

# **Fnk**

**Um ambiente de programação visual  
para processamento de som e imagem**

**José Fernando B. Caneiro**

**Centro Universitário Senac  
São Paulo, 2008**

JOSÉ FERNANDO BALDO CANEIRO

# **Fnk**

**Um ambiente de programação  
visual para projetos multimídia**

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 Baldo Caneiro

Título: Fnk, um ambiente de programação visual para projetos multimídia

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



A meus pais, que sempre me apoiaram,

E a meus colegas de faculdade, gigantes a despertar.



# AGRADECIMENTOS

Agradecimentos especiais para Ricardo Cabello, Elise Roese, Marcelo Petriaggi, Fernando Magalhães, André Luiz Mariano e Gabriela Simões pelo apoio, além de meu orientador, Ruggero Ruschioni.



# 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 para processamento 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 se utilizando de uma 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 process 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.



# LISTA DE ILUSTRAÇÕES

Figura 1 - Desenhando uma flor passo-a-passo em Logo.....	23
Figura 2 - Software Max processando áudio.....	28
Figura 3 - Software Pure Data processando áudio.....	29
Figura 4 - Software vvvv criando imagens.....	30
Figura 5 - Software Processing criando imagens.....	31
Figura 6 - Um Nó único.....	44
Figura 7 - Tooltip mostrando dados de um conector.....	45
Figura 8 - Uma soma simples realizada no Fnk.....	45
Figura 9 - Um diagrama gerando símbolos com diferentes atributos.....	47
Figura 10 - Uma ligação entre dois nós.....	49
Figura 11 - Ordem de processamento de um diagrama.....	49
Figura 12 - Uma ligação de atraso criando um loop.....	52
Figura 13 - Seleção de dados condicionais.....	53
Figura 14 - Interface de edição principal do software.....	54
Figura 15 - Fnk sendo rodado a partir do navegador.....	56
Figura 16 - Website da ferramenta.....	57
Figura 17 - Taxa de adoção do Flash Player.....	59
Figura 18 - Versão do Flash usada por visitantes do website.....	60
Figura 19 - Menu de lições, e uma lição aberta abaixo.....	62
Figura 20 - Ajuda sobre o nó “Display” (em inglês).....	63
Figura 21 - Um exemplo sendo executado.....	67



# SUMÁRIO

<b>1 Introdução.....</b>	<b>17</b>
1.1 Arte contemporânea: o problema do museu.....	18
1.2 Tecnologia como suporte para a arte.....	19
1.3 Programação de computadores como instrumento de educação.....	21
<b>2 Panorama atual.....</b>	<b>27</b>
2.1 Softwares para interação.....	28
2.2 Cenário sócio-cultural .....	32
2.3 Ações online.....	35
<b>3 Metodologia de desenvolvimento.....</b>	<b>37</b>
3.1 Objetivos principais.....	38
3.2 Posicionamento do software.....	38
3.3 Público-alvo.....	39
3.4 Modelo de funcionamento.....	43
3.4.1 Lógica de execução.....	43
3.4.2 Representação visual.....	43
3.4.3 Representação de cada nó.....	46
3.4.4 Ordem de processamento .....	48
3.4.5 Questões comparativas.....	50
3.5 Interface gráfica de usuário.....	54
3.6 Ambiente de execução .....	55
3.7 Plataforma de desenvolvimento .....	58
3.8 Documentação e ajuda.....	61
3.9 Testes de usabilidade.....	64
3.9.1 Usuários.....	64

3.9.2 Formato e objetivos do teste.....	65
3.9.3 Erros apontados e correções.....	65
3.10 Exemplos avançados.....	67
<b>4 Considerações finais.....</b>	<b>69</b>
4.1 Manutenção de documentos online.....	70
4.2 Mais nós.....	70
4.3 Módulos de nós.....	71
4.4 Compartilhamento de documentos.....	71
4.5 Execução e embed de patches.....	72
4.6 Edição simultânea.....	72
4.7 Recursos da versão standalone.....	72
<b>Referências.....</b>	<b>75</b>
<b>Apêndice A – Lições internas.....</b>	<b>79</b>
<b>Apêndice B – Roteiro de testes iniciais.....</b>	<b>101</b>
<b>Anexo A – Estatísticas de versão Flash.....</b>	<b>105</b>
<b>Anexo B – Estatísticas de upgrade do Flash.....</b>	<b>107</b>
<b>Anexo C – Taxa histórica de adoção do Flash Player.....</b>	<b>109</b>

# 1 INTRODUÇÃO

O projeto Fnk consiste na construção de um ambiente de desenvolvimento de peças multimídia interativas através de uma linguagem de programação visual.

Para que o contexto no qual o projeto se insere fique claro, existem várias linhas de pesquisa e desenvolvimento que devem ser levadas em consideração; estas linhas transitam entre núcleos de arte, tecnologia, educação e experimentação.

## 1.1 Arte contemporânea: o problema do museu

Em meados da década de 90, a produção artística encarava um problema de posicionamento. O trabalho artístico – criado para o espaço de museus ou galerias – distanciava-se do público, tornando-se algo intangível, quase que sagrado; 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 se torna 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. Para Jacques Lacan, 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 (Cukiert, 2002). Esta teoria 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. É a partir desta premissa que a eletrônica começa a se aliar à produção artística; como os recursos de interação permitidos por interfaces controladas digitalmente ultrapassam o que era possível através de meios convencionais, 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 suportes eletrônicos, no entanto, há uma convergência entre a arte que busca novos meios de interação através do uso de recursos eletrônicos e da tecnologia digital que busca novos meios de expressão através da arte; pesquisas sobre o processo criativo e interativo – e o impacto da eletrônica 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 pontos de partida como esse 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 digital, 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 *interator*.

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

É nesse contexto que plataformas de desenvolvimento de peças multimídia buscam se inserir, ao oferecer ao artista ferramentas de desenvolvimento que possibilitam o controle de diferentes tecnologias; tornar este controle acessível é tarefa de tais ferramentas, bem como deste projeto.

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

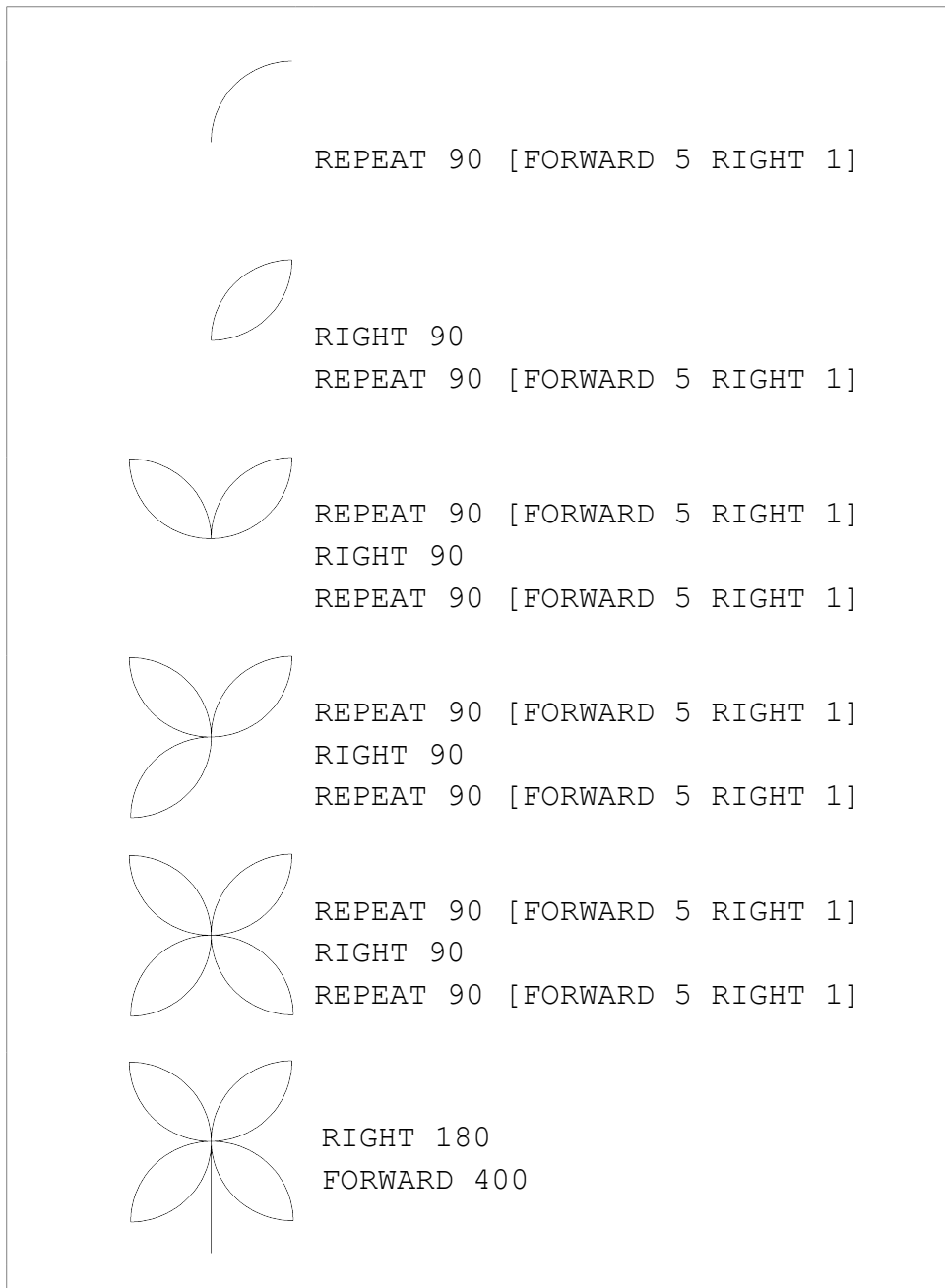
Além de ser usada como suporte para a arte, no entanto, meios digitais também têm traçado outro caminho em paralelo, desta vez no círculo acadêmico: como ferramenta no auxílio da educação.

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 (Clements, 2002). 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 é uma curva de aprendizado mais suave, seguindo uma filosofia de que não deveria haver *limites* impostos à linguagem (Muller, 1997); o usuário deveria ver o resultado do que estava tentando realizar de forma imediata. Mais do que servir de introdução à programação de computadores através de uma linguagem específica, no entanto, o propósito original da linguagem era funcionar como um auxílio ao ensino de conceitos envolvidos na matemática computacional.

Nesse sentido, a linguagem não deixou de obter sucesso. Porém, o que se destacava na mesma era a facilidade de interação com dispositivos gráficos: 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, percorre uma superfície de papel reproduzindo os desenhos num suporte físico ao invés da tela de um computador.



*Figura 1 - Desenhando uma flor passo-a-passo em Logo*

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 com sucesso (Papert, 1980). Assim, no final da década de 60, o Logo tornara-se uma ferramenta madura de educação, não devido à sua capacidade de aplicação prática – na construção de aplicativos reais – mas sim em sua facilidade de engajar o usuário, em especial crianças, e ao modo como estimulava a experimentação em um "ambiente rico que estimula a reflexão sobre a matemática e sobre as rotinas de solução de problemas de um indivíduo" (Clements, 2002, p. 163).

Outro sinal da popularidade do Logo no meio acadêmico 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 (Kelleher, 2005). Talvez o melhor exemplo desta tendência tenha sido 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 (Resnick, 1996).

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 (Kay, 2005).

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



## **2 PANORAMA ATUAL**

A convergência das áreas de pesquisa educacional e artística acabou contribuindo para o surgimento de um novo segmento de desenvolvimento de software: o de softwares para experimentação artística, menos voltados à produção comercial. Talvez a maior mudança iniciada por tal segmento tenha sido tornar os recursos de processamento de áudio e vídeo mais acessíveis para desenvolvedores amadores, removendo a necessidade de profundos conhecimentos técnicos em eletrônica digital ou programação de computadores.



dados uns aos outros, de forma visual, determinando assim o funcionamento de cada programa (que, dentro do Max, é chamado de *patch*).

Além disso, nos últimos anos, diversas extensões foram lançadas para este software, como o *MSP* – para síntese e processamento de áudio digital – e o *Jitter*, para processamento de vídeo. Somando-se a isso, inúmeros periféricos de entrada e saída de dados podem interagir com o software. Tudo isso contribuiu para que o Max se transformasse numa 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 (Puckette, 2005).

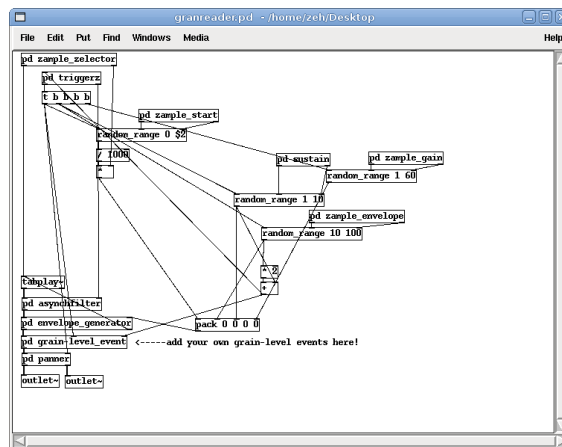


Figura 3 - Software Pure Data processando áudio

Além do próprio Max, vários outros softwares foram criados com objetivos semelhantes. Um deles é o *Pure Data* (ou *Pd*), criado também por Miller Puckette

em 1996; com foco semelhante, Pure Data utiliza também um ambiente de programação visual para síntese e processamento de áudio em tempo real. A grande diferença em comparação ao Max é o fato de não possuir um modelo de exploração comercial, já que se trata de um software gratuito – feito "como código aberto desde o princípio" (Puckette, 2005, p.6) – ao contrário do Max.

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.

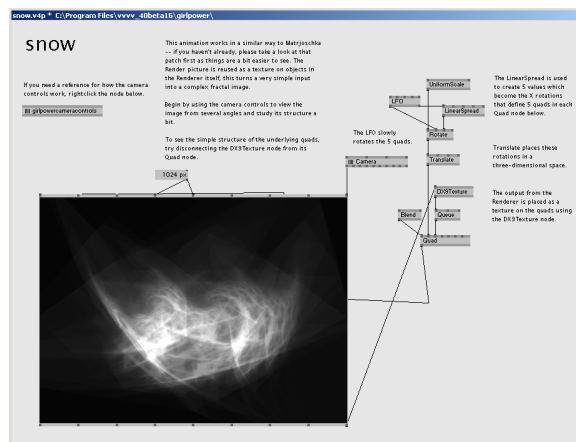


Figura 4 - Software vvvv criando imagens

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 na criação de módulos de processamento de dados baseados em nós e ligações, trabalhando assim um diagrama de *fluxo de dados* que permite chegar a um resultado final através da manipulação contínua de dados. Devido a isso, este modo de desenvolvimento se assemelha mais a uma linguagem funcional do que uma linguagem de programação imperativa (Stoy, 1974).

Entretanto, 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.

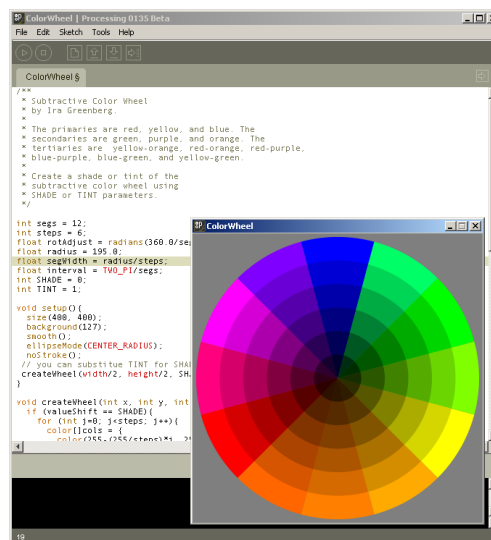


Figura 5 - Software Processing criando imagens

Independente dos paradigmas de desenvolvimento adotados, estes softwares acabaram não só servindo como solução para a criação de peças interativas, mas também para a popularização deste segmento de desenvolvimento, inclusive dentro do campo acadêmico. Muitos outros softwares poderiam ser citados aqui – como exemplos claros, podemos citar *LabVIEW*, *ROBOLAB* (ambiente utilizado para programação de *Lego Mindstorms*), *Lily*, *Quartz Composer*, e outros. Os softwares citados anteriormente, no entanto, são os softwares mais significativos na área que este projeto aborda, e o foco desta contextualização.

