

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

SonData: um *Toolkit* para Sonorização de Dados Interactiva

João Miguel Azevedo de Menezes

Mestrado em Multimédia

Orientador: Doutor Carlos Alberto Barbosa da Cunha Mendonça Guedes

Julho 2012

Resumo

O objectivo deste projecto era proporcionar um *toolkit* para sonorização de dados interactiva, dirigido a todos os praticantes interessados em sonorizar dados. No entanto, também deveria oferecer instrumentos úteis para a comunidade académica da sonorização de dados.

Neste sentido, desenvolvemos SonData, um *toolkit* que promove funções para a sonorização interactiva de conjuntos de dados complexos e multidimensionais. Para tal, SonData oferece módulos para as técnicas de *Parameter Mapping Sonification* (PMSOn) e *Model-Based Sonification* (MBS). Que tenhamos conhecimento, não existe outro *toolkit* para sonorização de dados que promova o uso de MBS ou destas duas técnicas em simultâneo.

SonData é desenvolvido em Max/MSP, um ambiente gráfico de programação facilmente acessível a um grande número de utilizadores. Para além disto, integra ainda a framework Jamoma, promovendo a (1) modularidade, (2) conectividade, (3) robustez e (4) rápida prototipagem no contexto da sua aplicação.

Abstract

The aim of this work was to provide an Interactive Data Sonification toolkit, targeted at all practitioners interested in sonifying data. However, it should also provide a set of tools that are useful for the academic community of Data Sonification.

For this purpose we developed SonData, which provides tools for the interactive sonification of complex and multidimensional datasets. In order to do so, SonData provides modules for both Parameter Based Sonification (PMSon) and Model-Based Sonification (MBS). To our knowledge, no other software for Data Sonification integrates MBS, nor both of these techniques in conjunction.

SonData was developed in Max/MSP integrating the Jamoma framework. Max/MSP is a high level programming language which is easily accessible to a large audience of target-users. In addition, it integrates the Jamoma framework, providing (1) modularity, (2) connectivity, (3) robustness, and (4) rapid prototyping in its application context.

Agradecimentos

Ao Professor Doutor Carlos Guedes, pela disponibilidade, apoio e confiança que sempre me transmitiu, não só no âmbito desta tese, mas também no meu percurso académico.

Aos meus pais e irmãs pelo amor e apoio demonstrado em todos os momentos da minha vida.

Por último, mas não menos importante, à Roxanne Leitão. Para além do amor e sábios conselhos, por estar sempre lá.

Índice

1. Introdução	19
Motivação	19
Objectivos	20
Estrutura do documento	20
2. Sonorização	23
O som como meio de transmissão de informação	23
Definição	24
Aplicações	26
Alarmes, alertas e avisos	26
Monitorização de Processos	26
Análise exploratória de dados	27
Meio artístico	29
Sumário	30
Técnicas	30
Audificação	30
Representação icónica	31
Parameter Mapping Sonification (PMSon)	31
Mapeamento	32
A - Polaridade	33
B - Escala	33
Discussão	33
Model-Based Sonification (MBS)	34
Configuração	35
Dinâmicas	35
Excitação	36
Variáveis acústicas	36
Conclusão	37

3. Sonorização Interactiva	40
Definição.....	40
Interacção em Parameter Mapping Sonification(PMSon)	40
Interacções de Navegação	40
Interacções de Mapeamento	41
Seleccção interactiva de dados	42
Interacção em Model Based Sonification (MBS).....	42
Conclusão.....	46
4. SonData	48
Introdução	48
Jamoma framework.....	49
Convenções gerais	50
Arquitectura	50
Estrutura de um módulo	51
Interface	51
Componentes internos.....	53
Documentação	54
Módulos	56
Aquisição e armazenamento	57
Interacção	59
Mapeamento	63
Síntese de som	65
Utilitários	67
5. Conclusão	68
Sugestões para estudo futuro	69
6. Referências	70
7. Apêndice	75

Lista de figuras

Figura 2.1 - Computação da técnica de PMSon	32
Figura 2.2 - Função de mapeamento de parâmetro	32
Figura 3.1 - Sonograma de dados	43
Figura 3.2 - Modelo de Sonorização de GNG	44
Figura 3.3 - Adaptação a superfície multitoque	44
Figura 3.4 - Modelo de Sonorização da curva principal	45
Figura 3.5 - Interface <i>Audio-haptic ball</i>	45
Figura 4.1 - Aparência geral de um módulo	50
Figura 4.2 - Componentes do interface de um módulo	51
Figura 4.3 - Cabeçalho de um módulo.....	51
Figura 4.4 - <i>Widgets</i>	52
Figura 4.5 - Primeira camada interna de um módulo	53
Figura 4.6 - Segunda camada interna de um módulo	54
Figura 4.7 - <i>Patch</i> de ajuda de um módulo	55
Figura 4.8 - Página de referência de um módulo	55
Figura 4.9 - Metodologia proposta por SonData	56
Figura 4.10 - Módulo <code>jmod.son.url</code>	58
Figura 4.11 - Módulo <code>jmod.son.table</code>	58
Figura 4.12 - <i>Panel</i> <code>jmod.son.table</code>	58
Figura 4.13 - Módulo <code>jmod.son.mgraph</code>	59
Figura 4.14 - <i>Panel</i> <code>jmod.son.mgraph</code>	59
Figura 4.15 - Módulo <code>jmod.son.brush</code>	60
Figura 4.16 - <i>Panel</i> <code>jmod.son.brush</code>	60

Figura 4.17 - Módulo jmod.son.knearest	61
Figura 4.18 - <i>Panel</i> jmod.son.knearest	61
Figura 4.19 - Módulo jmod.son.model	62
Figura 4.20 - <i>Panel</i> jmod.son.model	63
Figura 4.21 - Módulo jmod.son.mapper	64
Figura 4.22 - Módulo jmod.son.linkmodel	64
Figura 4.23 - Módulo jmod.son.pulsar~	65
Figura 4.24 - Módulo jmod.son.resonators~	66
Figura 4.25 - Módulo jmod.son.fof~	66
Figura 4.26 - Módulo jmod.son.karplus~	66
Figura 4.27 - Módulo jmod.son.getcol~	67
Figura 4.28 - Módulo jmod.son.datetime	67

Lista de tabelas

Tabela 4.1 — Módulos de interação, mapeamento e respectivas técnicas 57

Tabela 4.2 — Módulos de Interação em PMSon e respectiva tipologia 57

Abreviaturas

GNG - *Growing Neural Gas*

MBS - *Model-Based Sonification*

PMSon - *Parameter Mapping Sonification*

1. Introdução

A presente tese apresenta SonData, um *toolkit* modular para a sonorização de dados interactiva nos paradigmas de *Parameter Mapping Sonification* e *Model Based Sonification*.

Este capítulo apresenta uma breve descrição desta tese, ilustrando os seus objectivos e a motivação que levou ao seu desenvolvimento.

1.1. Motivação

Desde cedo que o sistema auditivo é um instrumento fundamental na percepção que temos do mundo. Este, graças ao processo evolucionário, é hoje um órgão complexo e poderoso na interpretação do universo que nos rodeia. Ao identificarmos fontes sonoras, palavras e melodias mesmo em condições ruidosas, estamos a fazer uso de capacidades cognitivas extraordinárias. Capacidades essas, a que Hermann et al. (2011) se referiram como “um processo supremo de reconhecimento de padrões que nem os computadores mais recentes são capazes de reproduzir”.

Nas ultimas décadas, os cientistas têm tirado partido destas capacidades, aplicando-as na análise de dados complexos e multidimensionais, outrora representados visualmente. Para tal, as relações dos dados são traduzidas para relações perceptíveis num sinal acústico, de modo a facilitar a sua comunicação e interpretação (Kramer, 1992). Este processo é conhecido como sonorização de dados, e recorre ao nosso sistema auditivo como canal principal na interpretação dos dados a serem comunicados.

Os dados passíveis de serem sonorizados, dados quantitativos, são imensos e de de naturezas igualmente distintas. Isto levanta questões sobre qual a melhor estratégia para a sua representação sonora, por outras palavras, quais os atributos acústicos que melhor representam os dados sobre análise. Esta é uma questão frequente no desenvolvimento de uma sonorização, e um bom conhecimento da natureza dos dados é fundamental para o sucesso do mapeamento definido.

Actualmente existem vários sistemas para a sonorização. Por um lado, sistemas bastante eficazes na sonorização de um tipo específico de dados (EEG¹, sísmicos, entre outros), por outro, sistemas de uso geral com capacidade para lidar com vários tipo de dados. Estas com o amadurecimento do campo, têm vindo a tornar-se obsoletas.

A crescente capacidade de processamento dos computadores actuais permitiu o desenvolvimento de novos paradigmas de interacção para com um sistema, assim como o aparecimento da técnica de *Model-Based Sonification* (MBS; Secção 2.4.4). Estes paradigmas têm-se revelado bastante úteis no processo de sonorização, com especial contribuição para a análise exploratória de dados (Secção 2.3.3). No entanto, o seu potencial tem sido essencialmente explorado num âmbito académico, o que em torno restringe o seu acesso à comunidade em geral. Actualmente, não existe nenhuma ferramenta prontamente disponível que implemente os actuais paradigmas de interacção nem os princípios da técnica de MBS.

Tendo em conta este contexto, foi desenvolvido SonData, uma plataforma modular para a sonorização interactiva na técnica de *Parameter Mapping Sonification* (PMSon) e *Model-Based Sonification* (MBS).

1.2. Objectivos

Os objectivos propostos para este projecto focaram essencialmente a arquitectura modular da aplicação e a sua contribuição para a alteração de rotinas de mapeamento na sonorização bem como os paradigmas de interacção para com o sistema. Nesse contexto foram cumpridos os seguintes objectivos:

- Identificação dos diferentes cenários de interacção nas técnicas de *Parameter Mapping Sonification* e *Model Based Sonification*;
- Desenvolvimento de um *toolkit* modular, multiplataforma e *open-source* para sonorização de dados interactiva.

1.1. Estrutura do documento

Este documento é constituído por cinco capítulos.

Este capítulo, que sendo o primeiro, descreve a motivação e os objectivos subjacentes ao seu desenvolvimento.

¹ Electroencefalografia

O segundo capítulo apresenta a disciplina da sonorização de dados, ilustrando os contextos da sua aplicação e técnicas, com particular ênfase na técnica de *Parameter Mapping Sonification* e *Model-Based Sonification*. Adicionalmente, vão sendo referenciados alguns sistemas que apliquem as mesmas.

O terceiro capítulo descreve a componente interactiva na sonorização de dados, ilustrando o actual estado da arte relativamente aos cenários de interacção presentes na técnica de PMSon e MBS.

O capítulo quatro começa por descrever a arquitectura global do *toolkit* proposto neste documento. De seguida, são descritos os módulos mais relevantes, ilustrando as suas funções e como estes se relacionam entre si.

Por último, o capítulo cinco apresenta as principais conclusões obtidas com o desenvolvimento do projecto, e ainda, apresentadas algumas sugestões para um estudo futuro.

2. Sonorização

Neste capítulo, serão ilustradas as vantagens e desvantagens inerentes ao uso do sistema auditivo na interpretação de dados complexos. Posteriormente, são apresentadas duas definições para a sonorização de dados e discutida qual a adoptada. Adicionalmente são descritos os actuais contextos de aplicação da sonorização de dados, os sistemas utilizados para os mesmos e as suas técnicas.

2.1. O som como meio de transmissão de informação

De seguida serão apresentadas algumas vantagens e implicações existentes no uso da audição para interpretação de dados. Para tal, será feita uma comparação entre o campo de sonorização e visualização de dados.

Ao compararmos os dois campos (sonorização e visualização de dados), um dos factores que mais se destaca é o tempo. Quando interpretamos um gráfico, o tempo não nos é apresentado no domínio contínuo, isto é, cada indivíduo faz a leitura do gráfico a uma velocidade distinta, já na sonorização, os dados são revelados à medida que são ouvidos, dando ao ouvinte uma representação fiel do seu comportamento ao longo do tempo. Adicionalmente, o sistema auditivo é bastante sensível às alterações temporais, mesmo as mais subtis. Esta capacidade é bastante útil na identificação de tendências e padrões, que numa representação visual estariam “escondidos”.

Outro aspecto importante do sistema auditivo é o facto de conseguirmos identificar diferentes “objectos auditivos” simultaneamente. Frequentemente, ao ouvirmos uma música focamos a nossa atenção num determinado som ou instrumento. Dependendo da complexidade da peça e da experiência do ouvinte, esta discriminação é feita de forma mais ou menos eficaz. Porém, numa representação visual, se tentarmos interpretar vários gráficos ou vídeos que nos são apresentados em tempo real, esta capacidade reduz significativamente. Esta capacidade cognitiva, faz da sonorização uma ferramenta bastante útil na análise de dados multidimensionais, permitindo ao ouvinte analisar a sonorização como um “todo” bem como focar a sua atenção numa dimensão em particular, compreendendo a sua relação com as restantes dimensões.

Hoje sabemos que a nossa capacidade de memória para timbres e sequências sonoras abstractas é inferior em relação à capacidade de memorizar gráficos abstractos (Dingler et al., 2008), logo, em alguns casos a identificação de padrões pode tornar-se mais difícil com a sonorização. No entanto, está provado que músicos com formação conseguem detectar com maior precisão as variações numa sonorização (Newhoff et al., 2002). Isto deve-se à capacidade para aprender e melhorar a discriminação de estímulos auditivos. Para ilustrar esta capacidade, Hermann et al. (2011) deram o exemplo do ouvido de um mecânico experiente, que consegue identificar com precisão o problema de um carro através do barulho do motor, enquanto um ouvinte inexperiente apenas conseguiria dizer que “alguma coisa não está bem com o motor do carro”. Esta capacidade deve-se ao processo de aprendizagem a que o mecânico foi submetido ao longo do tempo, dando-lhe uma excelente habilidade auditiva na identificação de problemas no seu domínio. Isto sugere que “ao ser providenciada uma fase de treino e linhas de orientação para a audição de dados, o nosso cérebro tem o potencial de surgir com novas e úteis formas de ouvirmos os dados” (Hermann et. al. 2011).

As vantagens da sonorização são claras, particularmente quando aplicada em conjuntos de dados multidimensionais, permitindo ao ouvinte a capacidade de agrupar estes dados a dois níveis: Acústico (frequência, duração, timbre, entre outros) e espacial (através da espacialização áudio). No entanto, existem algumas limitações no uso da sonorização, uma delas deve-se à sua interferência com outros tipos de comunicação auditiva, nomeadamente a fala. Por esta razão, ao ouvirmos uma sonorização com a tarefa de identificar as relações dos dados, particularmente se estes forem multidimensionais, é recomendável um ambiente que não requeira muita comunicação verbal, já numa tarefa de monitorização de processos ou alarmes auditivos, esta necessidade não é tão acentuada (Secções 2.3.1, 2.3.2).

2.2. Definição

Sendo a sonorização de dados um campo relativamente recente, a sua definição encontra-se actualmente em progresso. De seguida serão apresentadas duas definições para a sonorização de dados. No final, será discutida a definição utilizada no desenvolvimento desta tese.

Definição A

Numa primeira fase, Kramer et al., (1997) definiram a sonorização de dados como “o uso de áudio não-verbal para transmitir informação, mais precisamente, a sonorização é a transformação de relações de dados para relações perceptuais num sinal acústico com o propósito de facilitar a comunicação ou interpretação.”

Esta definição foi a que reuniu maior consenso entre a comunidade, no entanto, com o amadurecimento do campo e o aparecimento de novos paradigmas, a

definição apresentada por Kramer revela algumas lacunas. Por exemplo Kramer et al. (1997) excluem claramente o uso de áudio verbal, no entanto, a comunidade tem tirado partido desta, recorrendo a sistemas de síntese de fala, gerada através da articulação de formantes de vogais. Hermann et al. (2006) desenvolveram uma sonorização bastante eficaz para a análise de dados de EEG através da articulação de formantes de vogais.

Para além do aparecimento de novas técnicas, outra razão que motivou a revisão da taxonomia do campo foi o aparecimento de composições musicais conotadas como sonorizações. Apesar de ambas serem compostas por som organizado, o principal objectivo da peça musical não é transmitir com a máxima clareza informação relativa aos dados sob análise, ao contrário da sonorização. Por estas razões, Hermann (2008) apresenta uma série de condições que considera necessário reunir para podermos considerar uma sonorização. Estas, compõem a seguinte definição:

Definição B

Uma técnica que use dados como *input* e gere sinais sonoros (eventualmente em resposta a uma facultativa excitação adicional ou desencadeamento) pode ser chamada de sonorização, se e só se:

A. O som reflecte propriedades ou relações objectivas nos dados de entrada;

B. A transformação é sistemática. Isto significa que existe uma definição precisa sobre de que modo os dados (e interacções facultativas) provocam a alteração do som;

C. A sonorização é reproduzível: introduzidos os mesmos dados e interacções idênticas (ou desencadeamentos) o som resultante tem que ser estruturalmente idêntico.

D. O sistema possa ser intencionalmente usado com diferentes dados, e também em repetição com os mesmos dados.

Resumindo, a “sonorização é a geração de som dependente de dados, se a transformação for sistemática, objectiva e reproduzível.” (Hermann, 2008)

A reunião destas condições (dependência de dados, objectividade, sistematicidade e reproduzibilidade), resultam de uma série de discussões entre Hermann e membros da comunidade e permitem fazer uma clara distinção entre a sonorização e outras práticas que façam uso das mesmas técnicas. No entanto, devemos ter em conta que a eficácia de uma sonorização não depende unicamente da reunião das condições apresentadas por Kramer. Para além disto, mais importante do que o termo utilizado, é a capacidade que a representação tem (ou não) de transmitir claramente os dados sob análise (Walker & Nees, 2011). Neste

documento, é adoptada a definição dada por Hermann, uma vez que é a mais completa e aquela que melhor define o papel da sonorização no contexto actual.

2.3. Aplicações

Nesta secção são apresentadas as principais áreas de aplicação da sonorização de dados, com especial interesse na exploração de dados e meio artístico, uma vez que é a que sairá mais beneficiada com o desenvolvimento deste *toolkit*. Será também descrito o papel da sonorização no meio artístico.

2.3.1. Alarmes, alertas e avisos

Nos sistemas de alarme, alertas e avisos, o som serve como meio para indicar ao ouvinte que alguma coisa aconteceu ou está para acontecer, esta, requer uma acção imediata ou num curto intervalo de tempo. Existem várias razões para o uso do som neste tipo de sistemas, por um lado, os nossos ouvidos não podem simplesmente fechar-se e ignorar o som (ao contrário do que acontece na visão), por outro lado, os alarmes conseguem despertar a atenção do ouvinte de uma forma bastante eficaz. Aparelhos como o telefone, detectores de fumo, a buzina do carro, campainha da porta, entre outros, são bons exemplos de sistemas de alerta auditivos (Hermann, 2002).

Contudo, esta técnica tem algumas limitações. Por um lado, a sua eficácia, depende de um conhecimento prévio do alerta por parte do ouvinte. Isto é, não sendo dada uma informação acústica adicional, a associação do alerta com a sua causa requer um conhecimento prévio do contexto onde este actua. Por outro lado, estes alertas não têm capacidade para transmitir muita informação. Por exemplo, ao ouvirmos a campainha da porta, sabemos que alguém está à porta, no entanto, não sabemos quem é nem o que pretende, o mesmo acontece quando ouvimos o alarme do microondas, sabemos que o tempo que definimos expirou, no entanto não sabemos se a comida está pronta ou não (Walker & Nees, 2006). Porém, existem alguns exemplos de alertas auditivos que transmitem alguma informação adicional. Desde telefones, que permitem definir um som de alerta para um contacto específico, permitindo ao ouvinte saber quem está a telefonar através do alerta ouvido, ou uma sirene de bombeiros, onde o número de toques é dado em função do tipo de desastre.

2.3.2. Monitorização de Processos

Na monitorização de processos, o ouvinte detecta eventos (representados por sons) discretos que ocorrem quando é atingido algum tipo de limite num sistema. Como acontece com as funções de alerta, para que o ouvinte identifique qual o

significado de cada evento (som), é necessário ter um conhecimento prévio do seu significado naquele contexto.

O nosso sistema auditivo oferece grandes vantagens em relação ao nosso sistema visual na monitorização de processos. Kramer et al. (1997), identificaram duas: Por um lado, o facto de o utilizador ter a sua visão livre, podendo executar outras tarefas em simultâneo, por outro lado, aquilo que Kramer (1992) denominou como “*backgrounding capacity*”, a capacidade de ignorar um determinado ambiente acústico, ao mesmo tempo que estamos suficientemente atentos a mudança significativas nesse mesmo ambiente. Esta é uma capacidade que usamos com bastante frequência. Ao conduzir um carro, por exemplo, somos capazes de colocar o som do motor num nível de pouca prioridade, onde passa praticamente despercebido, no entanto, se este fizer um ruído estranho, a sua prioridade passa para primeiro plano (Walker & Nees, 2006).

