II Encontro de Iniciação Científica e Tecnológica
II EnICT
ISSN: 2526-6772
IFSP – Câmpus Araraquara
26 e 27 de Outubro de 2017

ESTUDO DE UM SISTEMA DE ÁUDIO BINAURAL PARA UTILIZAÇÃO EM EQUIPAMENTO DE ACESSIBILIDADE PARA DEFICIENTES VISUAIS

IVAN ZUCCHI DOS SANTOS[1], RENÊ DE SOUZA PINTO[2], FABIANA FLORIAN[3]
[1]Graduando em Engenharia de Computação, Universidade de Araraquara – Uniara.

Área de conhecimento: Arquitetura de Sistemas de Computação – 1.03.04.02-9

RESUMO: O projeto assistivo SoundSee mapeia obstáculos físicos para pulsos sonoros binaurais a fim de auxiliar pessoas com deficiência visual na detecção dos mesmos. O sistema utiliza a plataforma modular de software SSR, capaz de processar demandas de áudio com baixa latência, porém com massivo consumo de processamento, degradando sua performance em sistemas embarcados com menos recursos disponíveis. Este trabalho tem como objetivo promover um estudo da implementação do software SSR e seus módulos (bibliotecas), assim como as interfaces com diferentes servidores de áudio do sistema operacional Linux para que se encontre pontos de melhoria de performance e forneça diretivas para o desenvolvimento de uma solução unificada e customizada a partir do SSR que satisfaça os requisitos desejados para a plataforma assistiva.

PALAVRAS-CHAVE: SoundSee; SSR; otimização; áudio; binaural.

INTRODUÇÃO

A audição binaural, ou seja, por meio de dois ouvidos, é o que permite aos animais identificar a localização geográfica de origens de emissões sonoras. Golfinhos e morcegos, por exemplo, utilizam-se de biosonares e se orientam pela ecolocalização, emitindo pulsos e sendo capazes de perceber a distância de obstáculos pelo retorno obtido.

O SSR (*SoundScape Renderer*) é uma ferramenta para processamento e reprodução em tempo real de áudio espacial que foi inicialmente desenvolvida no Laboratório de Qualidade e Usabilidade da Universidade Técnica de Berlin e atualmente é mantido e melhorado no Instituto de Telecomunicações da Universidade de Rostock (GEIER;AHRENS; SPORS, 2008), ambos na Alemanha. A ferramenta está licenciada sob a GNU GPL v3 (software livre), executando em sistemas operacionais derivados do UNIX e provendo uma variedade de renderizadores que podem ser utilizados separadamente devido a sua arquitetura modular, como ilustra a Figura 1. Estes renderizadores permitem a aplicação de diversos algoritmos, dentre eles o binaural, com possibilidade de controle dinâmico, seja por interface gráfica, ilustrada na Figura 2, ou via software (através de biblioteca).

Diversas aplicações utilizam o SSR, como o TWO!EARS, um *framework* para modelagem de áudio que se baseia na ligação estrutural da percepção binaural com julgamento e reação, partindo do preceito de que o ouvinte humano desenvolve seu conceito de mundo através da interação exploratória. Uma das mais recentes aplicações consiste em um sistema assistivo para deficientes visuais, denominado SoundSee. O SoundSee utiliza os princípios de ecolocalização e audição binaural para gerar sons que permitam a localização de objetos e obstáculos, especialmente acima da linha da cintura, impossíveis de serem detectados por bengalas. Nele, um software realiza a conversão de dados captados por sensores para uma transmissão de áudio em tempo real que mapeia as distâncias captadas, tornando-as em pulsos binaurais. Para realizar essa conversão, o projeto utiliza o software SSR com o módulo binaural. Entretanto, a performance do sistema tende a ser degradada em ambientes de menor poder computacional, especialmente dispositivos embarcados, devido ao alto consumo de processamento da plataforma SSR.