## 2.2 Cenário sócio-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 (2006, c. 1, p. 3, tradução livre):

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 impedi-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.

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 surgido 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 *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; conseqüentemente, 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**

Outro fator que deve ser notado hoje é a presença online de comunidades de desenvolvimento baseadas nas ferramentas utilizadas para a criação de peças multimídia interativas. É o caso dos *websites* dos próprios softwares: 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, como a necessidade de se criar uma peça interativa, as plataformas de desenvolvimento disponíveis acabam se tornando elas mesmas agregadoras de indivíduos com interesses em comum. Assim, plataformas que buscam difundir-se entre o meio artístico ou experimental precisam não só oferecer ferramentas eficientes para artistas e desenvolvedores, mas também levar em consideração que estes mesmos artistas e desenvolvedores precisam de um ponto de encontro central, onde eles possam discutir e aprender aspectos da ferramenta proposta, bem como ter contato com o conteúdo que está sendo produzido por outros desenvolvedores através da ferramenta.

### **3 METODOLOGIA DE DESENVOLVIMENTO**

Considerando-se o contexto citado anteriormente, o projeto Fnk consiste na criação de um ambiente de programação visual para análise, processamento e síntese de áudio e vídeo em tempo real. Este projeto parte do pressuposto que a rotina de criação de trabalhos deste tipo pode se beneficiar de plataformas de desenvolvimento altamente visuais, em especial quando tais plataformas permitem que o resultado de um processo interativo possa ser conferido em tempo real.

### **3.1 Objetivos principais**

O principal objetivo do projeto é criar uma saída satisfatória para que o desenvolvimento e a prototipagem de peças multimídia ocorra de forma rápida e eficiente. Este objetivo se desdobra em duas tarefas principais: primeiro, criar uma plataforma para desenvolvimento atraente, de modo a facilitar o aprendizado e a experimentação; e segundo, certificar-se de que o acesso a esta plataforma seja o mais fácil possível.

### **3.2 Posicionamento do software**

Consideradas as metas principais do projeto, o projeto proposto visa atuar principalmente como ferramenta complementar às plataformas de desenvolvimento já existentes na área de programação visual. A grande diferença está no posicionamento deste projeto como ferramenta intermediária, mais voltada para o aprendizado e a experimentação, do que como ferramenta para produção profissional.

Seria impossível estabelecer uma comparação com todas as alternativas de ambientes para desenvolvimento multimídia existentes; no entanto, em comparação às plataformas de desenvolvimento citadas anteriormente, algumas diferenças se destacam. Antes de mais nada, tais plataformas sofrem – cada qual à sua maneira – com certas limitações que acabam por torná-las pouco atraentes para usuários iniciantes, fazendo com que a prática de desenvolvimento em cada

uma destas fique restrita a usuários com conhecimentos mais avançados sobre síntese visual e sonora. Como problemas gerais, e com base exclusivamente nos objetivos citados nesta proposta, podemos citar a restrição a certos sistemas operacionais (como o vvvv, restrito a ambientes Windows), a adoção de modelos comerciais que requerem a compra do software para uso (caso do Max) ou licenças para uso comercial (caso do vvvv), e finalmente, a necessidade de instalação de cada plataforma como um aplicativo separado – o que restringe sua execução dentro de ambientes mais restritos, onde o usuário não tenha acesso administrativo ao computador sendo usado, como um ambiente acadêmico.

É importante ressaltar que não é parte do objetivo deste projeto *superar* tais ferramentas em suas próprias áreas de atuação. Algumas das desvantagens citadas, na verdade, acabam sendo um efeito de suas reais vantagens: o programa vvvv, por exemplo, só está disponível sob a plataforma Windows por utilizar recursos de aceleração de hardware disponíveis exclusivamente nesta plataforma (DirectX), mas que em contrapartida o transformam numa das soluções mais eficientes para processamento de imagens em tempo real; da mesma forma, não há nada de errado num modelo de exploração comercial para softwares dessa área, mas para este projeto, considera-se que restrições deste tipo impostas à disponibilidade do ambiente sejam características negativas.

### **3.3 Público-alvo**

Baseado neste modelo, o projeto é voltado para desenvolvedores *iniciantes*, que

não tenham ainda experiência com o uso de ambientes de programação visuais. Assim, o ambiente proposto busca funcionar mais como um degrau entre diferentes modos de experimentação e criação de peças interativas; ao mesmo tempo que ele possa não ser tão eficiente no uso do hardware disponível quanto outras ferramentas mais avançadas, ele busca compensar esta deficiência pela facilidade de acesso ao mesmo.

Talvez essa proposta seja melhor explicada através de casos paralelos existentes. Nesse sentido, um bom exemplo é o aplicativo *FontStruct*, criado pela empresa Alemã *FontShop*. Trata-se de uma ferramenta para criação de fontes que é executada a partir de um browser, sem a necessidade de instalação de qualquer aplicativo em separado além do *plugin* Flash; ou seja, tudo que o usuário precisa fazer para criar suas próprias fontes é visitar o website da própria ferramenta. Apesar de bastante limitado em termos de recursos – uma vez restringe o usuário à criação de fontes modulares, baseadas num grid – este aplicativo atingiu relativo sucesso rapidamente, devido em maior parte à facilidade de acesso à ferramenta. Como escreve Fagone (2008, tradução livre), "[A] vasta maioria dos usuários do FontStruct não são designers profissionais, mas entusiastas de fontes". Esse apelo para o amadorismo, no caso do FontStruct, acabou se concretizando numa ferramenta que é "puramente agradável - algo para deslanchar a criatividade", ainda que sem um objetivo claro – "FontStruct é sobre diversão e experimentação descompromissada, a pura diversão de criar formas tipográficas". Finalmente, e talvez mais importante, o autor nota que "Serendipidade é o ponto. FontStruct te força a ter a mente aberta, a ter carinho pelas formas com o qual você se depara".

Esse convite à experimentação, coisa que talvez não fosse desejada numa ferramenta de desenvolvimento mais séria, é um dos principais pontos positivos num aplicativo como FontStruct.

Talvez esse papel de *entusiasta amador* fique mais claro quando levamos em consideração a abundância de novas aplicativos que utilizam a web como sua plataforma de execução. Novamente, o principal atrativo de tais ferramentas é a facilidade com que qualquer um tem acesso a elas, mais do que a oferta de recursos inéditos. Outro bom exemplo nesse sentido é o aplicativo *Picnik*, um website que permite a usuários realizarem retoques e edições em suas próprias imagens, ou imagens oriundas de sites sociais, como *Flickr*, *Facebook* e outros. Este aplicativo não pretende ser superior em recursos a ferramentas convencionais de edição de imagens já existentes, como o *Adobe Photoshop*, mas sim, ser uma saída mais conveniente para uma gama de usuários amadores. Os objetivos da ferramenta ficam claros nesta resposta dada por Jonathan Sposato – CEO do *Picnik.com* – numa entrevista (McKee, 2007, tradução livre):

McKee: Conte-me sobre Picnik... o que é? Por que você o construiu?

Jonathan: Picnik dá superpoderes de edição de fotos num browser a "pessoais reais". É a maneira mais fácil de se fazer algo legal com uma foto se você quiser recortá-la, reduzir olhos vermelhos, ser criativo e adicionar uma gama louca de efeitos, imprimi-la, salvá-la para o Flickr, ou mandá-la por email para alguém. Basicamente qualquer verbo que você quiser aplicar a uma foto, nós gostaríamos que o Picnik cumprisse. Mas a beleza real do Picnik é que tudo isso acontece num browser, o que significa que não há absolutamente nada a baixar ou instalar, e sua

experiência é interligada sem obstáculos com qualquer outro lugar na web onde você pode obter uma foto, ou guardar uma foto.

Por que nós o construímos? Em termos gerais nós somos um time altamente motivado a abraçar os desafios difíceis e incomuns. Nós simplesmente enxergamos uma grande oportunidade em duas coisas: dar às pessoas métodos fabulosos de edição de imagem de uma maneira nunca antes vista, fazendo-o fácil e gratuito (finalmente); e criando um tipo de aplicativo da próxima geração que fosse rico e dinâmico, e que ainda rode dentro de um browser.

Embora seja uma ferramenta com um propósito mais específico – edição de imagens – fica claro através dessa resposta que o Picnik busca atuar de forma mais descompromissada, de forma a dar a usuários diversas possibilidades de exploração, mais do que um caminho fixo para a realização de tarefas complexas específicas, como é o caso de ferramentas de edição profissionais.

Este mesmo indeterminismo é abraçado no projeto Fnk. Embora esta proposta seja mais técnica, uma vez que exige certo conhecimento de processamento de imagem, som e lógica, o foco está no usuário amador, descompromissado, mas interessado em criar algo novo através do uso de uma ferramenta que até então não estava a seu alcance; e, talvez, funcionando até como introdução para algumas das plataformas citadas anteriormente, uma vez que o conhecimento adquirido através do uso do projeto proposto seria aproveitado ao se utilizar uma ferramenta como vvvv ou similares.

## **3.4 Modelo de funcionamento**

### **3.4.1 Lógica de execução**

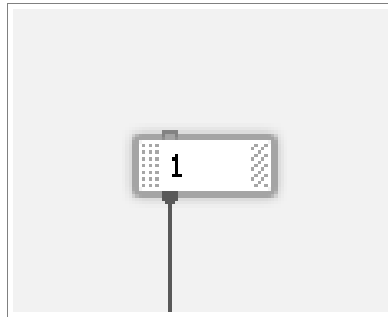
Em termos técnicos, o ponto chave do projeto é permitir a programação *visual* de processos multimídia. Dentro dessa proposta, o mecanismo de processamento utilizado pela ferramenta é baseado numa estrutura de fluxo de dados, como presente em algumas das plataformas de desenvolvimento citadas anteriormente. Este paradigma de desenvolvimento foi escolhido por oferecer uma solução mais concreta para problemas de lógica, evitando a mudança de paradigmas que é necessária durante o início do processo de programação baseado em linguagens imperativas (Johnston, 2004).

Outro detalhe importante do ambiente de programação proposto é que a execução dos programas é constante, ou seja, há distinção entre *tempo de design* e *tempo de execução* – diagramas criados dentro do ambiente são executados ao mesmo tempo em que são criados ou editados. Devido a isso, não existe qualquer etapa de compilação ou pré-compilação envolvida, mas sim uma constante interpretação do diagrama atual.

### **3.4.2 Representação visual**

Também a representação da linguagem se dá de forma gráfica, através de *nós* que representam cada uma das transformações possíveis dentro do ambiente, e de *ligações* entre cada um desses nós. Cada uma das ligações presentes nos diagramas

programados transporta diversos tipos de dados de um nó para outro, como números, texto, imagens etc.



*Figura 6 - Um Nó único*

A criação e o manuseio desses nós e de suas ligações permite a criação de diagramas que determinam como cada peça criada dentro da ferramenta se comporta. Isso contribui para uma estrutura de desenvolvimento mais amigável, uma vez que este projeto considera que "a facilidade com que tarefas são realizadas [em ambientes visuais] fazem uma grande diferença em como [suas linguagens] se comparam com outras linguagens", além de que ambientes de programação visual "permitem ao desenvolvedor realizar o planejamento e a implementação seguindo a ordem de sua preferência, e portanto, tornando o processo de desenvolvimento mais livre e mais fácil" (Johnston, 2004, tradução livre).

Cada nó possui também diversos *conectores*, de número fixo para tipo de nó diferente. É a partir destes conectores que ligações podem ser realizadas entre cada nó; cada conector serve um fim específico, de acordo com sua ordem, e pode carregar um tipo único de dado. A fim de facilitar o desenvolvimento, uma

descrição do propósito de cada conector, os tipos de dados que ele pode receber ou enviar, e os dados atualmente lá armazenados são mostrados através de um *tooltip* quando o usuário passa o mouse por cima do conector. Isso ajuda a manter a área utilizada por cada nó como a mínima possível, mas ainda possibilitando uma fácil leitura dos recursos e parâmetros de cada nó.

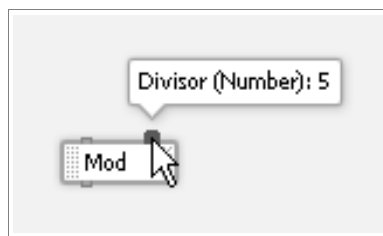


Figura 7 - Tooltip mostrando dados de um conector

Os diagramas criados sempre possuem início(s) e um fim(s) bem definidos, pontos onde o tráfego de dados inicia e termina; tais dados são transmitidos de nó para nó seguindo a ordem e o caminho estabelecido por suas ligações. Por exemplo, a expressão matemática  $x=1+1$  e seu resultado são representados dentro do ambiente proposto desta forma:

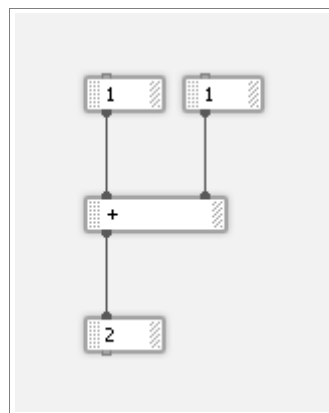


Figura 8 - Uma soma simples realizada no Fnk

Assim, podemos pensar num diagrama de fluxo de dados como sendo a representação gráfica de uma função matemática: temos dados iniciais (cada um dos valores ou parâmetros utilizados numa função) e dados finais (o valor retornado pela própria função).

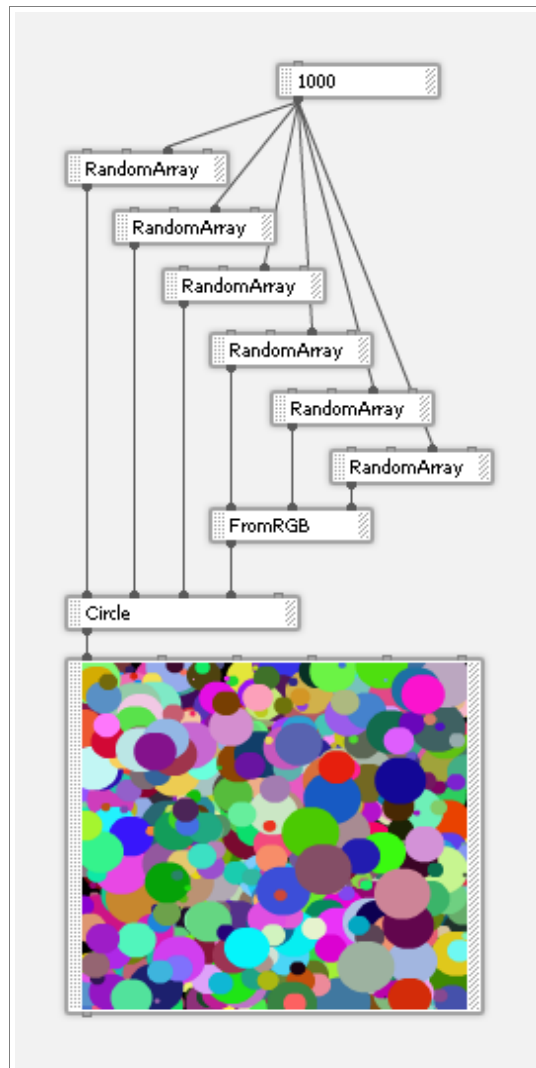
Em relação à disposição dos nós e ligações, o programa funciona como uma ferramenta de edição gráfica, semelhante a softwares como *Adobe Illustrator*, *FreeHand*, *CorelDRAW* e outros; ou seja, a disposição de nós e ligações é livre, podendo estes serem criados, movidos, redimensionados e apagados a qualquer momento.

