligações, criando assim um diagrama de fluxo de dados. Essa solução tende a proporcionar uma curva de aprendizado mais suave para o desenvolvedor, estimulando a experimentação e o descobrimento de novas técnicas e métodos através da combinação de diferentes módulos.

No entanto, outros softwares optaram por seguir um caminho diferente. É o caso do *Processing*, um ambiente de programação criado por Casey Reas e Benjamin Fry, do Laboratório de Mídias do MIT, em 2002. Embora seu objetivo seja também permitir o desenvolvimento de experimentos visuais e interativos de forma simples e rápida, este software se assemelha mais a uma linguagem de programação convencional, servindo então como uma solução mais técnica para a criação de peças multimídia interativas.

Software Processing  
executando um sketch  
para síntese de imagem



Independente dos paradigmas de desenvolvimento adotados, estes softwares acabaram não só servindo como solução para o desenvolvimento de peças interativas, mas também para a popularização deste segmento de desenvolvimento, inclusive dentro do campo acadêmico.

## 2.2 Cenário socio-cultural

Todos esses fatores desencadearam um redescobrimto da produção de obras que poderiam ser apreciadas pelo público em geral, devido à sua natureza *interativa*, ao contrário de simplesmente celebradas com base em fatores que poderiam ser controlados por uma ditadura artística. Iniciada pela installation art, essa reversão da característica de distanciamento do público que a produção artística vinha mantendo não deixou de ser notada por Coulter-Smith:

Em agosto de 2006 eu entrei no Lentos Kunstmuseum em Linz e tive a típica experiência de museu de arte. Visitantes bem-vestidos da classe média alta ponderando sobre objetos belos e/ou interessantes, paredes brancas impecáveis, um silêncio discreto, guardas por toda parte, fotografias proibidas (afinal, não queremos roubar a preciosa alma da arte). Então andei pelo Nibelungenbrücke até o Ars Electronica Center. Eles não o chamam de galeria, muito menos de museu. É uma extraordinária galeria de arte não só porque é dedicada à arte interativa, mas porque é barulhenta. O pessoal vestido de laranja altamente visível não está aqui para impedí-lo de tocar nas obras em exibição, mas para ajudá-lo a tocar. É o oposto do museu de arte. O amante de arte inveterado experimentará algo similar a um choque cultural, ou um ataque de pânico. Ele ou ela irá provavelmente correr para a porta. E existem as crianças. Claro que permitimos a entrada de crianças em galerias de arte ou museus também, mas como nos tempos Vitorianos, elas tem de ser vistas e não ouvidas. No Ars Electronica Center, elas tem livre circulação. [...] Mas o Ars Electronica Center é provavelmente um extremo grande demais para os críticos. O Palais de Tokyo oferece um equilíbrio entre a segurança máxima do, digamos, Tate Modern, em Londres, ou do The Whitney, em Nova Iorque, e a total liberdade do desejo evidente no Ars Electronica Center. Se mais galerias e museus de arte seguissem os passos do Palais de Tokyo, isso seria por si só um desenvolvimento extremamente positivo. (2006, c. 1, p. 3, tradução livre).

Há, assim, uma separação da produção artística tradicional e a produção de obras interativas: elas possuem não só espaços distintos, como também públicos e áreas

de atuação. Não é de se estranhar, então, que tenham surgidos espaços e festivais recorrentes inteiramente dedicados à arte interativa.

Talvez o melhor exemplo deste movimento seja o *Ars Electronica*, um festival anual sobre arte, tecnologia e sociedade iniciado em Linz, Áustria, em 1979, e considerado hoje o maior expoente da arte digital no mundo. Este festival tem em sua essência a interdisciplinaridade, e busca promover, de acordo com seu website, "um encontro aberto de experts internacionais das artes e ciências, com uma audiência de repertório e interesses altamente diversos". Além do festival, o grupo mantém o já citado *Ars Electronica Center* – considerado uma espécie de *museu do futuro* – e o *Ars Electronica Futurelab*, um laboratório de pesquisa e desenvolvimento sobre arte, tecnologia e sociedade.

Diversos outros festivais e iniciativas semelhantes existem ao redor do mundo, como o *Transmediale*, realizado em Berlim, Alemanha, também anualmente, ou o o *Memefest*, realizado na Eslovênia. Nestes e em outros casos, o foco na arte eletrônica é o que os diferencia de eventos ou exposições artísticas mais tradicionais; consequentemente, o envolvimento do público recebido em tais eventos é maior do que se esperaria do público de museus ou galerias de arte mais comuns.

Também encontramos ótimos exemplos no Brasil. Um deles é o *FILE* (Festival Internacional de Linguagem Eletrônica), que desde 2000 promove um festival anual voltado para a arte eletrônica em São Paulo, bem como simpósios e eventos performáticos centrados na arte midiática. Hoje, o FILE está presente em diversas outras cidades do Brasil e do mundo, tornando-se uma das referências mundiais de iniciativas de arte eletrônica e interativa.

Outro ótimo exemplo nacional é *Emoção Art.ficial*, a bienal internacional de arte e tecnologia do Itaú Cultural, criada em 2002. Seguindo os moldes de uma mostra mais tradicional, e contando também com um simpósio, este evento busca trazer produções de arte interativa, interface e cibernética para o público, e conta com trabalhos que já fazem parte do acervo áudio-visual de arte e tecnologia do Itaú Cultural, bem como novos trabalhos nacionais e internacionais.

## 2.3 Ações online

Devido à popularização deste tipo de produção entre o meio criativo, diversas comunidades *online* surgiram baseadas principalmente nas ferramentas utilizadas para criar peças multimídia interativas. É o caso dos *websites* dos próprios softwares para criação de peças interativas: todos possuem *fóruns* para discussão e troca de informações sobre técnicas e aplicações, fazendo com que o acesso a um grupo de desenvolvimento baseado numa destas plataformas seja algo fácil para qualquer um com acesso à Internet.

