A profícua relação entre Pure Data e áudio para games

Leonardo Porto Passos
PPG em Música / Instituto de Artes
Universidade Estadual de Campinas - Unicamp
Piracicaba/SP, Brasil
leoportopassos@gmail.com

José Fornari
CPG / DM / IA
Universidade Estadual de Campinas - Unicamp
Campinas/SP, Brasil
fornari@unicamp.br

***Resumo* — O Pure Data (Pd) é um linguagem visual de programação em tempo real de código aberto para desenvolvimento multimídia, processamento e geração de som, o que o torna uma excelente ferramenta para compositores e *designers* de som para a criação de trilha e efeitos sonoros para games por conta das vantagens que apresenta: economia de armazenamento digital por reduzir ou eliminar a necessidade de *assets* de áudio, geração de efeitos sonoros por síntese sonora, composição musical em tempo real, criação de ambiências randômicas, manipulação de arquivos de áudio, espacialização, gratuidade e comunidade ativa e colaborativa. Com a possibilidade de integração do Pd com o Unity pela biblioteca LibPdIntegration, as vantagens de utilização do Pd se ampliam e se tornam mais acessíveis e funcionais. O presente artigo apresenta uma pesquisa descritiva de revisão da bibliografia apresentada para compreender e descrever quais são as vantagens de utilização do Pd na área de áudio para games.**

Palavras-chave — Pure Data, Áudio para games, Trilha sonora, Efeitos sonoros, Paisagens sonoras.

I. INTRODUÇÃO

A partir da linguagem de programação Max,[1] criada por Miller Puckette no Ircam por volta de 1985 [1], foi desenvolvido, em 1988, o ambiente gráfico The Patcher, para criação musical em tempo real com controladores MIDI [2]. Em 1997, quando Puckette estava na Universidade da Califórnia, foi lançada a primeira versão do Pure Data (Pd), com a proposta de fornecer os principais recursos do Max e suporte à edição de estruturas de dados compostas de forma mais sofisticada do que a propiciada pelo Max [3]. O Pd ainda possui similaridades com outros *softwares*/linguagens de programação de áudio, como o Csound e o SuperCollider, que também são *open source* e escritos em linguagem C, mas possuem interface mais voltada à escrita de códigos.

A versão inicial do Pd evoluiu para o Pd-extended, que foi descontinuado em sua versão 0.43.4 – lançada em 25/01/2013 (ainda disponível para *download*),[2] e desde então não oferece mais suporte – para dar lugar ao Pd Vanilla. A principal diferença entre ambos é que o Pd-extended vinha com diversas bibliotecas já instaladas e muitas delas eram carregadas automaticamente, e consequentemente ocupava mais espaço em disco: seu instalador possuía 44.7 MB contra apenas 6.9 MB do Pd Vanilla (versões 0.43.4 e 0.51-4, respectivamente). Para que a versão Vanilla também possa oferecer recursos aprimorados, é possível fazer *download* de diversas bibliotecas externas, também chamadas de ***deken***.[3]

Atualmente em sua versão experimental 0.54 e versão estável 0.51-4 (09/06/2021),[4] o Pd é um ambiente de programação visual de código aberto para o desenvolvimento de *softwares* de forma gráfica, o que reduz drasticamente a escrita de linhas de código. Com programação de fluxo de dados (*dataflow programming*), na qual os pulsos de sinais são processados conforme percorrem os objetos, o Pd pode ser usado para controlar, processar e sintetizar sons, vídeos, gráficos 2D e 3D (com uso de *dekens*) e manipular dispositivos de entrada e MIDI. As funções algorítmicas são representadas por caixas visuais chamadas **objetos**, colocados dentro de uma janela de ***patches***[5] chamada **canvas**. O fluxo de dados entre os objetos é obtido por meio de conexões visuais chamadas ***patch cords***. Cada objeto executa uma tarefa específica, que pode variar em complexidade, desde operações matemáticas de nível muito baixo até funções complicadas de áudio ou vídeo, como reverberação, transformações FFT (*fast Fourier transform*, ou transformada rápida de Fourier) ou decodificação de vídeo [4].

