# Motor de Jogos e Multimídia

## Sistema de imagem e áudio de uma *game engine*

Slides por:
Gil Barbosa Reis (FoG - ICMC)
Gustavo Ferreira Ceccon (FoG - ICMC)


Este material é uma criação do
Time de Ensino de Desenvolvimento de Jogos Eletrônicos (TEDJE)
Filiado ao grupo de cultura e extensão Fellowship of the Game (FoG), vinculado ao ICMC - USP

# Objetivos

- ➔ Mostrar o que são recursos e como gerenciá-los
- ➔ Introduzir como áudio e imagem são representados
- ➔ Explicar como áudio e imagem são utilizados em jogos
- ➔ Mostrar alguns efeitos de transformação de áudio e imagens

# Índice



# 1. Introdução

# 1. Introdução

- Recursos
  - São arquivos adicionais e conteúdo estático que o código utiliza
  - Exemplos: imagens, texturas, modelos 3D, materiais, sons, músicas, fontes, shaders...

# 1. Introdução

- ➔ Carregamento
  - Síncrono (bloqueante)
  - Assíncrono (não bloqueante - loading screen)
  - Fonte: memória, disco, internet
  - Streaming

# 1. Introdução

- ➔ Quando e o que carregar?
  - Tudo de uma vez (Mario)
  - Carregar em pontos estratégicos (FPS)
  - Constantemente (mundo aberto)

# 1. Introdução

- Armazenamento e Distribuição
  - Ter muitos arquivos pode ser problemático
    - Disco tem que fazer muitos seeks e arquivos pequenos ocupam tamanho mínimo
  - Solução: compactar em um ou mais arquivos grandes

# 1. Introdução

- Compactação
  - Arquivo contínuo - carregamento mais rápido
  - Pode ou não haver compressão
  - Arquivos a serem compactados juntos podem seguir Game Design

# 1. Introdução

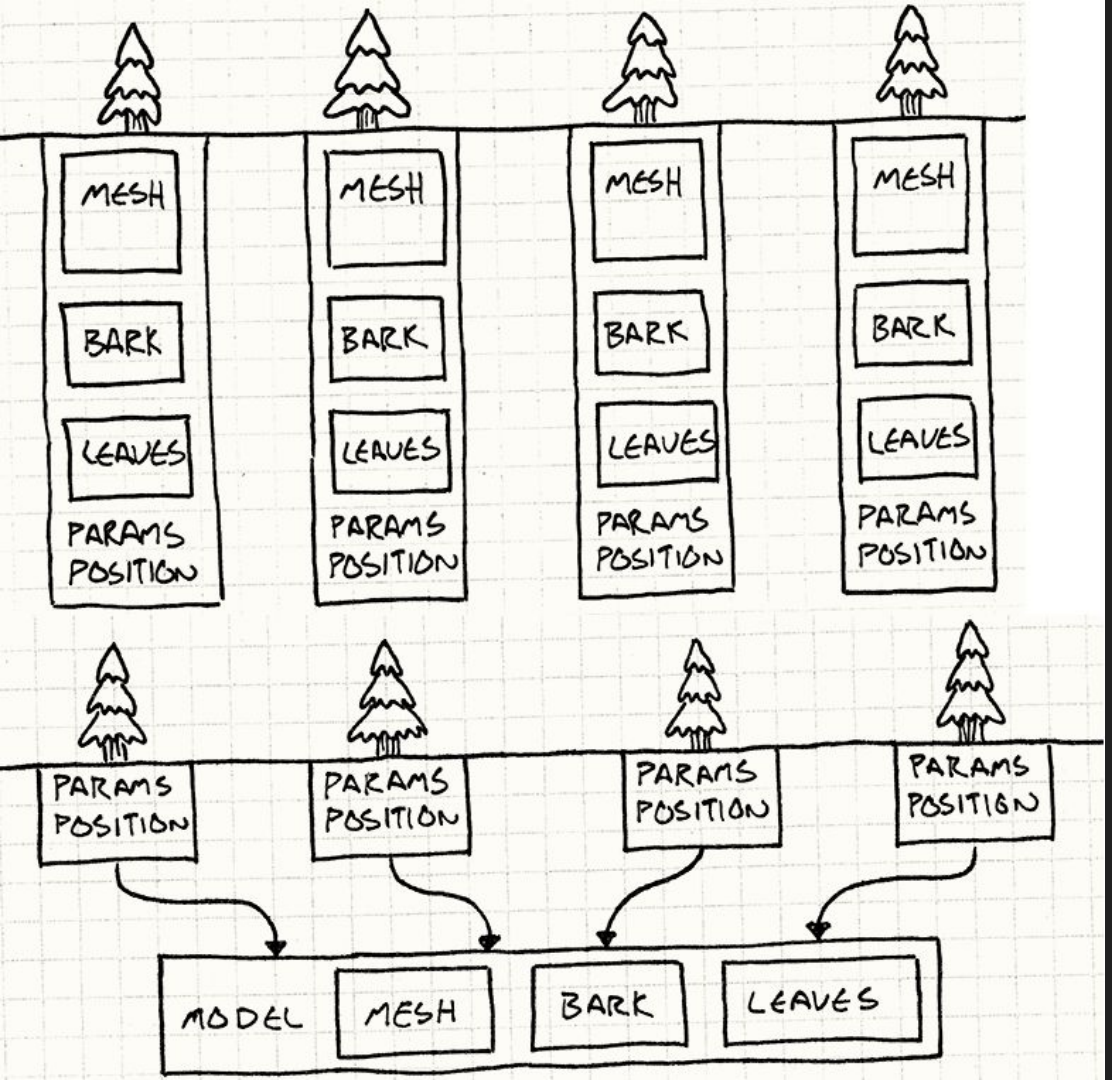
- Compressão
  - Reduzir o tamanho do arquivo, geralmente tirando redundância
  - Pode ser:
    - Raw
    - Lossy
    - Lossless