O uso da sonorização na monitorização de processos está presente em vários âmbitos, do hospitalar, ao da monitorização de processos fabris. Fitch e Kramer (1992) desenvolveram um estudo onde a tarefa era manter um “paciente” (computador) vivo. Para tal, foi feita a monitorização da condição do paciente durante uma cirurgia. Essa monitorização podia chegar até 8 variáveis (contínuas) e era transmitida pelo meio auditivo e visual. Após a análise de resultados, os autores concluem que a rapidez e eficácia do sistema auditivo eram muito superiores aos resultados conseguidos com o sistema de monitorização visual. Gaver et al. (1991) desenvolveram uma simulação onde os sujeitos, operários de uma fábrica virtual, controlavam e alteravam determinado processo reagindo de acordo com a monitorização que lhes era transmitida pelos meios visual e auditivo. À semelhança da conclusão de Fitch e Kramer (1992) no estudo referenciado anteriormente, ao comparar os resultados obtidos com os dois tipos de monitorização, os autores concluíram que o meio auditivo reduzia a taxa de erro na elaboração do processo ao mesmo tempo que acelerava o tempo de reacção.

2.3.3. Análise exploratória de dados

Frequentemente, usamos as nossas capacidades auditivas com o propósito exploratório. Por exemplo, ao fazermos uma análise do estado de um piso através do som dos nossos passos (Hermann, 2002), ou ao percutirmos uma superfície de modo a tentar identificar o seu material (vidro, madeira, plástico, entre outros), se esta é oca ou maciça, estamos a fazer uso de avançadas capacidades cognitivas que nos permitem concretizar com eficácia estes processos exploratórios. Estas capacidades complexas, que nos permitem classificar e identificar variações ocorridas no som, foram ilustradas por Hermann (2002) através de um exemplo do som de um carro, onde ao mesmo tempo que fazemos a classificação sonora (o barulho do motor) conseguimos interpretar variações nas suas categorias (altura, nível, rugosidade do som) e identificar quais as suas causas (a velocidade do carro,

por exemplo). No final deste exemplo, Hermann afirma que “mais importante, é que estamos aptos a desenvolver novas categorias e interpretar sons até então desconhecidos” (Hermann, 2002). Esta habilidade na segmentação e classificação de sons outrora desconhecidos, é particularmente útil na análise exploratória de dados (Secção 2.3.3). É através desta prática que o ouvinte pode efectuar tarefas como identificação de tendências, análise da estrutura e a inspecção exploratória do conjunto de dados.

Ao contrário da monitorização de processos e dos alertas auditivos, onde a informação é condensada de modo a capturar um estado momentâneo, a tarefa de identificação de tendências requer uma análise holística dos dados sob análise (Hermann, 2002). Esta análise permite a codificação integral da informação contida no conjunto de dados, dando ao ouvinte a possibilidade de perceber o seu comportamento ao longo do tempo, se estes aumentam ou diminuem, por exemplo. Walker e Nees (2006), comparam a noção de tendência à noção de contorno melódico de uma peça musical. Nesta, o ouvinte pode ouvir a peça como um todo, ou dedicar maior interesse a uma determinada secção. O mesmo acontece na identificação de tendências através da sonorização, o ouvinte pode estar empenhado em perceber a tendência geral dos dados ou focar a sua atenção num determinado intervalo de tempo.

Os sistemas mais comuns para a identificação de tendências são os gráficos auditivos. Nestes, os dados representados no eixo vertical (y) são traduzidos para atributos acústicos (altura, por exemplo), sendo estes comunicados ao ouvinte ao longo do tempo, representado pelo eixo horizontal (x). Exemplo de um sistema que implemente este tipo de gráficos é *Sonification Sandbox* (Walker & Cothran 2003).

A inspecção exploratória ocorre quando o ouvinte explora os dados sem nenhuma questão predefinida, ou seja, a exploração é feita sem nenhum tipo de tarefa designada à partida, ao contrário do que acontece na identificação de tendências e na identificação da estrutura.

Na identificação da estrutura dos dados, o ouvinte explora a estrutura global dos dados e as complexas relações entre as diversas variáveis. Estas relações são identificadas a partir de interacções para com o sistema, fazendo da Sonorização Interactiva uma prática importante neste tipo de análise. Os sistemas desenvolvidos para este tipo de análise, recorrem à técnica de *Model-Based Sonification* (Secção 2.4.4), onde os dados são representados sob a forma de um modelo físico virtual, cujas dinâmicas (ver 2.4.4.2) estão associadas a atributos acústicos. Quando excitado, o modelo responde acusticamente de acordo com as dinâmicas subjacentes, permitindo ao ouvinte compreender as relações entre os dados que o compõem.

A necessidade de interacção na análise exploratória varia de acordo com a tarefa em mãos. Por exemplo, a identificação de tendências num conjunto de dados, onde o ouvinte tenta identificar um aumento ou diminuição dos dados quantitativos ao longo do tempo, é uma tarefa que não requer modos de interacção muito avançados. Algum controlo de navegação (Secção 3.2.1) pode ser útil, tornando a tarefa mais prática para o ouvinte, no entanto o seu sucesso depende mais do

mapeamento definido (Secção 3.2.2) do que da componente interactiva presente na ferramenta. Já na identificação da estrutura e inspecção exploratória, onde o som ouvido é resultado da interacção para com o modelo virtual, a presença de uma forte componente interactiva é fundamental para a execução da tarefa. Na Secção 3.3 deste documento, são descritos os vários modos de interacção na técnica de MBS e discutido o seu papel no contexto actual da sonorização.

2.3.4.Meio artístico

Graças aos avanços tecnológicos ocorridos nas últimas décadas, o uso da sonorização (ou de processos muito semelhantes aos usados na sonorização), tem-se vindo a expandir para o domínio artístico, nomeadamente, no âmbito de computer music e artes performativas. Peças musicais feitas a partir de conjuntos de dados provenientes do processo de electroencefalografia (EEG) (*“Listening to the mind listening: Concert of sonifications at the Sydney Opera House”*, 2004) e dados globais dos mercados financeiros e saúde (*“Global music - The world by ear”*, 2006) são alguns dos exemplos mais recentes do uso da sonorização neste contexto.

Existem várias semelhanças entre uma sonorização e uma composição musical que faça uso de processos de sonorização, no entanto as suas premissas são bem distintas. Ambas são compostas por som organizado e possuem uma relação com os dados sob análise, há até sonorizações que podem ser ouvidas como peças musicais (Vickers & Hogg, 2006), no entanto, operam com propósitos diferentes. O principal objectivo da sonorização é transmitir com a máxima clareza as relações contidas no conjunto de dados sob análise, enquanto que na composição musical, o seu principal objectivo prende-se mais com o aspecto estético da peça e com o processo de uso de dados para a composição da mesma. Este tipo de prática, onde o compositor faz uso de dados como meio de criação, deve ser chamada de *“data-inspired music”*, ou *“data-controlled music”* (Hermann, 2008).

Outros processos semelhantes aos da sonorização, podem ser encontrados na área das artes performativas. Frequentemente é usada a análise de imagem, como meio de extracção de dados quantitativos relativos ao movimento de uma ou mais pessoas. Um bom exemplo deste tipo de casos, é (*“Etude for Unstable Time”*, 2003), uma peça cujo material acústico é gerado a partir da análise de movimento de um bailarino. Para tal, Guedes recorre ao uso de algoritmos (*m.objects*; Guedes 2005) que extraem informação relativa ao movimento do bailarino através de uma câmara de vídeo. Os dados extraídos são enviados para um algoritmo de geração sonora e mapeados para atributos musicais, como o ritmo por exemplo. O facto de o som ser gerado através de dados relativos à análise de imagem, torna esta prática bastante semelhante à da sonorização, no entanto, como acontece com o uso da sonorização no processo musical, o seu objectivo primário não é transmitir com clareza os dados sob análise.

2.3.5. Sumário

Em suma, nesta secção foram ilustradas as áreas de aplicação da sonorização de dados, e referenciados alguns exemplos. Nos alertas e alarmes, bem como na monitorização de processos, a sonorização assinala eventos discretos, isto é, os dados sob análise atingem um determinado estado e é emitido um sinal sonoro. Já na análise exploratória de dados, a sonorização é frequentemente ouvida continuamente, isto é, o ouvinte analisa os dados em função do tempo, podendo assim identificar padrões nos dados sob análise. O mesmo acontece na sua aplicação no meio artístico, que apesar de não ter o objectivo de representar claramente os dados sob análise, aplica processos bastante semelhantes aos da sonorização de dados.

O trabalho apresentado no Capítulo 4 deste documento procura dar resposta às necessidades existentes no domínio da análise exploratória de dados, promovendo diferentes cenários de interacção para com o sistema e representação dos dados sob análise. Todavia, ao contrário de um sistema hermético e dedicado à comunidade científica, a sua flexibilidade e abertura devem promover a sua aplicação no meio artístico, contribuindo para contextos que exijam o mapeamento de dados para atributos acústicos.

2.4. Técnicas

Apesar deste projecto contribuir particularmente para as técnicas de *Parameter Mapping Sonification* (PMSon; Secção 2.4.3) e *Model-Based Sonification* (MBS; Secção 2.4.4), nesta secção iremos descrever de uma forma geral as técnicas usadas para a sonorização de dados e exemplos que ilustrem a sua aplicação.

2.4.1. Audificação

A técnica mais simples para a tradução de dados para um sinal acústico é a audificação. Nesta técnica, uma variável de um determinado conjunto de dados é “transposta” para uma frequência audível, entre 50 Hz e 20000 Hz. Esta transposição opera em cada medição do conjunto de dados, transformando-a numa amostra de áudio. Esta amostragem é normalmente feita a uma taxa de cerca de 40 KHz, isto significa que para um segundo de representação sonora são necessários cerca de 40000 medições (valores). Por esta razão, são necessários conjuntos de dados bastante extensos para uma audificação de poucos segundos (Hermann, 2002).

Geralmente, a audificação é a primeira abordagem no processo de sonorização e existem diversos sistemas para a sua implementação. *Sonipy* e *Sonifyer* (Worrall et

al., 2007; Dombois, 2008) são exemplos de sistemas eficazes no processo de audificação.

2.4.2. Representação icónica

A representação icónica divide-se em dois tipos: (1) *Earcons* e (2) ícones auditivos. (1) *Earcons* são “tons sintéticos e abstractos que podem ser usados em combinações estruturadas para criar mensagens auditivas” (Blattner et al., 1989). Estes sons curtos, são frequentemente aplicados na relação entre humano-computador. O som musical que ouvimos quando ligamos o computador é um bom exemplo deste tipo de representação.

Enquanto os *earcons*, recorrem ao uso de sons sintéticos abstractos, os (2) ícones auditivos, recorrem a sons não verbais presentes no nosso quotidiano, como meio de representação de uma acção análoga. A sua aplicação é feita maioritariamente nas aplicações de computadores, dando uma ligação intuitiva sobre a acção levada a cabo pelo utilizador (Brazil & Fernström, 2011). Por exemplo, quando “esvaziamos a reciclagem” do nosso computador, é emitido o som de papel a ser “esmagado”. Este som simbólico, remete o utilizador para um processo de associação entre a operação efectuada e o som ouvido.

No contexto da sonorização, este tipo de representação icónica é frequentemente usada em conjunto com a técnica de PMSon. Nestes casos, a sua função é fornecer informação contextual, adicional à sonorização. Estudos demonstram que o uso de sons contextuais que assinalem a progressão temporal da sonorização (análogos aos “traços” representados no eixo do x num gráfico visual), ou tons de referência (“beep”, por exemplo) que assinalem valores importantes nos dados a comunicar (valor mínimo e máximo, por exemplo), permitem uma maior precisão na identificação da tendência dos dados (Secção 2.3.3) e momentos de interesse na sonorização (Smith & Walker, 2002). No entanto, a eficácia desta técnica requer um conhecimento prévio do significado do *earcon* ou ícone sonoro e do contexto onde este actua.

2.4.3. Parameter Mapping Sonification (PMSon)

A técnica de *Parameter Mapping Sonification* é a técnica mais comum na sonorização de dados. Enquanto as técnicas apresentadas anteriormente requerem conjuntos de dados extensos (audificação) ou eventos discretos (ícones sonoros e ícones auditivos), a técnica de PMSon pode dar resposta a conjuntos de dados com tamanhos arbitrários, tornando-a numa técnica bastante flexível (Hermann, 2002).

Esta técnica consiste no mapeamento de características dos dados para parâmetros de síntese de som de modo a perceber as relações dos dados sob análise (Hermann et al., 2011). Um exemplo intuitivo e eficaz é o termómetro auditivo. A

temperatura é mapeada para a frequência de um sinal acústico, permitindo identificar claramente o aumento ou diminuição da mesma (identificação de tendências, ver 2.3.3).

2.4.3.1. Mapeamento

Hermann (2002) formalizou o mapeamento do seguinte modo: fornecido um conjunto de dados com d dimensões $\{x_1, \dots, x_N\}$, um evento acústico é descrito através da função geradora de sinal $f: \mathbb{R}^{m+1} \rightarrow \mathbb{R}^q$ que faz a computação de um sinal sonoro de q canais como uma função de tempo $f: s(t) = f(\vec{p}; t)$. Onde $\vec{p}$ é um vector de dimensão m e representa atributos acústicos que são parâmetros da geração de sinal ao qual é adicionado 1, um número real que representa a dimensão temporal. Adicionalmente, q representa o número de canais (altifalantes) disponíveis para a sonorização (para uma sonorização em *stereo*, $q=2$). A técnica de PMSon é então computada por:

$$s(t) = \sum_{i=1}^N f(g(\vec{x}_i), t), \quad (2.1)$$

onde $\mathbb{R}^d \rightarrow \mathbb{R}^m$ é a função de mapeamento do parâmetro.

Existem diversos tipos de funções de mapeamento com comportamentos distintos. As funções linear, sigmóide e exponencial, são apenas alguns exemplos de funções de mapeamento utilizadas na técnica de PMSon. Na sonorização de conjuntos de dados com uma série temporal, o uso de diferentes funções é bastante útil, permitindo ao ouvinte aplicá-las no mapeamento entre valores de dados e parâmetros acústicos como amplitudes e envolventes espectrais. Deste modo, o utilizador pode articular a sonorização no tempo de formas distintas (Hermann et al., 2011). A Figura 2.2 representa o mapeamento entre valores dos dados e um determinado parâmetro auditivo utilizando duas funções: linear e sigmóide.

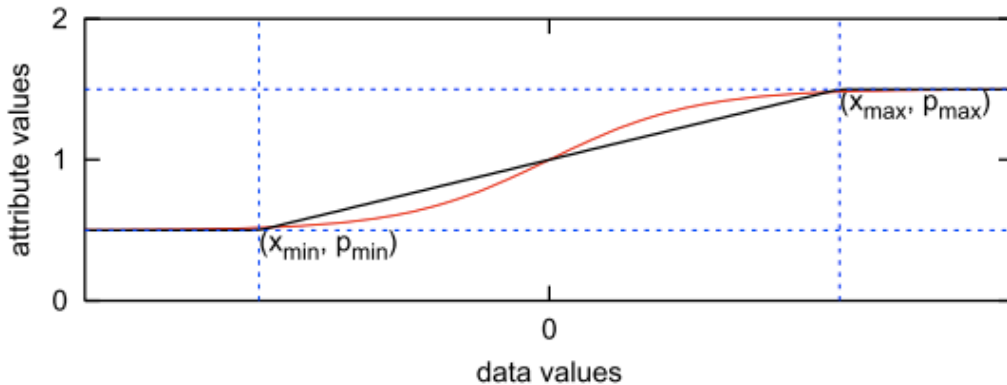


Figura 2.2 - Função de mapeamento do parâmetro - Representação do mapeamento entre valores de dados (x) e atributos acústicos (y) utilizando uma função linear (linha preta) e uma função sigmóide (linha vermelha). (Hermann, 2002).

A escolha da função de uma função para o mapeamento entre atributos de dados e parâmetros de síntese de som, representa apenas o primeiro passo no mapeamento. Adicionalmente, é necessário definir qual a sua (A) polaridade e (B) escala.

A - Polaridade

Polaridade é o termo usado para descrever a direcção do mapeamento dos dados para um parâmetro sonoro. Numa polaridade positiva, uma sequência de valores crescentes causa o aumento do parâmetro mapeado, no entanto, o utilizador pode querer mapear a mesma sequência de valores para a diminuição do valor do parâmetro, neste caso a polaridade é invertida. Ao definir a polaridade do mapeamento, o utilizador deve ter em conta qual a direcção que melhor comunica os dados. Se considerarmos o mapeamento entre a temperatura e a frequência de um sinal sonoro, certamente a maioria dos ouvintes achará óbvio que o aumento da temperatura cause um aumento na frequência. É com base nesta opinião predominante que a polaridade deve ser definida.

B - Escala

Na maioria dos casos, os valores dos dados sob análise não se encontram no âmbito de um determinado parâmetro sonoro. Logo, é necessário escalar os valores no domínio dos dados para o domínio do parâmetro auditivo. Novamente, o mapeamento da temperatura para a frequência de um sinal sonoro exemplifica bem este tipo de casos. Por muito extremos que sejam, os valores da temperatura nunca entram num âmbito de frequência audível (20 kHz - 20,000 Hz). Logo, para que as mudanças na frequência sejam perceptualmente significativas, é necessário escalar os valores dos dados sob análise para valores que causem alterações perceptualmente significativas no parâmetro mapeado.

2.4.3.2. Discussão

A escolha de um mapeamento que traduza de forma clara os dados sob análise é um processo complexo que depende em grande parte de aspectos perceptuais. Factores como a não-linearidade do ouvido humano, as interacções perceptuais e as diferentes resoluções auditivas, representam a origem dos desafios na escolha de um mapeamento eficaz (Hermann, 2002).

As interacções perceptuais e a não-linearidade do ouvido humano comprometem a qualidade perceptual dos atributos acústicos. Por exemplo, o utilizador pode mapear determinada característica dos dados para um atributo que a represente claramente (frequência, por exemplo), porém, variações ocorridas noutro atributo auditivo (duração, por exemplo) irão afectar a qualidade perceptual do atributo outrora eficaz (frequência). A não-linearidade é responsável por um fenómeno

semelhante, no entanto, as variações da qualidade perceptual do parâmetro auditivo não se devem às interacções com os restantes atributos, mas sim às variações ocorridas no mesmo. Este tipo de fenómenos representam o maior desafio na técnica de PMSon, contribuindo para o risco de ambiguidade da interpretação dos dados ou mesmo a sua incompreensão (Hermann, et al., 2011).

No sentido de contrariar esta tendência, foram criadas algumas linhas de orientação. Kramer (1994), apresentou alguns princípios organizacionais para a representação sonora de dados. Walker e Kramer (Walker & Kramer, 2005) apresentaram uma série de mapeamentos classificados de acordo com a sua eficácia (Intuitivo, *okay*, mau e aleatório) para a monitorização de processo de uma fábrica de cristais (simulada). Adicionalmente, existem ainda algumas linhas de orientação para a sonorização de gráficos e tabelas (Brown et al., 2003).

Para além das limitações perceptuais, estão as limitações técnicas impostas pelos sistemas existentes. As interacções de mapeamento são bastante úteis na “procura” de atributos acústicos eficazes para os dados a comunicar, no entanto, os sistemas existentes estão longe de promover interacções eficazes. Na Secção 3.2.2 deste documento, descrevemos com mais detalhe este tipo de interacções e a sua contribuição para um mapeamento eficaz.

Apesar de ser uma técnica versátil e com grandes capacidades na comunicação de dados multi-dimensionais, a diversidade de tipos dados aliada aos contextos onde a PMSon actua fazem de cada mapeamento, um caso único. Logo, apesar de úteis, as linhas de orientação existentes são incapazes de dar resposta às adversidades que surgem nos diferentes contextos da sua aplicação. Por esta razão, é justo dizer que o sucesso de um bom mapeamento depende: por um lado do ouvinte, que deve estar familiarizado com o mapeamento definido de modo a atribuir-lhe o significado correcto e compreender as interacções perceptuais entre os diferentes atributos acústicos, por outro lado, dos sistemas para a sonorização. Estes devem permitir o desenvolvimento de estratégias que superem os desafios ilustrados anteriormente, providenciando flexibilidade na sua arquitectura, permitindo a alteração de estratégias de mapeamento e a interacção para com o mesmo.

2.4.4. Model-Based Sonification (MBS)

Num ambiente real, o ser humano está apto a interagir com sistemas físicos e a extrair significado do som resultante dessa mesma interacção. Por exemplo, ao agitarmos uma garrafa de água, o som resultante está directamente relacionado com as propriedades físicas do objecto (plástico, vidro, metal, entre outros) e as suas dinâmicas (a quantidade de água, por exemplo). O mesmo acontece quando batemos numa superfície de modo a saber o seu material, o som ouvido codifica as propriedades físicas do objecto quando excitado, permitindo ao ouvinte identificar a sua composição. Hermann (2000) referiu-se a este “modo” de audição como “*analytical everyday listening*”.

A técnica de MBS é particularmente útil para a análise exploratória de dados (Secção 2.3.3) e foca-se na relação entre a resposta acústica do sistema em função da excitação sofrida, tirando partido da capacidade que o ser humano tem em extrair significado dessa mesma resposta. Para que esta capacidade cognitiva possa ser aplicada no domínio da sonorização de dados, é necessário recorrer a um modelo virtual (modelo de sonorização) capaz de evoluir no tempo e consequentemente gerar uma resposta acústica.