Adicionalmente, além do conteúdo gerado online – discussões, galerias de trabalhos, tutoriais sobre técnicas específicas, etc – também é comum que tais comunidades online tenham algum impacto adicional fora da rede, como quando são utilizadas para a organização de palestras, workshops, cursos ou outros eventos voltados para a ferramenta em questão.

Comunidades focadas em certas ferramentas utilizadas em instalações interativas – como componentes de hardware utilizados para interação – também surgem naturalmente, criando certo cruzamento entre os grupos de usuários de cada recurso. Assim, é comum que num website dedicado ao componente *Arduino*, por exemplo, existam desenvolvedores que utilizem Processing e Max discutindo algum aspecto sobre a implementação de trabalhos interativos.

Por isso, mais do que simplesmente oferecer uma solução para um problema técnico – a necessidade de se criar uma peça interativa – as plataformas de desenvolvimento disponíveis acabam se tornando elas mesmas agregadores, centros ao redor do qual pessoas podem criar discussões, aprender, e difundir conhecimento; o desenvolvedor interessado em utilizar um software específico não só cria uma peça interativa, mas tem contato com peças criadas por outras pessoas, técnicas utilizadas, e novas idéias que surgem a partir desta discussão.

### 3 DEFINIÇÕES DE INTERAÇÃO

Softwares interativos podem ser definidos como software cujo comportamento muda de acordo com mudanças em seus parâmetros de entrada. Assim, a resposta do software a estímulos externos é o que determina sua capacidade de interação.

Softwares convencionais geralmente necessitam de complicados métodos de operação para a realização de alguma tarefa específica, como o uso de menus ou comandos disponíveis em sua interface. Em projetos artísticos multimídia interativos, ou na *arte interativa*, no entanto, essa interação geralmente se dá de forma muito mais análoga: através de gestos captados por uma câmera, sons emitidos pelo interator, outros sensores específicos, etc.

Devido ao amadurecimento das ferramentas disponíveis para criação de apresentações interativas, cabe aqui uma classificação destes sistemas interativos. Fundamentalmente baseado em Rowe (1993), mas adaptado para sistemas multimídia (não só sonoros), o intuito deste sistema é funcionar como auxiliar à compreensão dos programas que poderão ser criados dentro do projeto proposto, como os exemplos citados no próximo capítulo.

#### 3.1 Controle

O primeiro critério de classificação diz respeito ao *controle* de uma peça multimídia. Peças multimídia podem ser controladas por *sequências pré-programas* – como uma animação, por exemplo – ou por *parâmetros de interação*.

No caso de sequências pré-programas, todo o processo criativo seria realizado do lado do artista, durante o processo de desenvolvimento do programa. Ao criar resultados específicos, criam-se obras estáticas, mas que reagem a parâmetros pré-determinados, como tempo de execução, dados sequenciais lidos de algum arquivo, etc.

Já quando parâmetros controlados externamente são usados – geralmente, a partir de alguma informação gerada pelo interator e captada por algum tipo de sensor ou periférico de entrada – o processo se torna mais interativo, e o resultado na peça é atribuído a métodos algorítmicos de combinação destes parâmetros.

A princípio, pode parecer que só a última destas características trata realmente de sistemas interativos. No entanto, a verdade é que produções interativas acabam lidando com os dois modos de controle, cada qual à sua maneira e em diferentes níveis, e saber classificá-los é importante para estabelecer uma representação de cada produção. Por exemplo, imaginemos uma produção interativa que utilize uma música que é tocada indefinidamente, gerando efeitos que podem então ser controlados pela presença do interator em frente a uma câmera ou sensor de movimento. Neste exemplo, a alimentação sonora – a música que é lida e tocada – é baseada em dados *pré-definidos*, ou seja, uma sequência estática. Os efeitos, por sua vez, são controlados pelo interator, sendo assim *interativos*.

Ao mesmo tempo, peças interativas podem ter sido criadas com o intuito de serem controladas somente pelo desenvolvedor, seja em tempo real – durante sua execução – seja durante o desenvolvimento. No primeiro caso, a peça acaba funcionando como um software – um programa para controle de vídeos para VJs, por exemplo – enquanto que no segundo caso, o eventual interator acaba se tornando um mero espectador de um processo interativo que ocorreu previamente entre o desenvolvedor e a ferramenta de desenvolvimento.

### **3.2 Processamento**

Outra classificação importante deve ser feita acerca do tipo de resposta gerada pela apresentação multimídia baseada num dado, e dos métodos empregados para seu processamento: *transformativos*, *generativos*, ou *sequenciais*.

Métodos *transformativos* são usados quando material multimídia pré-existente é utilizado para criar um novo material. Este é um dos principais métodos usados em peças multimídia interativas, uma vez que certos dados coletados – como entrada de vídeo, ou imagens – podem ser *transformados* antes de serem apresentados de volta ao interator. Assim, no projeto proposto, diversos métodos de transformação estarão disponíveis para que entrada de áudio e vídeo possa ser processada e reusada.

Métodos *generativos* são usados quando o material multimídia é gerado pelo processo interativo programado pelo desenvolvedor, baseado em informações simples ou outros parâmetros externos; ao invés de simplesmente realizar uma transformação em alguma informação multimídia já existente, esses dados são usados para se gerar uma nova informação baseada em métodos de *síntese*, sendo

então descartados. Nesta proposta, assim como em ambientes multimídia, tais métodos geralmente fazem uso de informações numéricas para controlar parâmetros de entrada de processos generativos.