O Pd não foi desenvolvido especificamente para games, mas é uma ótima ferramenta que pode ser utilizada tanto para prototipagem quanto para a composição e programação do áudio de um game, como apresentado em [5] e [6], com exemplos práticos. Mas antes da abordagem do assunto, tratemos de tecer um breve histórico do áudio nos games.

II. BREVE HISTÓRIA DO ÁUDIO NOS GAMES

Antes dos videogames, existiam os *penny arcade* e os *pinball* (conhecidos no Brasil como "fliperamas"), máquinas eletromecânicas de jogos que ainda não contavam com sons eletroacústicos, apenas sinos e campainhas, que serviam para atrair os jogadores e provocar seu entusiasmo [7]. Os primeiros videogames, *Tennis for Two* (1958) e *Spacewar!* (1961), não possuíam sons, portanto, os recursos gráficos vieram antes, por isso o nome **video**games. O primeiro videogame com som foi o arcade *Computer Space* (1971), que trazia "Sons de batalha espacial: motores de foguetes e propulsores, disparos de mísseis, explosões" ("*Space battle sounds: rocket and thruster engines, missiles firing, explosions*").[6]

Após as primeiras experiências sonoras rudimentares com os *bleeps* e *bloops* do arcades, e com a grande popularização dos videogames decorrente dos primeiros consoles, como o Magnavox Odyssey (1972), a versão doméstica do Pong (1975) e o estrondoso sucesso do Atari 2600 (1977), começaram a surgir evoluções no áudio para games.

Em 1979, a Mattel apresentou o sistema Intellivision, com um gerador de harmonias sonoras em três partes. Em 1982, a Atari lançou o console 5200, com o processador de áudio dedicado Pokey, cujo *chip* usava quatro canais separados, com controle de *pitch*, volume e distorção. Novos consoles vieram e aumentaram a quantidade de canais de áudio. Em 1986, a Sega trouxe o Master System, com geradores de som

[1] Disponível em: <https://cycling74.com/products/max>.
[2] Disponível em: <https://puredata.info/downloads/pd-extended>.
[3] Ver: <https://puredata.info/docs/faq/deken>.
[4] Disponíveis em: <http://msp.ucsd.edu/software.html>.
[5] Unidades modulares e reutilizáveis de código escritas em Pd e usadas como programas independentes, que podem conter inúmeros *patches* interligados.
[6] Disponível em: <https://flyers.arcade-museum.com/?page=thumbs&db=videodb&id=1531>. Acesso em: 19 jul. 2021.

monofônicos usando quatro oitavas cada, que podia fornecer três canais de onda quadrada e um canal de ruído. E em 1989, o Mega Drive (conhecido como Sega Genesis nos EUA) vinha com 10 vozes com saída estéreo. Dali em diante, os processadores de áudio seguiram sua evolução, adaptando *chips* de sintetizador, processadores de 16-bits, mais vozes, mais memória, algoritmos de compressão e descompressão melhores e até processadores de efeitos internos [8].

Na mesma época, outros tipos de desenvolvimento ocorriam nos jogos de computador, com placas de som separadas com *chips* de sintetizador que permitiam arquivos com pequenas mensagens com instruções ao dispositivo sobre quais sons tocar e quando tocá-los, a partir de um banco com 128 sons, e capacidade de tocar até 16 notas por vez. Ou seja, iniciava-se o uso do padrão MIDI (*Musical Instrument Digital Interface*) nos games [8].

No início da década de 1990, surgiram os jogos em CD-ROM, um grande avanço em relação à síntese FM das placas de som dos computadores de então. Com a nova tecnologia, compositores e *designers* de som podiam agora gravar efeitos sonoros, instrumentos ao vivo e vozes paras os jogos [7].