# 2. Gerenciamento de Recursos

# 2. Gerenciamento de recursos

- Aspectos importantes
  - Memória
  - Descarregamento quando não mais utilizado
  - Reutilização: Flyweight

# Flyweight

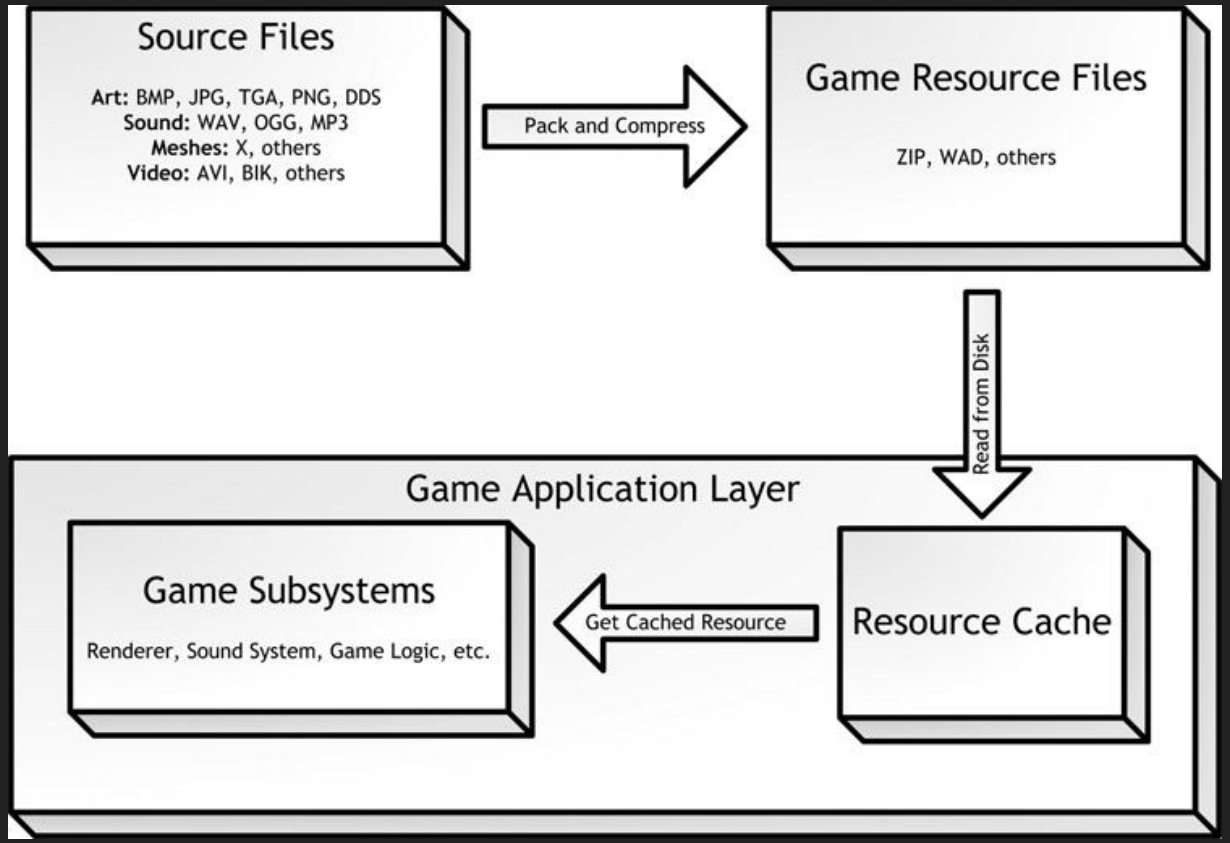



Robert Nystrom. Game Programming Patterns.

# 2. Gerenciamento de recursos

- Gerenciamento automático
  - Cache - guarda recursos mais utilizados
    - Recursos podem ter prioridades
    - Algoritmos de substituição
  - Gerenciamento de memória (ex.: garbage collection)
  - Recarregamento de recursos em tempo de execução

# Cache de recursos




McShaffry, Mike & Graham, David. Game Coding Complete 4th Edition

# 3. Imagem

# 3. Imagem

- ➔ Paleta de cores
  - ◆ Como representar valores em cores reais (LED), é o mapeamento de cor
  - ◆ Cada monitor/software usa a própria paleta de cores, por isso jogos podem parecer diferentes em outros monitores

# 3. Imagem

- Color space
  - Mesmo conceito de paleta, é um mapeamento de cor
  - Gamma (luminosidade): exponencial
  - sRGB (vermelho, verde, azul): não linear
  - Adobe RGB (vermelho, verde, azul)
  - HSV/HSB (Hue, Saturation, Value/Brightress)