Segundo a metodologia apresentada por Hermann e Ritter (1999) o processo de desenvolvimento de um modelo de sonorização é essencialmente composto por quatro passos: a sua (A) configuração, (B) dinâmicas, (C) excitação, e as (D) variáveis auditivas.

2.4.4.1. Configuração

A configuração do modelo é o primeiro passo no desenvolvimento de um modelo de sonorização e consiste na definição dos seus elementos dinâmicos. Este passo, faz a “ponte” entre os dados multidimensionais no seu estado abstracto e estático para um domínio de um modelo virtual que quando excitado produz som (Hermann, 2011).

2.4.4.2. Dinâmicas

Uma vez escolhida a configuração do modelo, é necessário introduzir as dinâmicas que foram definidas na configuração. Estas, irão descrever a sua evolução temporal. Hermann (2011) descreve as dinâmicas de um modelo como “equações de movimento que descrevem as mudanças do estado do vector em função do tempo”. Isto é, como é que o próximo estado $\vec{s}(t + \Delta t)$ é computado a partir do estado actual $\vec{s}(t)$.

De modo a simular a interacção no mundo real, as dinâmicas são inspiradas em leis da física. Os modelos de sonorização existentes têm feito uso de sistemas de oscilação-acoplada (*mass-spring systems*), princípios físicos como a energia cinética e potencial ou princípios acústicos como meio de inspiração para as suas dinâmicas (Hermann, 2002). Um bom exemplo é o sonograma de dados (Secção 3.3), onde os pontos representados num espaço euclidiano contêm uma determinada massa e estão acoplados a uma mola (*spring*). O movimento e posição dos pontos (massas) é descrito em função da força exercida pela mola acoplada, para tal, as dinâmicas aplicadas neste modelo provêm da física e mecânica dos sistemas de oscilação-acoplada.

Uma vez que as dinâmicas do sistema ditam o comportamento do modelo quando excitado, que por sua vez produz som, este passo é um passo fundamental no desenvolvimento do modelo de sonorização. No entanto, como em qualquer modelo físico, estas dinâmicas só se manifestam quando é introduzida energia no sistema, por outras palavras, quando o sistema é excitado.

2.4.4.3. Excitação

Num ambiente real, um objecto físico encontra-se normalmente em estado de equilíbrio, isto é, a sua energia atinge um valor mínimo e lá permanece até sofrer uma excitação. O mesmo acontece no modelo de sonorização, encontra-se inicialmente num estado de equilíbrio onde não produz qualquer som, no entanto, ao ser excitado, é introduzida energia no sistema (Hermann, 2011). Este reage física e acusticamente de acordo com a excitação dada, as dinâmicas subjacentes e as variáveis acústicas.

Existem diversos tipos de interacção com um modelo. Desde interacções vulgares como premir ou arrastar o cursor, como acontece no sonograma de dados por exemplo, até modos de interacção mais tangíveis que recorrem a controladores externos que permitam agitar, espremer e deformar um modelo (Hermann et al., 2002).

Frequentemente adaptamos a nossa interacção para com um objecto físico, em função da sua configuração e dinâmicas. Por exemplo, ao vermos um tambor, temos a clara noção que a melhor forma de excitar a membrana é percutindo-a, ao invés de a friccionar por exemplo. Esta intuição também deve surgir perante o modelo de sonorização, para tal, ao definir o tipo de excitação, é necessário ter uma noção clara de qual o tipo de interacção adequado à estrutura e dinâmicas definidas para o modelo em causa. Na secção 3.3 deste documento são descritos alguns dos métodos de interacção para com um modelo de sonorização.

2.4.4.4. Variáveis acústicas

Para que o modelo responda acusticamente a uma dada interacção, é necessário relacionar as suas dinâmicas com atributos acústicos. A forma mais simples de traduzir as dinâmicas para som é “transpondo-as” para uma frequência audível, como acontece na técnica de audificação (ver 2.4.1). No entanto, a prática mais comum é relacionar diferentes variáveis do modelo com diferentes atributos acústicos.

Uma vez que o modelo é análogo a um objecto físico, os elementos definidos na configuração podem e devem influenciar a escolha do material sonoro. Neste sentido, a síntese por modelação física tem um forte potencial para fortalecer a relação entre a excitação e a resposta acústica do sistema. No entanto, maior parte dos modelos aplicam métodos de síntese que também são usados na técnica de PMSon (2.4.3). Nestes casos, o modelo serve de intermediário entre o utilizador e o mapeamento definido para com o algoritmo de síntese.

Em suma, para o desenvolvimento de um modelo de sonorização é necessário definir:

- Configuração - Quais os elementos dinâmicos que compõem o modelo?

- Dinâmicas - Quais as dinâmicas que descrevem a evolução temporal do modelo?
- Excitação - De que modo é feita a interacção para com o modelo?
- Variáveis acústicas - Quais as variáveis que contribuem para a geração da resposta acústica do modelo?

2.4.4.5. Conclusão

Este Capítulo cobriu de um modo geral o actual contexto da sonorização de dados. Foram revistas as suas áreas de aplicação e técnicas existentes, com particular interesse no uso da sonorização na análise exploratória de dados, no meio artístico e descritas as técnicas de PMSon e MBS (Secções 2.3, 2.4). Neste sentido, foi ilustrado o processo de mapeamento da técnica de PMSon (Secção 2.4.3) e as suas implicações, bem como, a metáfora subjacente a um modelo de sonorização para MBS (Secção 2.4.4).

A metodologia apresentada por Hermann e Ritter (1999) ilustra os passos necessários no desenvolvimento de um modelo de sonorização. A configuração, onde o utilizador define os elementos dinâmicos do sistema, isto é, as suas propriedades físicas. As dinâmicas do modelo, que descrevem o comportamento do modelo no tempo e a excitação, onde é definido o modo de interacção para com o modelo. Adicionalmente, é necessário definir a “ligação” entre as variáveis do modelo e os atributos de síntese bem como o papel do ouvinte na sonorização.

O sucesso do modelo de sonorização, depende da estratégia adoptada em cada passo da metodologia descrita anteriormente. À semelhança da construção de um instrumento musical, onde cada fase tem como objectivo final melhorar a sua ergonomia e qualidade acústica, cada fase do desenvolvimento de um modelo deve ter sempre presente o objectivo de comunicar os dados sob análise.

Hermann (2002), aponta diversas vantagens na técnica de MBS em relação à PMSon. As que mais se destacam são, por um lado, o facto de fazermos uso de capacidades auditivas que usamos no nosso quotidiano, por outro, o facto do modelo conter poucos parâmetros, permitindo ao ouvinte uma interpretação intuitiva com um período de aprendizagem menor. O mesmo não se verifica na técnica de PMSon, onde o mapeamento, na maioria dos casos é bastante complexo e requer um maior período de aprendizagem.

Adicionalmente, outro factor que contribui para uma aprendizagem mais eficaz é a reutilização do modelo. Uma vez definido, o modelo pode ser usado na sonorização de diferentes tipos de dados, evitando deste modo, a necessidade de um novo mapeamento para cada sonorização. Este é um aspecto importante, que permite ao ouvinte criar uma relação próxima com o modelo em causa.

Actualmente, não existem princípios para a avaliação de um modelo de sonorização. Porém, a avaliação da sua performance em termos de fluxo, fadiga e

nível de compreensão são aspectos referidos na agenda de investigação proposta por Hermann (2011). Adicionalmente, o autor refere ainda a necessidade de desenvolvimento de uma *toolbox* de modelos de sonorização que resultem da literatura existente no campo.

3. Sonorização Interactiva

3.1. Definição

Sonorização interactiva é “a disciplina de exploração de dados através da manipulação interactiva da transformação de dados para som” (Hermann & Hunt, 2004). Nesta prática, o utilizador opera num ciclo contínuo entre o sistema e o resultado sonoro, remetendo o ouvinte para um processo de aprendizagem fundamental para uma mais profunda compreensão dos dados sob análise.

Existem diferentes tipos de interacção com um sistema de sonorização, estes, variam de acordo com a técnica utilizada e o contexto da sua aplicação. Nesta secção, são apresentados os cenários de interacção existentes nas técnicas de PMSon e MBS (Secções 2.4.3, 2.4.4).

3.2. Interacção em *Parameter Mapping Sonification* (PMSon)

Existem três tipos distintos de interacção na técnica de PMSon:

- A. Interacções de Navegação;
- B. Interacções de Mapeamento;
- C. Selecção interactiva de dados;

3.2.1. Interacções de Navegação

As interacções de navegação, permitem ao utilizador navegar na sonorização obtida através de funções como reproduzir, interromper, avançar ou recuar, bem como seleccionar uma área específica no conjunto de dados. Apesar de não afectar directamente os atributos acústicos e permitir interacções rudimentares (Hunt & Hermann, 2011) o nível de navegação desempenha um papel importante na análise dos dados no domínio do tempo, do mesmo modo que tem uma função prática, permitindo a selecção e reprodução de regiões que manifestem maior interesse.

Este tipo de interacção é bastante comum nos sistemas existentes, no entanto, poucas oferecem as mesmas possibilidades. Por exemplo, as funções triviais (reproduzir, interromper, avançar, recuar) estão presentes em aplicações como *Sonification Sandbox*, *Interactive Sonification Toolkit (IST)* e *Personify* (Walker and Cothran, 2003; Pauletto & Hunt, 2004; Barrass, 1995), no entanto, apenas *IST* e *Personify* possibilitam seleccionar uma região específica do conjunto de dados e proceder à sua sonorização. No caso de *IST*, a região é seleccionada sob uma área cujo comprimento representa a duração total da sonorização, em *Personify*, cujo propósito é sonorizar dados sísmicos e dados provenientes de EEG, são apresentados três modos de interacção: no primeiro modo, à semelhança de *IST* o utilizador uma região com o cursor e ainda reproduzir, parar, avançar e recuar na sonorização. No segundo modo, orientado para a sonorização de dados sísmicos, a navegação é feita através de uma representação geográfica das principais estações sísmicas existentes. O utilizador navega no mapa através do cursor, podendo seleccionar a estação pretendida e ouvir como é que determinado evento sísmico foi medido naquela localização. O terceiro e último modo é orientado para a sonorização de dados EEG, para tal, o utilizador dispõem de uma representação visual da posição de cada eléctrodo em relação à cabeça humana. O utilizador pode seleccionar qual o ou os eléctrodos que quer “ouvir” num dado momento da sonorização.

3.2.2. Interacções de Mapeamento

As interacções de mapeamento permitem ao utilizador alterar o mapeamento adoptado e os parâmetros relacionados com o mesmo (Hunt & Hermann, 2011). Ao fazê-lo em tempo real, isto é, à medida que a sonorização é ouvida, o utilizador refina o mapeamento e analisa as alterações acústicas provocadas pelas suas acções. Sendo o mapeamento a essência da PMSon (Secção 2.4.3), este ciclo interactivo é fundamental para o desenvolvimento de um mapeamento eficaz na comunicação dos dados sob análise.

No entanto, a maioria dos sistemas existentes restringe este tipo de interacção, impossibilitando que esta seja feita em tempo-real. Esta lacuna está presente em sistemas como *Sonart*, *Sonifyer*, *Personify*, *Sonification Sandbox*, *Listen*, *IST* e *Sonipy* (Yeo et al., 2004; Dombois, 2008; Barrass, 1995; Walker & Cothran, 2003; Wilson & Lodha, 1996; Pauletto & Hunt, 2004; Worrall et al., 2007). Nalguns casos, como o de *Sonipy*, um *toolkit* para sonorização desenvolvido em módulos de Python², e *IST*, desenvolvido em *Pure Data*³, é possível adicionar modos de interacção em tempo real devido à abertura do sistema. Porém o desenvolvimento de um interface

² www.python.org

³ www.puredata.info

intuitivo para as interacções de mapeamento é um processo complexo e que requer bons conhecimentos das linguagens de programação envolvidas (Hunt & Hermann, 2011).

3.2.3. Selecção interactiva de dados

Como o próprio nome indica, a selecção interactiva de dados permite ao utilizador seleccionar qual o subgrupo dos dados sob análise a serem sonorizados (Hunt & Hermann, 2011). Para este tipo de interacção é necessário um interface que represente visualmente os dados do mesmo modo que permita a sua selecção.

Sonic Brushing (Hunt & Hermann, 2011) foi o termo dado por a um interface onde os dados são representados através de um gráfico de dispersão de duas dimensões. Neste, o utilizador selecciona os pontos num plano cartesiano de acordo com a posição de um círculo (de diâmetro variável). O sistema detecta quais os pontos que se encontram dentro do círculo e procede à sua sonorização, caso não existam pontos dentro do círculo não ocorre sonorização. Riedenklau et al. (2010) recorreram a este interface no desenvolvimento de um sistema que permite a um deficiente visual ouvir um gráfico de dispersão. Dado a sua condição, o utilizador interage com o sistema ao mover um objecto físico sobre uma superfície interactiva. A posição do objecto, dita a posição do círculo de selecção (*brush*).

Apesar de ser bastante útil na compreensão dos dados, particularmente na compreensão das distâncias e agrupamentos, não se encontra presente nos sistemas existentes. A sua aplicação tem sido feita num âmbito académico, onde o interface é implementado numa determinada linguagem de programação e orientado para uma determinada aplicação.

3.3. Interacção em *Model Based Sonification* (MBS)

Enquanto na PMSon (Secção 2.4.3) a componente interactiva pode ser encarada como uma função adicional, na técnica de MBS (Secção 2.4.4) a componente interactiva é parte constituinte do sistema de sonorização (Hunt & Hermann, 2011).

Uma vez que a técnica de MBS, faz uso das nossas capacidades em extrair significado a partir de sons resultantes de interacção com objectos físicos, a componente interactiva deve oferecer cenários semelhantes a este tipo de interacção. De seguida, iremos descrever os actuais modos de interacção para com um modelo de sonorização.

Grande parte dos sistemas, recorre ao uso do rato como meio de interacção. Por exemplo, o sonograma de dados (Hermann & Ritter, 1999), onde os dados são representados por pontos (massas) associados a molas (*springs*), ao premir o cursor

sob o interface o utilizador provoca uma “onda de choque”. Esta tem origem nas coordenadas onde o cursor foi premido e propaga-se no plano a uma velocidade contínua. O sistema é excitado quando a onda de choque intersecta a posição de cada ponto (massa), introduzindo energia nas suas equações de movimento e fazendo-o vibrar em torno da posição da mola associada (Fig. 3.1). O mesmo acontece no modelo de sonorização de cristalização de dados (Hermann & Ritter, 2005), o utilizador pressiona o cursor sobre um gráfico de dispersão de duas dimensões que desencadeia a sonorização dos pontos que estão num raio previamente definido.

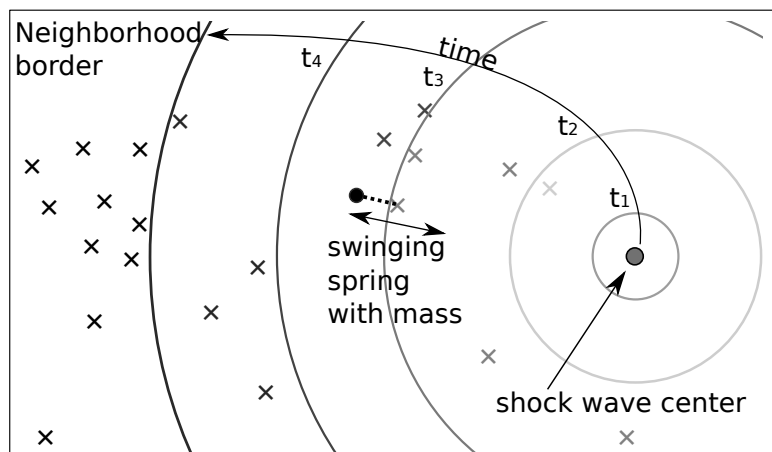


Figura 3.1 - Sonograma de dados - Propagação da onda de choque no sonograma de dados. Os pontos de dados são excitados à medida que a onda de choque intersecta a sua posição (Hermann & Ritter, 1999).

Na sonorização de *Growing Neural Gas* (Hermann & Ritter, 2004), cujo objectivo é aplicar o algoritmo de GNG em conjuntos de dados com grandes dimensões de modo a reduzir a sua complexidade ao mesmo tempo que preserva a sua estrutura, o utilizador dispõem de um modelo virtual composto por nódulos, que representam neurónios da rede GNG, ligados entre si através de *edges* (Fig. 3.2). Quanto maior for o número de ligações de um nódulo, maior é a dimensionalidade dos pontos de dados subjacentes. Neste modelo, cada nódulo contribui com um som cujos atributos acústicos variam de acordo com o seu comportamento quando excitado e número de *edges* associados.

A excitação ocorre quando o utilizador prime sobre um nódulo, introduzindo-lhe energia que se traduz numa força nas suas equações de movimento. O autor compara esta interacção à acção de bater uma barra metálica, onde energia cinética do sistema altera pontualmente. Uma vez que os nódulos se encontram ligados entre si, a energia introduzida num nódulo é transferida para os restantes, permitindo ao ouvinte compreender dimensionalidade intrínseca dos dados sob análise (Hermann & Ritter, 2004). Mais tarde, este modelo foi adaptado de forma a sofrer excitação táctil (Kolbe et al., 2010). Dispondo de uma superfície multitoque, o utilizador excita o modelo ao “tocar” no nódulo pretendido (Fig. 3.3).

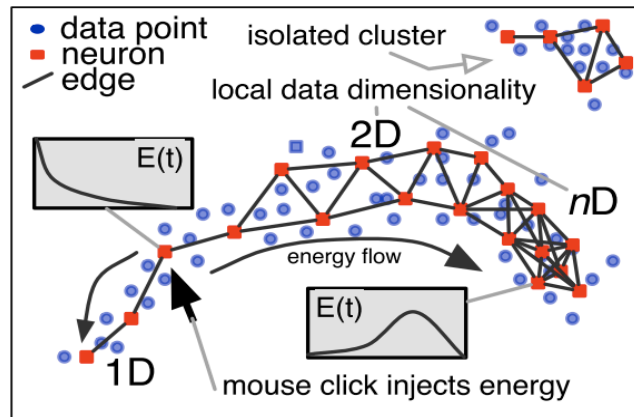


Figura 3.2 - Modelo de sonorização de GNG - O utilizador introduz energia do modelo ao pressionar o rato sobre um neurónio (massa). A energia é transferida para os restantes neurónios através dos edges (Hermann & Ritter, 2004).

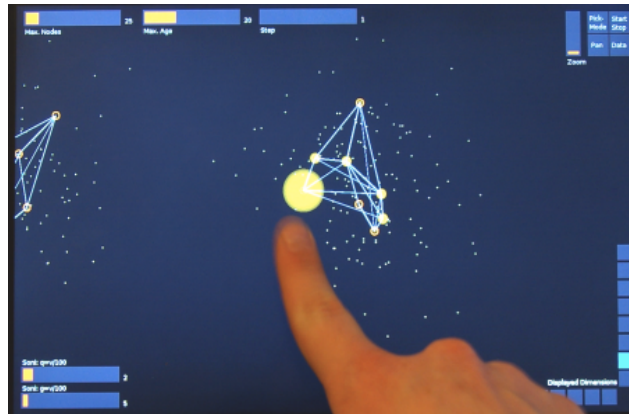


Figura 3.3 - Adaptação a superfície multitouch - O modelo de sonorização de GNG adaptado a uma superfície multitouch de modo a sofrer uma interacção táctil (Kolbe et al., 2010).

Outro modelo que faz uso do rato como meio de interacção é o modelo de sonorização da curva principal (Hermann et al., 2000). Neste, ao invés de um premir o cursor, o utilizador interage com o modelo ao arrastar o cursor sobre a curva principal dos dados sob análise (Fig. 3.4). Ao intersectar pontos de dados que se encontram directamente sob a trajectória da curva, o sistema procede à sonorização dos mesmos.

Todavia, apesar de úteis, este tipo de interacções estão longe das interacções que temos com objectos físicos. Ao usarmos as mãos para interagir com um objecto, temos a possibilidade de o fazer de forma contínua, agitando ou deformando o objecto por exemplo. No sentido de aplicar este tipo de interacções contínuas para com um modelo de sonorização, Hermann et al., (2002), desenvolveram o interface “*Audio-haptic Ball*” (Fig. 3.5). Com uma forma arredondada para ser facilmente manuseado com mão e equipado com um acelerómetro e sensores que medem a

pressão exercida pelos dedos do utilizador, o interface permite interações como bater, arranhar, espremer, agitar e deformar o modelo de sonorização associado.