Finalmente, métodos *sequenciais* tratam da combinação de diferentes fontes para gerar uma nova informação. Neste caso se encaixam produções que permitem ao interator decidir por diversos caminhos de narrativa, ou controlar elementos que são utilizados de forma diferente em processos adicionais de transformação. Apresentações de *video jockey* (VJ) – ambientes interativos controlados exclusivamente por um interator, que por sua vez determina a ordem em que determinadas sequências de vídeo são apresentadas - é um bom exemplo deste método de processamento.

É importante ressaltar que, também nesta linha de classificação, as definições utilizadas permitem certo cruzamento, variando de produção em produção. A linha que divide cada método de processamento dos dados é bastante fina, e é comum que certos processos utilizados em produções interativas desafiem a classificação.

### **3.3 Mensagem**

A última classificação – e talvez onde os métodos de classificação adotados neste projeto mais se distanciem das definições de Rowe – trata da classificação da *mensagem* criada pela produção interativa. Numa peça interativa, a mensagem criada nem sempre é controlada somente pelo artista que a desenvolve – o interator pode ter grande parte no resultado final. Nesse contexto, é possível determinar o quanto uma peça interativa é controlada *pelo interator* ou *pelo desenvolvedor*.

Quando controlada pelo *desenvolvedor*, a obra acaba se tornando algo em separado, com interação mínima. Nesse caso, a mensagem passada pela obra pode ser bastante restrita, confinada à gama de interações permitidas pelo desenvolvedor – por exemplo, uma peça que permite ao interator alternar entre diferentes imagens que são mostradas. O conteúdo é previsível e, neste caso, a possibilidade de interação fica restrita à sequência criada pelo interator para visualização das imagens; o interator funciona mais como receptor da mensagem do que como criador.

Já quando o controle está nas mãos do *interator*, a mensagem da obra acaba se tornando mais aberta, sendo ela mais baseada no processo interativo do que numa

sequência pré-programa. É o caso de peças interativas que dão ao interator pleno controle de suas ações – por exemplo, uma peça que permite ao interator criar desenhos dentro de um ambiente pouco controlado. O resultado é imprevisível, e o interator atua muito mais como criador de uma nova mensagem do que como receptor.

## **4 PROPOSTA**

O projeto Fnk consiste na criação de um ambiente de programação visual para análise, processamento e síntese de áudio e vídeo em tempo real, dado o contexto citado.

### **4.1 Objetivo principal**

Instalações e apresentações de arte e tecnologia utilizam som e imagem de forma única; por isso, acredita-se que a rotina de criação de trabalhos desse tipo pode se beneficiar de plataformas de desenvolvimento altamente visuais, especialmente quando tais plataformas permitem que o resultado de um processo interativo possa ser conferido em tempo real.

Assim, o principal objetivo do projeto é encontrar uma saída eficiente para que o desenvolvimento e a prototipagem de peças multimídia ocorra de forma rápida e eficiente. Este objetivo se desdobra em dois propósitos importantes: primeiro, criar uma plataforma para desenvolvimento atraente, que permita uma curva de aprendizado suave e o aprendizado gradual das ferramentas disponíveis; e segundo, certificar-se de que o acesso a esta plataforma seja fácil, não só em termos de presença – estar disponível em tantos computadores e situações quanto possível – mas também através de uma interface amigável e transparente, ajudando o desenvolvedor tanto quanto necessário numa tarefa que, por definição, possui grande complexidade.

### **4.2 Segmentação e posicionamento**

Em relação às plataformas de desenvolvimento já existentes, a grande diferença está no posicionamento como ferramenta intermediária, por não exigir muitos recursos de hardware e por ser algo mais voltado para aprendizado e experimentação. Seria impossível estabelecer uma comparação com todas as alternativas de ambientes para produção multimídia existente; no entanto, em comparação às plataformas de desenvolvimento já citadas, existem algumas propriedades que se destacam.

Embora extremamente eficientes, tais plataformas sofrem – cada qual à sua maneira – com certas limitações que acabam por torná-los pouco atrativos para usuários iniciantes. Isso pode contribuir para esta categoria de usuários tenha certo

desinteresse em tais plataformas, evitando que os benefícios que elas possam trazer sejam prontamente reconhecidos.

Com base *exclusivamente nos objetivos citados nesta proposta*, podemos citar como desvantagem das plataformas avaliadas anteriormente:

- Max: software comercial; manuseio pouco amigável; disponível somente para OS X e Windows.
- PuraData: manuseio pouco amigável.
- vvvv: manuseio pouco amigável; disponível somente para Windows.
- Processing: ambiente programado de forma "clássica".

É importante repetir que não é parte do escopo deste projeto *superar* tais ferramentas em suas próprias áreas. Algumas das desvantagens citadas acima, na verdade, acabam sendo um efeito de suas reais *vantagens*: o ambiente vvvv, por exemplo, só está disponível sob a plataforma Windows por utilizar recursos de aceleração de hardware disponíveis exclusivamente nesta plataforma, mas que em contrapartida o transformam numa das soluções mais eficientes para processamento de imagens em tempo real; já o ambiente Processing pode utilizar uma abordagem mais clássica através de uma linguagem de programação escrita, mas o faz de forma a permitir uma maior gama de recursos que podem ser controlados com facilidade.

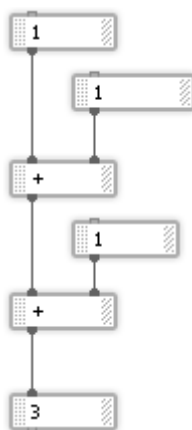
Ao contrário, o objetivo deste projeto é criar uma solução mais focada no rápido desenvolvimento e experimentação, e por isso, naturalmente voltada para desenvolvedores iniciantes. Assim, ao invés de substituir outras plataformas, este projeto busca funcionar mais como um degrau ou um elo de ligação entre diferentes modos de experimentação e criação de peças interativas.