Na mesma década de 1990, houve outro importante progresso no áudio para games, com a chegada do *surround*, que dá ao ouvinte a percepção de que os sons provêm de fontes sonoras em um espaço tridimensional (3D). E por volta do mesmo período, surgiram os primeiros consoles que traziam áudio com qualidade de CD, caso do Sega Saturn e do Sony PlayStation, ambos de 1994 [7].

A partir daí, novos avanços ocorreram em relação aos recursos de áudio nos games, principalmente no que se refere à taxa de amostragem (*sample rate*), que é o número de vezes que o som é amostrado por segundo, ou o número de medições por segundo. Uma taxa de amostragem de qualidade de CD, de 44,1 KHz, significa que 44.100 amostras por segundo foram registradas. Hoje, os consoles possuem processamento de áudio com capacidade para áudio de 16-bits – que equivalem à 2^16 = 65536 níveis discretos de intensidade sonora, equivalente a uma relação sinal-ruído SNR = $20.\log_{10}$ (65536) =~ 96dB –, com taxa de amostragem máxima de 48 kHz, melhor do que o áudio de um CD [7].

Atualmente, ainda há problemas a serem superados no áudio para games, como questões que envolvem orçamento, tempo e as crescentes expectativas dos jogadores e desenvolvedores. Além disso, há também os problemas tecnológicos, já que o desenvolvimento de *software* não conseguiu acompanhar as atualizações de *hardware* [7].

Um conceito que vêm sendo a tônica quando se trata de sons para games é o áudio dinâmico (também chamado de adaptativo, gerativo, interativo ou procedural), que diz respeito à adaptabilidade da música e dos efeitos sonoros às ações do jogador e a tudo o que está acontecendo no game.

Como a sucessão de fatos e as ações ocorridas dentro de um game não são possíveis de serem antecipadas como num filme, cuja progressão é linear, foram desenvolvidas técnicas para que a trilha, os efeitos sonoros e as vozes de um game reajam em tempo real aos acontecimentos do jogo e às decisões do jogador, esteja ele apenas explorando o mundo do jogo, recolhendo itens ou travando batalhas. O resultado aumenta o poder de imersão do game e enriquece a experiência do jogador, ao permitir que desenvolvedores e equipe de áudio tenham mais controle sobre o clima do jogo, e por isso o uso do áudio dinâmico ver se tornando um padrão no desenvolvimento de games [8].

Por conta dos padrões da indústria de games, ainda é dedicada aos recursos sonoros apenas uma pequena fração do orçamento, armazenamento, processamento e pessoal em um jogo em comparação ao que é destinado aos recursos de vídeo [9]. A questão é que, ainda que os recursos gráficos costumem ter maior relevância em um videogame, com os recursos sonoros a experiência proporcionada ao jogador é ampliada significativamente e recebe novas perspectivas. Não à toa, os videogames se tornaram uma **multi**mídia. As músicas de games vem despertando cada vez mais interesse e é crescente o número de trilhas sonoras disponibilizadas separadamente. Ademais, os sons possuem autonomia e peculiaridades suficientes para proporcionar experiências significativas por si só, e assim, não são poucos os games cujo conteúdo é transmitido por meio do som, caso dos audiogames.

III. VIDEOGAMES E AUDIOGAMES

Audiogames possuem seu conteúdo transmitido de maneira integral (*audio-only-games*) ou predominante (*audio-based-games*) por meio do som [10]. Os jogadores precisam se concentrar na audição para interagir com o sistema do game: percorrer o espaço virtual do jogo, compreender e executar as ações necessárias e interpretar *feedbacks* [11].