Adicionalmente, o diagrama que está sendo executado possui diferentes modos de visualização. Assim, a visualização normal do ambiente – o diagrama sendo editado – pode ser substituída por uma representação gráfica criada por um dos nós do próprio diagrama, e vista em tela cheia – basicamente, uma troca entre um modo de *desenvolvedor* e um modo de *execução pública*. Isso dá ao ambiente a possibilidade de executar produções interativas sem expor a própria plataforma de edição ao interator.

### **3.4.3 Representação de cada nó**

Diversos tipos de nós estão disponíveis na linguagem, cada qual pertencendo a uma categoria distinta. Considerando-se a estrutura onde cada nó funciona como um mecanismo de transformação de dados, o projeto conta com uma lista nós de transformação especialmente voltados para a criação de peças multimídia, tais

como cálculos, manipulação de strings, criação de gráficos, etc. Estes nós podem ser considerados o *vocabulário* da linguagem, cada um lidando com um recurso diferente na programação de diagramas dentro do ambiente.



*Figura 9 - Um diagrama gerando símbolos com diferentes atributos*

A representação destes nós se dá de forma escrita, através de nomes descritivos

para cada nós, no caso de nós de transformação, ou através da representação dos valores ali contidos, para nós de entrada e saída de dados. Embora soe como uma saída atraente, este projeto considera que uma representação gráfica para cada nó diferente através de ícones poderia criar diagramas confusos e de difícil entendimento; uma vez que leva-se em consideração a expansão da linguagem no futuro, espera-se que a quantidade de nós de transformação permaneça em constante crescimento.

Assim, foi preferida uma representação mais abstrata mas que permitisse uma leitura clara do propósito de cada nó, daí a escolha dos nós serem representados por seu próprio nome. Além disso, a ferramenta permite a desenvolvedores mais avançados a criação de novos nós a partir da digitação do nome do nó escolhido, tornando o processo de criação de diagramas mais rápido caso o desenvolvedor já saiba o que quer criar; assim, ao invés de criar duas representações diferentes (gráfica e textual), o ambiente mantém a unidade da linguagem ao adotar a representação textual de forma exclusiva.

### **3.4.4 Ordem de processamento**

Ligações em diagramas criados neste projeto seguem basicamente um caminho de cima para baixo. Esta é uma convenção criada pela forma como cada nó funciona – conectores superiores são conectores de entrada, enquanto que conectores inferiores são conectores de saída. Não existe qualquer restrição quanto ao posicionamento de nós dentro de um diagrama, no entanto; é perfeitamente

possível que dados sejam transmitidos de nós na base de um diagrama para nós localizados acima.

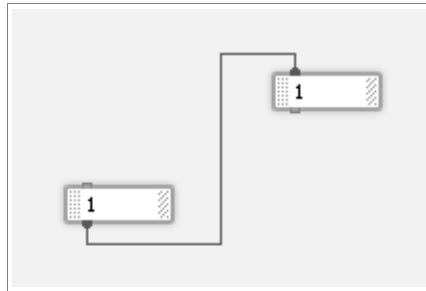


Figura 10 - Uma ligação entre dois nós

Do ponto de vista lógico, um diagrama de fluxo de dados é processado de forma sequencial. Em cada ciclo de processamento do diagrama – executado diversas vezes por segundo – cada nó presente na estrutura é processado uma única vez. Além disso, cada nó só poderá ser processado uma vez que seus dados de entrada tenham sido recebidos com sucesso através de ligações durante o ciclo atual. Assim, há uma *árvore de dependências* que deverá ser levada em consideração para que o diagrama seja processado de forma correta.

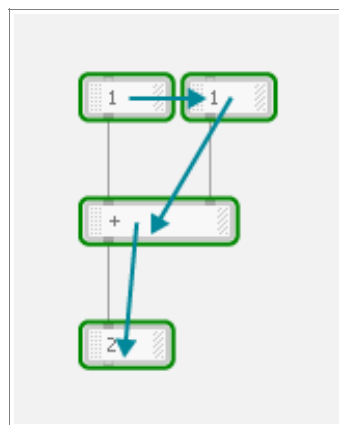


Figura 11 - Ordem de processamento de um diagrama

Esta estrutura de execução de fluxo de dados *acíclica* – isto é, sem ciclos diretos (Dennis, 1973) – tenta estabelecer um paradigma de execução mais previsível para o desenvolvedor, uma vez que o fluxo de recebimento, transformação e transmissão dos dados é determinado diretamente pelas próprias ligações criadas, independente da posição física de cada nó.

### **3.4.5 Questões comparativas**

É interessante notar, no entanto, que pela própria filosofia adotada em linguagens de fluxo de dados, certos conceitos que parecem óbvios quando consideramos linguagens de programação imperativas – em especial, *loops* – se tornam algo bastante incomum e de difícil implementação, uma vez que um certo dado não pode ficar em constante transmissão (Culler, 1986); a árvore de dependências de um diagrama precisa ter uma hierarquia definida de forma objetiva. Neste caso, linguagens baseadas em fluxo de dados que permitam o uso de dados de saída como dados de entrada – numa espécie de retroalimentação – precisam criar *pontos de espera* onde o processamento de um diagrama é interrompido até seu próximo ciclo; estes pontos funcionam como inícios e fins simulados para um diagrama.

Existem diferentes formas de implementar este conceito. O software Max, por exemplo, ao adotar um método de processamento diferente, baseado na posição dos nós no diagrama – que é executado da direita para a esquerda, de cima para baixo – faz com que essas esperas sejam automaticamente criadas, dependendo

unicamente da direção física adotada por cada ligação; assim, loops tornam-se possíveis dentro de sua estrutura de forma clara. Um efeito colateral desta solução, no entanto, é a dificuldade de prever como a execução de um diagrama se dará, uma vez que a ordem de execução não é perfeitamente clara para o desenvolvedor, podendo variar de acordo com o número de diagramas usados ou sua posição.

Esta incerteza é ausente num software como o vvvv, que adota um modelo mais rígido de construção de diagramas: loops não são permitidos, a menos que passem por um nó especial de espera – chamado de *FrameDelay* – que retém dados por um ciclo, sem gerar dependências imediatas para os nós ligados a ele.

O modelo adotado no projeto Fnk para permitir loops semelhantes pode ser definido como um misto destas duas soluções. A princípio, todo tipo de ligação é permitida; no entanto, quando uma ligação é realizada entre dois nós que estão em posições diferentes na lista de execução – isto é, dois nós que já fazem parte do mesmo ciclo de execução, mas numa ordem diferente que a nova ligação exige – é criada uma ligação *atrasada*, isto é, um ligação que interrompe a retransmissão de dados até o próximo ciclo, e que não gera dependências imediatas para os nós ligados a ela.

Este recurso permite que loops sejam criados de forma que cada execução do diagrama seja equivalente a uma *iteração* de um ciclo fechado de nós estabelecidos pelas ligações. Assim, no diagrama abaixo – equivalente a algo como  $x=x+1$  – a execução contínua faria com que o segundo nó aumentasse de valor

indefinidamente, graças à ligação atrasada (indicada por uma coloração diferente).

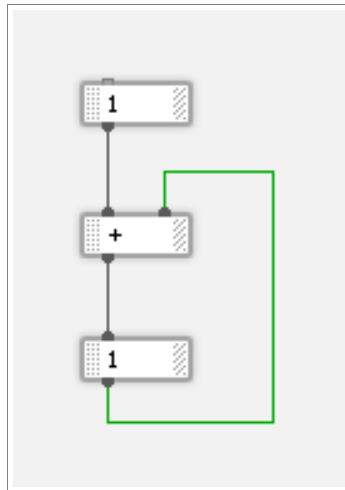


Figura 12 - Uma ligação de atraso criando um loop

Outra diferença de linguagens de fluxo de dados com relação a linguagens imperativas é a ausência de condicionais de execução. Numa linguagem de fluxo de dados, não existe ponto único de execução – para todos os efeitos, todos os nós são processados uma vez a cada ciclo. Assim, não há qualquer roteamento de processos possível. Há, no entanto, a possibilidade da criação de condicionais que permitem a troca entre diferentes tipos de dados, de acordo com operadores lógicos ou outros métodos de comparação.

Esses nós que permitem a verificação entre diferentes condições e a escolha de um determinado valor em muito se assemelham a condicionais *inline* usadas em outras linguagens de programação, ou até mesmo softwares de planilha de cálculo, como o *Microsoft Excel*. Como referência, em muitas linguagens, o diagrama

abaixo poderia ser representado como  $mensagem = 1 > 2 ? "Maior" : "Menor"$ .

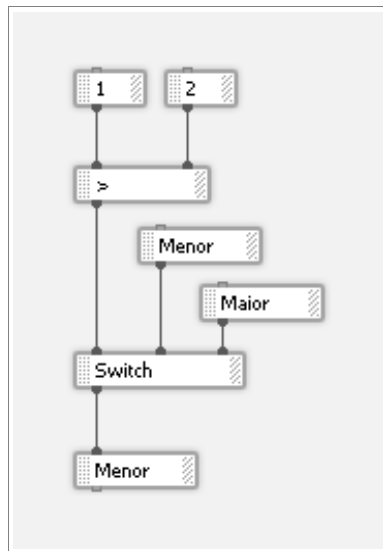


Figura 13 - Seleção de dados condicionais

Nestes nós condicionais, só um dos parâmetros de entrada é preservado e passado adiante; demais parâmetros são descartados. Isso não quer dizer que qualquer processo de transformação deixe de ser executado em dados que serão posteriormente descartados; todos os nós de um diagrama são processados, mesmo que sejam eventualmente inutilizados. Este é um desvio de outros métodos mais complexos que permitem que diferentes *blocos de código* sejam criados e ignorados conforme a necessidade (Culler, 1986); em contrapartida, o mecanismo de execução interno suprime a execução de nós que não tiveram seus parâmetros de entrada modificados, fazendo que os dados de saída gerados anteriormente sejam retransmitidos ao invés, evitando assim o consumo de recursos de processamento em atividades desnecessárias. Em outras palavras, o mecanismo de processamento da linguagem realiza um *armazenamento* dos dados

de saída, e continua utilizando-os até que seja necessário criar novos dados.

### 3.5 Interface gráfica de usuário

A interface gráfica de usuário do ambiente proposto oferece ao desenvolvedor um espaço onde novos diagramas podem ser criados através da disposição de nós e ligações. Devido à sua natureza gráfica, este processo de criação busca principalmente proporcionar ao desenvolvedor amplo espaço para edição.

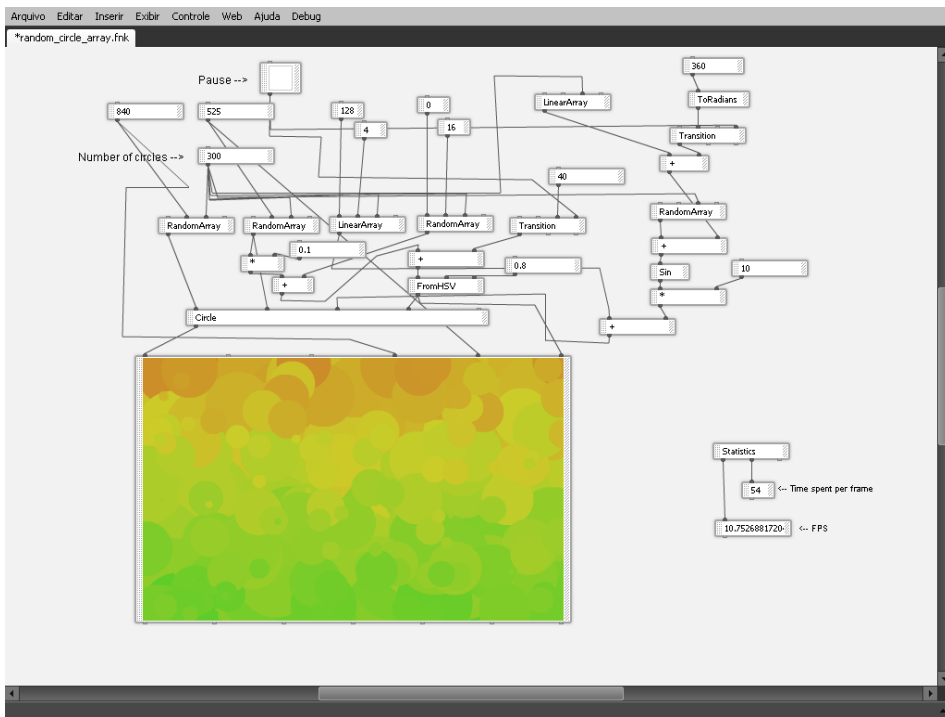


Figura 14 - Interface de edição principal do software

Assim, a interface busca ser bastante clara, com praticamente todo o espaço disponível sendo usado para a edição dos diagramas. A principal exceção é o topo da interface, usado para o menu do programa e a lista de abas contendo os

arquivos atualmente abertos.

Recursos inicialmente invisíveis da interface – acionados através de atalhos de teclado ou mouse – também proporcionam acesso mais rápido a determinados recursos do ambiente, como criação de novos nós e edição de suas propriedades.

Finalmente, por se tratar de um projeto sem âmbito local, a interface do editor está disponível tanto em Português do Brasil quanto em Inglês, dentro da mesma versão do programa. O editor usa como padrão inicial a linguagem do sistema do usuário, mas disponibiliza uma opção para mudança da linguagem no menu (ativada após reinicialização da ferramenta).

### **3.6 Ambiente de execução**

Parte da proposta considerada é que o ambiente Fnk deve ser acessado tão facilmente quanto quanto possível. Como solução a este problema, ele possui dois modos de execução diferentes.

A primeira – e principal – forma de utilizar o ambiente de desenvolvimento é *online*, através de um navegador da Internet comum, visitando o website de onde a ferramenta é executada; ou seja, sem a necessidade de instalação de quaisquer aplicativos adicionais ou arquivos incomuns no computador do usuário. Isso não só possibilita a qualquer micro acessar o ambiente a partir de qualquer lugar (desde que conectado à Internet), mas também permite que usuários com recursos de administração limitados em seus computadores – caso típico de computadores

usados dentro de um ambiente acadêmico – tenham acesso à ferramenta.

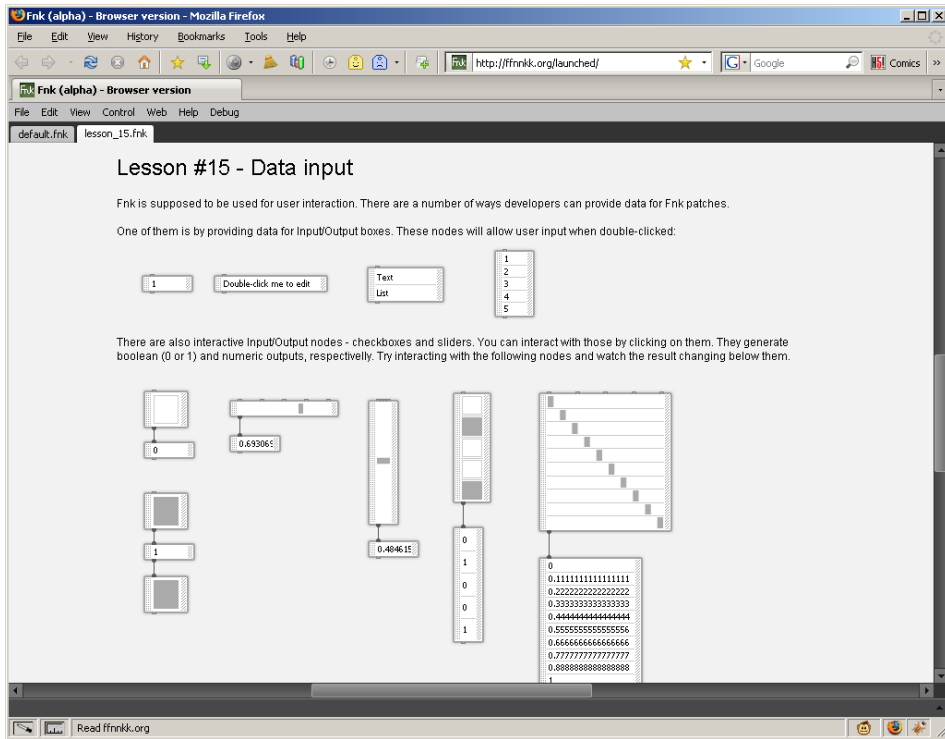


Figura 15 - Fnk sendo rodado a partir do navegador

Este modo de execução segue uma tendência de certos aplicativos transferirem para web devido exatamente a estas facilidades de execução. Além dos exemplos citados anteriormente – FontStruct e Picnik – podemos citar muitos outros aplicativos executados na web, em sua maioria aplicativos lançados recentemente: por exemplo, *Aviary* (pacote de ferramentas para edição de imagens), *Hobnox* (aplicativo para síntese de som através da combinação de sintetizadores virtuais), *Pixlr*, *SumoPaint* (editores de imagens profissionais), *Pikifx* (editor de fotos para adição de efeitos), *Sliderocket*, *Slideshare* (editores e agregadores de slideshows), *Noteflight* (criador de partituras), *GoogleDocs* (editor de documentos), e outros.