### **4.3 Método de execução**

Assim, solução proposta para tais fins é a construção de uma plataforma para a programação visual de processos multimídia. Dentro dessa proposta, através do manuseio de *nós e ligações* entre cada um dos nós, o ambiente de desenvolvimento proposto permitirá a criação de diagramas que determinam como cada programa funcionará. Acredita-se que esta abordagem proporciona um ambiente mais amigável de desenvolvimento; como proposto por Jonhston (2004), nesta estrutura

de programação, "a facilidade com que as tarefas são realizadas fazem uma grande diferença na comparação com outras linguagens".

Isso quer dizer que a linguagem funcionará baseada numa estrutura de fluxo de dados, ou seja, cada uma das ligações presentes nos diagramas programados transportará diversos tipos de dados, como números, texto, imagens, etc; adicionalmente, cada um dos nós disponível realizará algum tipo de transformação nos dados recebidos, gerando assim novos dados que são transmitidos para outros nós seguindo o caminho estabelecido por suas ligações (Stoy, 1974). Por exemplo, a expressão  $(1+1)+1$  e seu resultado seriam representados dentro do ambiente proposto aproximadamente desta forma:

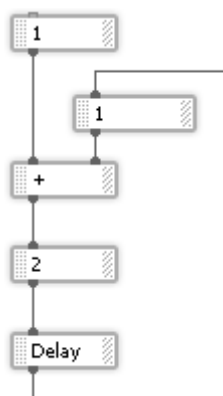


Do ponto de vista técnico, um diagrama de fluxo de dados é processado de forma sequencial. Em cada ciclo de processamento do diagrama – que deverá ser executada diversas vezes por segundo – cada nó presente na estrutura é processado uma vez. No entanto, cada nó só poderá ser processado uma vez que seus dados de entrada – recebidos através de *ligações* de entrada – tiverem sido recebidos com sucesso durante a iteração atual. Assim, há uma *árvore de dependências* que deverá ser levada em consideração, fazendo com que o diagrama seja processado de forma correta.

Esta estrutura de execução de fluxo de dados *acíclica* (Dennis, 1973) estabelece um panorama de desenvolvimento mais previsível para o desenvolvedor, uma vez que o fluxo de recebimento, transformação e transmissão dos dados é facilmente previsível: determinado pelas próprias ligações criadas, e independente da posição física de cada nó.

É interessante notar, no entanto, que pela própria filosofia adotada neste tipo de execução, certos conceitos que parecem básicos quando consideramos linguagens de programação convencionais – em especial, *loops* – se tornam algo bastante incomum e de difícil implementação, uma vez que um certo dado não pode ficar em constante transmissão – a árvore de dependências do diagrama precisa ter uma hierarquia definida de forma objetiva.

Neste caso, linguagens de fluxo de dados podem adotar nós de *espera* que permitem ao desenvolvedor criar pontos onde dados são retidos até a próxima iteração. Por exemplo, a expressão  $a = a + 1$  executada seguidamente – em loop – poderia ser representada desta forma (utilizando-se um nó de espera hipotético, intitulado "*Delay*"):

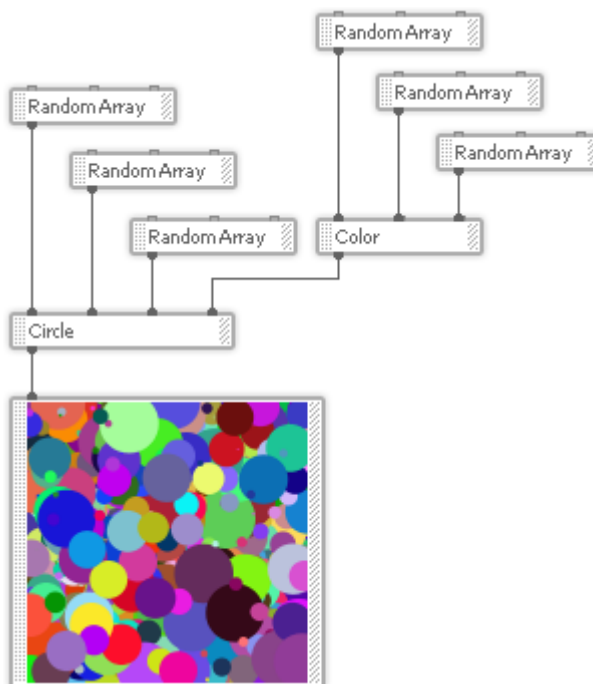


No exemplo acima, o segundo parâmetro empregado na soma seria constantemente modificado de forma a receber o valor da soma total realizada anteriormente. Ele não seria modificado imediatamente – no mesmo ciclo de processamento da soma – devido ao nó de espera empregado. Assim, o caminho de transformações executadas em cada ciclo de processamento sempre tem começo e fim definidos.

Considerando-se esta estrutura final, o projeto proposto também deverá contar com uma lista nós de transformação especialmente voltados para a criação de peças multimídia. Embora a realização de cálculos matemáticos simples – como mostrado nos exemplos acima – seja uma necessidade, a base final da linguagem proposta deverá contar com nós de transformação focados no processamento de imagem e som, e portando, eles serão a tradução dos reais recursos da linguagem.

Por exemplo, para criar uma estrutura que gera uma série de círculos de tamanho, posição e cores aleatórios, poderíamos utilizar um comando hipotético intitulado de

"Circle", alimentado por diversos parâmetros numéricos, e finalmente desenhados num nó criado especialmente para visualização de imagens:



Outro detalhe importante do ambiente de programação proposto é que a execução dos programas será constante, ou seja, não existirá distinção entre *tempo de design* e *tempo de execução* – diagramas criados dentro do ambiente serão executados ao mesmo tempo em que são criados ou editados. Devido a isso, não existe qualquer tipo de compilação ou pré-compilação envolvida, mas sim uma constante interpretação do diagrama atual, aliada a alguns recursos de ordenação da árvore de dependências e de armazenamento de dados não modificados para melhor performance.