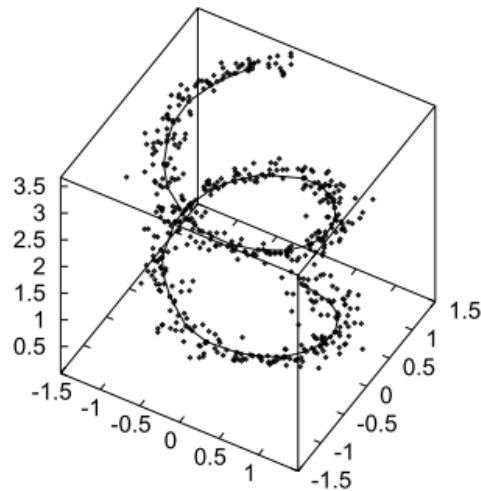


Figura 3.4 - Modelo de sonorização da curva principal. (Hermann et al., 2000)

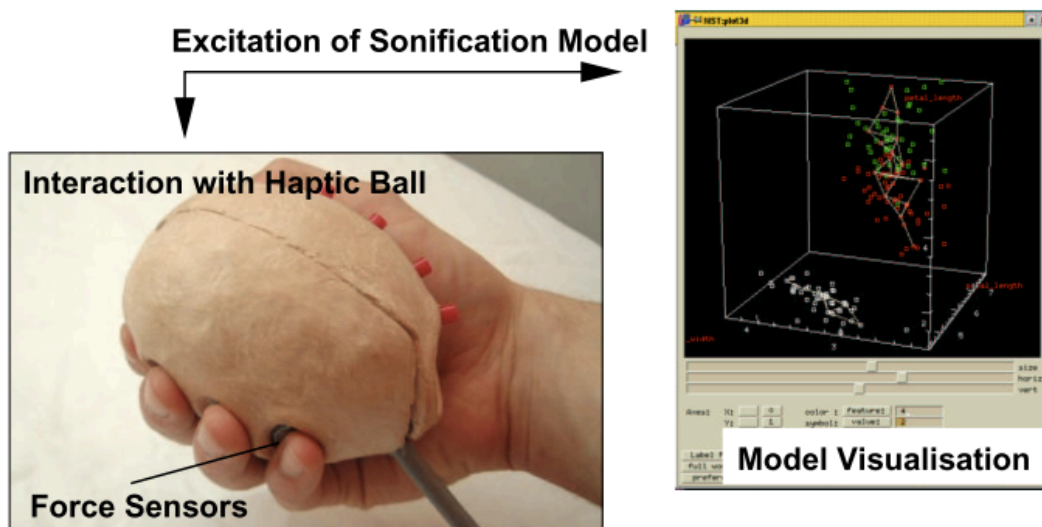


Figura 3.5 - Interface *Audio-haptic Ball* - *Audio-haptic Ball* e respectivo modelo. (Hermann et al., 2002)

3.4. Conclusão

Neste capítulo, foram revistos os modos de interacção na técnica de PMSon e MBS. Foram também referenciados alguns sistemas e modelos que permitem a sua utilização.

A componente interactiva na técnica de PMSon (Secção 2.4.3), permite ao utilizador navegar interactivamente na sonorização, interagir com o mapeamento e parâmetros relacionados e seleccionar interactivamente os dados a sonorizar. Porém, à excepção da navegação interactiva, os sistemas existentes estão longe de promover este tipo de funções de um modo eficaz. Isto deve-se, por um lado, à ausência de um interface eficaz para as interacções de mapeamento (ver 3.2.2), Por outro lado, à inexistência de sistemas que permitam a selecção interactiva de dados. Apesar da sua utilidade na compreensão dos dados sob análise, interfaces como *Sonic Brushing* (Hunt & Hermann, 2011) têm sido aplicados unicamente no âmbito académico da sonorização interactiva.

O mesmo acontece com os modelos de sonorização. Estes resultam de um processo de investigação e são detalhadamente apresentados à comunidade, no entanto, a sua aplicação reduz-se ao meio académico. Isto é, não existe nenhum sistema disponível para a comunidade que forneça um modelo de sonorização.

Por tudo isto, estamos perante uma lacuna por parte dos sistemas na implementação dos diferentes modos de interacção para com a técnica de PMSon e modelos de sonorização para MBS .

4. SonData

No sentido de colmatar as lacunas descritas no Capítulo anterior, foi desenvolvido SonData, um *toolkit* para a Sonorização Interactiva nas técnicas de PMSon e MBS (Secções 2.4.3, 2.4.4). Neste capítulo, é descrita a metodologia adoptada para o desenvolvimento de SonData bem como as suas funcionalidades. Adicionalmente é feita uma conclusão em torno do trabalho desenvolvido e ainda uma sugestão para estudos futuros.

4.1. Introdução

SonData, é uma biblioteca para Max/MSP⁴ que consiste numa série de módulos que oferecem soluções para procedimentos e funcionalidades no domínio da sonorização interactiva.

Os módulos encontram-se organizados categoricamente de acordo com a sua função, esta, tem em conta as necessidades que o utilizador encontra no processo de desenvolvimento de uma sonorização interactiva.

Tendo em conta o constante aparecimento de novos cenários de interacção na sonorização de dados, pretende-se que SonData, ao contrário de um *toolkit* “fechado”, seja aberto e expansível, permitindo a integração de novos módulos e funcionalidades que se revelem importantes no contexto da sonorização interactiva. No sentido de reforçar esta intenção, a sua integração foi feita através da *framework* Jamoma⁵.

Para além dos objectos nativos do Max e componentes Jamoma, SonData recorre a algumas bibliotecas e objectos externos, sendo estas:

A. Biblioteca FTM & Co (http://ftm.ircam.fr/index.php/Main_Page) - Consiste numa biblioteca de objectos externos cujo principal objectivo é auxiliar a manipulação de estruturas de dados complexas, promovendo o uso de matrizes, sequências, dicionários, entre outros objectos.

⁴ www.cycling74.com

⁵ www.jamoma.org

B. Abstracções mc.PMPD (<http://mathieu.chamagne.free.fr/max.htm>) - Série de objectos externos que promovem a criação de modelos físicos no ambiente MaxMSP e Pure Data.

C. Objecto Resonators~ e SDIF-Buffer (cnmat.berkeley.edu/downloads) - Objecto externo que permite a criação de um modelo de ressonância. Este, pode ser definido manualmente, ou através de um ficheiro SDIF.

D. Objecto *munger~* (music.columbia.edu/percolate/) - Objecto externo que implementa síntese granular num determinado sinal áudio.

4.2. Jamoma *framework*

Jamoma, é uma *framework* modular para o desenvolvimento de módulos de alto nível no ambiente gráfico de programação Max/MSP/Jitter (Place & Lossius, 2006). Esta, surgiu com a premissa de facilitar a partilha de *patches* entre a comunidade, promovendo uma abordagem sistemática e estandardizada à programação em Max/MSP/Jitter. Esta é promovida através do *Jamoma Interface Guide* (JIG) como uma recomendação para questões comuns relacionadas com a construção, manipulação e interface de um *patch* de Max. No entanto, é importante ter em conta que a sua abordagem sistemática não introduz restrições ao desenvolvimento e expansão de cada módulo.

O utilizador dispõem de uma colecção de módulos de alto nível que operam nos domínios de processamento de dados, áudio e vídeo. Para além disto, existe ainda uma uma colecção de objectos externos e abstracções (*patches* de Max que se comportam como objectos externos) para o desenvolvimento de um módulo. Estes de acordo com a terminologia da *framework*, denominam-se componentes (Place & Lossius, 2006).

Cada módulo é constituído por três camadas distintas: O (A.) interface, que providencia *feedback* visual sobre o estado do módulo, (B.) a camada de declaração de parâmetros e mensagens, composta pelos componentes *jcom.parameter* e *jcom.message*, e ainda (C.) o algoritmo que descreve a tarefa lógica do módulo. Este, é frequentemente implementado sob a forma de um *subpatch* guardado separadamente.

Outros factores que contribuíram para a escolha da *framework* Jamoma, são:

A. Comunicação entre módulos através do protocolo *Open Sound Control* (OSC; Wright & Freed, 1997), esta, segundo a convenção proposta pelo Jamoma é feita através do primeiro *inlet* e *outlet* de cada módulo;

B. Geração automática de documentação HTML sobre um determinado módulo;

C. Biblioteca externa composta por módulos desenvolvidos pela comunidade (Jamoma UserLib). Por tudo isto, Jamoma foi a *framework* adoptada para a implementação de SonData.

4.3. Convenções gerais

De acordo com a terminologia da *framework* Jamoma, inerente ao *toolkit* aqui apresentado, a nomeação de um módulo é composta por “jmod.” seguida do nome do módulo. No sentido de promover alguma distinção entre módulos nativos do Jamoma e os introduzidos por este *toolkit*, ao prefixo nativo “jmod.”, acrescentámos “son.”. Deste modo, um módulo de SonData é nomeado de acordo com a sua função e sempre precedido por “jmod.son.”. Caso seja um módulo de processamento ou geração de um sinal áudio, o nome do módulo é sucedido por “~”.

Adicionalmente, tendo em conta a comunidade de sonorização interactiva e a sua contribuição para futuras expansões de SonData, todos os termos, funções, parâmetros e documentação dos módulos encontram-se em inglês.

4.4. Arquitectura

Usufruindo da arquitectura promovida pelo Jamoma, SonData é constituído por uma série de módulos organizados categoricamente de acordo com a sua função. Um módulo, não é mais do que um *patch* de Max previamente programado e embebido num interface gráfico que permite ao utilizador manipular intuitivamente as suas funções e parâmetros.

Cada módulo é representado por um objecto rectangular de cor cinzenta que opera num *patch* de Max. A sua aparência geral é ilustrada na Figura 4.1, porém, os controlos disponíveis para o utilizador variam de acordo com a sua função.

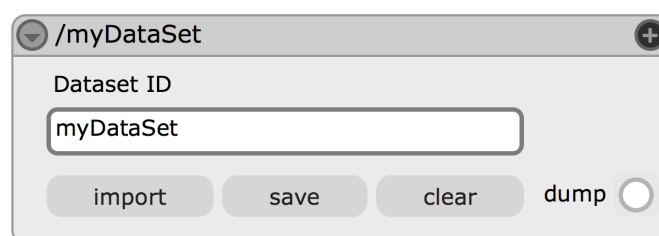


Figura 4.1 - Aparência de um módulo de SonData.

4.4.1. Estrutura de um módulo

A estrutura de um módulo de SonData, é composta por duas componentes distintas: o interface, onde o utilizador controla os parâmetros relativos à sua função e os componentes internos que implementam o algoritmo que providencia essas mesmas funções. De seguida, são descritas cada uma destas componentes.

4.4.1.1. Interface

O interface de um módulo é constituído por duas componentes principais: o (A) Cabeçalho, e (B) o corpo do módulo. Adicionalmente contém ainda as entradas (*inlets*), que se situam na parte superior do módulo e as saídas (*outlets*) na parte inferior.

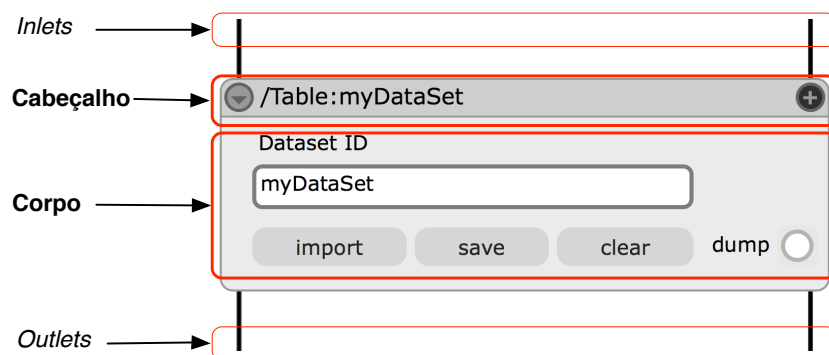


Figura 4.2 - Componentes do interface de um módulo.

A. Cabeçalho



Figura 4.3 - Cabeçalho de um módulo.

O cabeçalho de um módulo (Fig. 4.3) é constituído por três secções distintas. A primeira secção, é composta pelo menu do módulo que é acessível ao pressionar com o rato no canto superior esquerdo do mesmo. Este permite ao utilizador funções como, guardar e importar *presets* do módulo no formato XML, aceder aos componentes internos do módulo (ver 4.4.1.2), abrir o *patch* de ajuda e documentação HTML, entre outras. Este menu é providenciado pelo Jamoma e encontra-se descrito em detalhe em (Place & Lossius, 2006).

A segunda secção, é composta pelo nome do módulo. Como referido na Secção 4.3, ao nome de cada módulo, precede-se “jmod.son.”, porém, por motivos de

clareza, este (“jmod.son.”) encontra-se omitido no cabeçalho. Adicionalmente, devido à convenção existente no protocolo de comunicação OSC, ao nome do módulo, o Jamoma acrescenta automaticamente “/”.

A terceira secção é composta por pequenos botões situados na área direita do cabeçalho. Estes, de acordo com a terminologia do Jamoma, denominam-se de *widgets* e a sua presença varia consoante a função do módulo. A Figura. 4.4, representa os *widgets* presentes entre os vários módulos de SonData.



Figura 4.4 - Widgets -(da esquerda para a direita: *gain*, *mix*, *mute*, *bypass*, *panel*)

Em módulos onde ocorra o processamento ou geração de um sinal áudio, *widgets* como: (A) *gain*, (B) *mix*, (C) *mute*, e (D) *bypass*, providenciam funções bastante úteis, sendo:

- A. *Gain* - Como o próprio nome indica, permite ao utilizador ajustar o ganho do sinal de saída do módulo.
- B. *Mix* - Este *widget* opera em módulos que processem um dado sinal áudio, permitindo ajustar a relação entre o sinal de entrada e o sinal de saída (resultante do processo aplicado).
- C. *Mute* - Este *widget* tem apenas um estado (activo ou inactivo). Por defeito encontra-se inactivo, quando activado, anula o sinal de saída do módulo. Adicionalmente, desliga o processamento do módulo, permitindo assim poupar recursos computacionais.
- D. *Bypass* - Tem a mesma função que Mute, no entanto, está presente em módulos que recebam um sinal audio. Quando activo, o processamento do módulo é desligado e apenas é ouvido o sinal de entrada.

Adicionalmente, existe ainda o *panel*. Ao ser pressionado, este *widget* “abre” uma nova janela (*panel*) que contém um interface de controlo para além do corpo do módulo.

B. Corpo

O corpo do módulo representa a área onde estão disponíveis os controlos relativos aos seus parâmetros. (Fig. 4.2). Em alguns módulos, para além dos controlos presentes no corpo do módulo, o utilizador pode aceder a outras funções através do envio de uma mensagem para os *inlets*. Toda a informação relativa ao controlo de cada módulo está presente na sua pagina de referência (Secção 4.5).

4.4.1.2. Componentes internos

A segunda componente de um módulo é constituída pelos seus componentes internos. Podemos dividir esta componente em duas camadas distintas: (A) os componentes Jamoma, e o (B) algoritmo.

A. A primeira camada (Fig. 4.5) é constituída componentes da *framework* Jamoma que fazem a comunicação entre os controlos disponíveis e o algoritmo. A comunicação é feita remotamente através do componente “jcom.hub”, providenciado pela *framework* Jamoma.

B. A segunda camada (Fig. 4.6) é composta pelo algoritmo responsável pela função do módulo.

A estrutura interna de um módulo de Jamoma e a sua comunicação são descritas em detalhe em (Place & Lossius, 2006).

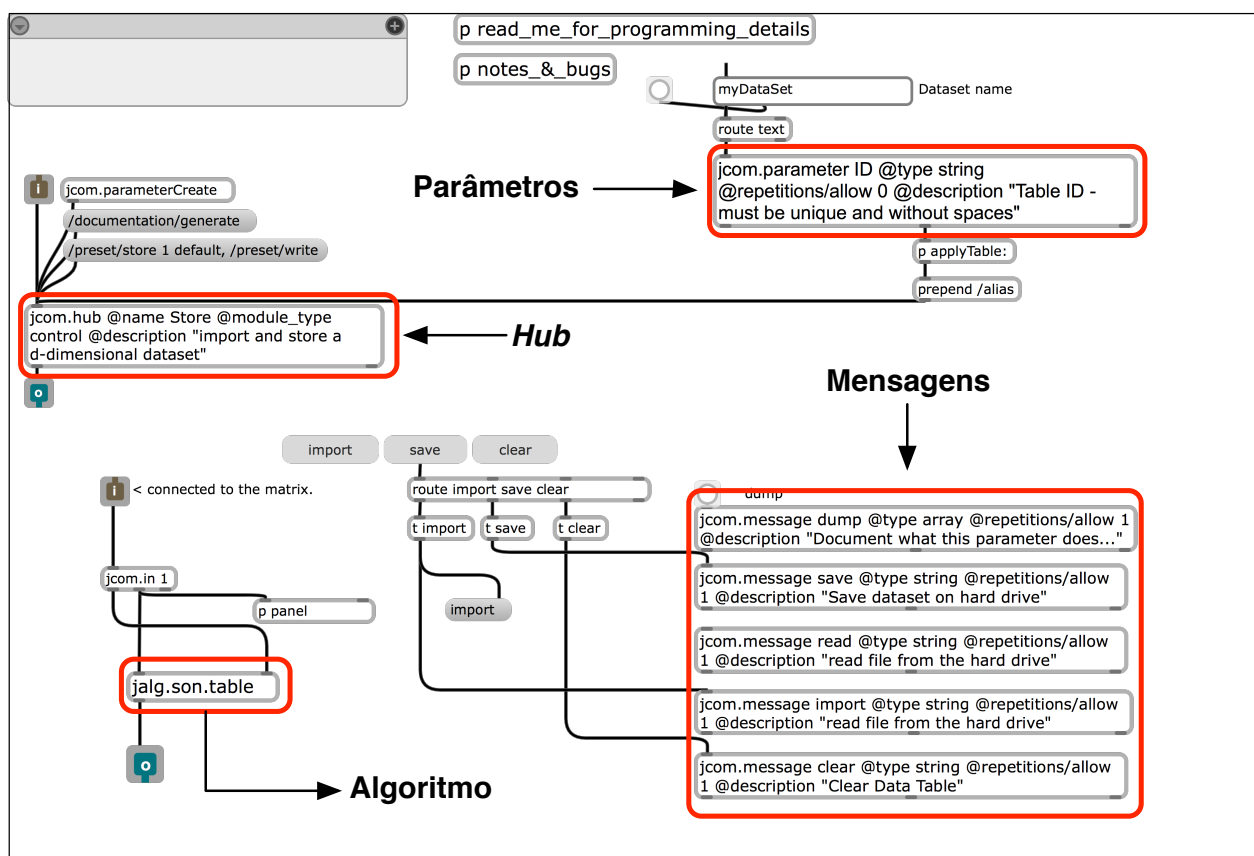


Figura 4.5 - Primeira camada interna de um módulo.

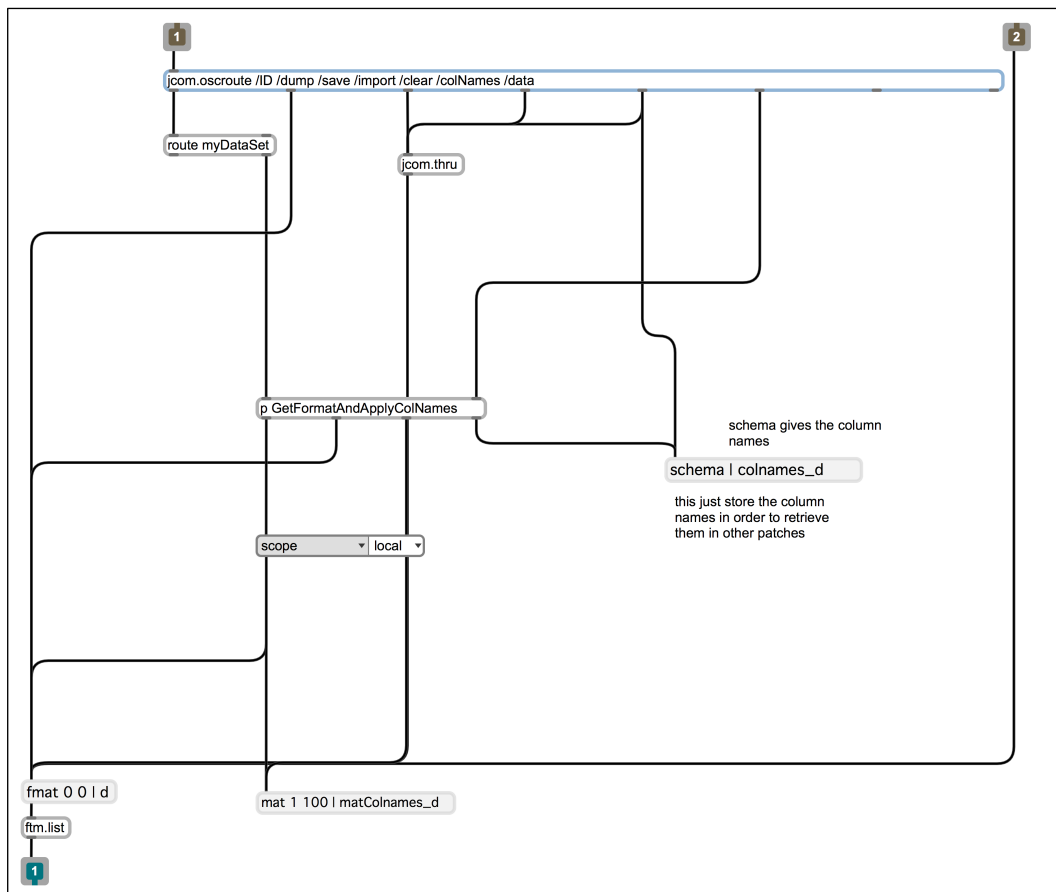


Figura 4.6 - Segunda camada interna de um módulo.