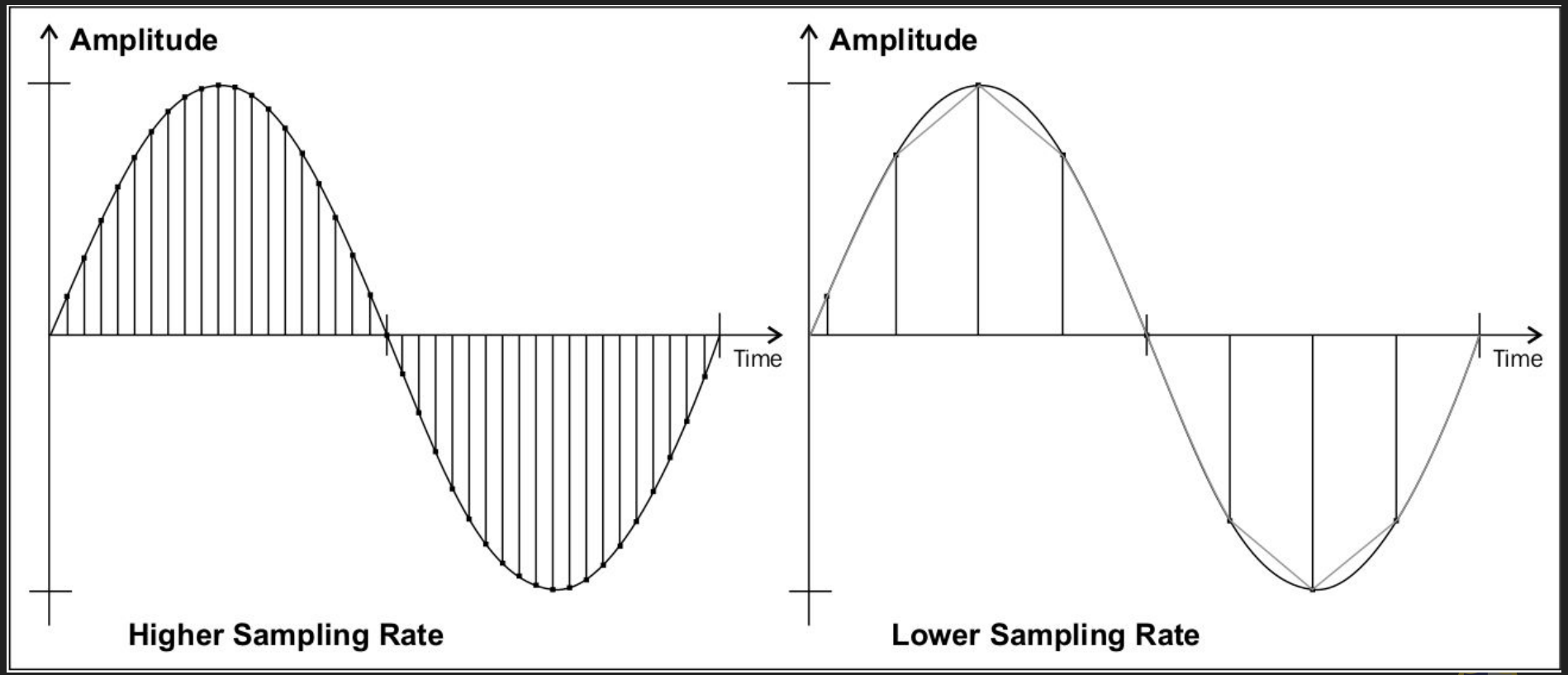
# 3. Imagem

# 4. Áudio

# 4. Áudio

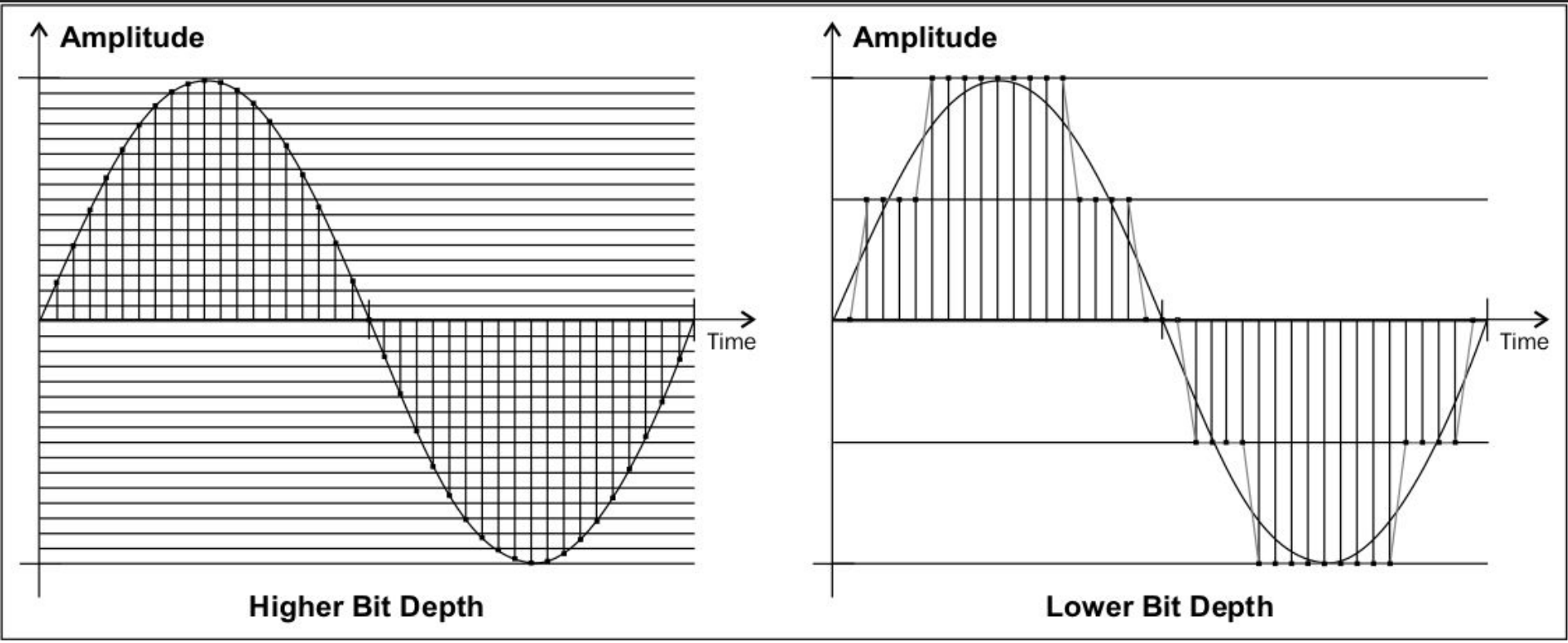
- Representação
  - Taxa de amostragem: discretização de tempo
  - Quantização: discretização de um sinal contínuo
    - Salva cada amostra como um número
    - Formatos mais comuns: Inteiros de 16 bits, Ponto flutuante de 32 bits (valores entre -1.0 e 1.0)
  - Sinal digital é chamado de Pulse Code Modulation (PCM)