O segundo método de acesso ao ambiente é através de uma versão separada, *standalone*, criada para ser instalada e então executada numa máquina específica, como um aplicativo comum. Esta versão está disponível através de download, e embora tenha as mesmas características da versão online, ela tem como vantagem adicional a independência de uma conexão à Internet. Ela é considerada um modo alternativo de acesso à ferramenta, no entanto, uma vez que exige a instalação do software.

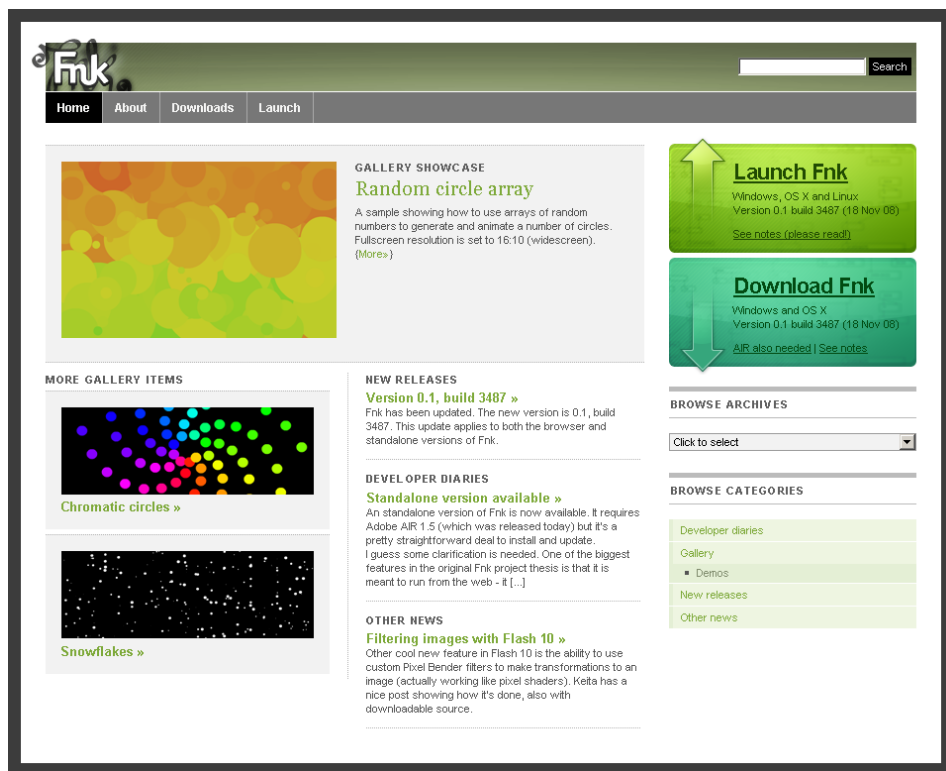


Figura 16 - Website da ferramenta

Ambas as versões estão disponíveis no website do projeto – [www.ffnkk.org](http://www.ffnkk.org) – que funciona também como diário de desenvolvimento da ferramenta.

### 3.7 Plataforma de desenvolvimento

Considerando-se o objetivo de permitir o acesso tanto através de uma versão *online*, executada a partir do navegador Internet, e uma versão *standalone*, foi escolhida a plataforma *Adobe Flash* para desenvolvimento da ferramenta. Além da familiaridade do autor com desenvolvimento para esta plataforma, esta escolha se justifica por algumas razões adicionais.

A primeira é permitir um desenvolvimento multiplataforma eficiente, sem a necessidade de recompilações ou adaptações para diferentes sistemas operacionais, uma vez que executáveis criados nesta plataforma são executados por uma máquina virtual própria. Assim, o projeto está disponível para sistemas Windows, Mac OS X e Linux, sem diferenças entre cada versão (além de certas questões de acessibilidade próprias, como atalhos de teclado). Da mesma forma, o uso de uma plataforma de virtualização separada evita que grandes incompatibilidades entre browsers – em especial mecanismos de renderização de conteúdo HTML e de execução JavaScript – interfira na experiência de edição.

Outra razão é que, embora seja dependente da instalação de um *plug-in* adicional em browsers, a penetração da plataforma Flash é hoje a mais alta entre tecnologias semelhantes. Para efeitos comparativos, sua penúltima versão – Flash Player 9 – está instalada de 90% dos computadores com acesso à web de acordo com estudos trimestrais realizados pela consultoria Millward Brown (Brown, 2008).

Já o projeto Fnk faz uso da versão mais recente – Flash Player 10 – lançada oficialmente em em 15 de Outubro de 2008; esta escolha foi motivada por recursos gráficos e sonoros importantes introduzidos nesta versão (Gilbertson, 2008). Por se tratar de um período de transição, esta versão ainda está pouco difundida, mas projeções baseadas na popularidade de versões anteriores indicam que ela deverá atingir acima de 90% de penetração em menos de um ano.

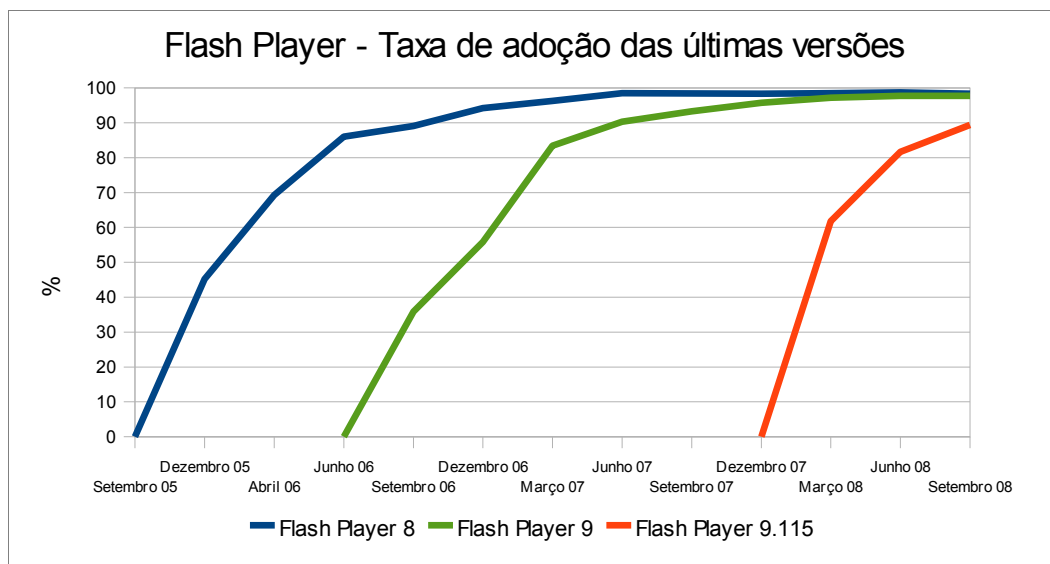
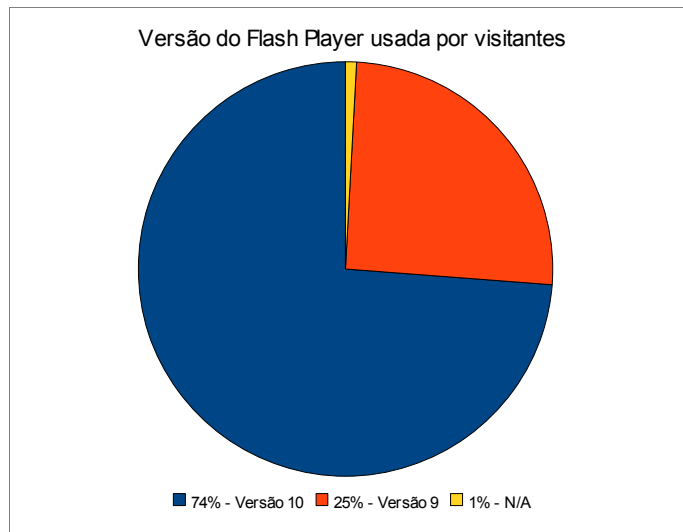


Figura 17 - Taxa de adoção do Flash Player

Adicionalmente, embora não seja indicativo suficiente do público-alvo da proposta como um todo, esta tendência de rápida penetração da tecnologia Flash é reforçada por testes preliminares realizados no website do projeto: ainda em fase de testes *alpha*, 74% dos visitantes do website do projeto durante o mês de novembro já possuíam a versão 10 do Flash Player instalado (Anexo A).



*Figura 18 - Versão do Flash usada por visitantes do website*

Além disso, para quem já possui alguma versão do Flash Player instalado, a atualização é feita automaticamente pelo website do aplicativo, caso o usuário permita. Como referência, no mês de novembro, cerca de 8% das visitas totais ao website optaram por fazer a atualização para o Flash 10 (Anexo B).

Vale citar como contraponto o fato de que a atualização do Flash Player geralmente não é possível dentro de um ambiente acadêmico: uma vez que atualizações deste tipo podem ser vistas como a instalação de software, o sistema operacional Windows não permite a atualização automática de plugins (ao contrário do sistema OS X). Neste caso, acreditamos que a tendência dentro de ambientes acadêmicos é que atualizações deste tipo sejam eventualmente realizadas pela própria administração assim que certas versões da tecnologia atingem um certo índice de popularidade.

Em comparação ao Flash, a tecnologia Microsoft SilverLight – concorrente mais próxima na área de tecnologias distribuídas para execução em navegadores no lado cliente – encontra-se num estágio bem inicial, sem penetração suficiente para se tornar uma saída viável neste momento; é difícil obter números precisos sobre esta tecnologia, mas a penetração do *player* SilverLight está entre 17.6% (Hanes, 2008) e 25% (Microsoft, 2008), independente da versão.

Uma alternativa que faria mais sentido do que SilverLight hoje seria o uso da tecnologia Java. Embora considerada mais rica em recursos e processamento do que a tecnologia Flash, esta alternativa também não foi levada em consideração por possuir problemas adicionais de incompatibilidade entre versões (Steinmeyer, 2005) e dificuldades de se obter informações precisas sobre a penetração de diferentes versões da tecnologia.

Finalmente, esta decisão sobre a tecnologia utilizada fica melhor fundamentada quando consideramos algumas das plataformas escolhidas por aplicativos online criadas recentemente, como os citados anteriormente – Aviary, Hobnox, FontStruct, Pixlr, Pikifx, Sliderocket, Picnik, SumoPaint e Noteflight são baseados em Flash, além de Onyx-VJ, SlideShowPro.net, Rsizr, Go!Animate e outros.

### **3.8 Documentação e ajuda**

Por se tratar de uma ferramenta que busca funcionar de maneira introdutória à produção de peças multimídia, parte dos objetivos do projeto é disponibilizar

informações iniciais sobre a linguagem e seu editor de forma simples. Isso se manifesta através de um menu que contém uma série de lições sobre a linguagem e sua interface na ferramenta de edição.

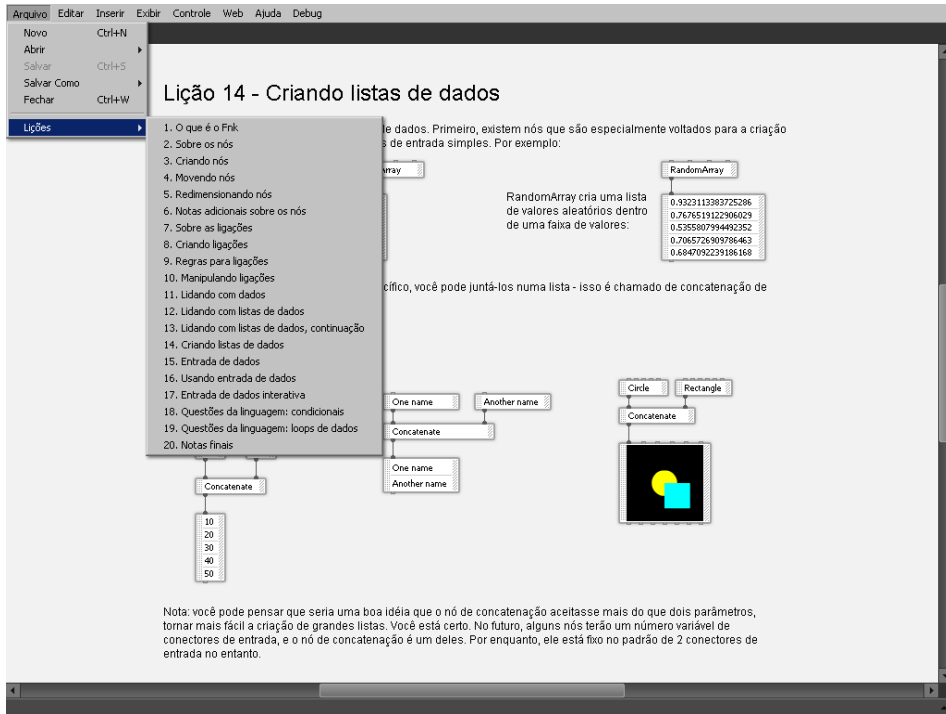


Figura 19 - Menu de lições, e uma lição aberta abaixo

Estas lições foram criadas como documentos do próprio Fnk. Isto permite ao usuário editar os nós apresentados diretamente, bem como copiar e colar exemplos que são disponibilizados no decorrer das lições no próprio corpo de texto.

Além destas lições iniciais, a linguagem também permite que desenvolvedores utilizem referências que estão atreladas a cada nó específico: cada nó possui um

documento próprio – que pode ser acessado através da tecla F1 ou do menu “Referência de Contexto” no menu “Ajuda” – e que contém informações adicionais sobre o uso de cada nó. Este recurso visa oferecer um modo mais focado de aprendizado ao usuário, uma vez que ele pode escolher tópicos para pesquisa a partir do uso dos nós em que ele tem mais interesse.