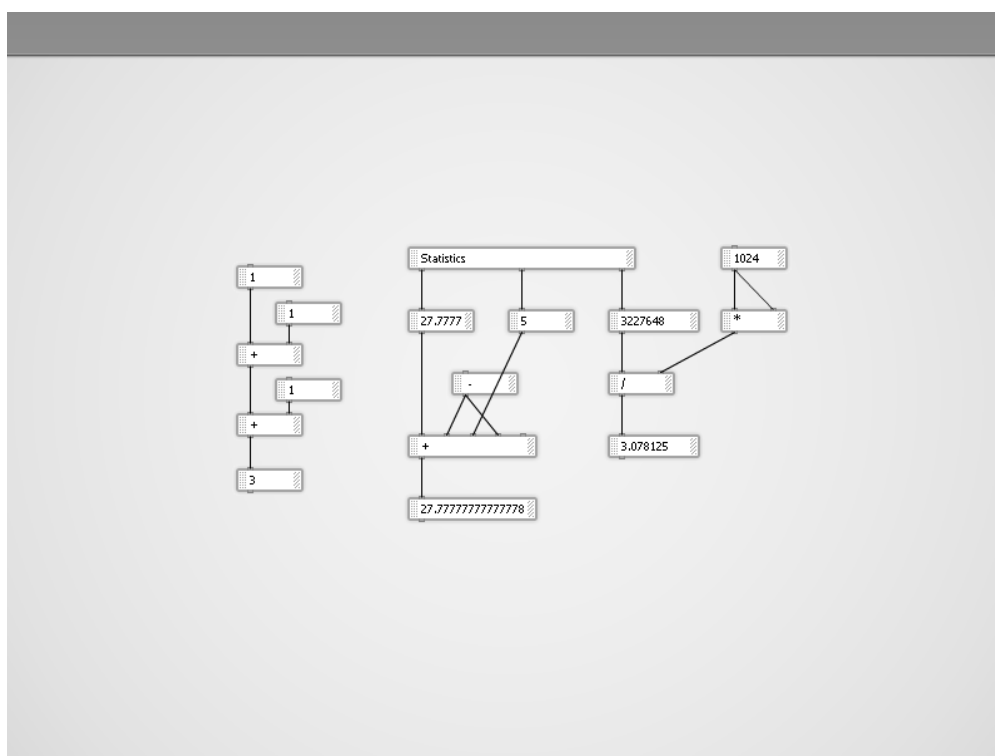
Adicionalmente, em casos específicos, um diagrama sendo executado poderá ter modos de visualização alternativos – por exemplo, no exemplo acima, a imagem gerada poderia ser visualizada diretamente em tela cheia, sem que o resto do diagrama estivesse visível. Isso dá ao ambiente a possibilidade de executar produções interativas sem expor a própria plataforma de edição ao interator, ou ainda, de utilizar novas interfaces gráficas para interação.

#### 4.4 Interface gráfica de usuário

A interface gráfica de usuário do ambiente proposto deverá oferecer ao desenvolvedor um espaço onde novos programas possam ser criados através da

disposição de nós e ligações. Devido à sua natureza gráfica, este processo de criação deve proporcionar ao desenvolvedor amplo espaço para edição, bem como permitir a escolha dos nós a serem adicionados de uma lista pré-definida de algum modo simples e rápido. A edição das propriedades dos nós já posicionados deve ser também simples, bem como a consulta à referência e ajuda – que deverá estar facilmente disponível para auxiliar o desenvolvimento.

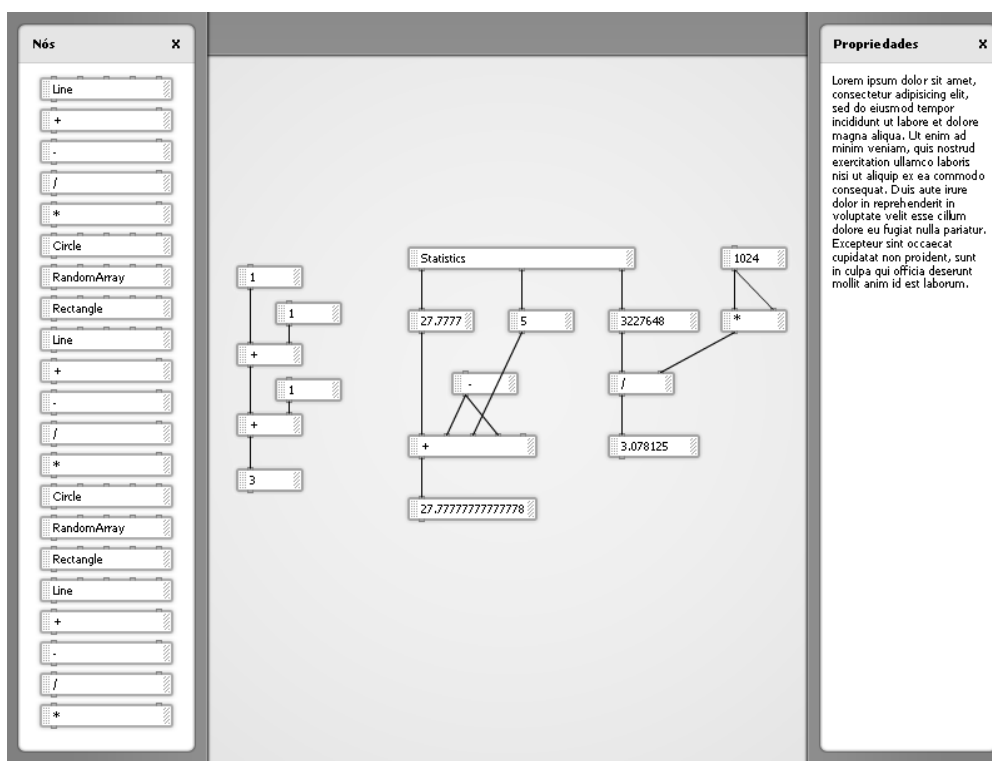
O trabalho de criação da interface final será feito em conjunto com o desenvolvimento da plataforma. Cabem aqui, no entanto, algumas considerações acerca dos elementos necessários da interface, utilizando-se um mock-up da interface como exemplo.