Por conta do número limitado ou inexistente de elementos visuais, a paisagem sonora[7] tem grande relevância nos audiogames, já que é um importante recurso para a construção de seu universo ficcional, em conjunto com as descrições, a narração e os diálogos. Assim, a percepção sonora e a habilidade de ouvir são fundamentais. Profissionais de áudio para games têm se interessado cada vez mais pelos audiogames por conta de suas possibilidades de experimentação, do envolvimento direto com a programação do game e dos constantes avanços das técnicas e tecnologias, como o áudio dinâmico e o áudio binaural.

A diminuição ou inexistência de informações visuais pode ampliar habilidades, como a memória e a capacidade de concentração, e também servir de introdução aos conceitos e princípios musicais e de estudos do som, tornando-se uma valiosa aplicação para a área da cognição musical [10].

IV. OS SONS NOS GAMES

É possível categorizar os sons quanto à sua **diegese**, ou seja, o universo espacial-temporal no qual se desenrola a história [16]. Os sons podem ser [17]:

A. *Diegéticos*

Sons presentes no universo ficcional do game e podem ser ouvidos tanto pelo jogador quanto pelos personagens.

B. *Extradiegéticos*

Sons ouvidos pelo jogador, mas que não estão presentes no universo ficcional, e assim, não são ouvidos pelos personagens.

[7] Conceito popularizado por R. Murray Schafer: "Uma paisagem sonora consiste em eventos ouvidos e não em objetos vistos" [12]. Trata-se do equivalente sonoro de uma paisagem visual.

C. *Metadiegéticos*

Sons subjetivos que fazem parte apenas da imaginação, sonho, visão, fantasia de um personagem e são ouvidos apenas na mente do personagem em questão e pelo jogador.

Existem algumas variações no que concerne aos tipos de sons utilizados em games [13, 14, 15]. Buscamos aqui definir, de maneira precisa e sucinta, os elementos sonoros utilizados em games, subdivididos em quatro categorias:

A. *Música (ou trilha sonora)*

Pode ter poucos segundos ou vários minutos de duração e ser executada uma única vez (*one shot*) ou ser reiniciada quando o arquivo de áudio chega ao fim (*loop*). É extradiegética quando faz parte apenas do *background* do game ou do menu, diegética quando é uma música em execução no universo ficcional do jogo (como uma banda ou um disco tocando em um bar) e metadiegética se é ouvida apenas na mente de um personagem.

B. *Efeitos sonoros (ou SFX, do inglês* sound effects*)*

Também podem ser do tipo *one shot* ou *loop*. Existem dezenas ou centenas em um jogo e geralmente são disparados por um "gatilho" ou evento, ou seja, algum acontecimento dentro do jogo. Podem ser diegéticos, como um personagem que anda, quando um personagem pega um objeto, quando um inimigo entra em cena etc.; extradiegéticos, como os sons de menu ou de interface do usuário; ou metadiegéticos, quando acontecem apenas na mente de um personagem.

C. *Ambiências (ou paisagem sonora)*

Também são efeitos sonoros, mas não são tão diretamente reativos, pois não são modificados de acordo com as ações diretas do jogador. Simulam sons de multidão, natureza, animais, maquinário (uma indústria, por exemplo) ou quaisquer sons que componham a paisagem sonora de um ambiente interno ou externo. Geralmente rodam em *loop* quando um jogador entra em um determinado ambiente e param quando o jogador sai deste ambiente.

D. *Vozes*

a) Diálogos: as falas dos personagens, sejam monólogos ou diálogos.

b) Narração: falas mais específicas, de um narrador que não aparece em cena (*voice-over*).

c) Sons não linguísticos: sons emitidos por um personagem, mas que não caracterizam uma fala, como grunhidos, gritos, urros, gemidos etc.