4.5. Documentação

Cada módulo de SonData é documentado por um (A) *patch* de ajuda (Fig. 4.7), e uma (B) página de referência (Fig. 4.8).

A. *Patch* de ajuda

O *patch* de ajuda é uma representação de todas ou das funções principais de um módulo. Este, como acontece com os objectos de Max, surge quando o utilizador pressiona o rato sobre o módulo mantendo pressionada a tecla “*alt*”.

B. Página de referência

Para além do *patch* de ajuda, cada módulo contém uma página de referência no formato HTML. Esta descreve a função do módulo, bem como as propriedades de parâmetros e mensagens que compõem o mesmo.

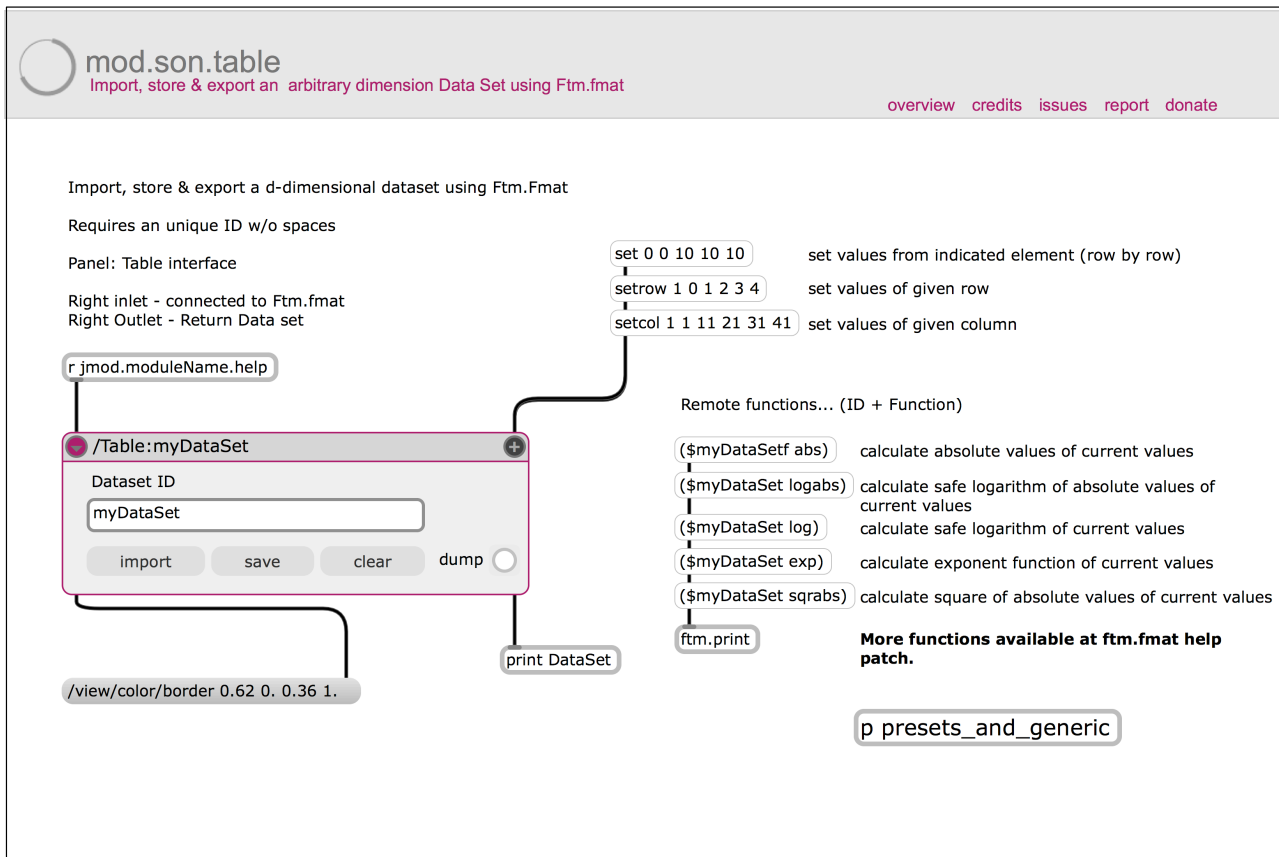


Figura 4.7 - Patch de ajuda de um módulo.



mod.son.table

import and store a d-dimensional dataset

[Table of Contents](#) |
 [Index of Modules](#) |
 [Credits](#) |
 [Donate](#) |
 [Jamoma.org](#)

Configuration

Module Type: control

Algorithm Type: default

Interface Size: 300x105 pixels

Number of signal inlets: 1

Number of signal outlets: 0

Parameters

/name	/type	/range/bounds	/range/clipmode	/ramp/drive	/ramp/function	/dataspace	/dataspace/unit/native	/repetitions/allow	/description
ID	string	N/A	none	none	linear	none	none	0	Dataset ID - must be unique and without spaces
colNames	generic	0.000000 1.000000	none	none	linear	none	none	0	set name for each column
view/freeze	boolean	0 1	none	none	linear	none	none	0	Turn off the updating of user interface elements when parameter change. This may be done to conserve CPU resources.
view/highlight	string	N/A	none	none	linear	none	none	0	Highlight the module with a color tint such as red, green, or similar.

Messages

/name	/type	/range/bounds	/range/clipmode	/ramp/drive	/ramp/function	/dataspace	/dataspace/unit/native	/repetitions/allow	/description
clear	string	N/A	none	none	linear	none	none	1	Clear Data Table
csv	boolean	0 1	none	none	linear	none	none	0	check this in before import an csv file
documentation/generate	generic	0.000000 1.000000	none	none	linear	none	none	1	Generate a html documentation page for this module and save it to disk. The argument 'tex' creates a Latex document.
dump	array	N/A	none	none	linear	none	none	1	Dump dataSet
import	string	N/A	none	none	linear	none	none	1	read file from the hard drive
init	none	N/A	none	none	linear	none	none	1	Initialize a module completely to the original state.
preset/clear	none	N/A	none	none	linear	none	none	1	Clean all presets, providing a blank slate for saving new presets.
preset/copy	array	N/A	none	none	linear	none	none	1	Create a new preset (2nd argument) by copying the contents of another preset (1st argument)
preset/default	none	N/A	none	none	linear	none	none	1	Open the default preset file and recall the first preset in that file.
preset/dump	none	N/A	none	none	linear	none	none	1	Dump all preset names.
preset/interpolate	array	N/A	none	none	linear	none	none	0	Interpolate between two named presets (argument 1 and 2) using a ratio (float in the range [0.0, 1.0]) specified as the third argument.
preset/mix	array	N/A	none	none	linear	none	none	0	Mix list of pairs of (preset name, mix value) using a ratio (float).
preset/post	none	N/A	none	none	linear	none	none	1	Post all presets to the Max window.
preset/read	generic	0.000000 1.000000	none	none	linear	none	none	1	Open an xml-preset file and recall the first preset in that file. An optional argument defines the file to open.
preset/recall	generic	0.000000 1.000000	none	none	linear	none	none	1	Recall a preset by number - you can also choose presets from the module menu.
preset/store	array	N/A	none	none	linear	none	none	1	Store a preset by number in memory. All presets present in memory will be written to disk when you send a /preset/write message to the module.
preset/storecurrent	none	N/A	none	none	linear	none	none	1	Store on the last recalled or stored preset
preset/storenext	none	N/A	none	none	linear	none	none	1	Store a preset in the next preset slot. Handy so that you do not need to specify a preset number manually.
preset/write	generic	0.000000 1.000000	none	none	linear	none	none	1	Write an xml-preset file to disk. An optional argument defines the file to open.
preset/writesavein	none	N/A	none	none	linear	none	none	1	Write on same xml-preset file.
save	string	N/A	none	none	linear	none	none	1	Save dataset on hard drive
view/color/border	array	N/A	none	none	linear	none	none	0	The border color of the module in the format RGBA where values range [0.0, 1.0].
view/color/contentbackground	array	N/A	none	none	linear	none	none	0	The background color of the module in the format RGBA where values range [0.0, 1.0].
view/color/toolbarbackground	array	N/A	none	none	linear	none	none	0	The background color of the module's toolbar in the format RGBA where values range [0.0, 1.0].
view/color/toolbartext	array	N/A	none	none	linear	none	none	0	The color of the module's toolbar text in the format RGBA where values range [0.0, 1.0].

Figura 4.8 - Página de referência de um módulo.

4.6. Módulos

Nesta secção serão ilustrados os principais módulos que compõem cada uma das categorias descritas anteriormente. Estes e os restantes módulos encontram-se descritos em detalhe no apêndice deste documento.

Com o propósito de facilitar a navegação e selecção do módulo adequado para uma determinada função no processo de sonorização interactiva, os módulos que compõem SonData encontram-se organizados pelas seguintes categorias:

- Aquisição e armazenamento;
- Interacção;
- Mapeamento;
- Síntese de som;
- Utilitários;

Os módulos presentes nas categorias acima descritas foram desenvolvidos de modo a funcionarem hierarquicamente, isto é, apesar de constituírem um *toolkit* modular, encontram-se distribuídos segundo uma estrutura hierárquica que define o processo de desenvolvimento de um sistema de sonorização. A Figura 4.9 procura ilustrar este processo, onde os dados contidos nos módulos de aquisição e armazenamento — que representam a base necessária ao processo de sonorização — são introduzidos nos módulos de interacção. É através destes, que os dados são posteriormente mapeados para parâmetros providenciados pelos módulos presentes na categoria de síntese de som.

Grande parte da interacção para com um sistema de sonorização desenvolvido em SonData é promovida pelos módulos presentes nas categorias de interacção e mapeamento. São estas categorias que permitem ao utilizador fazer uso dos cenários de interacção em PMSon e MBS (Secções 3.2; 3.3). O resultado destas interacções é então mapeado para parâmetros de síntese de som, promovendo assim a resposta acústica do sistema em função da interacção dada.

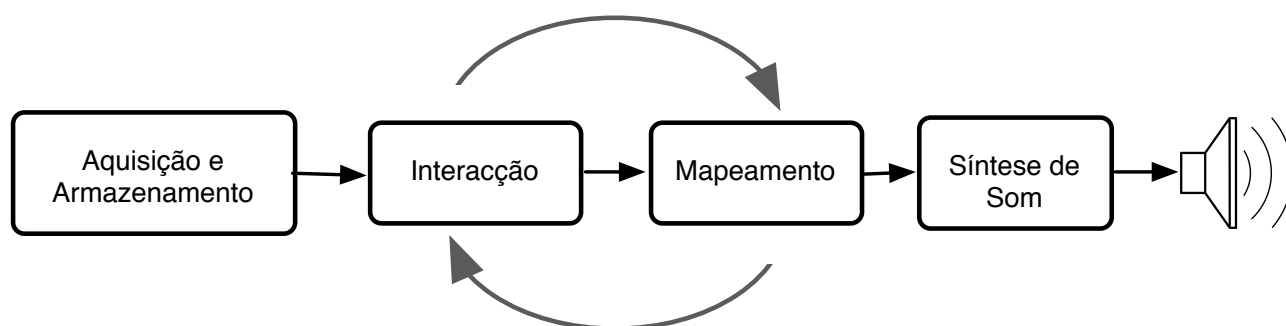


Figura 4.9 - Metodologia proposta por SonData.

Na Tabela 4.1 são apresentados os módulos presentes nas categorias de interacção e mapeamento de acordo com as técnicas para a qual foram desenvolvidos. Adicionalmente, na Tabela 4.2 são apresentados os módulos para a interacção em PMSon, distribuídos de acordo com o tipo de interacção que promovem. Os tipos de interacção na técnica de PMSon, encontram-se descritos em detalhe na Secção 3.2.

	Parameter Mapping Sonification	Model-Based Sonification
Interacção	•Jmod.son.brush •Jmod.son.graph •Jmod.son.mgraph •Jmod.son.Knearest	•Jmod.son.model
Mapeamento	•Jmod.son.mapper	•Jmod.son.linkModel

Tabela 4.1 - Módulos de interacção, mapeamento e respectivas técnicas.

	Navegação	Mapeamento	Seleccção Interactiva
Módulos	•Jmod.son.graph •Jmod.son.mgraph	•Jmod.son.mapper •(Módulos de síntese)	•Jmod.son.graph •Jmod.son.mgraph

Tabela 4.2 - Módulos de interacção em PMSon e respectiva tipologia.

4.6.1. Aquisição e armazenamento

Esta categoria é constituída pelos seguintes módulos:

- Jmod.son.url;
- Jmod.son.table;

De seguida, são ilustradas as funcionalidades específicas de cada um, bem como a sua relação com os restantes módulos de SonData. Uma documentação detalhada destes módulos pode ser consultada no apêndice deste documento.

Jmod.son.url

Este módulo permite ao utilizador fazer *download* de um conjunto de dados localizado num determinado URL (<http://>, <https://>, <ftp://>, e <file:///>). O ficheiro descarregado é armazenado numa directoria definida pelo utilizador. Permite também a edição do conjunto de dados.



Figura 4.10 - Jmod.son.url.

O segundo *outlet* devolve os dados adquiridos, permitindo ao utilizador armazenar os mesmos num módulo [jmod.son.table](#) (Fig. 4.11).

Jmod.son.table

Módulo que permite armazenar um conjunto de dados de dimensão arbitrária. Os dados são representados numa tabela (Fig. 4.12), acessível através do *panel widget* (Secção 4.4.1.1) que se situa no canto superior direito do módulo. Adicionalmente, permite atribuir um nome às colunas.



Figura 4.11 - Jmod.son.table.

Os dados podem ser inseridos de três modos: (1) o utilizador importa um conjunto de dados situado no disco rígido, (2) os dados inseridos provêm de [jmod.son.url](#), e (3) os dados são inseridos através de mensagens inseridas no *inlet* direito. A terminologia destas mensagens encontra-se documentada no *patch* de ajuda.

A comunicação entre [Jmod.son.table](#) e os módulos de interacção (Secção 4.6.2) é feita remotamente através do protocolo de comunicação OSC. Deste modo, o ID de cada módulo ([jmod.son.table](#); Fig. 4.11) deve ser único.

Separate by "~"

A-B-C-D-E

	A	B	C	D	E
0	5.1	3.5	1.4	0.2	1
1	4.9	3	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5	3.6	1.4	0.2	1
5	5.4	3.9	1.7	0.4	1
6	4.6	3.4	1.4	0.3	1
7	5	3.4	1.5	0.2	1
8	4.4	2.9	1.4	0.2	1
9	4.9	3.1	1.5	0.1	1
10	5.4	3.7	1.5	0.2	1
11	4.8	3.4	1.6	0.2	1
12	4.8	3	1.4	0.1	1
13	4.3	3	1.1	0.1	1
14	5.8	4	1.2	0.2	1
15	5.7	4.4	1.5	0.4	1

Figura 4.12 - *Panel* Jmod.son.table.

4.6.2. Interacção

Esta categoria é constituída pelos seguintes módulos:

- Jmod.son.graph;
- Jmod.son.mgraph;
- Jmod.son.knearest;
- Jmod.son.brush;
- Jmod.son.model;

De seguida, são apresentados os módulos principais desta categoria e ilustradas as funções específicas de cada um. Estes e os restantes módulos encontram-se detalhadamente documentados no apêndice deste documento.

Jmod.son.mgraph

Interface para visualização e reprodução de conjuntos de dados contidos em jmod.son.table (Fig. 4.11). Permite representar um máximo de cinco gráficos de linhas em simultâneo. O interface que representam os gráficos (Fig. 4.14) é acessível através do *widget panel* (ver 4.4.1.1) que se situa no canto superior direito do módulo.

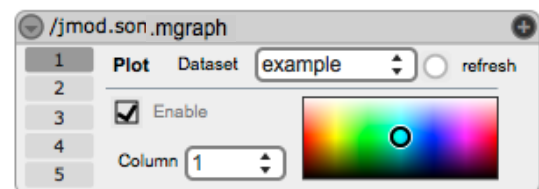


Figura 4.13 - Jmod.son.mgraph

A reprodução dos dados representados pode ser feita de dois modos: (1) reprodução contínua cuja duração é definida pelo utilizador (em segundos) ou ao (2) o arrastando o cursor sobre a *playhead* do gráfico. Permite ainda seleccionar e reproduzir uma região específica do conjunto.



Figura 4.14 - Panel jmod.son.mgraph.

Jmod.son.brush

Interface para selecção interactiva de dados (ver 3.2.3). Os dados armazenados em `jmod.son.table` (Fig. 4.11) são representados num plano cartesiano (Fig. 4.16), acessível através do *widget panel* (ver 4.4.1.1) que se situa no canto superior direito do módulo.

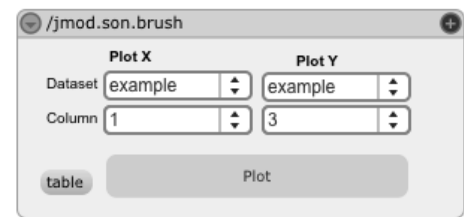


Figura 4.15 - Jmod.son.brush.

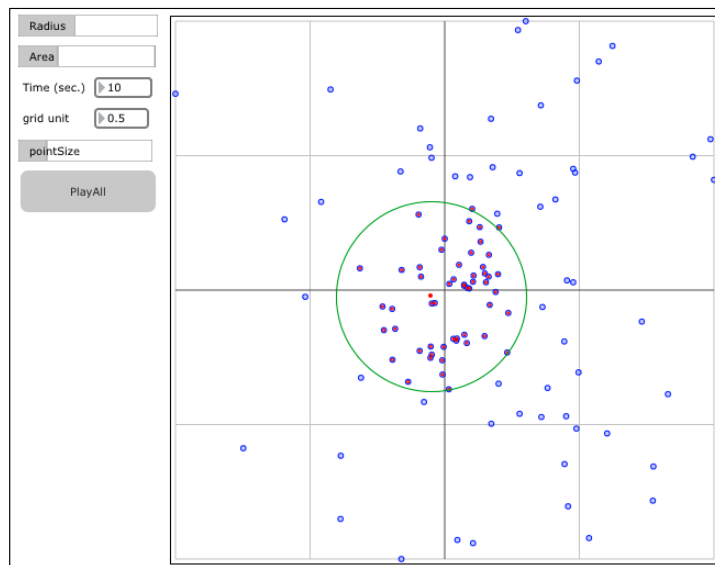


Figura 4.16 - *Panel* Jmod.son.brush.

À semelhança do interface *Sonic Brushing* (ver 3.2.3), a selecção é feita através de um círculo cujo raio é definido pelo utilizador. Adicionalmente, permite que o círculo se expanda de forma equidistante e com uma velocidade constante (definida pelo utilizador). Esta expansão pode ser feita em todo o gráfico, seleccionando todos os pontos, ou num raio definido pelo utilizador. Os dados subjacentes aos pontos seleccionados são devolvidos pelo *outlet* direito do módulo.

Jmod.son.knearest

Interface para selecção interactiva de dados (ver 3.2.3). Recorre ao objecto `mnm.knn` (FTM & Co), que implementa o algoritmo *k nearest neighbor* (Cover & Hart, 1967), um algoritmo de reconhecimento de padrões útil na classificação do conjunto de dados sob análise.

Os dados armazenados em `jmod.son.table` (Fig. 4.11), são representados por pontos num plano cartesiano (Fig. 4.18), acessível através do *widget* (ver 4.4.1.1) que se situa no canto superior direito do módulo.

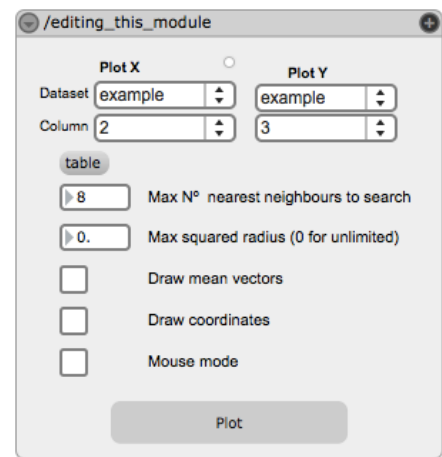


Figura 4.17 - Jmod.son.knearest.

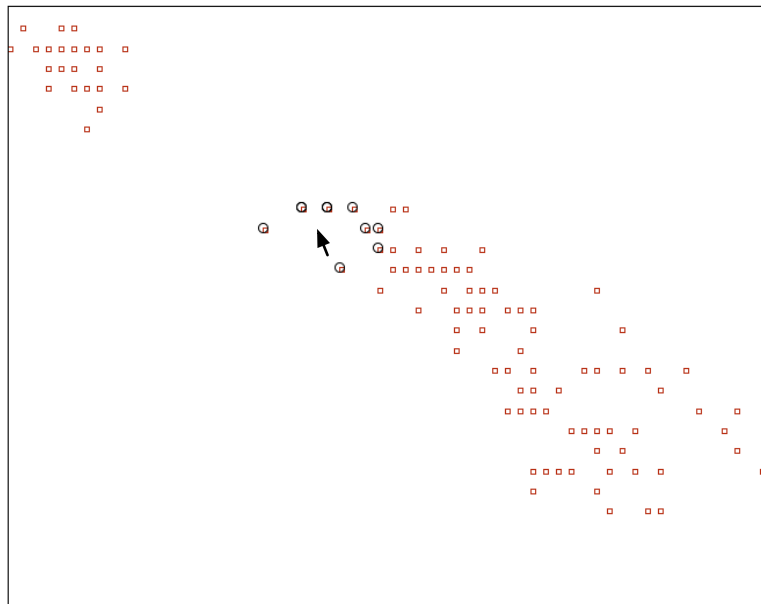


Figura 4.18 - Panel Jmod.son.knearest.