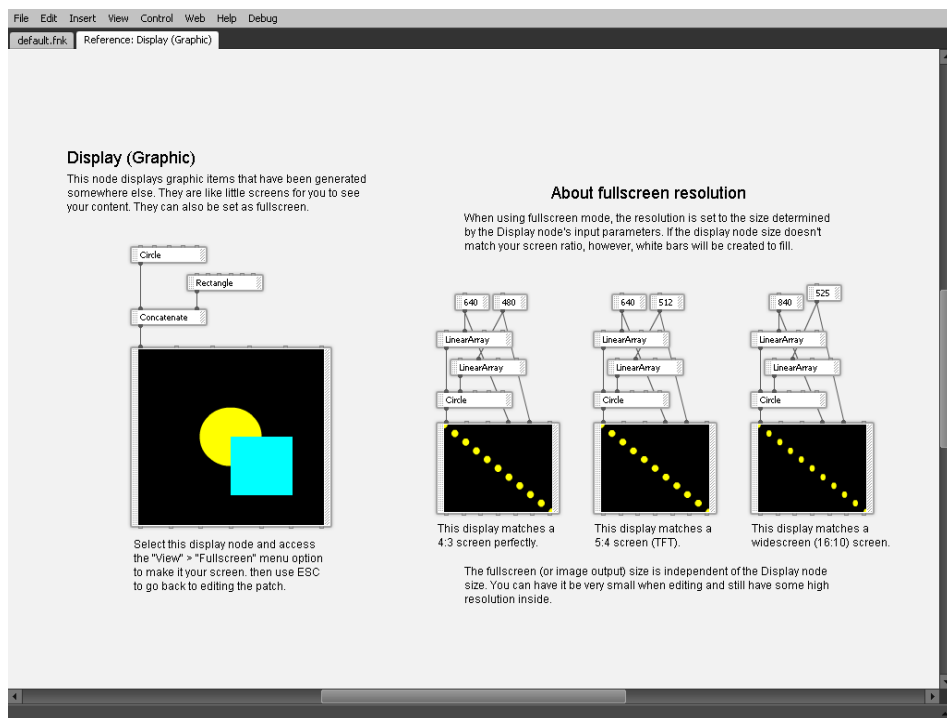


Figura 20 - Ajuda sobre o nó “Display”(em inglês)

Assim como as lições, os documentos de referência funcionam também como um documento criado na própria ferramenta, servindo tanto como explicação da funcionalidade de um nó quanto como ponto de partida para patches criados pelo usuário (muito embora nem todos os nós possuam esta referência atualmente).

## **3.9 Testes de usabilidade**

De modo a verificar a eficiência da interface de edição básica da ferramenta, uma rodada de testes de usabilidade foi realizada. Ela consistiu de um teste exploratório que buscava definir o modelo de comportamento de usuários diante de certas tarefas, e que passos deveriam ser tomados para solução de quaisquer problemas encontrados.

### **3.9.1 Usuários**

Com base nos usuários esperados da ferramenta, participantes de diferentes perfis foram utilizados. São eles:

- Usuário A: estudante universitário, trabalha com comunicação visual. Usuário de sistemas Macintosh. Utiliza com maior frequência softwares de edição de imagem como Illustrator, Photoshop e InDesign. Não possui conhecimentos de programação.
- Usuário B: estudante universitário, trabalha com motion graphics. Usuário tanto de sistemas Macintosh quanto Windows. Utiliza com maior frequência o software Maya. Possui certo conhecimento de programação, inclusive com Max/MSP.
- Usuário C: estudante universitário, trabalha com design gráfico para a web. Usuário de sistemas Macintosh. Utiliza com maior frequência o software Photoshop. Não possui conhecimentos de programação, mas conhece o

software Max/MSP.

- Usuário C: diretor de arte, trabalha com design gráfico para a web e produção sonora. Usuário de sistemas Macintosh. Utiliza com maior frequência o software Photoshop e ferramentas de edição de áudio. Não possui conhecimentos de programação.

### **3.9.2 Formato e objetivos do teste**

Este teste exploratório iniciava-se com o editor aberto, e um documento disponível para edição. Uma breve descrição da linguagem e seu propósito era dada, sem nenhuma grande introdução sobre a interface. Em seguida, uma lista de tarefas era passada ao participante, contendo tarefas relacionadas a:

- Movimentação ao redor da tela
- Seleção e remoção de nós
- Criação de novas ligações
- Abrir/Salvar arquivos

O roteiro completo deste teste encontra-se como apêndice deste trabalho.

### **3.9.3 Erros apontados e correções**

Embora tivesse uma natureza exploratória e fosse bastante breve, vários problemas foram encontrados na interface através desse teste.

Um problema em especial foi apontado por todos os participantes do teste: no projeto original, para a criação de novas ligações entre dois nós, usuários deveriam clicar num conector para iniciar a ligação, e em seguida, clicar num segundo conector para terminá-la. Todos os usuários, no entanto, esperavam que este processo fosse realizado através de clicar e arrastar (*drag'n drop*), ou seja, clicando num conector inicial, e então arrastando a ligação para um segundo conector. A solução, neste caso, foi modificar o processo de criação de ligações para permitir que novas ligações fossem criadas através de *drag* do mouse.

Outras observações apontam para características interessantes do uso de aplicativos online por parte dos participantes. Também em sua totalidade, os participantes evitaram utilizar atalhos de teclado, como atalhos para abrir arquivos, salvar arquivos ou mover a tela de edição – os participantes não se sentiam seguros para arriscar o uso de atalhos a menos que tivessem certeza de seu funcionamento. Ao mesmo tempo, algumas características normalmente só encontradas em aplicativos comuns – como seleção através de retângulos desenhados pelo mouse na área de edição – foram assimiladas bem rapidamente.

Finalmente, vale apontar este teste exploratório foi realizado durante o início do processo de desenvolvimento – somente alguns aspectos dos recursos básicos de edição foram medidos. Testes mais longos e que buscam medir a real eficiência do usuário seriam necessários para validação de algumas decisões de interface.

### 3.10 Exemplos avançados

Por se tratar de um ambiente de desenvolvimento consideravelmente aberto, foi necessário definir o escopo ao qual o projeto iria responder num estágio inicial. Assim, dentro da proposta de aplicação multimídia, vários exemplos de aplicações multimídia finais foram planejados e criados, de modo que uma grande gama de nós diferentes fossem necessários para manipulação de números, strings, áudio e imagem.

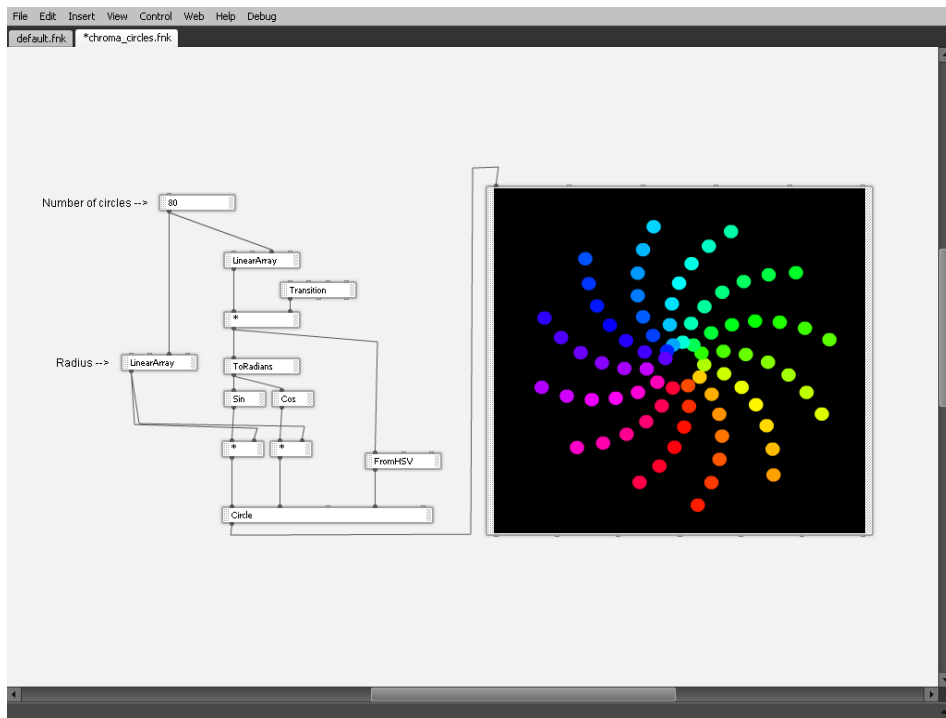


Figura 21 - Um exemplo sendo executado

Estes exemplos são auto-explicativos e estão disponíveis nos arquivos entregues junto deste trabalho.



## **4 CONSIDERAÇÕES FINAIS**

O desenvolvimento do projeto ocorreu sem muitas surpresas e consideramos o resultado satisfatório como estágio inicial tanto para uma plataforma de programação visual como para um editor rodando a partir da Internet. Entretanto, no decorrer de sua implementação, diversos recursos e idéias adicionais foram surgindo, fazendo com que o escopo do projeto crescesse de forma acentuada. Muitas destas idéias tiveram de ser descartadas, ou implementadas de forma breve, para que o tempo limitado de desenvolvimento fosse focado em necessidades mais urgentes da ferramenta. Aqui estão listados alguns destes recursos, bem como idéias que deverão ser levadas em consideração em eventuais evoluções futuras do projeto.

## 4.1 Manutenção de documentos online

O projeto inicial considerava realizar a edição de arquivos de patches diretamente a partir do website, isto é, sem a necessidade de acesso a arquivos no *lado cliente*, no computador do usuário. Arquivos seriam criados e ligados a uma conta de usuário, estando sempre acessíveis de qualquer lugar, desde que o usuário estivesse conectado à Internet e devidamente autorizado em sua conta. Este é o comportamento geralmente escolhido por aplicativos executados a partir de websites.

Este recurso foi posto de lado, no entanto, em prol de um modo único de gerenciamento de arquivos: acesso arquivos locais. Este modo de manutenção de arquivos permitiu que o desenvolvimento fosse mais rápido, uma vez que funcionalidades do *lado servidor* (banco de dados, mecanismos de autorização e criação de conta, etc) não tiveram de ser criados.

## 4.2 Mais nós

A lista de nós atuais pode ser definida como o vocabulário da linguagem utilizada pelo editor Fnk. Este vocabulário conta atualmente com cerca de 60 tipos de nó diferentes para a criação ou manipulação de diversos tipos de dados. Ainda assim, ele pode ser considerado como um vocabulário bastante inicial, uma vez que foi criado com exemplos muito específicos em mente; muitos mais nós se fariam necessário na tentativa de enriquecer as possibilidades da linguagem. Nesta lista

se encontram especialmente nós para interação com dispositivos de hardware, como *Arduino*, *Make*, *joysticks*, e outros.

### **4.3 Módulos de nós**

Talvez a maior falha na forma como a linguagem Fnk funciona atualmente é a ausência de recursos que possibilitem o agrupamento de diferentes nós. Em sua forma ideal, este recurso permitiria que *subpatches* fossem criados e reutilizados em separado, funcionando de forma semelhante a funções em outras linguagens. Desenvolvedores poderiam assim utilizar a própria linguagem para criar seus próprios nós, com parâmetros de entrada e saída específicos, e reutilizá-los conforme necessário.

### **4.4 Compartilhamento de documentos**

Outro recurso normalmente encontrado em aplicativos online é a possibilidade de se compartilhar arquivos, isto é, permitir que documentos criados por um usuário e atrelado à sua conta fossem abertos ou vistos por outras pessoas, através de um simples endereço Internet que levasse ao documento. Este recurso seria especialmente útil por facilitar a difusão de conhecimento, uma vez que autores poderiam compartilhar seu trabalho com inúmeras pessoas sem a necessidade de disponibilizar arquivos em separado para transferência. Este recurso depende diretamente do item anterior, e por isso, não foi considerado.

## 4.5 Execução e *embed* de patches

Ainda dentro da idéia de compartilhamento de arquivos, uma extensão natural seria permitir que o editor funcionasse também como um *player* em separado, isto é, permitisse que usuários *embutissem* documentos criados no Fnk em páginas HTML de sua própria autoria – assim como já é feito hoje, com o conteúdo de websites como YouTube, SlideShare, e muitos outros. Tal funcionalidade tornaria especialmente fácil a desenvolvedores demonstrarem patches criados no projeto, uma vez que nem mesmo a versão online do editor teria de ser aberta – o resultado da execução de um patch poderia ser diretamente inserida num artigo criado num blog, por exemplo.

## 4.6 Edição simultânea

Outro recurso que está diretamente ligado à edição de arquivos online e seu comportamento é a possibilidade de edição simultânea de um mesmo arquivo por diversos usuários. Dentro dessa proposta, encontrada hoje em alguns editores de arquivos online como *Google Docs*, *Buzzword* e outros, diferentes usuários poderiam editar um mesmo documento, de forma cooperativa.

## 4.7 Recursos da versão *standalone*

A versão instalável, atualmente, comporta-se meramente como uma versão paralela à versão online, com os mesmos recursos e limitações em relação à linguagem. Ela deveria, no entanto, utilizar alguns recursos que são exclusivos à

sua camada de execução, tal como a leitura e gravação irrestrita de arquivos locais (possibilitando a exportação de arquivos de animação quadro-a-quadro, por exemplo), bem como sendo melhor adaptada para cada sistema operacional diferente (utilizando menus nativos de cada plataforma, por exemplo).



# REFERÊNCIAS

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, NY, EUA, v. 3, n. 2, 2002. Disponível em: <[http://www.gse.buffalo.edu/RP/PDFs/ECE\\_Comp\\_Math.pdf](http://www.gse.buffalo.edu/RP/PDFs/ECE_Comp_Math.pdf)>. Acesso em: 17 abril 2008.

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.

CUKIERT, Michele; PRISZKULNIK, Léia. Considerações sobre o eu e o corpo em Lacan. **Estudos de Psicologia**, v. 7, n.1, p. 143-149, janeiro 2002. Disponível em: <<http://www.scielo.br/pdf/epsic/v7n1/10961.pdf>>. Acesso em: 13 setembro 2008.

CULLER, David E. **Dataflow Architectures**. Laboratory for Computer Science, M.I.T., fevereiro 1986. Disponível em: <<http://csg.csail.mit.edu/pubs/memos/Memo-261-1/Memo-261-2.pdf>>. Acesso em: 18 março 2008.

DENNIS, Jack B. FOSSEEN, John B. **Introduction to Data Flow Schemas**. Computation Structures Group, Project MAC, M.I.T., setembro 1973. <<http://csg.csail.mit.edu/pubs/memos/Memo-81/Memo-81-1.pdf>>. Acesso em: 30 março 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.

FAGONE, Jason. **The strange allure of making your own fonts**, junho 2008. Disponível em: <<http://www.slate.com/id/2192535>>. Acesso em: 3 novembro 2008.

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

GILBERTSON, Scott. **Flash Player 10: Dazzling Effects, Better Performance, Runs on Linux**, 14 Maio 2008. Disponível em: <<http://blog.wired.com/monkeybites/2008/05/flash-player-10.html>>. Acesso em: 15 maio 2008.

HANES, Michael. SilverLight Penetration, 14 outubro 2008. Disponível em: <<http://blog.mediawhole.com/2008/10/silverlight-penetration.html>>. Acesso em: 20 novembro 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 (CSUR)**, New York, NY, EUA, v. 36, n. 1, março 2004. Disponível em: <<http://www.ittc.ku.edu/~rsass/rcreading/johnston04.pdf>>. Acesso em: 25 fevereiro 2008.

KAY, Alan. **Squeak Etoys, Children & Learning**. Glendale, CA, EUA: Viewpoints Research Institute, 2005. Disponível em: <[http://www.squeakalpha.org/pdf/etoys\\_n\\_learning.pdf](http://www.squeakalpha.org/pdf/etoys_n_learning.pdf)>. Acesso em: 13 setembro 2008.

KELLEHER, Caitlin; PAUSCH, Randy. Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers. **ACM Computing Surveys (CSUR)**, New York, NY, EUA, v. 37, n. 2, junho 2005. Disponível em: <<http://reports-archive.adm.cs.cmu.edu/anon/anon/2003/CMU-CS-03-137.pdf>>. Acesso em: 13 setembro 2008.

MCKEE, Jake. **10 Questions with Jonathan and Peter of Picnik.com**, Outubro 2007. Disponível em: <<http://www.communityguy.com/1090/10-questions-with-jonathan-and-peter-of-picnikcom>>. Acesso em: 4 novembro 2008.

Microsoft: Website. **Microsoft Releases Silverlight 2, Already Reaching One in Four Consumers Worldwide**, 13 outubro 2008. Disponível em: <<http://www.microsoft.com/presspass/press/2008/oct08/10-13Silverlight2PR.msp>>. Acesso em: 20 novembro 2008.

BROWN, Millward. **Adobe Flash Player Version Penetration**, setembro 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: 20 novembro 2008.

