Arcabouço Orientado a Objetos para simulação Acústica na Plataforma AcMus

Mário Henrique Cruz Torres[1], **Fábio Kon**[1]

[1]Departamento de Ciência da Computação - Universidade de São Paulo

{marioct, kon}@ime.usp.br

Abstract. *In this poster we describe an Object Oriented framework, built upon AcMus System, which is being made in Java language and already counts with an ray tracing algorithm implementation. We also show the easinesses that our framework provide for its users, as simplicity in the comparison between calculated acoustic parameters from the measurement and of the simulation of a room.*

Resumo. *Neste poster descrevemos um arcabouço Orientado a Objetos desenvolvido dentro do ambiente AcMus, o qual está sendo feito totalmente na linguagem Java e já conta com a implementação de um algoritmo de traçado de raios. Mostramos também as facilidades que nosso arcabouço proverá para seus usuários, como a simplicidade na comparação entre parâmetros acústicos calculados a partir da medição e da simulação de uma sala.*

1. Introdução

Uma das propostas do projeto AcMus é construir um ambiente integrado para medição e simulação acústica de salas, o qual visa também a contribuir com ferramentas computacionais para melhorar a qualidade da acústica de salas [Iazzetta et al. 2001]. O sistema AcMus foi desenvolvido usando a linguagem de programação Java e a plataforma Eclipse. A linguagem Java tem como grandes características ser orientada a objetos e ser independente de plataforma, o que significa que programas escritos nesta linguagem funcionam em qualquer sistema operacional. Já a plataforma Eclipse provê um poderoso ambiente, baseado em componentes, para construção de aplicações com interfaces gráficas complexas [Foundation 2006].

O ambiente AcMus foi usado para realizar a medição acústica de salas de concerto na cidade de São Paulo. Os resultados obtidos contribuem para mostrar como o software está maduro na área de medição e cálculo de parâmetros acústicos de salas, como pode ser visto em [Figueiredo and Iazzetta 2005].

Dentro deste ambiente, construímos um arcabouço orientado a objetos para realizar a simulação acústica de salas. Tal arcabouço já conta com o método de traçado de raios e uma fonte sonora pseudo-aleatória implementados, de forma que é extremamente simples comparar os resultados de uma medição com os resultados de uma simulação, a fim de validar o modelo de simulação usado. O que também é uma novidade é este arcabouço ser licenciado sob a LGPL (*Lesser General Public Licence*), sendo único nesse sentido.

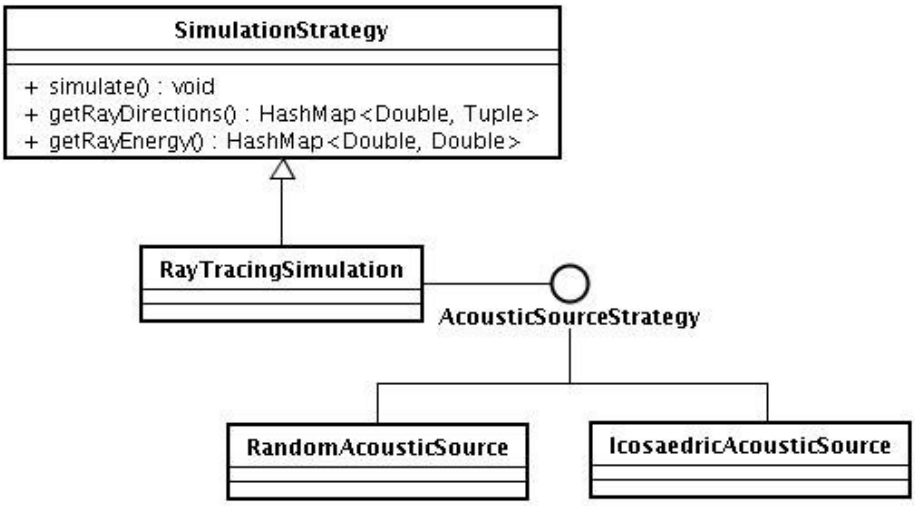

Figure 1. Diagrama de Classes que mostra a estrutura de simulação do arcabouço e a interface *AcousticSourceStrategy* usada para permitir um baixo acoplamento entre o método de simulação e a fonte sonora.

2. Arcabouço

Nosso arcabouço visa a facilitar a simulação e a comparação dos resultados obtidos com dados de uma medição. Para isso, implementamos uma estrutura orientada a objetos totalmente modularizada que permite ao usuário incluir, de maneira simples, seus próprios algoritmos de simulação do AcMus. Esta modularização é atingida com o uso do padrão de projeto (Strategy) [Gamma et al. 1995], o qual define uma família de algoritmos e nos permite trocar o algoritmo usado num certo momento independentemente do resto do sistema.

Já temos implementado um algoritmo de simulação acústica que utiliza o método de traçado de raios descrito por [Kulowski 1985] e uma fonte sonora pseudo-aleatória, na qual, por sua vez, são gerados raios por um método pseudo-aleatório. Ainda estamos desenvolvendo os métodos que acoplam o resultado obtido com a simulação ao cálculo de parâmetros acústicos e aos métodos de comparação entre a simulação e a medição da sala.

2.1. Arquitetura

Descrevemos a seguir as principais classes envolvidas no processo de simulação e a interface que deve ser implementada para criar novas fontes sonoras.