Propõe-se principalmente uma interface limpa e clara, onde praticamente todo o espaço disponível é usado para a edição dos diagramas. A principal exceção seria o topo da interface, usada para botões que realizam ações típicas – como salvar e abrir arquivos – e para uma lista de abas que daria acesso aos diferentes diagramas já abertos.

Outros elementos de interface também seriam necessários e estariam à disposição em certos momentos específicos. É o caso da lista de nós disponíveis para criação,

e da lista de propriedades do nó atualmente selecionado, mostrados abaixo, como exemplo, nas laterais.



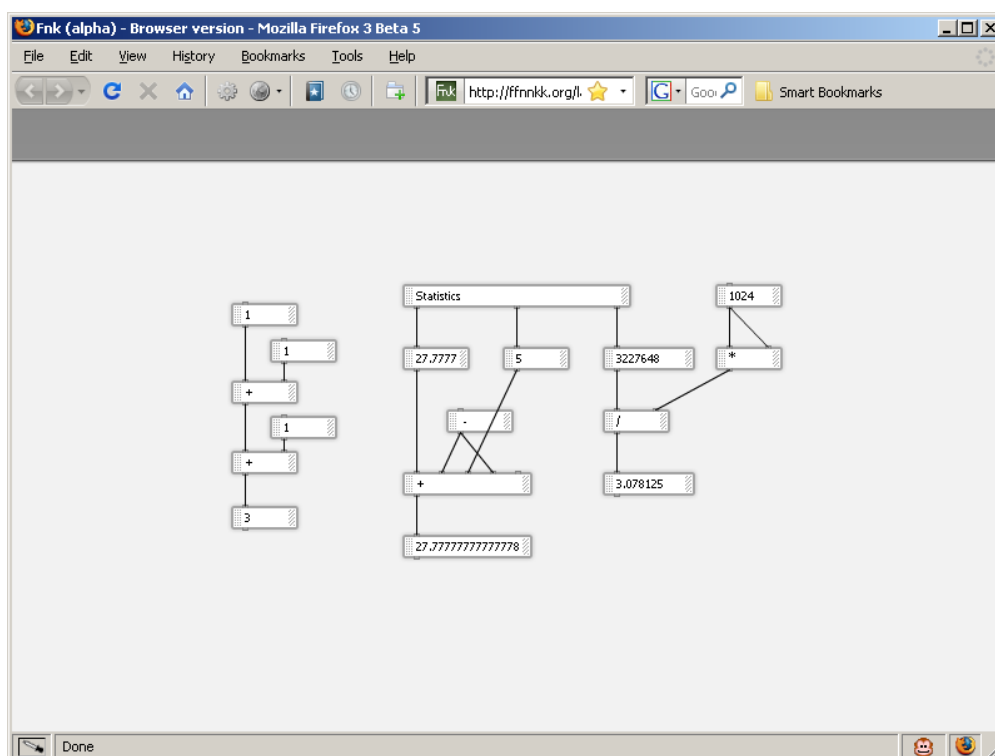
Recursos inicialmente invisíveis da interface – acionados através de atalhos de teclado ou mouse – também proporcionariam acesso a funções do ambiente, como criação de novos nós e edição de suas propriedades. Tais recursos teriam de ser feito claros de alguma forma, para que o desenvolvedor iniciante fizesse a transição para desenvolvedor experiente de forma gradual, realizando tarefas comuns de forma mais rápida.

## 4.5 Ambiente de execução

A plataforma proposta deverá também ser de fácil acesso no sentido de estar disponível tão facilmente para o desenvolvedor quanto possível. Assim, nesta proposta, poderá ser acessada de duas formas diferentes.

O primeiro – e principal – modo de utilizar o ambiente de desenvolvimento proposto será *online*, diretamente através de um *browser*, a partir da Internet, sem a necessidade de instalação de quaisquer aplicativos ou arquivos incomuns no computador do usuário. Isso não só dá a qualquer micro que utilize qualquer um dos sistemas operacionais mais populares a possibilidade de acessar o ambiente, a partir de qualquer lugar, mas também permite que usuários com recursos de

administração limitados em seus computadores – caso típico de computadores usados dentro de um ambiente acadêmico, por exemplo – tenham acesso ao ambiente. Arquivos editados também poderão ser gravados e lidos remotamente, num servidor central, permitindo que um mesmo programa criado na plataforma seja facilmente editado a partir de qualquer máquina cliente, sem a necessidade de transporte de arquivos por parte do desenvolvedor. Esta versão também permitiria a execução em tela cheia, tornando-a uma alternativa viável para a criação de instalações interativas.



O segundo método de acesso ao ambiente será através de uma versão separada, *standalone*, criada para ser instalada e então executada numa máquina específica, como um aplicativo comum. Esta versão estará disponível através de *download* do website da plataforma, e embora tenha as mesmas características da versão online, ela teria como vantagem adicional não depender de uma conexão à Internet para funcionar.