MULLER, Jim. **The great Logo Adventure - Discovering Logo On and Off the computer**. Madison, AI, EUA: Doone Publications, 1997. Disponível em: <<http://www.softronix.com/logo.html>>. Acesso em: 17 abril 2008.

PAPERT, Seymour. **Mindstorms: children, computers, and powerful ideas**. New York, NY, EUA: Basic Books, Inc., 1980.

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

PUCKETTE, Miller. MAX at Seventeen. **Computer Music Journal**, Cambridge, MA, EUA: The MIT Press, v. 26, n. 4, p. 31-43, junho 2005. Disponível em:

<<http://crca.ucsd.edu/~msp/Publications/dartmouth-reprint.pdf>>. Acesso em: 13 setembro 2008.

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

RESNICK, Mitch. **StarLogo: An Environment for Decentralized Modeling and Decentralized Thinking**. New York, EUA: ACM - Association for Computing Machinery, 1996. Disponível em: <[http://www.sigchi.org/chi96/proceedings/demos/Resnick/mjr\\_txt.htm](http://www.sigchi.org/chi96/proceedings/demos/Resnick/mjr_txt.htm)>. Acesso em: 13 setembro 2008.

RUBIN, Jeffrey. **Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests**. New York, EUA: John Wiley & Sons, 1994.

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

STENMEYER, 2005. **Development Platforms for Casual Games**, Gamasutra, 24 março 2005.

Disponível em:

<[http://www.gamasutra.com/view/feature/2266/development\\_platforms\\_for\\_casual\\_.php?page=1](http://www.gamasutra.com/view/feature/2266/development_platforms_for_casual_.php?page=1)>. Acesso em: 20 outubro 2008.

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

<<http://csg.csail.mit.edu/pubs/memos/Memo-110/Memo-110.pdf>>. Acesso em: 18 março 2008.

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

MAX/MSP/JITTER: Website. Disponível em:

<<http://www.cycling74.com/products/mmjoverview>>. Acesso em: 20 setembro 2008.

## **APÊNDICE A – Lições internas**

Para servir como introdução ao ambiente e à linguagem, o Fnk possui uma série de lições que estão disponíveis a partir do menu do programa. As lições cobrem desde as ferramentas de edição do programa até conceitos utilizados pela linguagem, em comparação com linguagens imperativas comuns ou outros ambientes semelhantes. Versões destas lições capturadas diretamente do programa estão disponíveis a seguir.


## Lição 1 - O que é o Fnk


Bem-vindo(a) ao Fnk!

Fnk é um ambiente de programação visual no qual você pode manipular números, texto, imagens e outros tipos de dados para criar resultados multimídia baseados em matemática. Ele é muito similar em funcionamento a programas como Max/MSP/Jitter, vvvv, Pure Data, e outros. Na verdade, Fnk traz muita inspiração destes programas, mas busca funcionar num ambiente Web.

Fnk usa uma estratégia de fluxo de dados para programação. Isso quer dizer que no Fnk, você está lidando com dados e sua constante transformação, ao contrário de linguagens imperativas comuns no qual você lida com o fluxo de execução e eventos.

Como resultado disso, no Fnk, você nunca está programando com comandos, mas sim com nós que geram, transformam e apresentam dados.

Por exemplo, aqui está um nó que representa um número: 

E aqui está um nó que representa uma string de texto: 

Finalmente, nós podem ter uma ligação entre si, como estes:



Quando nós estão ligados, eles transmitem informações adiante. No modelo de execução do Fnk, dados são criados e passados adiante conforme são transformados. É por isso que é uma linguagem de "fluxo de dados".

Uma coisa que é importante notar é que no Fnk, não há distinção entre tempo de edição/desenvolvimento, e tempo de execução. Não há compilação envolvida, então programas do Fnk (chamados de Patches) estão em constante execução, mesmo enquanto você ainda os está construindo.

Leia as próximas lições para descobrir mais sobre o Fnk.

## Lição 2 - Sobre os nós

Cada caixa que você vê no Fnk é um nó (em inglês, "node"). Nós são a fundação do Fnk, e podem ser interpretados como sendo um comando da linguagem do Fnk.

Existem diversos tipos de nós diferentes disponíveis no Fnk. Eles podem basicamente ser:

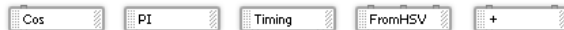
- Caixas de entrada/saída ("Input/Output boxes"): são nós usados para mostrar dados, ou permitir a edição dos mesmos (como um número ou algum tipo de texto). Você pode editar o conteúdo destes nós com um duplo-clique. Por exemplo:



- Nós de entrada/saída interativos ("Interactive I/O nodes"): permitem que o conteúdo seja editado através de interação com o nó, como uma caixa de seleção ("checkbox") ou uma barra de posicionamento ("slider"). Por exemplo, tente interagir com estes nós, clicando dentro deles:



- Nós normais: usados para criar transformações de dados, usando um ou mais (ou até mesmo nenhum!) parâmetro de entrada para criar novos dados. Aqui estão alguns nós diferentes:



- Nós de exibição ("Display nodes"): usados para a exibição de conteúdo gráfico; eles são como pequenas telas, no qual você pode ver o resultado da execução de seu diagrama. Por exemplo, aqui está um vazio:



## Lição 3 - Criando nós

Para criar um novo nó, você deve:

- Arrastar um nó da lista de nós - o painel à esquerda da área de edição; ou,
- Dar um duplo-clique em qualquer lugar da área de edição e selecionar um nó da lista.

Você perceberá que cada nó tem um nome específico, e uma categoria que ele representa. Existem, por exemplo, nós relacionados a manipulação de números (Number) ou texto (String). Outras categorias podem não estar diretamente relacionadas a um tipo específico de dado, no entanto.

Tente agora! Dê um duplo-clique na área abaixo para abrir a lista de novo nó. Então selecione algum nó da lista, ou comece a digitar o nome de um nó específico para selecionar mais rápido.

\* Nota: para manter a unidade da linguagem, os nós possuem seus nomes em inglês, independente da linguagem selecionada para a interface do programa.

## Lição 4 - Movendo nós

Manipular nós é fácil. Eles são como pequenas caixas gráficas que você pode mover ou redimensionar.

Na maioria dos nós, você pode clicar em qualquer parte do nó com seu mouse para selecioná-lo, depois arrastá-lo com o mouse para movê-lo. Você perceberá que o cursor do mouse se transforma numa seta cruzada para indicar que você pode movê-lo. Tente agora:



Já alguns nós requerem que você clique numa área específica para movê-los. É o caso de nós de exibição (Display) - como a grande área negra é reservada para interação, você deve clicar a área de seleção para movê-lo ao invés. A área de seleção é a barra à ESQUERDA do nó. Tente mover este nó:

Clique nesta área para selecionar ou mover o nó ----->



Outros nós não são interativos, mas também não deixarão você selecioná-los ao menos que você clique na mesma área designada para seleção - a área normal é selecionada para interação com o usuário. É o caso de checkboxes e sliders:

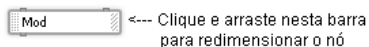


Finalmente, você perceberá que quando um nó está selecionado, sua cor muda para indicar a seleção. Então, quando estiver selecionado, você pode movê-lo usando as setas de seu teclado. Cada clique move os nós selecionados um pixel numa direção específica. Apertar SHIFT e então pressionar a seta move em incrementos de 10 pixels.

## Lição 5 - Redimensionando nós

Você pode também redimensionar nós. Isto ajuda a visualizar seu conteúdo.

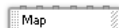
Para redimensionar nós, você pode simplesmente clicar na barra do lado direito de cada nó e arrastar. O cursor do mouse se torna uma seta diagonal e você pode arrastá-la para mudar o tamanho de cada nó. Cada nó tem um tamanho mínimo e máximo. Tente redimensionar este nó:



Você pode também redimensionar vários nós ao mesmo tempo. Tente selecionar todos os módulos abaixo - pressionando SHIFT e clicando em todos eles, ou desenhando um retângulo de seleção ao redor deles - e então redimensionar qualquer um dos nós. Você perceberá que todos os nós redimensionam-se junto, independente de seu tamanho original.



Você pode também redimensionar com seu teclado. Assim que você tiver um nó selecionado, pressione CTRL (Windows e Linux) ou COMMAND (Macintosh) e use as setas do teclado para redimensionar o nó em incrementos de um pixel. Pressionar a tecla SHIFT antes permite a você redimensionar por incrementos de 10 pixels. Tente agora: clique no nó abaixo para selecioná-lo, e use CTRL+setas ou COMMAND+setas para redimensioná-lo.



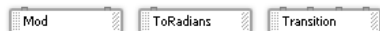
## Lição 6 - Notas adicionais sobre os nós

Outras regras clássicas para edição de gráficos se aplicam.

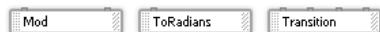
Para apagar um nó, selecione-o e pressione DEL/DELETE ou BACKSPACE.



Você pode copiar nós para a área de transferência (clipboard) selecionando-os e então pressionando CONTROL+C (Windows e Linux) ou COMMAND+C (Macintosh), e então colá-los com CONTROL+V ou COMMAND+V. Perceba que colar move os novos objetos para o centro da área de edição; para colá-los na mesma posição, pressione CONTROL+SHIFT+V ou COMMAND+SHIFT+V ao invés.



Você pode também apagar nós ao mesmo tempo que os copia para a área de transferência - para fazê-lo, selecione os nós e então pressione CONTROL+X (Windows e Linux) ou COMMAND+X (Macintosh).



Finalmente, para selecionar todos os nós, você pode usar o atalho CONTROL+A ou COMMAND+A.

Perceba que essas opções estão todas disponíveis no menu "Editar".

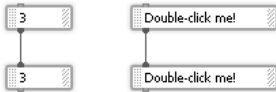
## Lição 7 - Sobre as ligações

Além dos nós, ligações (ou "links", em inglês) são também uma parte importante do ambiente Fnk. Ligações levam dados de um nó para outro. Veja este diagrama simples, por exemplo:



Estes são dois checkboxes ligados um ao outro. Quando o primeiro checkbox é mudado, seu valor é transmitido pela ligação, e chega ao segundo checkbox, mudando-o. Tente clicar dentro do primeiro checkbox e veja o segundo checkbox mudar de acordo.

Ligações transmitem todo tipo de dados. Os nós abaixo, por exemplo, lidam com números e texto. Tente executar um duplo clique em cada um deles para mudar seu valor: você vai perceber que o nó seguinte também muda seu valor de acordo.



Você perceberá que ligações sempre SAEM de e ENTRAM numa posição especial em cada nó. Estas posições são chamadas de Conectores (ou "Connectors" em inglês), e cada conector serve um propósito especial - eles são como os parâmetros de funções (no caso de conectores de entrada) ou seus valores de retorno (no caso de conectores de saída).

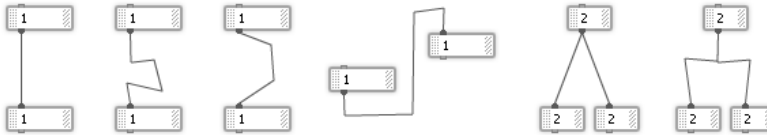
## Lição 8 - Criando ligações

Criar ligações é fácil: você simplesmente precisa clicar num conector, arrastar o mouse para outro conector, e então soltar o botão do mouse. Ou, você clicar num conector e soltar o botão do mouse, e então clicar num segundo conector para estabelecer a conexão. Tente agora: crie uma ligação entre os dois nós seguintes.



Perceba que a ordem de criação da ligação não é importante: ligar um conector de saída a um conector de entrada é o mesmo que fazê-lo de forma inversa, já que os dados sempre são enviados de um conector de saída de um nó para um conector de entrada em outro nó. É por isso que você não pode criar ligações entre dois conectores de entrada ou dois conectores de saída.

Ao criar ligações, você pode soltar e clicar o mouse na área de edição \*enquanto\* estiver desenhando uma ligação. Fazer isso cria um novo vértice para a ligação, permitindo a você criar ligações que não são linhas diretas. O caminho que cada ligação percorre não faz qualquer diferença na data que é transmitida - é um recurso meramente visual. Aqui estão algumas ligações de exemplo:



Em outras palavras, ao criar uma ligação, cada clique adicional na área de edição cria um novo vértice. Clicar BACKSPACE ou DELETE remove o último vértice criado (como a ferramenta de seleção poligonal do Photoshop).

Tente criar uma ligação entre os nós seguintes:



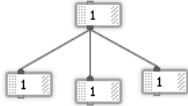
## Lição 9 - Regras para ligações

Ligações podem somente ser criadas entre dois conectores que carregam o mesmo tipo de dados. Você não pode ligar um conector com texto a um conector com números, por exemplo. É por isso que, quando você clica num conector para começar uma ligação, outros conectores com o mesmo tipo de dado são destacados; dessa forma, você pode saber que conectores aceitarão aquela ligação. Por exemplo, tentar criar uma ligação usando os nós seguintes não deixará criar ligações inválidas, de um nó de entrada de strings a um nó de entrada de números.

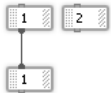


Lembre-se de que passar o mouse sobre um conector mostra que tipo de dado aquele conector manipula.

A maioria dos conectores também permite que você tenha múltiplas saídas (mas não múltiplas entradas!). Isto permite que os dados sejam duplicados e retransmitidos a diferentes nós, mudando todos os nós de acordo. Por exemplo:

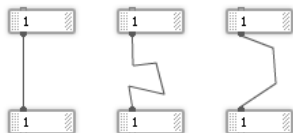


Tentar criar uma nova ligação num conector de entrada que já está ligado a outro nó remove a ligação existente. Para ver isto acontecer, tente ligar o nó com o número "2" abaixo ao nó da base: a ligação anterior é apagada e substituída pela nova.



## Lição 10 - Manipulando ligações

Ligações podem também ser selecionadas e manipuladas. Para selecionar uma ligação, clique na linha, ou desenhe uma seleção (clcando na área de edição) até que ela toque as linhas da ligação. Tente selecionar as ligações abaixo:



Pressionar DELETE ou BACKSPACE apaga as ligações selecionados.

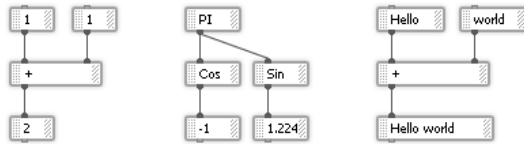
Outras regras para ligações - levadas em consideração ao copiar objetos para área de transferência, ou ao mover objetos dentro da área de edição - também se aplicam:

- Ao selecionar dois nós que estão ligados, qualquer ligação entre eles é automaticamente selecionada também.
- Você pode selecionar uma ligação e usar as setas do teclado para movê-la. No entanto, ligações só são copiadas para a área de transferência se ambos os nós o qual ela conecta forem copiados também.

Ligações não podem existir se elas não estiverem ligando dois nós. Se você apagar um nó, todas as ligações conectadas a ele são automaticamente apagadas também.

## Lição 11 - Lidando com dados

Programar no Fnk é feito através da constante transformação de dados. Isso se faz através do uso de nós que utilizam um ou mais parâmetros de entrada e que oferecem um número de conectores de saída. Por exemplo, aqui estão alguns nós de transformação simples:



Cada um destes parâmetros pode também carregar um tipo diferente de dado. Neste moment, os tipos de dados usados pelo Fnk são os seguinte:

- "Strings": todo tipo de texto
- "Number": todo tipo de número
- "Color": uma cor opaca, como uma cor que pode ser descrita usando-se o modelo RGB
- "Graphic": uma composição de gráficos vetoriais e imagens que descrevem um gráfico que pode ser exibido
- "Image": dados de cada pixel que compõe uma imagem

Adicionalmente, o Fnk usa números para descrever dados verdadeiro/falso (Booleanos) - 0 significa FALSO, 1 significa VERDADE. No caso de uma conversão, qualquer coisa abaixo de 1 é considerado falso; 1 ou mais alto é considerado verdade.