V. Possibilidades de uso do Pd nos games

A. *Áudio dinâmico*

Áudio (música ou SFX) que reage (por inicialização ou modificação) a *inputs* do jogador ou a mudanças no ambiente do jogo. Pode ser subdividido conforme seu nível de dinamicidade, ou seja, sua abertura estrutural ou possibilidade de mudança, manipulação e criação em tempo real [18]:

a) Áudio interativo: construído para responder a *inputs* estritamente relacionados às ações diretas do jogador, como um som de pulo ou disparo de arma.

b) Áudio adaptativo: adaptam-se à narrativa do jogo e vão além dos simples gatilhos acionados pelos *inputs*, com uso de parâmetros que não envolvem o controle direto do jogador, como a situação global do jogo, relações entre objetos e criaturas no ambiente, número de inimigos, intensidade dos acontecimentos, ação e estado do personagem etc.

c) Áudio procedural: não é pré-gravado, mas gerado por algum processo com certo grau de indeterminação, como os algoritmos.

B. *Gravação sonora*

Sons gravados por microfones e que, a princípio, são fixos, ou seja, serão os mesmos toda vez que forem reproduzidos (em *one shot* ou *loop*). Mas podem sofrer alterações (processamento) com técnicas de áudio dinâmico, como adição de reverberação, corte de frequências, mudança de *pitch* etc. Sua disposição "em fileira", de forma sequenciada, caracteriza a técnica de *sampling*, e se a reprodução de pequenos arquivos for disparada por controladores MIDI (ou *sequencers*), tem-se os sons sequenciados [17].

C. *Síntese sonora*

Sons criados do zero por *hardware* eletrônico ou *softwares* que simulam osciladores e filtros. Sintetizadores produzem formas de onda de áudio com características dinâmicas de formato, espectro e amplitude, e criam sons correspondentes a instrumentos reais ou sem referencial preexistente [17].

D. *Som estocástico*

Uso de dados aleatórios ou caóticos que podem ser filtrados para obter ordem por meio de regras matemáticas de estatística e distribuição. Podem ser usados métodos algorítmicos para geração de dados quase aleatórios com alto grau de complexidade e distribuição bem definida. Há aplicações específicas na composição, como determinar o comprimento das notas e a densidade melódica, ou síntese sonora para a obtenção, por exemplo, de som de chuva, tráfego urbano e texturas de passos. O som estocástico pode ser procedural ou interativo, com a entrada do usuário aplicada aos parâmetros da equação geradora ou a filtros subsequentes que operam nos dados gerados [17].

E. *Som algorítmico*

Processo que evolui de acordo com um conjunto simples de regras para a resolução de um problema em um número finito de etapas, um algoritmo, que neste caso, consiste em métodos matemáticos desenvolvidos para encontrar, por métodos iterativos, sequências com propriedades musicais úteis. Na computação normal, um algoritmo deve se concluir o mais rápido possível e retornar um valor no menor número de etapas, ao contrário do som algorítmico iterativo, que busca manter o algoritmo executando suas etapas pelo maior tempo possível, resultando em uma composição musical com modelo abstrato, baseada em formas emocionais dadas pelas regras de harmonia, melodia e ritmo [5].

F. *Espacialização*

Os métodos mais usuais envolvem o som *surround*, que consiste na utilização de fontes de áudio adicionais e independentes para simular ambientes sonoros 3D, com a disposição de um certo número de alto-falantes em torno do ouvinte, como o sistema 5.1 (muito comum em *home theatre*), que usa cinco canais de largura de banda total e um canal de efeitos de baixa frequência (o "ponto um"), ou o sistema 7.1, com sete alto-falantes e mais um *subwoofer* central (emissão das frequências graves). O uso do *surround* pode ser restritivo nos games, já que nem sempre é possível a disposição de diversos alto-falantes em torno do jogador. Por isso, os métodos de restituição 3D baseados no uso de fones de

ouvido, como a síntese binaural, são de grande relevância, já que são baratos e fornecem uma resolução espacial ideal. As técnicas de gravação binaural são caras, pois envolvem microfones especiais, e assim, o que nos interessa aqui é a síntese binaural, que consiste em filtrar as fontes sonoras com técnicas específicas conforme sua posição espacial, o que permite a simulação da especialização sonora por meio de fones de ouvidos comuns (intra-auricular ou *headphone*) [19].