Inicialmente, o ambiente será criado utilizando-se a tecnologia Adobe Flash versão 9, garantindo assim a possibilidade de execução em aproximadamente 97% das máquinas conectadas à Internet atualmente (Millward, 2008), bem como a disponibilidade nos sistemas Windows, OS X e Linux. Seu desenvolvimento, no entanto, prevê a mudança da versão alvo para a versão 10 da tecnologia – assim

que esta versão se torne disponível publicamente – visto que esta nova versão possui importantes recursos que serão aproveitados pelo projeto proposto.

## 4.6 Exemplos a serem construídos

Por se tratar de um ambiente de desenvolvimento consideravelmente aberto, é necessário definir o escopo ao qual o projeto proposto irá responder num estágio inicial. Assim, dentro da proposta de aplicação multimídia, vários exemplos de aplicações finais serão propostos como objetivo; em seguida, o processo de desenvolvimento determinará todos os tipos de nós de transformação necessários para que o fluxo de dados seja realizado com sucesso em tais exemplos, chegando finalmente a versões funcionais dos exemplos propostos.

Assim, mais do que seu impacto visual, é importante que os exemplos sejam escolhidos de modo que uma grande gama de nós diferentes sejam necessários. Afinal, o sistema será criado para que novos *programas* sejam criados, não só para que os exemplos propostos sejam possíveis.

### 4.6.1 Gerador de padrões aleatórios repetitivos

Este exemplo serviria para a criação de figuras gráficas aleatórias numa área determinada, criando assim uma textura baseada num algoritmo generativo, como por exemplo círculos de diferentes tamanhos seguindo alterações numa cor também determinada anteriormente. A repetição e a variação de certos parâmetros – bem como a restrição a certas variações de cores ou formas – geralmente permite a criação de padrões repetitivos, mas agradáveis ao olhar. Adicionalmente, este programa poderia ter seus parâmetros de geração alterados baseado no tempo de execução, criando assim padrões móveis, aparentemente abstratos, mas com profunda base matemática.

- Nós necessários: gerador de números aleatórios, gerador de formas gráficas, gerador de cores, painel gráfico.
- Tipos de dados necessários: números, formas gráficas, imagem, cores.

### 4.6.2 Jogo da Vida de Conway

Uma replicação do *jogo da vida* de John Conway – onde células de um grid nascem e morrem baseadas em regras específicas – é um bom exemplo da manipulação de matrizes de dados e de seu processamento constante. Assim, embora esta

simulação tenha pouca relação com projetos multimídia interativos, sua implementação serve como uma boa base de recursos que seriam necessários em aplicações mais complexas. A criação deste exemplo também cria uma necessidade quase imperceptível à primeira vista: a criação de nós de controle de fila de dados. Enquanto a criação de um jogo semelhante seja relativamente simples em linguagens de programação *imperativas* – que podem contar com loops condicionais – numa linguagem de fluxo de dados sua implementação torna-se um grande problema, uma vez que a base do jogo é a transformação contínua de um mesmo conjunto de dados. Assim, criar mecanismos para esta retroalimentação de dados funcione é também um dos principais objetivos deste exemplo.

- Nós necessários: condicionais; gerador de formas gráficas, painel gráfico com interface de usuário, controlador de fila, seleções.
- Tipos de dados necessários: booleanos, matriz de booleanos, formas gráficas, imagem.

#### 4.6.3 Detecção de batidas musicais

Outra aplicação bastante comum em softwares multimídia é a detecção do ritmo produzido a partir de uma entrada de áudio, geralmente chamada de *beat detection*. Com esse recurso, é possível criar aplicações gráficas que respondem a certos estímulos sonoros, criando apresentações cujas alterações em sua representação visual coincidem com momentos-chave do áudio como percebidos pelo interator. A proposta, então, seria criar uma peça altamente gráfica que utilizaria uma música como parâmetro, com sua representação gráfica sendo gerada em tempo real, criando assim uma espécie de sinestesia visual e sonora.

- Nós necessários: leitura de arquivo de som, execução de som, detector de frequências através de *fast fourier transform* (ou outro método semelhante), expressões matemáticas, condicionais, gerador de formas gráficas, gerador de cores, painel gráfico.
- Tipos de dados necessários: texto, som, números, formas gráficas, imagem, cores.

#### 4.6.4 Síntese de som

Síntese sonora é outro recurso que geralmente faz parte de projetos interativos multimídia. Embora este projeto se disponha a realizar síntese sonora, ele o fará em baixo nível – isto é, permitindo a criação de ondas sonoras, mas sem focar em

recursos muito específicos, como os encontrados em sintetizadores mais especializados. Assim, este exemplo permitirá que sons sejam criados utilizando diferentes frequências, amplitudes e formatos de onda, permitindo assim a síntese de som simples baseada em certas regras matemáticas ou parâmetros externos.

- Nós necessários: entrada numérica, seleção de lista, gerador de ondas sonoras, execução de som.
- Tipos de dados necessários: números, som, formas gráficas, imagem.

#### **4.6.5 Controle de vídeo a partir do teclado**

É também bastante comum que apresentações multimídia tenham de lidar com vídeo, usando-o como parte de uma composição gráfica ou controlando-o de alguma forma. Este exemplo propõe a criação de uma aplicação simples de mixagem de vídeo, onde diversos vídeos diferentes devem ser controlados através do teclado, criando assim um ambiente onde o interator pode escolher que sequência é mostrada a cada momento através de uma interface não-gráfica.

- Nós necessários: leitura de arquivo de vídeo, leitura de estado do teclado, condicionais, seleções.
- Tipos de dados necessários: texto, números, imagem.