## Lição 12 - Lidando com listas de dados

Além de dados únicos, a maioria dos nós no Fnk também lida com listas ("Arrays") de dados. Listas são um conjunto de dados: assim, ao invés de transmitir um dado único, ligações podem carregar uma lista inteira deles.

Como referência, aqui está um nó de entrada/saída contendo um dado único:

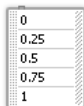


Mas aqui está o mesmo nó, carregando dois dados distintos ao invés. Ele mostra os dados na forma de uma lista editável:



Ou ainda mais, ou tipos diferentes de dados:

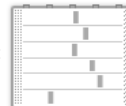
Uma lista de números:



Uma lista de checkboxes:



Uma lista de sliders:



Se você passar o mouse sobre um conector que contém uma lista de dados, ele mostrará o número de itens que estão lá contidos. Nós podemos então levar esses dados em consideração.

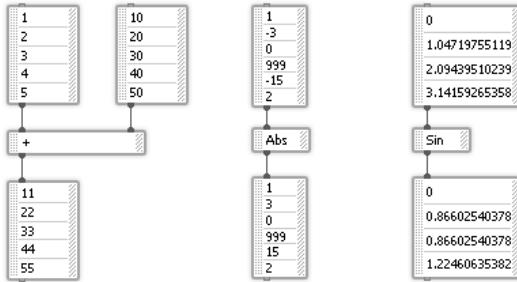
O mesmo se aplica a todos tipos de dados: ligações podem transmitir listas de gráficos, imagens, strings, etc.

\* Nota: usuários do programa WWW ([www.www.org](http://www.www.org)) reconhecerão listas como sendo análogas a "Spreads" no WWW. Ambos possuem a mesma idéia básica.

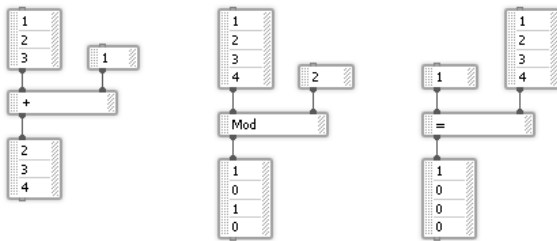
## Lição 13 - Lidando com listas de dados, continuação

No Fnk, quase todos os nós aceitam listas de dados como entrada também. Quando você usa listas de dados como parâmetros de entrada, isso faz com que o nó de transformação leve em consideração múltiplos dados de entrada e com que ele crie uma lista de dados como saída também, ao processar os dados de cada item separadamente. Isso torna listas um recurso poderoso de desenvolvimento no Fnk, já que permite ao desenvolvedor lidar com um grande número de dados de forma fácil.

Por exemplo, abaixo estão demonstradas algumas transformações aplicadas a listas de valores numéricos. Repare como cada resultado se relaciona com seus respectivos valores de entrada:



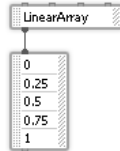
Nós geralmente levamos todos os parâmetros de entrada em consideração e então criamos uma lista de dados de saída que possui o mesmo número de itens que a maior das listas de entrada disponíveis. Quando o número de itens de entrada das listas não coincide, o nó circula nas listas de entradas sequencialmente. Abaixo estão alguns exemplos deste comportamento:



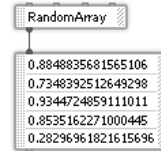
## Lição 14 - Criando listas de dados

Existem várias maneiras de se criar listas de dados. Primeiro, existem nós que são especialmente voltados para a criação de listas de dados baseados em parâmetros de entrada simples. Por exemplo:

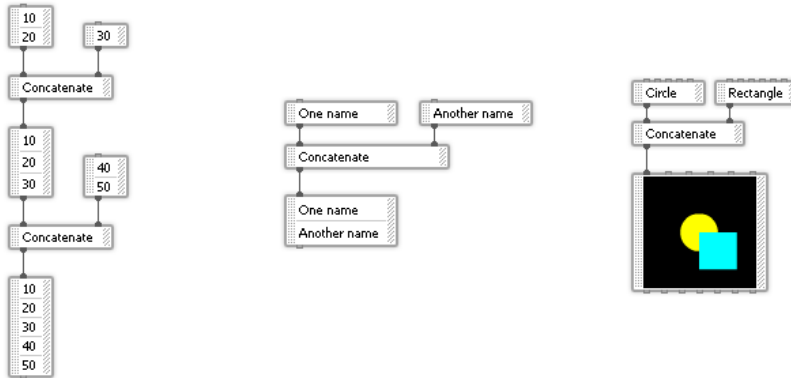
LinearArray cria uma lista de valores distribuídos igualmente por uma faixa de valores:



RandomArray cria uma lista de valores aleatórios dentro de uma faixa de valores:



E se você já tiver dados de um tipo específico, você pode juntá-los numa lista - isso é chamado de concatenação de dados. Aqui estão alguns exemplos:



Nota: você pode pensar que seria uma boa idéia que o nó de concatenação aceitasse mais do que dois parâmetros, tornar mais fácil a criação de grandes listas. Você está certo. No futuro, alguns nós terão um número variável de conectores de entrada, e o nó de concatenação é um deles. Por enquanto, ele está fixo no padrão de 2 conectores de entrada no entanto.

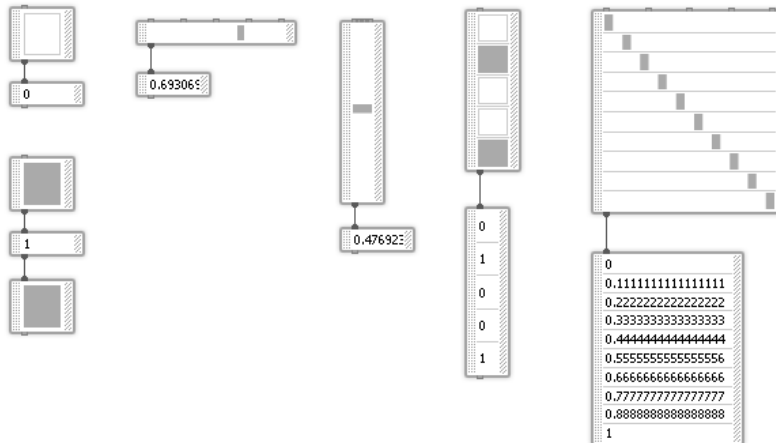
## Lição 15 - Entrada de dados

Um dos objetivos do Fnk é ser usado para interação. Existem algumas formas diferentes que desenvolvedores podem usar para criar dados para patches criados no programa.

Uma delas é através de nós de entrada e saída. Estes nós permitem a entrada de dados quando um duplo-clique é realizado em cada um deles:



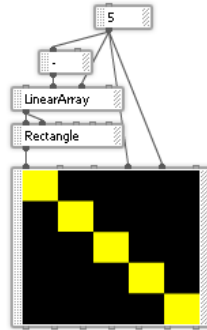
Existem também nós de entrada e saída interativos - checkboxes e sliders. Você pode interagir com estes nós ao clicar neles. Eles geram saídas Booleanas (0 ou 1) e numéricas, respectivamente. Tente interagir com os nós abaixo e veja o resultado nos nós seguintes.



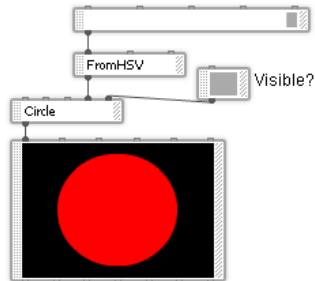
## Lição 16 - Usando entrada de dados

Entrada de dados pelo usuário é geralmente empregada para fornecer parâmetros baseados em texto ou números para a execução de um patch. Dentro do Fnk, é comum ter um número destes nós distribuídos de forma a ajudar a verificação dos valores que estão passando por algum conector, ou para permitir a modificação de parte do código visual. Por exemplo, mudar os parâmetros disponíveis abaixo muda a forma como cada diagrama se comporta:

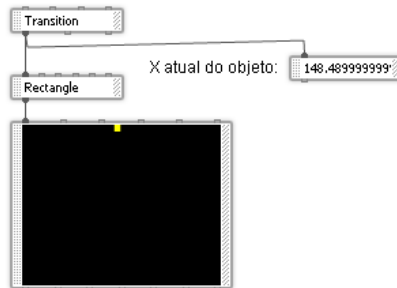
Mude o número de retângulos:



Arraste o slider para mudar a cor do círculo:

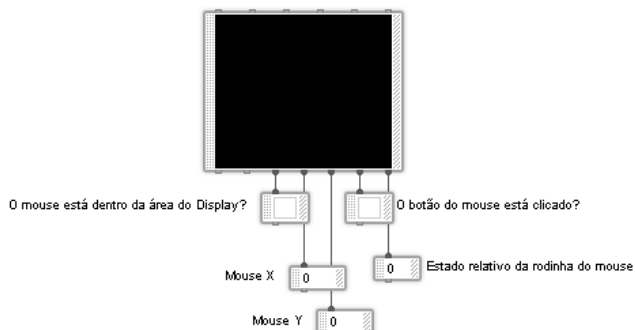


Ou, você pode usar estes nós de entrada e saída para checar os valores atuais de algo. Por exemplo, um nó é usado abaixo para mostrar a posição atual de um objeto.

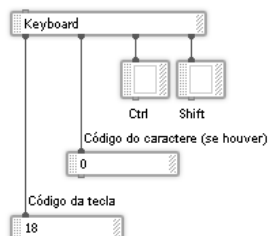


## Lição 17 - Entrada de dados interativa

Além de entrada de dados para o desenvolvedor, o Fnk também permite alguns modos interativos de manipulação de dados. Por exemplo, todos os nós de Display podem receber informações sobre o estado do mouse, e agir de acordo - inclusive quando está sendo usado em tela cheia. Tente over seu mouse sobre o nó abaixo e veja como os nós de dados ligados a ele reagem:



O Fnk também permite entrada de dados pelo teclado. Isto é feito através de uma leitura constante das teclas listadas. Tente pressionar uma tecla agora, e veja os nós seguintes mudarem - inclusive enquanto você está editando um patch.



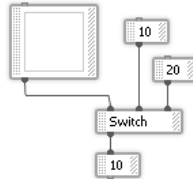
\* Nota: devido a limitações de segurança da plataforma Flash, nem todas as teclas funcionam no modo tela cheia. As únicas teclas suportadas em tela cheia são TAB (9), SPACE (32), e as setas (37, 38, 39 e 40). A versão standalone (instalável) permitirá todas as teclas no futuro, no entanto.

## Lição 18 - Questões da linguagem: condicionais

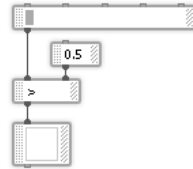
Por ser baseada num modelo de fluxo de dados, alguns recursos que podem soar óbvios em linguagens imperativas comuns não são muito práticos no Fnk.

Um destes recursos é o uso de condicionais. Enquanto em muitas linguagens você possa ter um bloco IF..ELSE (ou similares) decidindo o que deve ser executado e quando, no Fnk você nunca está lidando com o fluxo de execução, mas sim com o fluxo de dados. Isto quer dizer que TODO o código é executado, sempre; não existem condicionais de fluxo de código.

O que você pode usar, no entanto, são condicionais de dados que permitem a você escolher o dado que você quer - isto é feito através de nós de Switch ("chave"). Eles funcionam de maneira similar a condicionais "inline" que você encontra em muitas outras linguagens. Por exemplo, ligar/desligar o checkbox abaixo escolhe entre dois valores diferentes:



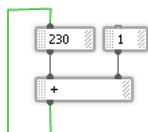
E operações condicionais são feitas por nós de comparação que retornam um valor Booleano (0 ou 1) que pode então ser usado por um nó de Switch. Por exemplo, arraste o slider abaixo - o checkbox liga quando o slider passa da metade.



## Lição 19 - Questões da linguagem: loops de dados

Outro recurso comumente encontrado em linguagens imperativas mas sem uma equivalência direta em linguagens de fluxo de dados são loops - como os que você cria com blocos FOR, WHILE, DO, etc. Isto é porque, no Fnk, cada nó é processado somente uma vez por cada ciclo de execução - você não pode ter o fluxo de execução preso e constantemente executando um mesmo grupo de nós. De forma similar, você não pode criar ligações que enviam dados de volta a nós que já foram executados - patches no Fnk têm de ser um diagrama acíclico, isto é, o patch precisa de inícios e fins bem definidos.

No entanto, no Fnk, você pode instruir ligações específicas para serem "atrasadas". Isto faz com que elas mantenham seus dados até o próximo ciclo de execução, e funcionem como um tipo de chave para loops. Este é um exemplo deste comportamento:



Execução constante do diagrama faz com que o nó aumente seu próprio valor, uma unidade por frame. Este tipo de ligação normalmente não seria permitida; no entanto, como a ligação que envia dos dados de volta ao começo do diagrama é uma ligação "atrasada", ela é válida. Perceba a cor verde da ligação - isso a identifica como uma ligação atrasada.

É importante, como referência, citar que outros ambientes de programação semelhantes têm diferentes soluções a este problema do loop. O Max, por exemplo, deixa desenvolvedores criarem qualquer tipo de ligação, mas a execução sempre segue a mesma ordem - direita para esquerda e cima para baixo. Isto quer dizer que a transformação dos dados depende da posição do nó - na prática, é como se ele possuísse ligações atrasadas, mas elas nunca são visíveis a menos que o desenvolvedor tenha conhecimento das regras de fluxo de dados no Max.

Já no WWV, a execução de nós segue a ordem das ligações. Esta é uma maneira mais previsível de lidar com transformações de dados, e é também a maneira com que o Fnk se comporta. No WWV, no entanto, loops não são permitidos automaticamente - você tem de passar seus dados através de um nó de atraso ("FrameDelay"), que então lida com o início e final de execução de cada loop de transformação de dados. E embora a solução usada no Fnk funcione de forma semelhante, ele o faz através de ligações atrasadas, ao invés de nós atrasados.

Você pode também definir uma ligação normal como sendo uma ligação atrasada - simplesmente use a opção "Editar > Ligação > Atrasada" no menu. Isto é útil se você quiser definir as áreas em que você quer que seu atraso ocorra. Você pode usar a mesma opção para remover o atraso de uma ligação, mas somente se isso for permitido - isto é, se não houver nenhum loop que dependa da ligação atrasada.

## Lição 20 - Notas finais

As lições básicas sobre o Fnk terminam aqui. Esperamos que elas tenham servido como uma boa introdução à linguagem e ao ambiente de programação.

Mais importante, se você quiser aprender mais sobre o Fnk, existem maneiras ainda melhores. Todos os nós devem possuir uma referência própria, que ensinam a você como usá-los através de exemplos; para acessar essa referência, simplesmente selecione um nó e clique F1 (ou use a opção "Ajuda > Referência Contextual" no menu).

Por exemplo, selecione qualquer um dos nós seguintes e aperte F1 para abrir um patch de referência sobre cada um deles:



Isto permite a qualquer um aprender no seu próprio ritmo e sobre os tópicos que quiserem - em geral, acessar a referência de algum nó no qual você tem interesse levará a diversos nós e exemplos relacionados, bem como informações mais aprofundadas sobre cada recurso da linguagem Fnk.

Adicionalmente, outros exemplos mais práticos devem estar disponíveis em breve para referência.



## **APÊNDICE B – Roteiro de testes iniciais**

Este é o roteiro que deverá ser seguido pelo *facilitador* durante testes de usabilidade do software. Ele contém indicações gerais de como o teste será realizado, bem como informações que deverão ser fornecidas aos participantes (na forma de frases em destaque e entre aspas).