SimulationStrategy

A classe abstrata *SimulationStrategy* deve ser implementada por todos algoritmos de simulação, pois ela é o ponto de ligação entre nosso arcabouço e os métodos de simulação. Essa classe define somente três métodos, sendo o método `generate()` chamado pelo arcabouço para realizar a simulação. Os algoritmos de simulação implementados em nosso arcabouço devem armazenar três informações a respeito da simulação: (i) o tempo e (ii) correspondente energia - que constituem a Resposta Impulsiva Energética (IR), que

é a resposta da sala a um determinado impulso de curta duração - e (iii) o raio que atinge o receptor sonoro. Estas informações são utilizadas para calcular os parâmetros acústicos e para realizar a auralização da simulação. Os outros dois métodos definidos na classe *SimulationStrategy* são usados justamente para recuperar tais informações. São eles: `getRayDirections()`, que devolve um mapa *hash* com os tempos e raios que atingiram o receptor, e `getRayEnergy()`, que devolve um mapa *hash* com os tempos e correspondentes energia dos raios que atingiram o receptor.

É possível ver na Figura 1 a especialização realizada pela classe *RayTracingSimulation* que implementa um algoritmo de traçado de raios implementado de acordo com o método descrito por Kulowski [Kulowski 1985].

AcousticSourceStrategy

Os métodos de simulação acústica que utilizam a técnica de traçado de raios são muito sensíveis à forma como é construída a fonte sonora. Kulowski propõe um modelo de fonte sonora pseudo-aleatória, cuja vantagem é a simplicidade de implementação. Já Lewers [Lewers 1993] discute a validade dos modelos de fontes sonoras, principalmente as geradas por algoritmos pseudo-aleatórios, propondo um modelo onde os raios são gerados a partir de várias subdivisões de um icosaedro. Lewers mostra também como a escolha da fonte sonora pode influenciar os resultados das simulações.

Tendo em vista as distintas necessidades para a escolha das características de uma fonte sonora, criamos nosso sistema usando o padrão de projeto *Strategy*. Com este padrão é possível mesclar o algoritmo de simulação com qualquer fonte sonora implementada ou desenvolver uma nova fonte sonora e usar os algoritmos de simulação para a verificação dos resultados. Esta versatilidade torna nosso arcabouço independente do algoritmo usado para a fonte sonora e permite que novos algoritmos sejam avaliados de forma extremamente simples. O que pode ser visto na Figura 1, onde mostramos uma outra fonte sonora, *IcosahedricAcousticSource*, a qual pode ser implementada de acordo com Lewers.

Para desenvolver um algoritmo de uma nova fonte sonora usando nosso arcabouço, basta criar uma classe que implemente a interface *AcousticSourceStrategy*, a qual tem somente dois métodos, que devolvem uma lista com os raios gerados:

- `generate()`
- `generate(int n)`

2.2. Histograma com a Resposta Impulsiva

Nosso arcabouço provê também uma maneira simples para se construirem gráficos com a resposta impulsiva da simulação. O Histograma com a Resposta Impulsiva pode ser gerado a partir dos valores obtidos com o método `getRayEnergy()`, implementado pelos algoritmos de simulação. Para visualizar este diagrama basta que o usuário clique sobre "Histograma da Resposta Impulsiva".

2.3. Cálculo de Parâmetros Acústicos a Partir da Simulação

Ao realizarmos uma simulação acústica, estamos interessados em saber como é ou como será o comportamento acústico de uma determinada sala. Medidas para descrever tal

comportamento são complexas e muitas vezes não refletem a sensação acústica percebida pelos ouvintes.

O ambiente AcMus apresenta as ferramentas necessárias e já avaliadas para realizar os cálculos de vários parâmetros acústicos, dentre eles: Tempo de Reverberação (RT60, RT30 e RT20), Força Sonora (G), Definição (D_t). Como nosso arcabouço foi desenvolvido integrado ao AcMus, utilizamos os parâmetros acústicos deste sistema. Para realizar o cálculo dos parâmetros acústicos, basta clicar na opção "gerar parâmetros acústicos" e todos parâmetros serão calculados e mostrados na tela.

3. Conclusão

Nosso arcabouço já permite que sejam feitas simulações acústicas de salas, dentro do ambiente AcMus, usando o método de traçado de raios e uma fonte sonora pseudo-aleatória. A integração entre a simulação realizada e os cálculos de parâmetros acústicos do AcMus é mais uma facilidade para o Engenheiro Acústico que não precisará se preocupar em usar vários programas independentes e em lidar com os formatos de arquivos específicos de cada um destes programas.

Por ser um programa livre, esperamos que, em pouco tempo, o AcMus conte com várias implementações de algoritmos de simulação, fato que permitirá a comparação de maneira simples de algoritmos de simulação, dando mais liberdade para o usuário escolher se precisará de mais qualidade na simulação ou tempo de processamento, por exemplo.

O código fonte de nosso arcabouço está disponível em `http://gsd.ime.usp.br/acmus`

References

Figueiredo, F. and Iazzetta, F. (2005). Comparative study of measured acoustic parameters in concert halls in the city of São Paulo. *Proceedings of the 2005 International Congress and Exposition on Noise Control Engineering, Rio de Janeiro, Brazil.*

Foundation, E. (2006). The Eclipse Platform. *Online at http://www.eclipse.org. Visitado em: maio 2007*.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Iazzetta, F., Kon, F., and Silva, F. (2001). AcMus: Design and Simulation of Music Listening Enviroments. *Anais do XXI Congresso da Sociedade Brasileira de Computação (CD-ROM), Fortaleza-CE.*

Kulowski, A. (1985). Algorithmic representation of the ray tracing technique. *Applied Acoustics*, 18(6):449–469.

Lewers, T. (1993). A combined beam tracing and radiant exchange computer model of room acoustics. *Applied Acoustics*, 38(2-4):161–178.