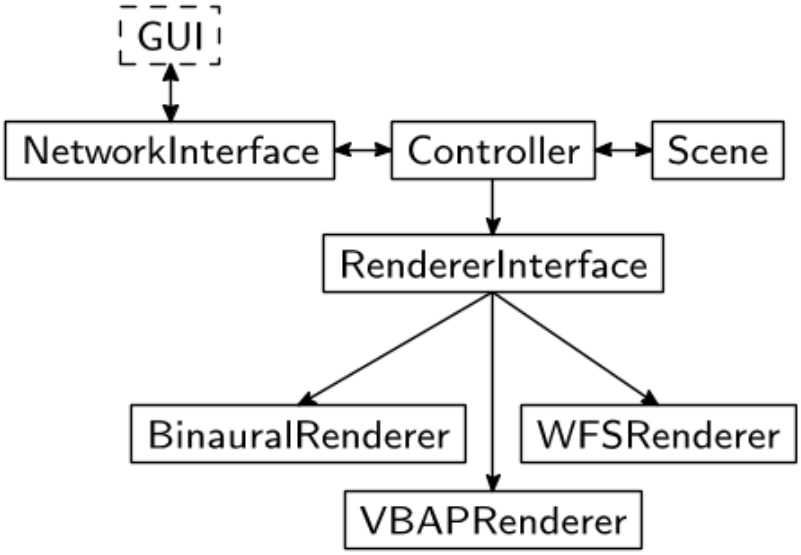

Figura 1. Arquitetura da aplicação SoundScape Renderer.

Fonte: Matthias Geier, 2008.

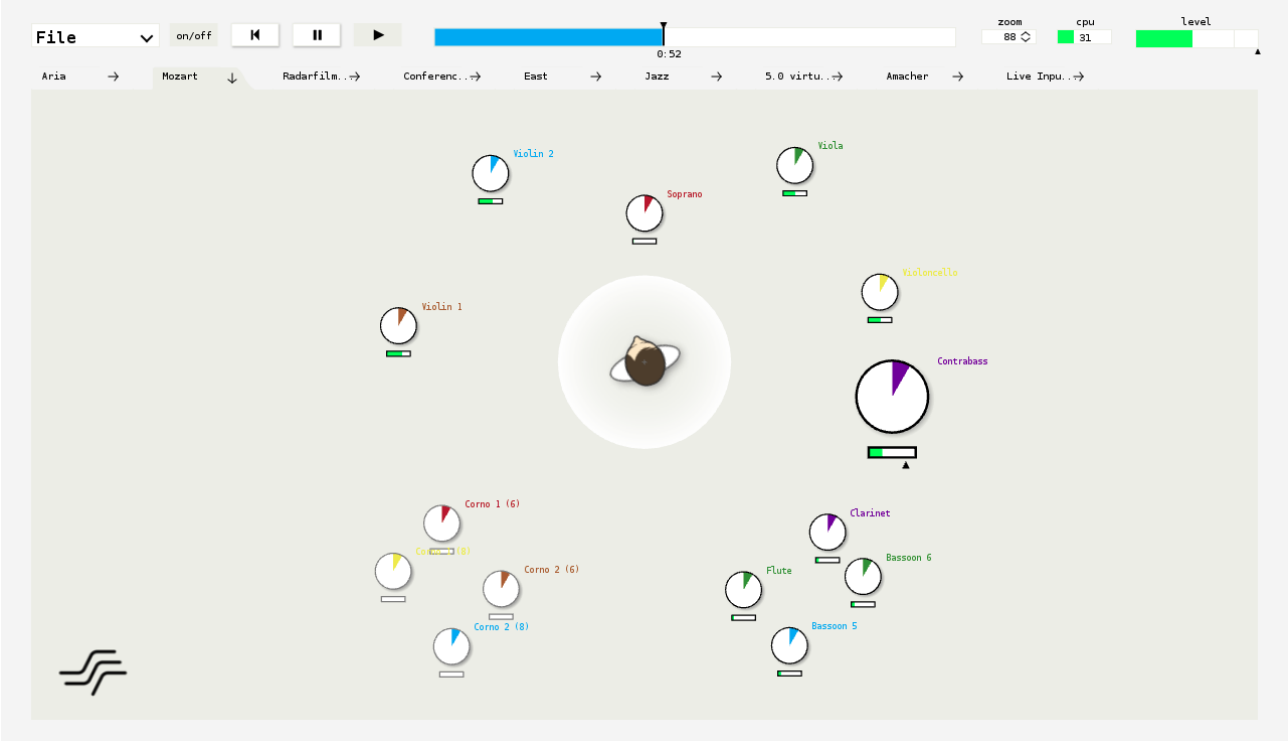

Figura 2. Interface gráfica da aplicação SoundScape Renderer.