### **Introdução e agradecimentos**

*“Obrigado por concordar em participar desta avaliação do software.”*

### **Objetivo, formato, e o papel do participante**

*“Você ajudará a completar uma série de tarefas dentro de cenário específicos; nosso objetivo é descobrir o quão fácil ou difícil você achará cada uma dessas tarefas. Você também será perguntado sobre sua experiência ao final desta seção.”*

### **O papel do facilitador**

*“Estou aqui para gravar suas reações e comentários no programa que você verá. Não poderei oferecer nenhuma sugestão ou dica; faça de conta que não estou aqui. Existirão momentos, no entanto, em que pedirei para você explicar porque você disse ou fez algo.”*

### **Pontos a manter em mente**

- “Este teste não é sobre você; estamos testando o software. Então, não se

preocupe em cometer enganos. Eles são esperados.”

- “Não existe resposta certa ou errada. Só queremos saber se o software funciona bem para você.”
- “Se você sentir que não pode completar uma tarefa com a informação que você recebeu, por favor nos avise. Eu perguntarei o que você faria numa situação no mundo real e depois irei ajudá-lo ou movê-lo para o próximo cenário.”
- “Estaremos gravando esta seção para estudos futuros se necessário. Seu nome não será associado aos dados ou resultados desta avaliação.”
- “Seria positivo se você falasse sobre o que está pensando ou fazendo, pensando em voz alta. Demonstrações de confusão, frustração, ou alegria são positivas.” (Demonstrar antes)
- “Se você cometer algum engano, continue.”
- “Ao usar o software, faça-o como você faria em sua casa ou escritório, da forma mais relaxada possível.”
- “Você tem alguma questão antes de iniciarmos?”

O facilitador não deverá demonstrar surpresa ou emoção. Não faça perguntas diretas sempre. Mantenha o foco no que os participantes esperam que aconteça.

### **Ficha do participante**

Após o teste, perguntar ao participante:

- Plataforma mais usada
- Programas mais usados, e seu nível de experiência em cada um deles
- Área de atuação profissional
- Nível de conhecimento de programação

### **Tarefas de teste**

Mostrar o patch básico.

*“Este é o software Fnk. Ele é usado para programação visual. Ele funciona através de nós e da ligação entre eles.”*

**1. Remoção de nós:** este teste visa medir de que forma o usuário seleciona o nó (clitando, ou desenhando um retângulo de seleção), e como ele faz para apagá-lo (backspace, delete, ou através de algum menu). *“Este nó está solto, sem estar ligado a nenhum outro nó. Apague-o.”*

**2. Movimentação da tela:** feito para medir qual o processo de movimentação da área de edição escolhido pelo usuário (barras de rolagem ou através do atalho espaço + drag'n drop como em ferramentas de edição). *“Você percebe que estas ligações levam a algum nó, que está fora da tela e além do seu alcance. Digamos que você queira visualizar este nó. O que você faria?”*

**3. Inserção de nós:** a idéia é ver qual o comportamento padrão do usuário para criar uma ligação sem saber mais sobre o recurso. *“Você pode perceber que ligações podem ser criadas entre dois conectores de nós distintos. Estes dois nós podem ser ligados por seus conectores. Tente criar uma nova ligação entre eles.”*

4. Manipulação de arquivos: idealizada para conferir se o usuário aciona os atalhos de teclado (como “Ctrl+O”, “Ctrl+S”) instintivamente, e quais, ou se vai direto ao menu. *“Salve as mudanças que você realizou num arquivo”, “Agora abra este mesmo arquivo.”*

# ANEXO A – Estatísticas de versão Flash

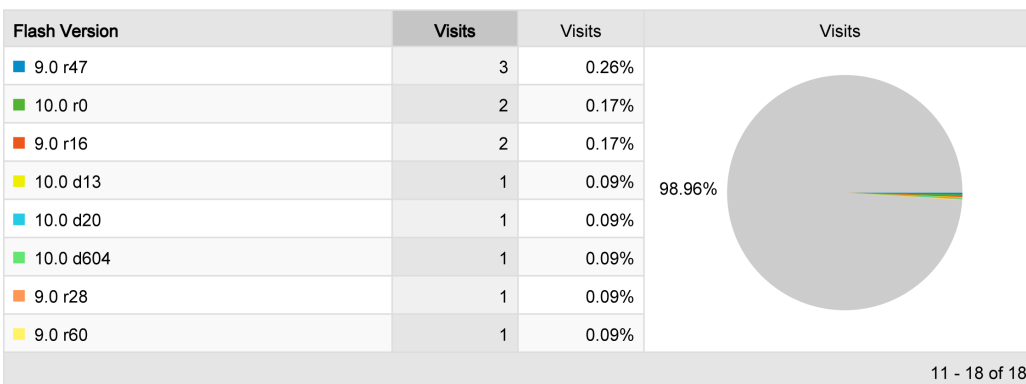
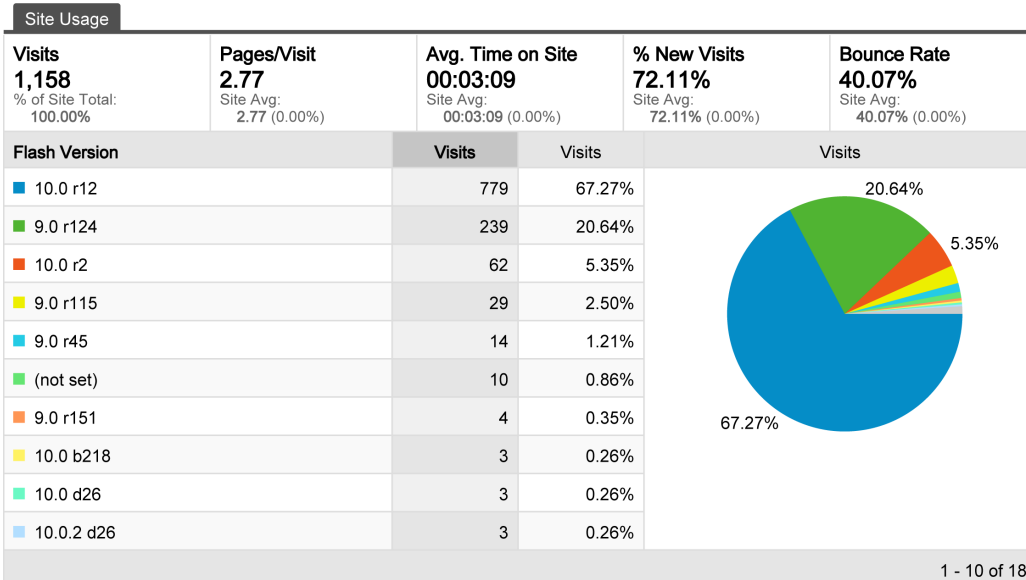
O gráfico abaixo, obtido através da ferramenta de estatísticas instalada no website do projeto, indica a versão do Flash Player encontrado em todas as visitas não-únicas realizadas ao website durante o mês de novembro. Visitas realizadas pelo autor são filtradas e excluídas destas estatísticas.

ffnkk.org

## Flash Versions

Nov 1, 2008 - Nov 19, 2008

1,158 visits used 18 flash versions





## ANEXO B – Estatísticas de upgrade do Flash

A tabela abaixo, obtida através da ferramenta de análise de *logs* instalada no website do projeto, lista os *referrals* do website durante o mês de novembro, isto é, de que sites os visitantes obtiveram o link para visitar o site do projeto. A quantidade de upgrades automáticos do Flash realizados pelo website (97) é indicada pelos dados da segunda linha (uma vez que a atualização do plugin realizada uma redirecionamento após a instalação com sucesso, imperceptível para o visitante).

<b>Statistics for:</b>	ffnkk.org	
<b>Last Update:</b>	20 Nov 2008 - 00:00	
<b>Reported period:</b>	Month Nov 2008	

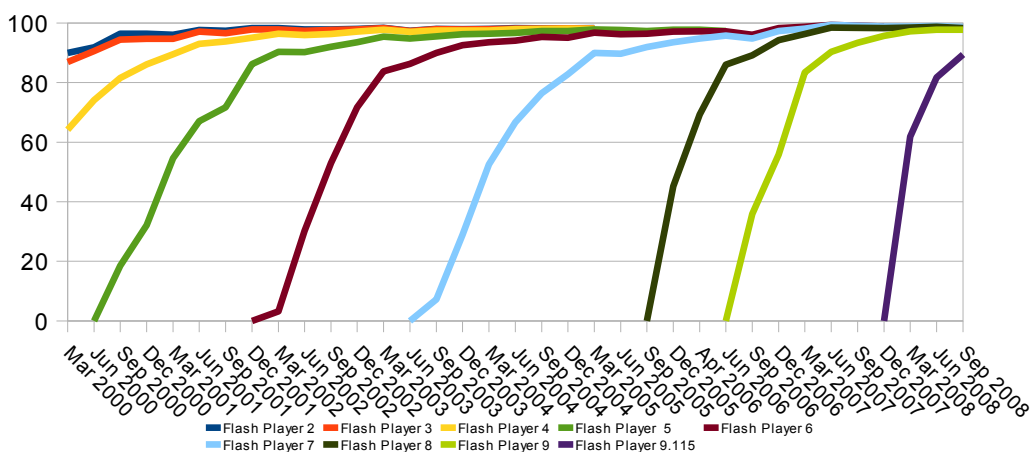
[Close window](#)

Links from an external page (other web sites except search engines)				
Total: 102 different pages-url				
	Pages	Percent	Hits	Percent
<a href="http://mrdoob.com/blog/post/609/">http://mrdoob.com/blog/post/609/</a>	149	19.4 %	149	19.4 %
<a href="http://www.macromedia.com/software/flash/about/installerRedirect...">http://www.macromedia.com/software/flash/about/installerRedirect...</a>	97	12.6 %	97	12.6 %
<a href="http://mrdoob.com/blog/">http://mrdoob.com/blog/</a>	92	11.9 %	92	11.9 %
<a href="http://zehfernando.com/2008/air-15-is-out-fnk-follows-suit/">http://zehfernando.com/2008/air-15-is-out-fnk-follows-suit/</a>	57	7.4 %	57	7.4 %
<a href="http://zehfernando.com">http://zehfernando.com</a>	51	6.6 %	51	6.6 %
<a href="http://www.netvibes.com">http://www.netvibes.com</a>	45	5.8 %	45	5.8 %
<a href="http://zehfernando.com/projects/">http://zehfernando.com/projects/</a>	28	3.6 %	28	3.6 %
<a href="http://twitter.com/home">http://twitter.com/home</a>	23	2.9 %	23	2.9 %
<a href="http://ricardocabello.com/blog/">http://ricardocabello.com/blog/</a>	23	2.9 %	23	2.9 %
<a href="http://labs.zeh.com.br/blog/?page_id=97">http://labs.zeh.com.br/blog/?page_id=97</a>	18	2.3 %	18	2.3 %
<a href="http://ricardocabello.com/blog/post/609/">http://ricardocabello.com/blog/post/609/</a>	18	2.3 %	18	2.3 %
<a href="http://kevincao.com">http://kevincao.com</a>	11	1.4 %	11	1.4 %
<a href="http://www.vimeo.com/2109843">http://www.vimeo.com/2109843</a>	8	1 %	8	1 %
<a href="http://mrdoob.com/rss/">http://mrdoob.com/rss/</a>	7	0.9 %	7	0.9 %
<a href="http://zehfernando.com/2008/letting-the-fnk-out/">http://zehfernando.com/2008/letting-the-fnk-out/</a>	7	0.9 %	7	0.9 %
<a href="http://hosted.zeh.com.br">http://hosted.zeh.com.br</a>	5	0.6 %	5	0.6 %
<a href="http://twitter.com/hysysk">http://twitter.com/hysysk</a>	5	0.6 %	5	0.6 %
<a href="http://hosted2.zeh.com.br">http://hosted2.zeh.com.br</a>	4	0.5 %	4	0.5 %
<a href="http://www.macromedia.com/software/flash/about/index.html">http://www.macromedia.com/software/flash/about/index.html</a>	3	0.3 %	3	0.3 %
<a href="http://zehfernando.com/category/flash/">http://zehfernando.com/category/flash/</a>	3	0.3 %	3	0.3 %
<a href="http://www.vimeo.com/2191017">http://www.vimeo.com/2191017</a>	3	0.3 %	3	0.3 %
<a href="http://twttrly.com">http://twttrly.com</a>	3	0.3 %	3	0.3 %
<a href="http://b.hatena.ne.jp/add?mode=confirm&amp;title=Fnk%20%28alpha%29%2...">http://b.hatena.ne.jp/add?mode=confirm&amp;title=Fnk%20%28alpha%29%2...</a>	3	0.3 %	3	0.3 %
<a href="http://www.macromedia.com/software/flash/about/so_redirect.swf">http://www.macromedia.com/software/flash/about/so_redirect.swf</a>	3	0.3 %	3	0.3 %
<a href="http://delicious.com/y.ocodop">http://delicious.com/y.ocodop</a>	3	0.3 %	3	0.3 %
<a href="http://vimeo.com/2191017">http://vimeo.com/2191017</a>	3	0.3 %	3	0.3 %
<a href="http://www.doom3world.org/phpbb2/viewtopic.php?t=22142">http://www.doom3world.org/phpbb2/viewtopic.php?t=22142</a>	3	0.3 %	3	0.3 %
<a href="http://twitter.com/freshface">http://twitter.com/freshface</a>	3	0.3 %	3	0.3 %
<a href="http://www.feedly.com/feedly">http://www.feedly.com/feedly</a>	3	0.3 %	3	0.3 %
<a href="http://beta.bloglines.com/b/view">http://beta.bloglines.com/b/view</a>	2	0.2 %	2	0.2 %



# ANEXO C – Taxa histórica de adoção do Flash Player

Os dados abaixo foram obtidos através da página oficial de informações sobre a tecnologia Flash – [http://www.adobe.com/products/player\\_census/flashplayer/version\\_penetration.html](http://www.adobe.com/products/player_census/flashplayer/version_penetration.html) – e pesquisas de versões anteriores desta página através do website <http://web.archive.org>.



	2000				2001				2002				2003				2004			
	Mar	Jun	Set	Dez	Mar	Jun	Set	Dez	Mar	Jun	Set	Dez	Mar	Jun	Set	Dez	Mar	Jun	Set	Dez
FP 2	89.9	91.8	96.4	96.4	96.0	97.6	97.4	98.3	98.3	97.8	97.8	98.0	98.4	97.4	98.0	97.9	98.0	98.3	98.2	98.2
FP 3	86.9	90.6	94.4	94.7	94.7	97.1	96.6	97.8	97.8	97.4	97.6	97.8	98.3	97.3	97.9	97.8	97.9	98.1	98.1	98.2
FP 4	64.2	74.1	81.5	86.1	89.5	93.0	93.8	95.1	96.4	95.9	96.3	97.1	97.7	97.0	97.5	97.5	97.5	97.9	98.0	98.1
FP 5		0.0	18.5	32.1	54.6	67.0	71.7	86.2	90.3	90.2	92.0	93.5	95.4	94.8	95.5	96.2	96.4	96.7	97.3	97.3
FP 6								0.0	3.2	30.2	53.0	71.7	83.8	86.3	90.0	92.5	93.5	94.1	95.3	95.1
FP 7														0.0	7.2	28.9	52.5	66.7	76.4	82.8
FP 8																				
FP 9																				
FP 9.115																				

	2005				2006				2007				2008		
	Mar	Jun	Set	Dez	Abr	Jun	Set	Dez	Mar	Jun	Set	Dez	Mar	Jun	Set
FP 2	98.3														
FP 3	98.2														
FP 4	98.1														
FP 5	97.8	97.6	97.3	97.7	97.7	97.3									
FP 6	96.8	96.2	96.4	97.1	97.2	97.2	96.0	98.3	98.7	99.3	99.1	98.8			
FP 7	90.0	89.7	91.9	93.5	94.8	95.8	94.8	97.3	98.1	99.3	99.1	98.8	98.8	99.0	98.6
FP 8			0.0	45.2	69.3	86.0	89.1	94.2	96.3	98.5	98.4	98.3	98.5	98.7	98.3
FP 9						0.0	35.9	55.8	83.4	90.3	93.3	95.7	97.2	97.7	97.7
FP 9.115												0.0	61.8	81.7	89.4