G. Prototipagem

Um problema comum enfrentado por compositores e *designers* de som é a falta de controle sobre a programação do áudio e como será o resultado final do conteúdo sonoro em um game, seja por desconhecimento técnico ou por conta das atribuições específicas de cada profissional em uma equipe de desenvolvimento de games. Por essas razões, a prototipagem de áudio é uma maneira eficaz de realizar testes para a experimentação e análise dos recursos sonoros de um game, pois deixa sob responsabilidade do profissional de áudio o controle sobre o comportamento dos recursos de áudio, sem a necessidade de supervisão e intervenção direta do programador, tornando mais eficiente o processo [20].

VI. INTEGRAÇÃO ENTRE PD & UNITY

Todas as técnicas e recursos apresentados na seção V são possíveis com o Pd, e felizmente existe uma opção gratuita e simples à disposição dos programadores de áudio para a integração do Pd com uma das principais *engines* do mercado.

Como o Pd não é nativamente incorporado a nenhuma *engine* ou *middleware* para programação de áudio, a integração por meio de um *plugin* ou biblioteca é a solução mais eficaz. Método este que foi utilizado pela Electronic Arts no desenvolvimento do game *Spore* (2008), que contou com uma equipe de programadores para incorporar o Pd à engine utilizada para a criação do game, o que possibilitou aos compositores a escrita de partituras procedurais [6].

Esta não é uma opção viável a todos, já que estúdios ou pequenas equipes independentes podem não possuir conhecimento técnico em nem orçamento suficientes para o desenvolvimento de *plugins* ou bibliotecas para integração do Pd à *engine* utilizada. Mas graças ao LibPdIntegration, a integração do Pd ao Unity é possível.

O LibPdIntegration é uma biblioteca desenvolvida por Niall Moody na Abertay University para incorporar *patches* do Pd ao Unity. Atualmente é compatível com Windows, OSX, Linux e iOS e funciona com versões recentes do Unity (2018 em diante). A biblioteca oferece suporte a várias instâncias, o que permite que os desenvolvedores executem vários *patches* em seus projetos do Unity, inclusive permitindo a construção de uma cena 3D com vários *patches* espacializados usando o código de áudio do Unity [21].

VII. CONCLUSÃO

O Pd é uma ótima ferramenta disponível a compositores, *designers* de som e programadores de áudio, que além de proporcionar soluções inovadoras em comunicação sonora para games, também traz uma considerável lista de vantagens: economia de armazenamento digital, geração de efeitos sonoros por síntese sonora, composição musical em tempo real, criação de ambiências randômicas, manipulação de arquivos de áudio, espacialização, gratuidade e suporte com uma extensa e colaborativa comunidade de usuários. Todos as técnicas e usos de sons apresentados no presente artigo são possíveis com o Pd, e sua integração com o Unity por meio da biblioteca gratuita LibPdIntegration torna sua utilização acessível e uma excelente alternativa para a prototipagem, programação e criação de recursos sonoros para games.

O LibPdIntegration carece de certos incrementos, como a compatibilidade com Android e WebGL, e iniciativas para essas atualizações seriam bem-vindas para a comunidade de desenvolvimento de games, o que permitiria novas possibilidades para programadores de áudio, compositores e designers de áudio na prototipagem e desenvolvimento final dos recursos de áudio para games, além de uma possível efetivação da utilização do Pd em games.

Como possibilidades futuras, estamos realizando testes e criando protótipos de microgames de áudio para serem utilizados dentro de um audiogame (*only-audio-game*, com *inputs* por comando de voz, áudio dinâmico, espacialização binaural, composição musical computacional em tempo real e síntese sonora para criação de SFX e ambiências randômicas) em desenvolvimento como estudo de caso para a dissertação de um curso de mestrado em andamento.