Fonte: Matthias Geier, 2008

OBJETIVOS

Este trabalho tem como objetivo avaliar a biblioteca SSR, especialmente o algoritmo de áudio binaural, bem como o funcionamento do processamento de áudio em tempo real no sistema operacional Linux e as diferentes maneiras de funcionamento de seus servidores de áudio (PulseAudio e Jack) para determinar potenciais abordagens para melhoria de performance em ambientes embarcados.

Por se tratar de um software complexo e com diversos módulos implementados, o SSR foi originalmente projetado para executar em uma plataforma PC, com alta capacidade de recursos, tais como memória e processamento. Entretanto, os recursos limitados em sistemas embarcados podem comprometer a utilização da ferramenta nestes ambientes. O SoundSee, apesar de estar implementado em um sistema de propósito geral (Linux), executa em uma plataforma embarcada, necessitando de um sistema personalizado para garantir performance e desempenho de tempo real. A ideia principal é que o estudo proposto determine diretivas para o desenvolvimento de software específico para o projeto que concilie a utilização do módulo de processamento de forma a utilizar a menor quantidade de recursos possível, permitindo melhor performance em ambientes de recursos limitados.

METODOLOGIA

Foi realizada uma pesquisa bibliográfica das ferramentas para utilização no desenvolvimento da aplicação, levantando vantagens e desvantagens do uso de cada uma, assim como o estudo do projeto da arquitetura de áudio do Linux (*ALSA*), o funcionamento de interfaces de áudio em geral, de servidores de áudio do sistema (*JACK*, *Pulseaudio*), o modo de atuação do *framework* de reprodução espacial de áudio (*SoundScape Renderer*) e o *framework* utilizado pelo mesmo para seu processamento (*APF)*.

JACK é um conector para softwares de áudio em ambientes Linux e tem como principais vantagens de sua utilização a facilidade com que interconexões entre aplicações de áudio são feitas e sua baixa latência, sendo bastante utilizado em aplicações de produção de áudio profissional. Para isso, porém, requer um poder de processamento relativamente alto. Seu uso em ambientes embarcados pode degradar a performance do sistema devido a baixa quantidade de recursos de hardware disponível.

ALSA (*Advance Linux Sound Architecture*) foi o nome dado ao projeto que abrange tanto os drivers de áudio em nível de kernel quanto a biblioteca a nível de usuário que os envolve. Interface nativa para os *drivers*, a biblioteca provê funcionalidades de áudio e MIDI para o SO e possui suporte eficiente para todos os tipos de interface de áudio, desde placas simples a interfaces de áudio multicanais de uso profissional. Esta se divide em duas, **Full ALSA** e **Safe ALSA**, sendo a primeira todo o conjunto de APIs e a segunda, um subconjunto mais portátil e compatível entre servidores de áudio que executam sobre o ALSA.