#### **4.6.6 Composição de imagem**

É igualmente importante que a imagem composta possa ser controlada antes de ser mesclada ao espaço de apresentações interativas. Este exemplo propõe demonstrar os recursos de controle de *matrizes de transformação* de imagens através de uma demonstração simples onde transformações bidimensionais – como rotação, escala, e movimentação – são aplicadas a uma imagem.

- Nós necessários: entrada numérica, leitura de arquivo de imagem, matriz de transformação de imagem.
- Tipos de dados necessários: números, formas gráficas, imagem.

#### **4.6.7 Filtros de imagem**

No processamento e síntese gráfica, filtros de modificação de imagem desempenham um papel extremamente importante. Assim, este exemplo consiste na construção de diversos métodos de filtragem e processamento de imagem,

utilizando-se um vídeo já existente como base de testes. Este é também um recurso de escopo aberto e provisório, uma vez que este recurso permitirá o uso de filtros customizados (normalmente chamados de *shaders*), possibilitando assim ao desenvolvedor criar seus próprios filtros ou trazê-los de outras localidades.

- Filtros propostos: blur (desfoque); threshold (limiar); glow (brilho); brilho; saturação; tom (hue); alteração de canais de cor.
- Nós necessários: entrada numérica, leitura de arquivo de vídeo, seleção de lista, condicionais, seleções.
- Tipos de dados necessários: números, imagem.

#### 4.6.8 Operações de imagem

Operações de imagem são operações matemáticas realizadas diretamente nos canais de cor de duas imagens. Diferentemente dos filtros, essas operações são determinadas por um operador único, e sempre levam em conta duas imagens diferentes (funcionando, então, como um modo de mistura entre imagens, ou um *blending mode*). Este exemplo demonstrará a possibilidade de se realizar operações como soma e subtração em imagens captadas através de uma câmera, de modo também a demonstrar uma das mais comuns soluções proporcionadas por este recurso: subtração do fundo de modo a obter a imagem separada do interator em frente a uma câmera.

- Nós necessários: entrada numérica, leitura de entrada de vídeo, seleção de lista, leitura de estado do teclado, condicionais, controlador de fila, seleções.
- Tipos de dados necessários: texto, números, imagem.

#### 4.6.9 Análise de imagem

Este exemplo propõe a análise de certas características da imagem, como geralmente necessário em certas aplicações interativas. Isto será feito através da criação de um programa que utilizará a câmera de vídeo para demonstrar os resultados de análises realizadas na imagem em tempo real.

- Cálculos propostos: localização de retângulos de cores na imagem (*color tracking*), análise de brilho e cor (histograma), análise de frequência de quadros.
- Nós necessários: entrada numérica, leitura de entrada de vídeo, seleção de lista, condicionais, controlador de fila, seleções.



## REFERÊNCIAS

MILLWARD Brown. **Adobe Flash Player Version Penetration**, março 2008.

Disponível em: <[http://www.adobe.com/products/player\\_census/flashplayer/version\\_penetration.html](http://www.adobe.com/products/player_census/flashplayer/version_penetration.html)>. Acesso em: 13 maio 2008.

ARS Electronica: Website. Disponível em: <<http://www.aec.at/en/index.asp>>. Acesso em: 7 maio 2008.

CANDY, Linda; EDMONDS, Ernest. Interaction in Art and Technology. **Crossings: Electronic Journal of Art and Technology**, Leicestershire, Inglaterra, v. 2, n. 1, março 2002. Disponível em: <<http://crossings.tcd.ie/issues/2.1/Candy>>. Acesso em: 14 abril 2008.

CLEMENTS, Douglas H. Computers in Early Childhood Mathematics. **Contemporary Issues in Early Childhood**, New York, EUA, v. 3, n. 2, 2002.

DENNIS, Jack B. FOSSEEN, John B. **Introduction to Data Flow Schemas**. Computation Structures Group, Project MAC, M.I.T., setembro 1973.

COULTER-SMITH, Graham. **Deconstructing Installation Art: Fine Art and Media Art, 1986-2006**. Southampton, Reino Unido: Casiad Publishing, 2006. Disponível em: <<http://www.installationart.net/>>. Acesso em: 14 abril 2008.

EMOÇÃO Art.ficial: Website. Disponível em: <[http://www.itaucultural.org.br/index.cfm?cd\\_pagina=2597](http://www.itaucultural.org.br/index.cfm?cd_pagina=2597)>. Acesso em: 7 maio 2008.

FILE Festival Internacional de Linguagem Eletrônica: Website. Disponível em: <<http://www.file.org.br>>. Acesso em: 7 maio 2008.

JOHNSON, Steven. **Emergence: The connected lives of ants, brains, cities, and software**. New York, EUA: Scribner, 2001.

JOHNSTON, Wesley M.; HANNA, J. R. Paul; MILLAR, Richard J. Advances in dataflow programming languages. **ACM Computing Surveys**, New York, EUA, v. 36, n. 1, março 2004.

MULLER, Jim. **The great Logo Adventure - Discovering Logo On and Off the computer**. Madison, AI, EUA: Doone Publications, 1997.

PROCESSING 1.0 (BETA): Website. Disponível em: <<http://processing.org>>. Acesso em: 14 abril 2008.

PURE DATA – PD Community Site: Website. Disponível em: <<http://puredata.info>>. Acesso em: 14 abril 2008.

ROWE, Robert; **Interactive Music Systems**. Cambridge, MA, EUA: The MIT Press, 1993.

STOY, Joseph E. **Proofs of Correctness of Dataflow Programs**. Computation Structures Group, Project MAC, M.I.T., setembro 1974.

VVVV a multipurpose toolkit: Website. Disponível em: <<http://vvvv.org>>. Acesso em: 14 abril 2008.

WIKIPEDIA the free encyclopedia. **Max (software)**. Disponível em: <<http://en.wikipedia.org/wiki/Max/MSP>>. Acesso em: 14 abril 2008.