REFERÊNCIAS

[1] E. Favreau *et al.*, "Software developments for the 4X Real-time System," in *1986 Proc. ICMC*, San Francisco, pp. 369-373.

[2] M. S. Puckette, "The Patcher," in *1988 Proc. ICMC*, San Francisco, pp. 420-429.

[3] M. S. Puckette, "Pure Data," in *1997 Proc. ICMC*, San Francisco, pp. 420-429.

[4] IEM. "Home." Pd Community Site. http://puredata.info/ (accessed Jul. 15, 2021).

[5] A. Farnell, "An introduction to procedural audio and its application in computer games," CiteSeerX Repository, September 2007. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.531.2707>.

[6] A. Farnell, *Designing sound*. Cambridge, MA, USA: MIT Press, 2010.

[7] K. Collins, *Game sound: an introduction to the history, theory, and practice of video game music and sound design*. Cambridge, MA, USA: MIT Press, 2008.

[8] A. Marks, *The complete guide to game audio: for composers, musicians, sound designers, and game developers*. 2th ed. Oxford, UK: Focal Press: 2009.

[9] M. D. Wilde, *Audio programming for interactive games: the computer music of games*. Oxford, UK: Focal Press, 2004.

[10] E. Rovithis, A. Mniestris, and F. Floros, "Educational audio game design: sonification of the curriculum through a role-playing scenario in the audio game 'Kronos'," in *Proc. 9th Audio Mostly*, New York, NY, USA, 2014, Article 21, pp. 1-6.

[11] E. Rovithis, F. Floros, and L. Kotsira, "Educational audio gamification: theory and practice," in *Proc. 17th Ecel*, Athens, Greece, 2018, pp. 497-505.

[12] R. M. Schafer, *A afinação do mundo*. São Paulo, Brazil: Unesp, 2001.

[13] L. H. M. Alves, J. M. Silva Jr., and C. S. Araújo, "Desenvolvimento de áudio para jogos com Unity e FMOD," in *Proc. 16th SBGames*, Curitiba, Brazil, 2017. Porto Alegre, Brazil: SBC, 2017, pp. 1289-1299.

[14] E. S. Boury and P. N. Mustaro, "Um estudo sobre áudio como elemento imersivo em jogos eletrônicos," in *Proc. 12th SBGames*, São Paulo, Brazil, 2013. Porto Alegre, Brazil: SBC, 2013, pp. 345-352.

[15] S. Huiberts, "Captivating sound: the role of audio for immersion in computer games," Ph.D. dissertation, Univ. of Portsmouth, Portsmouth, UK, 2010.

[16] C. Reis and A. C. M. Lopes, *Dicionário de teoria da narrativa*. São Paulo, Brazil: Ática, 1988.

[17] C. Gorbman, *Unheard melodies: narrative film music*. Bloomington, IN, USA: Indiana University Press, 1987.

[18] L. Meneguette, "Áudio dinâmico para games: conceitos fundamentais e procedimentos de composição adaptativa," in *Proc. 10th SBGames*, Salvador, Brazil, 2011. Porto Alegre, Brazil: SBC, 2011, pp. 1-10.

[19] D. Doukhan and A. Sédès. (Jul. 2009). CW_binaural~: a binaural synthesis external for Pure Data. Presented at 3th Pd Inter. Conv. [Online]. Available: https://puredata.info/community/conventions/convention09/doukhan.pd.

[20] L. Paul. "Audio prototyping with Pure Data". Game Developer. https://www.gamedeveloper.com/audio/audio-prototyping-with-pure-data (accessed Jul. 21, 2021).

[21] N. Moody. "LibPdIntegration v2.1.2". GitHub. https://github.com/LibPdIntegration/LibPdIntegration (accessed Jul. 21, 2021).