O *outlet* direito do módulo, devolve os (1) dados relativos aos pontos seleccionados, (2) a distância entre estes e o seu (3) índice na tabela de `jmod.son.table` (Fig. 4.12). Para além da sua função no âmbito de reconhecimento de padrões, à semelhança de `jmod.son.brush` (Fig. 4.15), também este módulo é útil na compreensão do agrupamentos dos pontos de dados sob análise.

Jmod.son.model

The screenshot shows a software interface titled "/myModule". It is divided into three main sections: "Model Setup", "Model Dynamics", and "Range".

- Model Setup:** Contains a "Dataset" dropdown menu set to "Example". Below it are two rows of controls: "Mass Position X" with a "col." label and a dropdown set to "ColA", and "Mass Position Y" with a "col." label and a dropdown set to "ColB". At the bottom left is a "Refresh Names" checkbox (unchecked) and a red "Reset" button.
- Model Dynamics:** Contains three rows of controls: "Masses Mass" with a "col." label and a dropdown set to "ColC", "Link's Rigidity" with a "col." label and a dropdown set to "ColD", and "Link's Damping" with a "col." label and a dropdown set to "ColE". At the bottom are two buttons: "LinkTable" and "MassTable".
- Range:** Contains two columns of range controls. The first column has "Min." and "Max." labels. The first row has values "1." and "100.". The second row has values "10." and "40.". The third row has values "4." and "50.". A large green "Build" button is at the bottom right.

Figura 4.19 - Jmod.son.model.

Um modelo de sonorização para MBS (Secção 2.4.4). A sua configuração (ver 2.4.4.1) encontra-se previamente definida e é composta por um sistema de massa-mola (*mass-spring system*). Os dados armazenados num módulo [jmod.son.table](#) (Fig. 4.11) são representados por uma estrutura de nódulos (massas) ligados entre si através de *edges* (molas; Fig. 4.20), acessível através do *widget panel* (ver 4.4.1.1) que se situa no canto superior direito do módulo.

As suas dinâmicas (ver 2.4.4.2) são fornecidas pelas equações inspiradas no comportamento físico e mecânico dos sistemas de massa-mola. Propriedades físicas como a massa dos nódulos, rigidez e amortecimento dos *edges* são definidos pelo conjunto de dados subjacente que se encontra armazenado no módulo [jmod.son.table](#) (Fig. 4.11).

Neste modelo, cada nódulo (massa) contribui com um som específico para a sonorização. Este, é representado de acordo com o movimento e velocidade do nódulo quando excitado.

A excitação (ver 2.4.4.3) ocorre ao ser aplicada uma força externa nas equações definidas nas dinâmicas do modelo. A força aplicada num nódulo é transferida para o/s *edge/s* que lhe estão acoplados, deste modo, o número e propriedades físicas (rigidez e amortecimento) dos *edges* acoplados ao nódulo excitado são determinantes no comportamento do resto da estrutura. O utilizador excita o modelo ao seleccionar e arrastar com o cursor sobre um determinado módulo da estrutura. Porém, devido ao protocolo de comunicação OSC inerente a cada módulo de SonData, a excitação do modelo pode ser feita através de outros interfaces externos para além do rato.

A ligação entre o comportamento do modelo e a as variáveis acústicas, providenciada pelos módulos presentes na categoria de síntese de som (Secção 4.6.4), é mediada pelo módulo Jmod.son.linkmodel (Fig. 4.22). Este, recebe informação relativa à força aplicada pela excitação, posição e velocidade de cada nódulo que compõem o modelo.

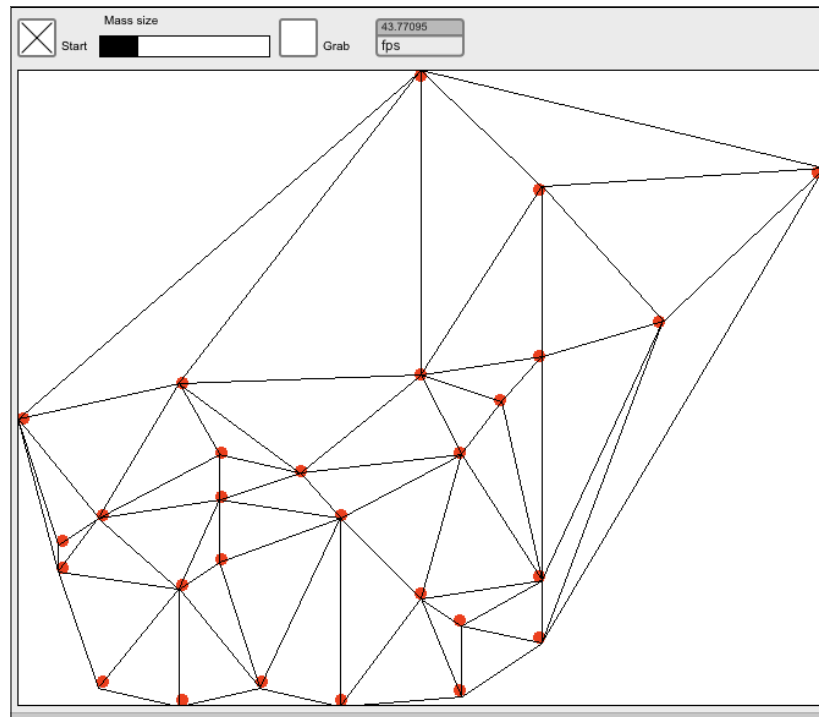


Figura 4.20 - *Panel Jmod.son.model*.

4.6.3. Mapeamento

Esta categoria é constituída pelos seguintes módulos:

- Jmod.son.mapper;
- Jmod.son.linkmodel;

De seguida, são ilustradas as funcionalidades de cada um, bem como a sua relação com os restantes módulos de SonData. Uma documentação detalhada dos módulos acima referidos pode ser consultada no apêndice deste documento.

Jmod.son.mapper

Módulo de mapeamento para PMSon (Fig. 4.21; ver 2.5.3.1). Permite receber até cinco valores em simultâneo e proceder ao mapeamento dos mesmos. Permite as

interacções de mapeamento descritas na Secção 3.2.2, deste modo, o utilizador pode alterar em tempo real o mapeamento definido e parâmetros relacionados.

No sentido de criar um interface eficaz para estas interacções, foi implementado um “conector” que permite estabelecer ou interromper o mapeamento entre os dados de entrada e um determinado parâmetro de síntese. Os restantes controlos são apresentados na forma de caixas de números e menus, referentes à função, escala e polaridade do mapeamento (ver 2.4.3.1).

A comunicação entre este módulo e os módulos de síntese (ver 4.6.4) associados é feita remotamente através do protocolo de comunicação OSC. Apesar de optimizado para a comunicação com os módulos presentes na categoria de síntese, este módulo pode ser facilmente utilizado no mapeamento com os restantes módulos do Jamoma e Jamoma UserLib.

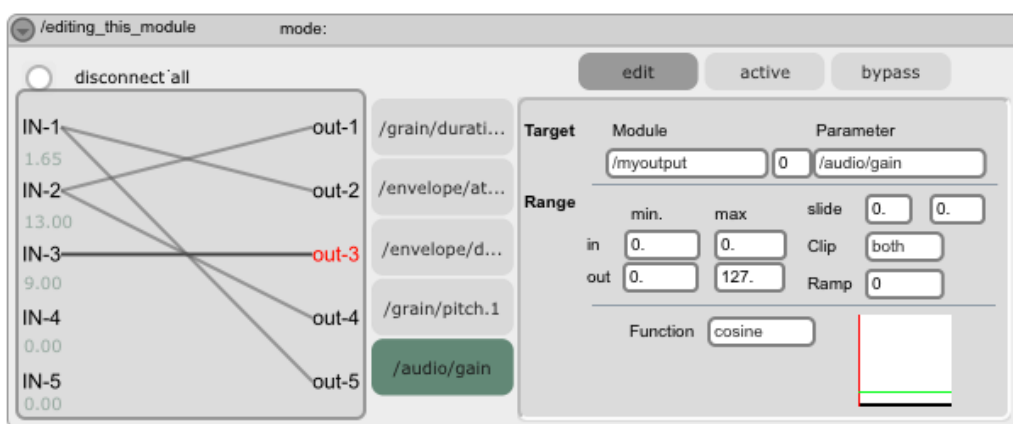


Figura 4.21 - Jmod.son.mapper.

Jmod.son.linkmodel

Módulo que estabelece a ligação entre o comportamento físico do modelo de sonorização presente em `jmod.son.model` e as variáveis acústicas (ver 2.4.4.4). Estas são providenciadas pelos módulos presentes na categoria de síntese de som (Secção 4.6.4).

Permite relacionar a posição e velocidade de cada massa, bem como a força aplicada pela excitação (ver 2.4.4.3) até um número máximo de cinco variáveis acústicas. Adicionalmente, o utilizador pode definir um âmbito para os valores das variáveis acústicas e a função de mapeamento.

À semelhança de `jmod.son.mapper` (Fig. 4.21), a comunicação entre este módulo e os módulos de síntese associados é feita remotamente através do protocolo de comunicação OSC.

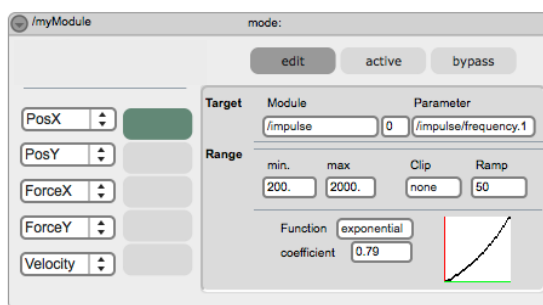


Figura 4.22 - Jmod.son.linkmodel.

4.6.4. Síntese de som

Esta categoria é constituída pelos seguintes módulos:

- Jmod.son.oscillator~;
- Jmod.son.impulse~;
- Jmod.son.granular~;
- Jmod.son.pulsar~;
- Jmod.son.waveshaper~;
- Jmod.son.subtractive~;
- Jmod.son.resonator~;
- Jmod.son.fof~;
- Jmod.son.karplus~

Estes são módulos polifónicos, isto é, permitem instanciar várias vozes. Deste modo, o número de parâmetros presentes no copo de cada módulo é multiplicado pelo número de vozes instanciada.

O endereçamento dos parâmetros para respectivas vozes é feito remotamente através dos módulos de mapeamento (Secção 4.6.3), ou através do envio de uma mensagem para o primeiro *inlet* do módulo. A terminologia destas encontra-se documentada no seu *patch* de ajuda (Secção 4.5).

De seguida, são ilustradas as funcionalidades dos módulos que representam maior importância nesta categoria, bem como, a sua relação com os restantes módulos de SonData. Uma documentação detalhada dos módulos acima referidos pode ser consultada no apêndice deste documento.

Jmod.son.pulsar~

Módulo com simples implementação de síntese pulsar. Recebe remotamente os valores provenientes dos módulos de mapeamento (Secção 4.6.3).

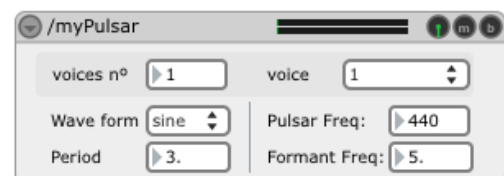


Figura 4.23 - Jmod.son.pulsar~.

Jmod.son.resonator~

Banco de filtros ressonantes, úteis para as variáveis auditivas na técnica de MBS (Secção 2.4.4). Permite o uso de modelos previamente definidos ou modelos específicos indicados pelo utilizador.

Pode ser excitado a partir de um sinal áudio externo ou através de um impulso de tamanho arbitrário. Adicionalmente, permite ainda importar ficheiros SDIF (*Sound Description Interchange Format*) de modo a expandir o leque de modelos disponíveis.

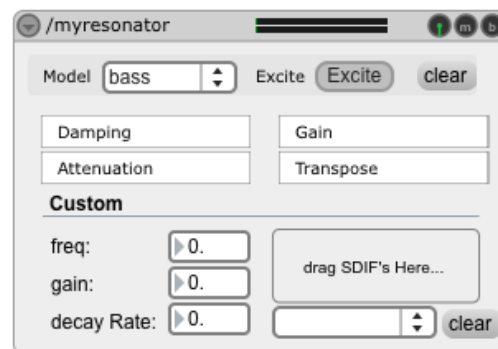


Figura 4.24 - Jmod.son.resonators~.

Jmod.son.fof~

Síntese de formantes gerados através de uma função de onda formante (FOF). Contém três formantes, no entanto, sendo um módulo polifónico, o número de formantes e respectivos parâmetros é multiplicado pelo número de vozes instanciadas.

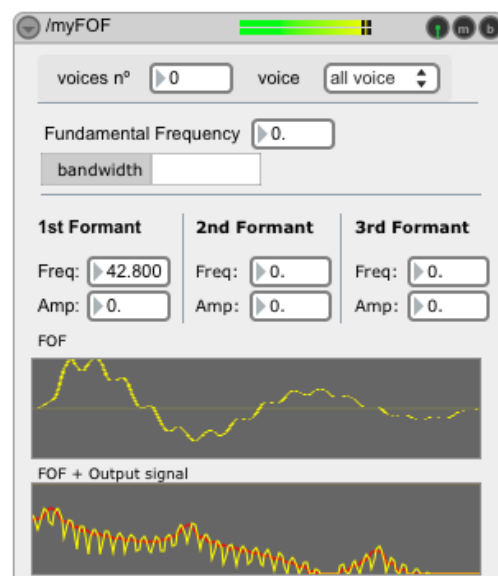


Figura 4.25 - Jmod.son.fof~.

Jmod.son.karplus~

Módulo de síntese por modelação física. Recorre ao algoritmo de *Karplus Strong* (Karplus & Strong, 1983) para emular o som de uma corda quando beliscada. O número de cordas e respectivos parâmetros é multiplicado pelo número de vozes instanciadas.

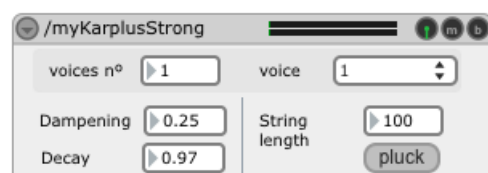


Figura 4.26 - Jmod.son.karplus~.

4.6.5. Utilitários

Esta categoria é constituída pelos seguintes módulos:

- Jmod.son.stats;
- Jmod.son.getcol;
- Jmod.son.datetime;
- Jmod.son.except;

Estes, representam funções que estão para além do âmbito da sonorização interactiva, mas que se revelam úteis no contexto de utilização de SonData. De seguida iremos descrever as funcionalidades de Jmod.son.getcol e Jmod.datetime. Uma documentação detalhada dos módulos acima referidos pode ser consultada no apêndice deste documento.

Jmod.son.getcol

Módulo que permite obter uma lista com os dados relativos a uma determinada coluna, de um conjunto armazenado em jmod.son.table (Fig. 4.11).

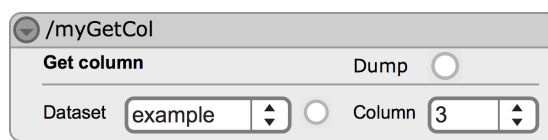


Figura 4.27 - Jmod.son.getcol.

Uma vez que a comunicação entre jmod.son.table e os módulos de interacção (Secção 4.6.2) é feita remotamente através do protocolo OSC, pode tornar-se difícil extrair os dados armazenados no formato de uma lista. Este módulo devolve os dados armazenados em Jmod.son.table sob a forma de uma lista, permitindo deste modo a integração de outros componentes (módulos Jamoma e objectos nativos ou externos do ambiente Max) exteriores a SonData.

Jmod.son.datetime

Módulo que devolve informação relativa à data e hora. O utilizador pode recorrer a este módulo e ao jmod.son.url (Fig. 4.10) para fazer o *download* de um conjunto de dados que é actualizado de minuto em minuto, por exemplo.

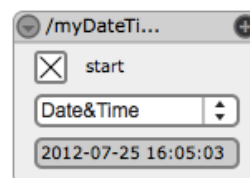


Figura 4.28 - Jmod.son.datetime.

5. Conclusão

O *toolkit* apresentado anteriormente — SonData — resulta da reunião de alguns cenários de interacção e práticas recorrentes na sonorização interactiva. Tendo uma maior contribuição para a técnica de PMSon (Secção 2.4.3), SonData permite (1) interacções de navegação, (2) a selecção interactiva de dados, e ainda (3) as interacções para com o mapeamento e parâmetros relacionados. Funções essas que foram descritas na Secção 3.2. Para além destas, providencia também um modelo de sonorização para a técnica de MBS, tratando-se assim do único sistema disponível para a comunidade que promove esta funcionalidade.

Apesar de ter sido desenvolvido para a sonorização interactiva, grande parte dos módulos de SonData podem ser utilizados nos mais variados contextos que exijam o mapeamento entre dados e parâmetros de síntese. Adicionalmente, a presença do protocolo de comunicação OSC em todos os módulos de SonData é um factor importante na conectividade de sistemas, tanto no contexto da sonorização interactiva como no meio artístico. No contexto da sonorização interactiva, esta conectividade é particularmente útil no uso de interfaces externos (superfície multitoque, acelerómetro, por exemplo) como meio de controlo de determinado parâmetro, seja ele de mapeamento, selecção interactiva ou mesmo a excitação de um modelo de sonorização. No meio artístico, esta conectividade pode facilitar processos de mapeamento, e geração sonora a partir de dados (relativos ao movimento de um bailarino, por exemplo) que provenham de um sistema externo (sensores, por exemplo).

O actual estado de SonData constitui uma base de trabalho suficientemente robusta e flexível para o desenvolvimento de uma sonorização eficaz, e ainda para motivar futuros aperfeiçoamentos e expansões por parte da comunidade. Neste sentido a sua integração na *framework* Jamoma representa um papel fundamental. A uniformidade da abordagem ao desenvolvimento de cada módulo, da sua documentação, e a própria robustez da *framework*, fazem desta, uma excelente plataforma para o desenvolvimento e distribuição de SonData. Só deste modo é possível obter um sistema robusto e ao mesmo tempo suficientemente flexível, de modo a adaptar-se às necessidades emergentes no campo da sonorização de dados interactiva.

Por último SonData é um *toolkit* multiplataforma e *open-source*, que se encontra actualmente disponível num repositório *online*, acessível em (<https://github.com/>

JoaoMenezes/SonData). Integra ainda a biblioteca de módulos externos Jamoma UserLib, distribuída com a *framework* Jamoma, permitindo deste modo um acesso imediato a um grande número de utilizadores.

5.1. Sugestões para estudo futuro

Os futuros desenvolvimentos de SonData devem ter como prioridade o aperfeiçoamento da relação entre a excitação do modelo (Fig. 4.19) e as variáveis acústicas. Neste sentido, a integração de uma nova categoria composta por módulos de síntese exclusivamente adaptados para MBS (Secção 2.4.4), poderia assegurar uma maior robustez na sua contribuição para a técnica e consequentemente, para com os dados sob análise. Uma vez aperfeiçoada a relação entre a excitação e a resposta acústica do modelo, seria de esperar a integração de uma categoria composta por modelos de sonorização resultantes da literatura existente no campo.

Posteriormente, uma função que SonData deve implementar a curto prazo, é um módulo para a análise de uma *feed* XML. Este, apesar de ser um método comum na partilha de dados pela internet, poucas vezes é construído com o rigor necessário. Deste modo, o desenvolvimento de um módulo flexível e intuitivo para a extracção de dados provenientes de uma *feed* XML, pode tornar-se num processo complexo.

Referências

- Barrass, S. 1995. Personify: a Toolkit for Perceptually Meaningful Sonification.
- Blattner, Meera M, Denise A Sumikawa, and Robert M Greenberg. 1989. "Earcons and Icons: Their Structure and Common Design Principles." *Human-Computer Interaction* 4 (1): 11-44.
- Brazil, E. and Fernström, M. (2011). Auditory icons. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, *The Sonification Handbook*, chapter 13, pages 325–338. Logos Publishing House, Berlin, Germany.
- Brown, Lorna M, Stephen A Brewster, Ramesh Ramloll, Mike Burton, and Beate Riedel. 2003. Design guidelines for audio presentation of graphs and tables. In *Proceedings of the international conference on auditory display*.
- Cover, T, and P Hart. 1967. "Nearest neighbor pattern classification." *Information Theory, IEEE Transactions on* 13 (1) (January): 21-27.
- Dingler, Tilman, Jeffrey Lindsay, and Bruce N Walker. 2008. LEARNABILITY OF SOUND CUES FOR ENVIRONMENTAL FEATURES: AUDITORY ICONS , EARCONS , SPEARCONS , AND SPEECH. In *Proceedings of the 14th International Conference on Auditory Display, Paris, France, June 24-27, 2008*, 1-6.
- Dombois, Florian. 2008. SONIFYER A Concept, a Software, a Platform. In *Proceedings of the 14th International Conference on Auditory Display*. Paris, France.
- Fitch, W T, and G Kramer. 1992. Sonifying the body electric: Superiority of an auditory over a visual display in a complex multivariate system. In *Auditory display: Sonification, audification, and auditory interfaces*, ed. G Kramer, 307-326. Addison Wesley Longman.
- Gaver, W, Randall B Smith, and Tim O Shea. 1991. "Effective sounds in complex systems: The ARKola simulation." ... *Human factors in computing systems: ...* 1991: 85-90.
- Guedes, Carlos. 2005. THE M-OBJECTS: A SMALL LIBRARY FOR MUSICAL RHYTHM GENERATION AND MUSICAL TEMPO CONTROL FROM DANCE MOVEMENT IN REAL TIME. In *Proceedings from the International Computer Music Conference*, 794-797. Barcelona, Spain.
- Hermann, T. 2002. Sonification for exploratory data analysis. PhD thesis, Bielefeld University, Bielefeld, Germany, <http://en.scientificcommons.org/34743641>.