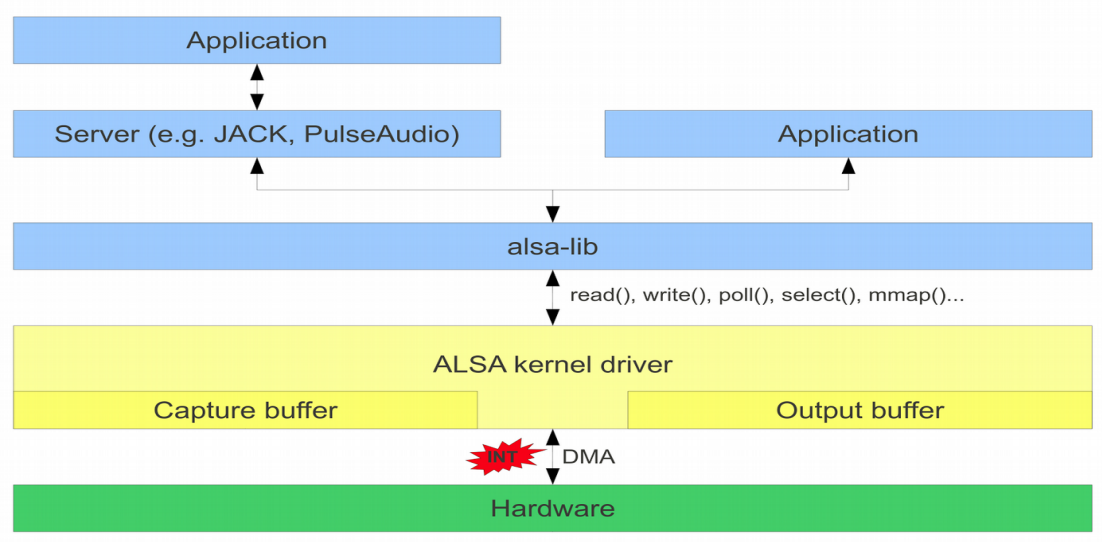

Figura 3. Cadeia de áudio do sistema operacional Linux.

Fonte: REMI LORRIAUX, 2011

Pulseaudio é um servidor de áudio para ambientes Linux *desktop* e embarcados que executam em espaço de usuário e sobre o ALSA. Atualmente, é o padrão instalado em ambientes Debian e seu propósito principal é prover uma interface em alto nível para o controle da conexão entre fontes (programas que reproduzem áudio) e placas de som. Necessita de menor poder de processamento se comparado ao JACK, porém, não é útil em ambientes de produção profissional de áudio.

A Figura 3 ilustra as divisões do ALSA e os níveis de chamadas de cada API. Como demonstrado, o acesso ao hardware só se dá por meio do *driver* a nível de kernel do ALSA, que tem seu acesso restrito a funções de nível maior presentes na biblioteca em nível de usuário. Sendo assim, aplicações que necessitem ativar funções de hardware precisam necessariamente o fazer por meio da utilização desta biblioteca.

SoundScape Renderer é um *framework* para renderização de áudio espacial em tempo real que é composto basicamente por sua própria aplicação *standalone* com interface de usuário (GUI) intuitiva, interface de conexão via *socket* e recursos como salvar e carregar cenas de áudio, e um *framework* modular que permite que seus renderizadores possam ser utilizados como bibliotecas em aplicações C++.

O **APF** (*Audio Processing Framework*) foi apresentado na Linux Audio Conference em Abril de 2012. Criado com o intuito de facilitar o processamento de áudio que estava sendo realizado no SSR e torná-lo genérico, o *framework* pode ser utilizado em outras aplicações C++.

Os requisitos desejados são que os resultados do estudo sejam propícios para o auxílio na criação da solução de software unificada para o projeto.

RESULTADOS E DISCUSSÃO

A habilidade do sistema auditivo humano de localizar sons se baseia em explorar as propriedades acústicas do próprio corpo, especialmente cabeça e os aurículos. Para obter-se a pressão sonora produzida por uma fonte de áudio x(t) em um dos ouvidos, precisa-se encontrar a Resposta de Impulso Relacionada à Cabeça (HRIR) da fonte para o receptor (ouvido). Essa reposta é representada na imagem abaixo como h(t) e é calculada por meio de uma transformada de Fourier H(f) chamada de Função de Transferência Relacionada

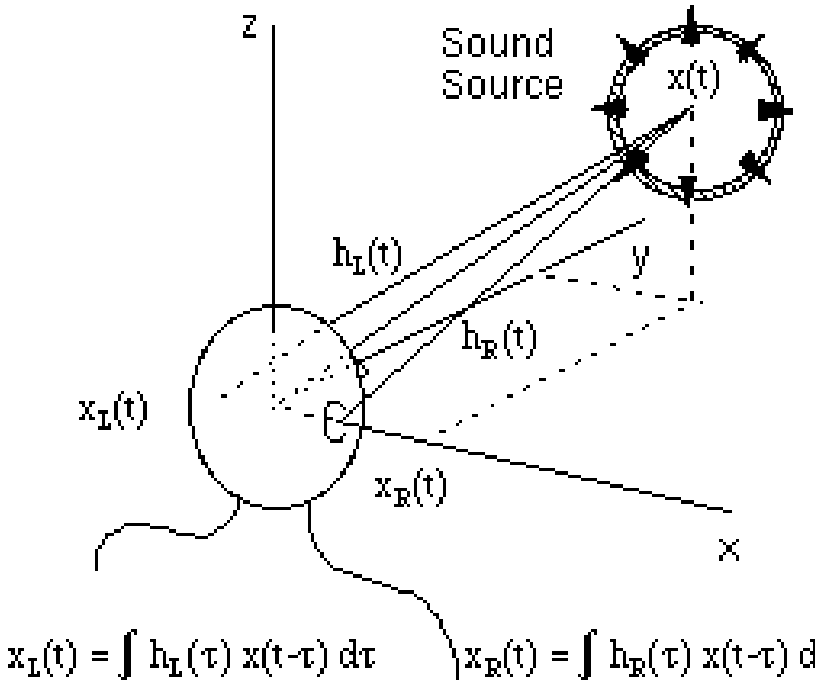

FIGURA 4. Head-Related Impulse Response.

Fonte: THOMPSON, D. 2011

à Cabeça (HRTF), que considera todos os ângulos de incidência das fontes sonoras para as posições dos aurículos na cabeça humana, como mostra a Figura 4.

No APF, a classe MimoProcessor (Multiple-input multiple-output Processor), como o nome descreve, é um processador abstrato de múltiplas entradas e saídas. Ela permite ao desenvolvedor realizar a implementação do processamento em si, sendo a classe central de processamento de áudio do *framework*.

O renderizador binaural é uma subclasse de MimoProcessor e utiliza-se de HRIRs para realizar o processamento do áudio recebido nos canais de entrada e recriar os sinais acústicos nos ouvidos, geralmente por meio de fones de ouvido estéreos. O processamento consiste em filtrar o sinal recebido da fonte sonora aplicando a HTRF e direcionando aos dois canais de saída.

Para que este processador seja utilizado por uma aplicação, é necessário especificar uma política de uso, seguindo o padrão *Policy Design*. A política mais adequada para o uso do renderizador como um módulo *standalone* é a de ponteiros, que disponibiliza uma função (audioCallback) para ser chamada manualmente pelo desenvolvedor quando se é necessário processar algum dado de entrada. Os argumentos recebidos pela função são: tamanho de bloco (número de quadros a serem processados como um bloco durante cada ciclo de áudio), uma série canais de entrada e dois canais de saída.

Para alterar a posição de uma fonte de áudio, basta aplicar um código do tipo:

```
auto∗ source = _engine . get_source(src_id);
source−>position = Position (x , y);
```

O mapeamento entre os canais de entrada é de 1-1 com as fontes de áudio e ambos são ordenados similarmente (channels[0] é a primeira fonte, channels[1] é a segunda...).

No diagrama da figura 5 é possível observar o fluxo do processamento de áudio na aplicação, onde há a captação das posições de fontes de áudio (realizada por sensores ultrassônicos) .