- Hermann, T. 2008. TAXONOMY AND DEFINITIONS FOR SONIFICATION AND AUDITORY DISPLAY. In *Proceedings of the 14th International Conference on Auditory Display*, 1-8. Paris, France.
- Hermann, T. (2011). Model-based sonification. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, *The Sonification Handbook*, chapter 16, pages 399–427. Logos Publishing House, Berlin, Germany
- Hermann, Thomas, Gerold Baier, Ulrich Stephani, and Helge Ritter. 2006. VOCAL SONIFICATION OF PATHOLOGIC EEG FEATURES. In *Proceedings of the 12th International Conference on Auditory Display, Londonth International Conference on Auditory Display, London, UK June 20 - 23*.
- Hermann, Thomas, and Andy Hunt. 2004. The Discipline of Interactive Sonification. In *Proceedings of the International Workshop on Interactive Sonification (ISon 2004)*, ed. Thomas Hermann and Andy Hunt. Bielefeld, Germany: Interactive Sonification Community.
- Hermann, Thomas, Jan Krause, and Helge Ritter. 2002. Real-Time Control of Sonification Models with an Audio-Haptic Interface. In *Proceedings of the International Conference on Auditory Display*, ed. R Nakatsu and H Kawahara, 82-86. Kyoto, Japan: ICAD.
- Hermann, Thomas, Peter Meinicke, and Helge Ritter. 2000. Principal Curve Sonification. In *Proceedings of the International Conference on Auditory Display*, ed. P R Cook, 81-86. International Community for Auditory Display (ICAD).
- Hermann, Thomas, and Helge Ritter. 1999. Listen to your Data: Model-Based Sonification for Data Analysis. In *Advances in intelligent computing and multimedia systems*, ed. G E Lasker, 189-194. Baden-Baden, Germany: Int. Inst. for Advanced Studies in System research and cybernetics.
- Hermann, Thomas, and Helge Ritter. 2004. Neural Gas Sonification - Growing Adaptive Interfaces for Interacting with Data. In *Proceedings of the Information Visualisation, Eighth International Conference*, 871-878. Washington, DC, USA: IEEE Computer Society.
- Hermann, Thomas, and Helge Ritter. 2005. “Crystallization sonification of high-dimensional datasets.” *ACM Trans. Appl. Percept.* 2 (4) (October): 550-558.
- Hermann, T., Hunt, A., Neuhoff, J. G. (2011). Introduction. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, *The Sonification Handbook*, chapter 1, pages 1–6. Logos Publishing House, Berlin, Germany
- Hunt, A, and T Hermann. 2004. THE IMPORTANCE OF INTERACTION IN SONIFICATION. In *computer music*, 1-8.
- Hunt, A. and Hermann, T. (2011). Interactive sonification. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, *The Sonification Handbook*, chapter 11, pages 273–298. Logos Publishing House, Berlin, Germany
- Karplus, K, and A Strong. 1983. “Digital synthesis of plucked-string and drum timbres.” *Computer Music J.* 7 (2): 43-55.

- Kolbe, Lukas, René Tünnermann, and Thomas Hermann. 2010. Growing Neural Gas Sonification Model for Interactive Surfaces. In *Interactive Sonification Workshop (ISon) Human Interaction with Auditory Displays*, ed. Roberto Bresin. Stockholm: KTH.
- Kramer, G. 1992. *An introduction to auditory display*. Ed. G Kramer. Addison Wesley Longman. Proceedings/1992/Kramer1992a.pdf.
- Kramer, G. 1994. Some organizing principles for representing data with sound. In *Auditory display: Sonification, Audification, and Auditory Interfaces.*, ed. G Kramer, 185-222. Addison Wesley Longman.
- Kramer, G, B Walker, T Bonebright, P Cook, J Flowers, N Miner, and Neuhoff. 1997. Sonification report: Status of the field and research agenda. *Prepared for the National Science Foundation by members of the International Community for Auditory Display*.
- Newhoff, John G, Rebecca Knight, and Joseph Wayand. 2002. Pitch change, sonification, and musical expertise: Which way is up? In *International Conference on Auditory Display*.
- Pauletto, Sandra, and Andy Hunt. 2004. A TOOLKIT FOR INTERACTIVE SONIFICATION. In *Proceedings of the 10th International Conference on Auditory Display*. Sydney, Australia.
- Place, Tim, and Trond Lossius. 2006. JAMOMA: A MODULAR STANDARD FOR STRUCTURING PATCHES IN MAX M/MSP. In *Proceedings of the International Computer Music Conference 2006*. New Orleans, USA.
- Riedenklaus, E, T Hermann, and H Ritter. 2010. Tangible Active Objects and interactive sonification as a scatter plot alternative for the visually impaired. In *Proc. ICAD 2010*.
- Smith, D R, and B N Walker. 2002. Tick-marks, axes, and labels: The effect of adding context to auditory graphs. In *Proceedings of the International Conference on Auditory Display*. Kyoto, Japan.
- Vickers, P, and B Hogg. 2006. Sonification abstraite/sonification concrete: An “aesthetic perspective space” for classifying auditory displays in the ars musica domain. In *Proceedings of the 12th International Conference on Auditory Display (ICAD2006)*, ed. Christopher Frauenberger Alistair D N Edwards Tony Stockman Louise Valgerður Nickerson and Derek Brock, 210-216. London, UK: Department of Computer Science, Queen Mary, University of London, UK.
- Walker, B.N., and M.A. Nees. 2006. 2 Theory of Sonification. In *Principles of Sonification: An Introduction to Auditory Display and Sonification*, 1-32.
- Walker, B. N. and Nees, M. A. (2011). Theory of sonification. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, *The Sonification Handbook*, chapter 2, pages 9–39. Logos Publishing House, Berlin, Germany
- Walker, Bruce N, and Gregory Kramer. 2005. “Mappings and metaphors in auditory displays: An experimental assessment.” *ACM Trans. Appl. Percept.* 2 (4) (October): 407-412.
- Walker, Bruce N., and Joshua T. Cothran. 2003. Sonification Sandbox: A graphical toolkit for auditory graphs. In *Proceedings of the Ninth International Conference on Auditory Display ICAD2003*, 161-163. Boston, USA.

- Wilson, C M, and S K Lodha. 1996. Listen: A data sonification toolkit. In *Proceedings of the 3rd International Conference on Auditory Display (ICAD96)*, ed. S P Frysinger and G Kramer. Palo Alto, CA, U.S.: International Community for Auditory Display.
- Worrall, D, M Bylstra, S Barrass, and R Dean. 2007. SoniPy: The Design of an Extendable Software Framework for Sonfication Research and Auditory Display. In *Proceedings of the 13th International Conference on Auditory Display (ICAD2007)*, ed. Gary P Scavone, 445-452. Montreal, Canada: Schulich School of Music, McGill University.
- Yeo, W. S.; Berger, J.; Wilson, R. S. 2004. A FLEXIBLE FRAMEWORK FOR REAL-TIME SONIFICATION WITH SONART. In *Proceedings of the 10th International Conference on Auditory Display (ICAD2004)*, 1-3. Sydney, Australia.

Apêndice

SonData: Manual de referência

As páginas que se seguem, apresentam a página de referência de cada módulo que compõem SonData. No entanto, por motivos de clareza, apenas são referenciados mensagens e parâmetros directamente relacionados com o contexto de SonData. Deste modo, mensagens e parâmetros nativos da *framework* Jamoma, também presentes na página de referência, encontram-se aqui omitidos.

Jmod.son.table - Import, store and save a d-dimensional dataset

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions /allow	/description
ID	string	N/A	none	none	linear	none	none	0	Dataset ID - must be unique and without spaces
colNames	generic	0.00000 0 1.00000 0	none	none	linear	none	none	0	set name for each column

Messages

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/repetitions/ allow	/description
clear	string	N/A	none	none	linear	none	none	1	Clear Data Table
dump	none	N/A	none	none	linear	none	none	1	Dump Data set
import	string	N/A	none	none	linear	none	none	1	read data file from the hard drive (.txt)

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/repetitions/ allow	/ enable	/description
dataset	array	N/A	none			none	none	1	1	return stored data set

Jmod.son.url - Provides asynchronous file download (http://; https://; ftp://, file:///)

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions /allow	/description
URL	array	N/A	none	none	linear	none	none	1	Insert URL

Messages

name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/repetitions/ allow	/description
clear	string	N/A	none	none	linear	none	none	1	clear URL text field
defaultDir	string	N/A	none	none	linear	none	none	1	choose download folder
download	generic	0.000000 1.000000	none	none	linear	none	none	1	bang to download
dump	none	N/A	none	none	linear	none	none	1	dump data (row-by-row)

edit	string	N/A	none	none	linear	none	none	1	edit dataset as text
------	--------	-----	------	------	--------	------	------	---	----------------------

Return

data	array	N/A	none			none	none	1	Return stored dataset as list.
------	-------	-----	------	--	--	------	------	---	--------------------------------

Jmod.son.mgraph - line graph player

Parameters

/name	/type	/range/ bounds	range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	dataspace/ unit/native	repetitions/ allow	description
ID	string	N/A	none	none	linear	none	none	0	Table ID - must be unique and without spaces
column	array	N/A	none	none	linear	none	none	1	choose column to plot in X axis
cursor	decimal	0.000000 1.000000	none	none	linear	none	none	0	GUI cursor, range/bounds= GUI size
dataSetID	array	N/A	none	none	linear	none	none	1	Name of table containing desired input for X axis
Data	array	N/A	none	none	linear	none	none	1	Data to store in table
duration	integer	1 10000	low	none	linear	none	none	1	Set sequence's Duration : Seconds
fgcolor	array	N/A	none	none	linear	none	none	0	set foreground color
loop	boolean	0 1	none	none	linear	none	none	0	Toggle to loop the reproduction
pause	boolean	0 1	none	none	linear	none	none	0	Toggle to pause reproduction
remote	boolean	0 1	none	none	linear	none	none	0	switch remote communication On & Off
resizeGraph	boolean	0 1	none	none	linear	none	none	0	toggle to resize graph window with patcher
rulerUnit	string	N/A	none	none	linear	none	none	0	Desired domain ruler unit
selection	array	N/A	none	none	linear	none	none	0	selected data region in GUI
viewGrid	array	N/A	none	none	linear	none	none	0	view domain/range grid ruler
viewRuler	array	N/A	none	none	linear	none	none	0	view domain/range ruler

Messages

clear	none	N/A	none	none	linear	none	none	1	Clear Graph Content
import	string	N/A	none	none	linear	none	none	1	Import data to graph table
play	boolean	0 1	none	none	linear	none	none	0	Toggle to play graph

Return

value	array	N/A	none			none	none	0	1	output data from the graph under analysis
-------	-------	-----	------	--	--	------	------	---	---	---

Jmod.son.mgraph - Multi line graph player

Parameters

/name	/type	/range/ bounds	range/ clipmode	/ramp/drive	/ramp/ function	/dataspace	dataspace/ unit/native	repetitions /allow	description
autoUpdate	boolean	0 1	none	none	linear	none	none	0	Toggle to enable auto update of graph data
column.1	array	N/A	none	none	linear	none	none	0	choose column to plot
column.2	array	N/A	none	none	linear	none	none	0	choose column to plot
column.3	array	N/A	none	none	linear	none	none	0	choose column to plot
column.4	array	N/A	none	none	linear	none	none	0	choose column to plot
column.5	array	N/A	none	none	linear	none	none	0	choose column to plot
cursor	decimal	0.000000 1.000000	none	none	linear	none	none	0	GUI cursor, range/ bounds= GUI size
data	array	N/A	none	none	linear	none	none	1	Data to store in table
datasetName	array	N/A	none	none	linear	none	none	0	Name of table containing desired input for X
duration	integer	1 10000	low	none	linear	none	none	1	Set desired sequence's duration in seconds
enable.1	boolean	0 1	none	none	linear	none	none	0	Enable graph
enable.2	boolean	0 1	none	none	linear	none	none	0	Enable graph
enable.3	boolean	0 1	none	none	linear	none	none	0	Enable graph
enable.4	boolean	0 1	none	none	linear	none	none	0	Enable graph
enable.5	boolean	0 1	none	none	linear	none	none	0	Enable graph
fgcolor.1	array	N/A	none	none	linear	none	none	0	set foreground color
fgcolor.2	array	N/A	none	none	linear	none	none	0	set foreground color
fgcolor.3	array	N/A	none	none	linear	none	none	0	set foreground color
fgcolor.4	array	N/A	none	none	linear	none	none	0	set foreground color
fgcolor.5	array	N/A	none	none	linear	none	none	0	set foreground color
loop	boolean	0 1	none	none	linear	none	none	0	Toggle to loop the reproduction
pause	boolean	0 1	none	none	linear	none	none	0	Toggle to pause reproduction

resizeGraph	boolean	0 1	none	none	linear	none	none	0	toggle to resize graph window with patcher
rulerUnit	string	N/A	none	none	linear	none	none	0	Desired domain ruler unit
selection	array	N/A	none	none	linear	none	none	0	selected data region in GUI
viewGrid	array	N/A	none	none	linear	none	none	0	view domain/range grid ruler
viewRuler	array	N/A	none	none	linear	none	none	0	view domain/range ruler

Messages

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	/dataspace/ unit/native	/repetitions/ allow	/description
clear	none	N/A	none	none	linear	none	none	1	dump module contents
dump	string	N/A	none	none	linear	none	none	1	dump module contents
import	string	N/A	none	none	linear	none	none	1	import data into table
init	none	N/A	none	none	linear	none	none	1	Initialize a module completely to the original state.
play	boolean	0 1	none	none	linear	none	none	0	Toggle to play reproduction
preset/copy	array	N/A	none	none	linear	none	none	1	Create a new preset (2nd argument) by copying the contents of another preset (1st argument)
preset/default	none	N/A	none	none	linear	none	none	1	Open the default preset file and recall the first preset in that file.
refresh	none	N/A	none	none	linear	none	none	1	dump module contents
save	string	N/A	none	none	linear	none	none	1	save current table
stop	string	N/A	none	none	linear	none	none	1	Stop reproduction

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	/dataspace/ unit/native	/ repetitions/ allow	/ enable	/description
ReproductionEnd	string	N/A	none			none	none	0	1	return 1 at the end of reproduction and 1 while playing
value/graph1	array	N/A	none			none	none	0	1	output data from the graph under analysis - graph 1
value/graph2	decimal	0.000000 1.000000	none			none	none	0	1	output data from the graph under analysis - graph 1
value/graph3	decimal	0.000000 1.000000	none			none	none	0	1	output data from the graph under analysis - graph 3
value/graph4	decimal	0.000000 1.000000	none			none	none	0	1	output data from the graph under analysis - graph 4

value/ graph5	decim al	0.000000 1.000000	none			none	none	0	1	output data from the graph under analysis - graph 5
------------------	-------------	----------------------	------	--	--	------	------	---	---	---

Jmod.son.kneares - K nearest neighbor

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions /allow	/description
NtoSearch	integ er	0 1	none	sched uler	linear	none	none	0	Max number of K neighbors to search
col/x	arra y	N/A	none	none	linear	none	none	1	choose column to plot in x axis
col/y	arra y	N/A	none	none	linear	none	none	0	choose column to plot in y axis
dataSet/ x	arra y	N/A	none	none	linear	none	none	1	Name of table containing desired input for X
dataSet/ y	arra y	N/A	none	none	linear	none	none	1	Name of table containing desired input for Y
drawCoord s	integ er	0 1	none	none	linear	none	none	0	draw point coordinates
drawMean Vectors	integ er	0 1	none	none	linear	none	none	0	draw mean vectors and split lines per tree node
maxRadiu s	decim al	0.00000 0 1.00000 0	none	sched uler	linear	none	none	0	max squared radius of nearest neighbors to search (0 for unlimited)
mouseMod e	integ er	0 1	none	none	linear	none	none	0	mouse over mode
view/ freeze	bool ean	0 1	none	none	linear	none	none	0	Turn off the updating of user interface elements when parameters change. This may be done to conserve CPU resources.
view/ highligh t	strin g	N/A	none	none	linear	none	none	0	Highlight the module with a color tint such as red, green, or similar.

Messages

clear	none	N/A	none	none	linear	none	none	1	dump module contents
data	gene ric	N/A	none	none	linear	none	none	1	data values to plot
mousePos	arra y	N/A	none	none	linear	none	none	1	set mouse position x,y
plot	none	N/A	none	none	linear	none	none	1	trigger to plot the interface

Return

dista nce	arra y	N/A	none			none	none	1	1	distance between points
indic es	arra y	N/A	none			none	none	1	1	indices of selected points

Data	array	N/A	none			none	none	1	1	data of selected points
------	-------	-----	------	--	--	------	------	---	---	-------------------------

Jmod.son.brush - Select points inside radius

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions /allow	/description
area	decimal	1.00000 0 100.000 000	both	none	linear	none	none	0	set Radius
col/x	array	N/A	none	none	linear	none	none	1	choose column to plot in x axis
col/y	array	N/A	none	none	linear	none	none	0	choose column to plot in y axis
dataSet/ x	array	N/A	none	none	linear	none	none	1	Name of table containing desired input for X
dataSet/ y	array	N/A	none	none	linear	none	none	1	Name of table containing desired input for Y
gridUnit	decimal	0.00000 0 1.00000 0	both	none	linear	none	none	0	set grid unit
onOff	boolean	0 1	none	none	linear	none	none	0	Turn module on/off
playAll	integer	0 1	none	none	linear	none	none	1	expand radius -play all points
playTime	integer	1 3600	low	none	linear	none	none	1	set time of reproduction in seconds
pointSize	decimal	2.00000 0 15.0000 00	both	none	linear	none	none	0	set points size
radius	decimal	0.05000 0 1.00000 0	both	none	linear	none	none	0	set Aural Radius

Messages

mouse	array	N/A	none	none	linear	none	none	1	Mouse position in lcd - xy coordinates
mouse/ pressed	boolean	0 1	none	none	linear	none	none	0	is mouse pressed? just for internal patching convenience
plot	none	N/A	none	none	linear	none	none	1	Plot the graph

Return

data	array	N/A	none			none	none	1	1 Return selected data
------	-------	-----	------	--	--	------	------	---	------------------------

Jmod.son.model - Sonification model - Nodes connected by edges via delaunay triangulation

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions /allow	/description
dataSet	array	N/A	none	none	linear	none	none	1	Name of table containing desired input for X
grab	boolean	0 1	none	none	linear	none	none	0	Grab mass with mouse
link/dampMin	decimal	0.000000 10000.00 0000	low	none	linear	none	none	0	set min out scale for Mass's mass
link/damping	array	N/A	none	none	linear	none	none	1	choose column to plot in y axis
link/rigidity	array	N/A	none	none	linear	none	none	1	choose column to plot in y axis
link/rigidityMax	decimal	0.000000 10000.00 0000	low	none	linear	none	none	0	set max out scale for Mass's mass
link/rigidityMin	decimal	0.000000 10000.00 0000	low	none	linear	none	none	0	set min out scale for Mass's mass
mass/dampMax	decimal	0.000000 10000.00 0000	low	none	linear	none	none	0	set max out scale for Mass's mass
mass/mass	array	N/A	none	none	linear	none	none	1	choose column to plot in y axis
mass/massMax	decimal	0.000000 10000.00 0000	low	none	linear	none	none	0	set max out scale for Mass's mass
mass/massMin	decimal	0.000000 10000.00 0000	low	none	linear	none	none	0	set min out scale for Mass's mass
mass/pos/x	array	N/A	none	none	linear	none	none	1	choose column to plot in x axis
mass/pos/y	array	N/A	none	none	linear	none	none	1	choose column to plot in y axis
massSize	decimal	0.001000 0.500000	both	none	linear	none	none	0	adjust mass's size
start	boolean	0 1	none	none	linear	none	none	0	start rendering the module

Messages

build	none	N/A	none	none	linear	none	none	1	trigger to build the physical model
mouse	array	N/A	none	none	linear	none	none	0	set mouse coordinates on LCD
nodesTable	none	N/A	none	none	linear	none	none	1	Show nodes table
refresh	none	N/A	none	none	linear	none	none	1	dump module contents

Return

appForce	array	N/A	none			none	none	0	1	applied force on each node
nodesPos	array	N/A	none			none	none	0	1	return nodes position (x, y)
nodesVel	array	N/A	none			none	none	0	1	return nodes velocity

Jmod.son.mapper - Continuous mapper (PMSon).