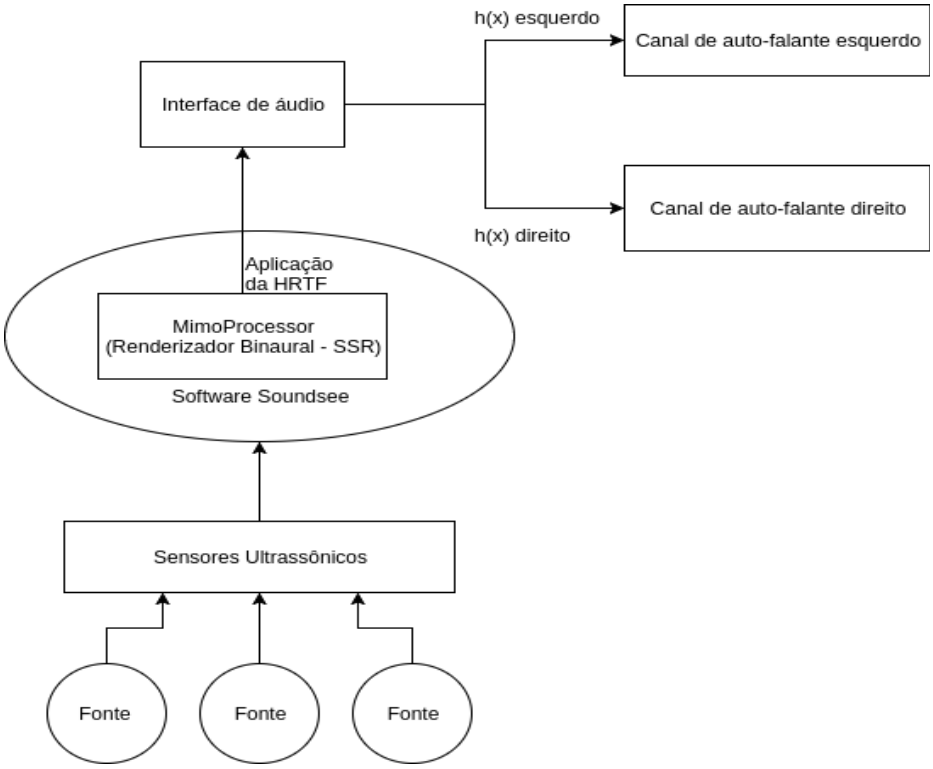

FIGURA 5. Diagrama do fluxo de processamento de áudio pelo software.

Fonte própria, 2017.

A partir do estudo efetuado, pode-se resumir algumas etapas para que uma aplicação faça o uso do renderizador binaural como um módulo independente para uma aplicação C++:

1) Definir uma política de uso (como a política de ponteiros – apf::pointer_policy<T*>);
2) Gerar o mapa de parâmetros de configuração (preferencialmente utilizando o dicionário apf::parameter_map);
3) Definir o tamanho de bloco a ser utilizado;
4) Fornecer ponteiros para matrizes de entrada e saída (podem ser criadas utilizando o *container* apf::fixed_matrix);

É importante ressaltar que estas etapas identificadas constituem um processo geral de desenvolvimento, o presente trabalho ainda encontra-se em fase de desenvolvimento e possui perspectiva de finalização em dois meses, prazo este em que ocorrerá a conclusão do curso de graduação do autor principal. Neste período cada etapa será detalhada em mais processos e recomendações de desenvolvimento para que o uso de módulos independentes possa ser implementado dentro da aplicação SoundSee, levando a economia de processamento e melhor performance em sistemas embarcados.

CONCLUSÕES

A arquitetura com que o *framework* SSR foi projetado permite que seus renderizadores sejam facilmente utilizados separadamente por outras aplicações, tornando possível a produção de um software unificado especificamente criado para o projeto SoundSee ou outros projetos que sejam implementados em sistemas embarcados.

O desenvolvimento do software unificado permitirá que não seja necessária a utilização de softwares terceiros para o funcionamento do projeto, promovendo uma economia de recursos de processamento e a garantia da manutenibilidade e escalabilidade. Este trabalho se propôs a facilitar o desenvolvimento, porém não efetivamente realizá-lo, postergando a implementação de fato para trabalhos futuros. A próxima etapa consiste nesta implementação da solução utilizando apenas os módulos necessários do SSR.

REFERÊNCIAS

[1] GEIER, M.; AHRENS, J.; SPORS, S. “**The SoundScape Renderer: A Unified Spatial Audio Reproduction Framework for Arbitrary Rendering Methods**”. Ernst-Reuter-Platz 7, 10587 Berlin, Germany, 2008. Disponível em: https://www.researchgate.net/publication/228652910_The_SoundScape_Renderer_A_unified_spatial_audio_reproduction_framework_for_arbitrary_rendering_methods

[2] GEIER, M.; AHRENS, J.; SPORS, S; HOHN, T; “**An Open-Source C++ Framework for Multithreaded Realtime Multichannel Audio**”. Ernst-Reuter-Platz 7, 10587 Berlin, Germany, 2012. Disponível em: http://lac.linuxaudio.org/2012/papers/19.pdf

[3] GEIER, M.; SPORS, S. “**Spatial Audio with the SoundScape Renderer**”. Ernst-Reuter-Platz 7, 10587 Berlin, Germany, 2012. Disponível em: http://spatialaudio.net/paper-spatial-audio-with-the-soundscape-renderer/

[4] LORRIAUX, R. “**Real-time Audio on Embedded Linux**” - Embedded Linux Conference, 2011. Disponível em: http://elinux.org/images/8/82/Elc2011_lorriaux.pdf

[5] THOMPSON, D. “**Head-Related Transfer Functions”.** Universidade de Davis, California. Disponível em: http://interface.cipic.ucdavis.edu/images/research/hrir.gif