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	/dataspace/ unit/native	/repetitions/ allow	/description
clipmode.1	string	N/A	none	none	linear	none	none	1	Choose the clipping mode (none, low, high, both)
clipmode.2	string	N/A	none	none	linear	none	none	1	Choose the clipping mode (none, low, high, both)
clipmode.3	string	N/A	none	none	linear	none	none	1	Choose the clipping mode (none, low, high, both)
clipmode.4	string	N/A	none	none	linear	none	none	1	Choose the clipping mode (none, low, high, both)
clipmode.5	string	N/A	none	none	linear	none	none	1	Choose the clipping mode (none, low, high, both)
connection	array	N/A	none	none	linear	none	none	0	Mapping Connector - input value to Target Module's parameter.
core/edit.1	integer	0 1	none	none	linear	none	none	1	Choose operation mode (active, edit, bypass)
core/edit.2	integer	0 1	none	none	linear	none	none	1	Choose operation mode (active, edit, bypass)
core/edit.3	integer	0 1	none	none	linear	none	none	1	Choose operation mode (active, edit, bypass)
core/edit.4	integer	0 1	none	none	linear	none	none	1	Choose operation mode (active, edit, bypass)
core/edit.5	integer	0 1	none	none	linear	none	none	1	Choose operation mode (active, edit, bypass)
function.1	string	N/A	none	none	linear	none	none	1	select mapping function
function.2	string	N/A	none	none	linear	none	none	1	select mapping function
function.3	string	N/A	none	none	linear	none	none	1	select mapping function
function.4	string	N/A	none	none	linear	none	none	1	select mapping function
function.5	string	N/A	none	none	linear	none	none	1	select mapping function

in/max.1	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	0	Maximum input value
in/max.2	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	0	Maximum input value
in/max.3	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	0	Maximum input value
in/max.4	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	0	Maximum input value
in/max.5	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	0	Maximum input value
in/min.1	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum input value
in/min.2	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum input value
in/min.3	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum input value
in/min.4	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum input value
in/min.5	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum input value
mode	string	N/A	none	none	linear	none	none	1	Mapping mode : -->:mono-directionnal, <-->:bi-directionnal, x-->:excluded from cues
out/max.1	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Maximum output value
out/max.2	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Maximum output value
out/max.3	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Maximum output value

out/max. 4	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Maximum output value
out/max. 5	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Maximum output value
out/min. 1	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum output value
out/min. 2	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum output value
out/min. 3	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum output value
out/min. 4	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum output value
out/min. 5	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Minimum output value
out/ module/ instance .1	integer	0 1	none	none	linear	none	none	1	Destination module instance of the mapping
out/ module/ instance .2	integer	0 1	none	none	linear	none	none	1	Destination module instance of the mapping
out/ module/ instance .3	integer	0 1	none	none	linear	none	none	1	Destination module instance of the mapping
out/ module/ instance .4	integer	0 1	none	none	linear	none	none	1	Destination module instance of the mapping
out/ module/ instance .5	integer	0 1	none	none	linear	none	none	1	Destination module instance of the mapping
out/ module/ name.1	string	N/A	none	none	linear	none	none	0	Destination module of the mapping
out/ module/ name.2	string	N/A	none	none	linear	none	none	0	Destination module of the mapping

out/ module/ name.3	string	N/A	none	none	linear	none	none	0	Destination module of the mapping
out/ module/ name.4	string	N/A	none	none	linear	none	none	0	Destination module of the mapping
out/ module/ name.5	string	N/A	none	none	linear	none	none	0	Destination module of the mapping
out/ module/ parameter.1	string	N/A	none	none	linear	none	none	0	The parameter that the mapped or scaled value will be passed to.
out/ module/ parameter.2	string	N/A	none	none	linear	none	none	0	The parameter that the mapped or scaled value will be passed to.
out/ module/ parameter.3	string	N/A	none	none	linear	none	none	0	The parameter that the mapped or scaled value will be passed to.
out/ module/ parameter.4	string	N/A	none	none	linear	none	none	0	The parameter that the mapped or scaled value will be passed to.
out/ module/ parameter.5	string	N/A	none	none	linear	none	none	0	The parameter that the mapped or scaled value will be passed to.
ramp.1	integer	0 1	none	scheduler	linear	none	none	1	Ramp time
ramp.2	integer	0 1	none	scheduler	linear	none	none	1	Ramp time
ramp.3	integer	0 1	none	scheduler	linear	none	none	1	Ramp time
ramp.4	integer	0 1	none	scheduler	linear	none	none	1	Ramp time
ramp.5	integer	0 1	none	scheduler	linear	none	none	1	Ramp time
slide/ down.1	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value decreases
slide/ down.2	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value decreases
slide/ down.3	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value decreases

slide/ down.4	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value decreases
slide/ down.5	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value decreases
slide/ up.1	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value raises
slide/ up.2	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value raises
slide/ up.3	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value raises
slide/ up.4	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value raises
slide/ up.5	decimal	0.00000 0 1.00000 0	none	scheduler	linear	none	none	1	Smoothing coef when value raises
targetTab	integer	1 5	both	none	linear	none	none	1	select tab in order to output the mapped value

Messages

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/repetitions/ allow	/description
disconnect All	string	N/A	none	none	linear	none	none	1	Disconnect existent connections

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/repetitions/ allow	/ enable	/description
out/ direct	decimal	0.000000 1.000000	none			none	none	0	0	direct result of the mapping

Jmod.son.oscillator~ - Simple oscillator

Configuration

Module Type: audio

Number of signal inlets: 0

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions/ allow	/description
allocVoice	integer	0 1	none	none	linear	none	none	0	choose voice to set parameters
bypass	boolean	0 1	none	none	linear	none	none	0	When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
gain	decimal	0.000000 0 127.000000	none	scheduler	linear	gain	midi	0	Set gain (as MIDI value by default).
mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony
oscillator/ frequency. 1	decimal	0.000000 0 1.000000 0	none	none	linear	none	none	1	Set frequency
oscillator/ onOff.1	boolean	0 1	none	none	linear	none	none	0	toggle on/off
oscillator/ waveForm. 1	integer	0 1	none	none	linear	none	none	1	choose waveform - sine, triangle, saw, noise

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions/ allow	/ enable	/description
audio/ amplitude. 1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude. 2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

Jmod.son.impulse~ - Impulse trigger and train generator

Configuration

jmod.son.impulse~
Module Type: audio

Number of signal inlets: 0
Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmod e	/ramp/ drive	/ramp/ function	/dataspace	dataspace/unit/ native	repetitions/allow	/enable	description
allocVoice	integer	0 1	none	none	linear	none	none	0		choose voice to set parameters
audio/bypass	boolean	0 1	none	none	linear	none	none	0		When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/gain	decimal	0.00000 0 127.000 000	none	scheduler	linear	gain	midi	0		Set gain (as MIDI value by default).
audio/mute	boolean	0 1	none	none	linear	none	none	0		When active, this attribute turns off the module's processing algorithm to save CPU
impulse/duration.1	decimal	0.00000 0 10000.0 00000	low	scheduler	linear	none	none	0		set impulse duration in milliseconds
impulse/frequency.1	decimal	20.0000 00 20000.0 00000	both	scheduler	linear	none	none	0		Set frequency
impulse/period.1	integer	1 10000	low	scheduler	linear	none	none	0		set pulse train period in milliseconds
impulse/train.1	boolean	0 1	none	scheduler	linear	none	none	0		Activate pulse train at regular intervals
impulse/trigger.1	integer	1 1	both	scheduler	linear	none	none	1		send 1 to generate a single impulse
impulse/waveForm.1	integer	0 4	both	none	linear	none	none	0		choose waveform - sine, triangle, saw, noise
numVoices	integer	0 500	low	none	linear	none	none	0		set number of voices - polyphony

jmod.son.karplus~ - simple string synthesis using karplus strong algorithm - uses gen~

Configuration

Module Type: audio

Number of signal inlets: 0

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
allocVoice	integer	0 1	none	none	linear	none	none	0		choose voice to set parameters
audio/ bypass	boolean	0 1	none	none	linear	none	none	0		When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/ gain	decimal	0.00000 0 127.000 000	none	scheduler	linear	gain	midi	0		Set gain (as MIDI value by default).
audio/ mute	boolean	0 1	none	none	linear	none	none	0		When active, this attribute turns off the module's processing algorithm to save CPU
karplus/ dampening .1	decimal	0.00000 0 1.00000 0	both	scheduler	linear	none	none	0		Set dampening
karplus/ decay.1	decimal	0.00000 0 1.00000 0	low	scheduler	linear	none	none	0		Set decay
karplus/ pluck.1	string	N/A	none	none	linear	none	none	1		trigger to pluck the string
numVoices	integer	0 500	low	none	linear	none	none	0		set number of voices - polyphony

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
audio/ amplitude .1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude .2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

jmod.granular~ - Granular Synthesizer using munger~

Configuration

Module Type: audio

Number of signal inlets: 2

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
1704_	integer	-1 1	both	none	linear	none	none	0		allows grains to play backwards and forwards (0), just forwards (1), or just backwards (-1). (0) by default.
allocVoice	integer	0 1	none	none	linear	none	none	0		choose voice to set parameters
audio/ bypass	boolean	0 1	none	none	linear	none	none	0		When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/ gain	decimal	0.00000 0 127.000 000	none	scheduler	linear	gain	midi	0		Set gain (as MIDI value by default).
audio/mix	decimal	0.00000 0 100.000 000	none	scheduler	linear	none	none	0		Controls the wet/dry mix of the module's processing routine in percent.
audio/ mute	boolean	0 1	none	none	linear	none	none	0		When active, this attribute turns off the module's processing algorithm to save CPU
grain/ delaylength_ms.1	integer	0 10000	low	scheduler	linear	none	none	0		constrain how far back in time the module looks for grains. in milliseconds
grain/ durVariation.1	decimal	0.00000 0 300.000 000	low	scheduler	linear	none	none	0		set grain duration in milliseconds
grain/ duration.1	decimal	0.00000 0 300.000 000	low	scheduler	linear	none	none	0		set grain duration in milliseconds
grain/ pitch.1	integer	20 20000	both	scheduler	linear	none	none	0		set grain pitch
grain/ pitchVariation.1	decimal	0.00000 0 1.00000 0	both	scheduler	linear	none	none	0		set grain pitch variation

grain/ position. 1	decim al	0.00000 0 1.00000 0	both	schedu ler	linear	none	none	0	sets the length (in ms) of the ramping envelope on each grain. if the grain is smaller than the ramptime, the ramptime will be scaled to half the grain size.
grain/ rampTime. 1	integ er	0 1000	low	schedu ler	linear	none	none	0	sets the length (in ms) of the ramping envelope on each grain. if the grain is smaller than the ramptime, the ramptime will be scaled to half the grain size.
grain/ rate.1	decim al	0.00000 0 10000.0 00000	low	schedu ler	linear	none	none	0	set grain period in miliseconds
grain/ rateVaria tion.1	decim al	0.00000 0 10000.0 00000	low	schedu ler	linear	none	none	0	set grain position
numVoices	integ er	0 500	low	none	linear	none	none	0	set number of voices - polyphony

Messages

clear	string	N/A	none	none	linear	none	none	0	impolitely clears the internal sample buffer
-------	--------	-----	------	------	--------	------	------	---	---

Return

audio/ amplitude .1	decim al	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude .2	decim al	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

jmod.son.pulsar~ - simple pulsar synthesis module - gen~

Configuration

Module Type: audio
Number of signal inlets: 0
Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspac e	dataspac e/unit/ native	repetition s/allow	/enable	descripti on
allocVoic e	integ er	0 1	none	none	linear	none	none	0		choose voice to set parameters
audio/ bypass	bool ean	0 1	none	none	linear	none	none	0		When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/ gain	decim al	0.000000 127.0000 00	none	schedu ler	linear	gain	midi	0		Set gain (as MIDI value by default).

audio/ mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony
pulsar/ formantFreq.1	decimal	1.000000 10000.00 0000	both	scheduler	linear	none	none	0	Formant Frequency (duty cycle)
pulsar/ freq.1	integer	20 20000	both	scheduler	linear	none	none	0	pulsar frequency
pulsar/ period.1	decimal	1.000000 10000.00 0000	low	scheduler	linear	none	none	0	Pulsar Period in milliseconds
pulsar/ waveForm.1	integer	0 1	none	none	linear	none	none	1	choose pulsar waveform

Return

audio/ amplitude.1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude.2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

jmod.waveshaper~ - stereo 2nd order iir filter based on biquad~

Configuration

Module Type: audio

Number of signal inlets: 2

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/	dataspace	dataspace /unit/ native	repetitions /allow	/enable	description
allocVoice	integer	0 1	none	none	linear	none	none	0			choose voice to set parameters
audio/ bypass	boolean	0 1	none	none	linear	none	none	0			When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/ gain	decimal	0.00000 0 127.000 000	none	scheduler	linear	gain	midi	0			Set gain (as MIDI value by default).
audio/mix	decimal	0.00000 0 100.000 000	none	scheduler	linear	none	none	0			Controls the wet/dry mix of the module's processing routine in percent.

audio/ mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	dataspace /unit/ native	repetitions /allow	/enable	description
audio/ amplitude.1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude.2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

jmod.subtractive~ - stereo 2nd order iir filter based on biquad~

Configuration

Module Type: audio

Number of signal inlets: 0

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	dataspace/ unit/native	/repetitions/ allow	/description
allocVoice	integer	0 1	none	none	linear	none	none	0	choose voice to set parameters
audio/ bypass	boolean	0 1	none	none	linear	none	none	0	When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/ gain	decimal	0.000000 127.000000	none	scheduler	linear	gain	midi	0	Set gain (as MIDI value by default).
audio/mix	decimal	0.000000 100.000000	none	scheduler	linear	none	none	0	Controls the wet/dry mix of the module's processing routine in percent.
audio/ mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
envelope/ attack.1	decimal	0.000000 1000.000000	none	scheduler	linear	none	none	0	specifies the initial values for the attack parameter. milliseconds.
envelope/ decay.1	decimal	0.000000 5000.000000	low	scheduler	linear	none	none	0	specifies the initial values for the decay parameter. milliseconds.

envelope/ release.1	decimal	0.000000 5000.000 000	low	scheduler	linear	none	none	0	set the initial values for the release parameter. milliseconds.
envelope/ sustain.1	decimal	0.000000 0 1.000000 0	low	scheduler	linear	none	none	0	sustain gain - set values for the sustain parameter.
envelope/ trigger.1	integer	0 1	none	none	linear	none	none	0	trigger impulse based on envelope parameters
filter/ filtergain	decimal	-24.000000 0 24.000000 0	both	scheduler	linear	gain	db	0	Gain (dB)
filter/ filtertype	string	N/A	none	none	linear	none	none	0	What kind of filter to use. Possible values: lowpass highpass bandpass bandstop peaknotch lowshelf highshelf
filter/ frequency	decimal	30.000000 0 11025.00 0000	both	scheduler	linear	time	Hz	0	Center frequency (Hz)
filter/q	decimal	0.000000 100.0000 00	both	scheduler	linear	none	none	0	Resonance (Q)
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony
subtractive/ frequency.1	decimal	0.000000 1.000000	none	none	linear	none	none	1	Set frequency
subtractive/onOff.1	boolean	0 1	none	none	linear	none	none	0	toggle on/off
subtractive/ waveForm.1	integer	0 1	none	none	linear	none	none	1	choose waveform - sine, triangle, saw, noise

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	/dataspace/ unit/native	/repetitions/ allow	/enable	/description
audio/ amplitude.1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude.2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

Jmod.son.resonator~ - Resonant filter bank using resonators~

Configuration

Module Type: audio

Number of signal inlets: 2

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions/ allow	/description
a_bang	generic	0.000000 1.000000	none	none	linear	none	none	1	a bang
allocVoice	integer	0 1	none	none	linear	none	none	0	choose voice to set parameters
audio/ bypass	boolean	0 1	none	none	linear	none	none	0	When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/gain	decimal	0.000000 127.0000 00	none	scheduler	linear	gain	midi	0	Set gain (as MIDI value by default).
audio/mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony
resonator/ attenuate	decimal	0.000000 102.0000 00	both	none	linear	none	none	0	attenuate the model, make it softer
resonator/ custom/ decayRate	decimal	0.000000 4.000000	both	none	linear	none	none	0	set decay rate triples
sdif/ stream	integer	0 100	low	none	linear	none	none	0	Sdif - stream number

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ datasp ace	/dataspace/ unit/native	/ repetitions/ allow	/ enab le	/description
audio/ amplitude .1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude .2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

jmod.son.fof~ - Three formants generated by FOF

Configuration

Module Type: audio

Number of signal inlets: 0

Number of signal outlets: 2

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	/dataspace/ unit/native	/repetitions /allow	/description
a_bang	generic	0.000000 1.000000	none	none	linear	none	none	1	a bang
allocVoice	integer	0 1	none	none	linear	none	none	0	choose voice to set parameters
audio/ bypass	boolean	0 1	none	none	linear	none	none	0	When active, this attribute bypasses the module's processing algorithm, letting audio or video pass through unaffected
audio/ gain	decimal	0.000000 127.000000	none	scheduler	linear	gain	midi	0	Set gain (as MIDI value by default).
audio/ mute	boolean	0 1	none	none	linear	none	none	0	When active, this attribute turns off the module's processing algorithm to save CPU
fof/ 1stAmp.1	integer	-60 0	both	none	scheduler	none	none	0	1st Formant - Amplitude
fof/ 1stFreq.1	decimal	10.000000 0 5000.000000	both	none	scheduler	none	none	0	1st Formant - frequency
fof/ 2ndAmp.1	integer	-60 0	both	none	scheduler	none	none	0	2nd Formant - Amplitude
fof/ 2ndFreq.1	decimal	10.000000 0 5000.000000	both	none	scheduler	none	none	0	2nd Formant - frequency
fof/ 3rdAmp.1	integer	-60 0	both	none	scheduler	none	none	0	3rd Formant - Amplitude
fof/ 3rdFreq.1	decimal	10.000000 0 5000.000000	both	none	scheduler	none	none	0	3rd Formant - frequency
fof/ bandwidth.1	decimal	10.000000 0 5000.000000	both	none	scheduler	none	none	0	set bandwidth in Hz
fof/ fundFreq.1	decimal	0.000000 3000.000000	both	none	@description	none	none	0	set fundamental Frequency in Hz
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony

fof/ 2ndAmp.1	integer	-60 0	both	none	schedule r	none	none	0	2nd Formant - Amplitude
fof/ 2ndFreq. 1	decimal	10.00000 0 5000.000 000	both	none	schedule r	none	none	0	2nd Formant - frequency
fof/ 3rdAmp.1	integer	-60 0	both	none	schedule r	none	none	0	3rd Formant - Amplitude
fof/ 3rdFreq. 1	decimal	10.00000 0 5000.000 000	both	none	schedule r	none	none	0	3rd Formant - frequency
fof/ bandwidth .1	decimal	10.00000 0 5000.000 000	both	none	schedule r	none	none	0	set bandwidth in Hz
fof/ fundFreq .1	decimal	0.000000 3000.000 000	both	none	@description	none	none	0	set fundamental Frequency in Hz
numVoices	integer	0 500	low	none	linear	none	none	0	set number of voices - polyphony

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	/dataspace/ unit/native	/repetitions /allow	/enable	/description
audio/ amplitude .1	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 1
audio/ amplitude .2	decimal	0.000000 1.000000	none			gain	linear	1	0	instant amplitude of the signal number 2

Jmod.son.stats - Simple statistics on data stream

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/dataspace	dataspace /unit/ native	repetitions /allow	/enable	description
/max	float	N/A	none			none	none	1	1	maximum value
/min	float	N/A	none			none	none	1	1	minimum value
/min	float	N/A	none			none	none	1	1	mean
/min	float	N/A	none			none	none	1	1	Standard deviation

Jmod.son.getcol - get data (as list) from a desired dataset (jmod.son.table)

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
column	array	N/A	none	none	linear	none	none	1		Name of desired column
dataSet	array	N/A	none	none	linear	none	none	1		Name of desired dataset
dump	none	N/A	none	none	linear	none	none	1		dump selected data

Return

data	array	N/A	none		none	none	1	1		return selected data
------	-------	-----	------	--	------	------	---	---	--	-------------------------

jmod.son.except - remove elements from data

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
undesiredElement	string	N/A	none	none	linear	none	1			type the undesired element

Messages

string	none	N/A	none	none	linear	none	1			return list without the given element
--------	------	-----	------	------	--------	------	---	--	--	--

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
parsedData	array	N/A	none		none	none	1	1		return data without undesired elements

jmod.son.datetime - Return date and time

Parameters

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace /unit/ native	repetitions /allow	/enable	descriptio n
a_bang	gene ric	0.00000 0 1.00000 0	none	none	linear	none	none	1		a bang
mode	strin g	N/A	none	none	linear	none	none	1		select mode: date; time; date&time
start	bool ean	0 1	none	none	linear	none	none	1		start the clock

Return

/name	/type	/range/ bounds	/range/ clipmode	/ramp/ drive	/ramp/ function	/ dataspace	dataspace/ unit/native	/repetitions/ allow	/enable	/description
date	array	N/A	none			none	none	1	1	return current date
time	array	N/A	none			none	none	1	1	return current time