# Taxa de amostragem



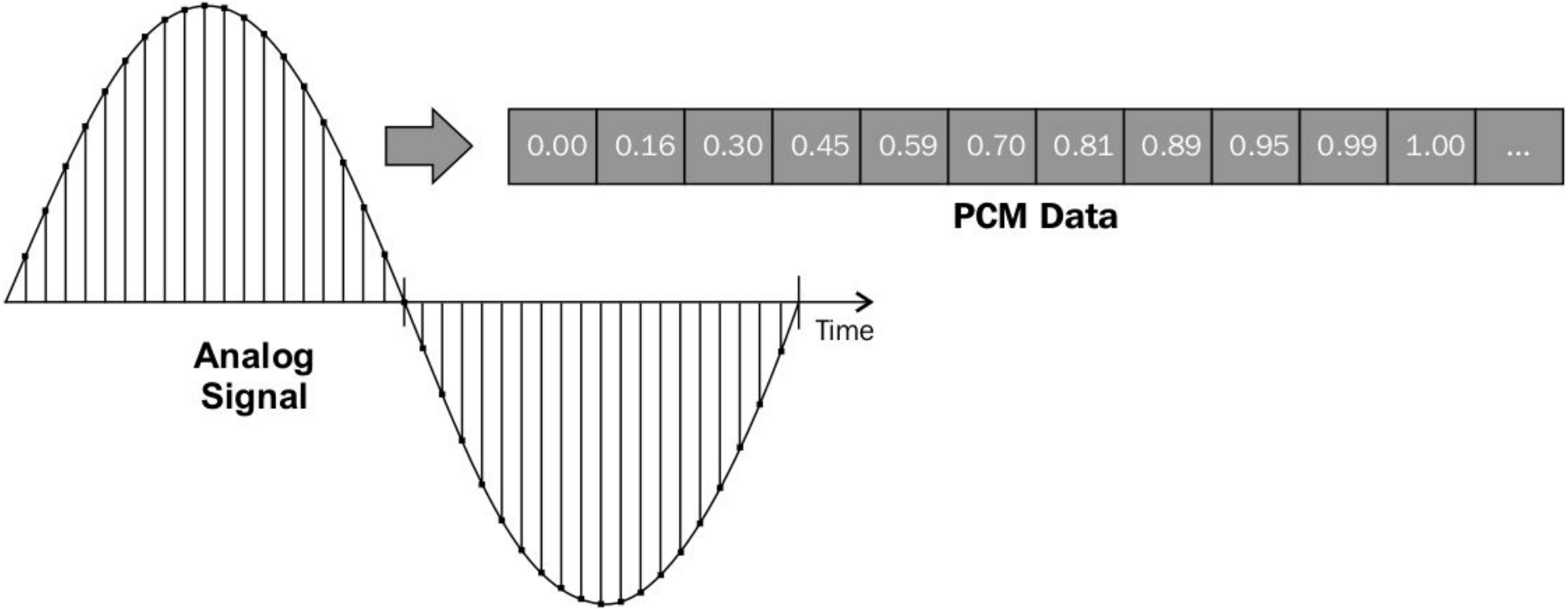

Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# Quantização (Profundidade de Bit)




Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# *Pulse Code Modulation* (PCM)

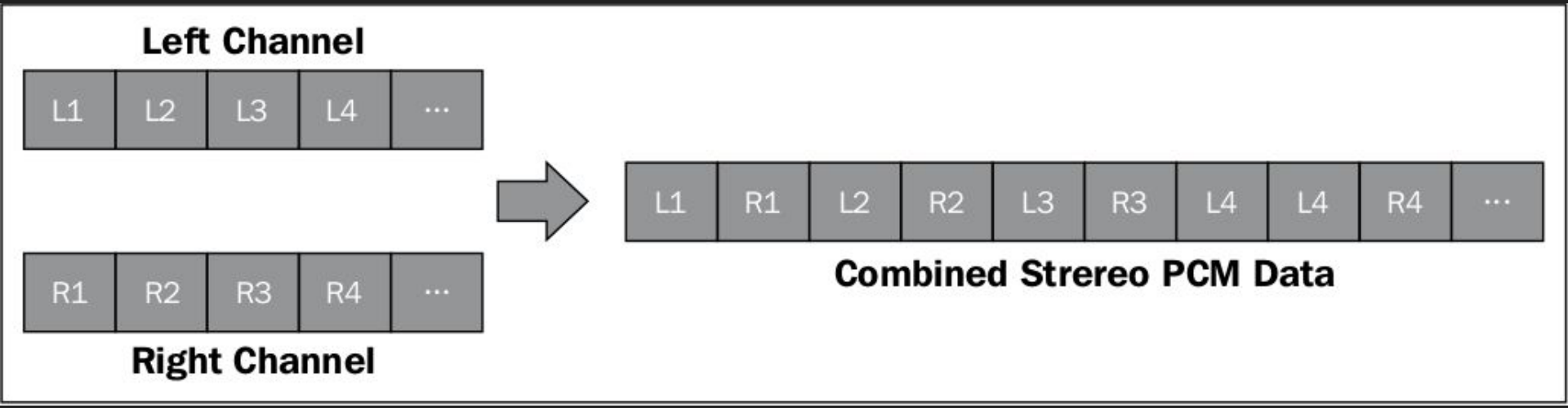



Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# 4. Áudio

- Representação de canais
  - Mono (1.0)
  - Stereo (2.0)
  - Surround (5.1, 7.1)

# Multicanais




Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# 4. Áudio

- Formatos
  - Sem compressão: RAW, WAVE, AIFF
    - Representa diretamente o PCM
  - Compressão sem perdas (lossless): FLAC
  - Compressão com perdas (lossy): MP3, OGG, ACC
    - Elimina componentes do som que seres humanos normalmente não percebem diferença

# 4. Áudio

- Formatos
  - Efeitos sonoros curtos (SFX) geralmente estão em formato PCM e são carregados completamente na memória
  - Músicas maiores normalmente estão em formato comprimido e é usado o streaming: carrega na memória conforme a música vai tocando

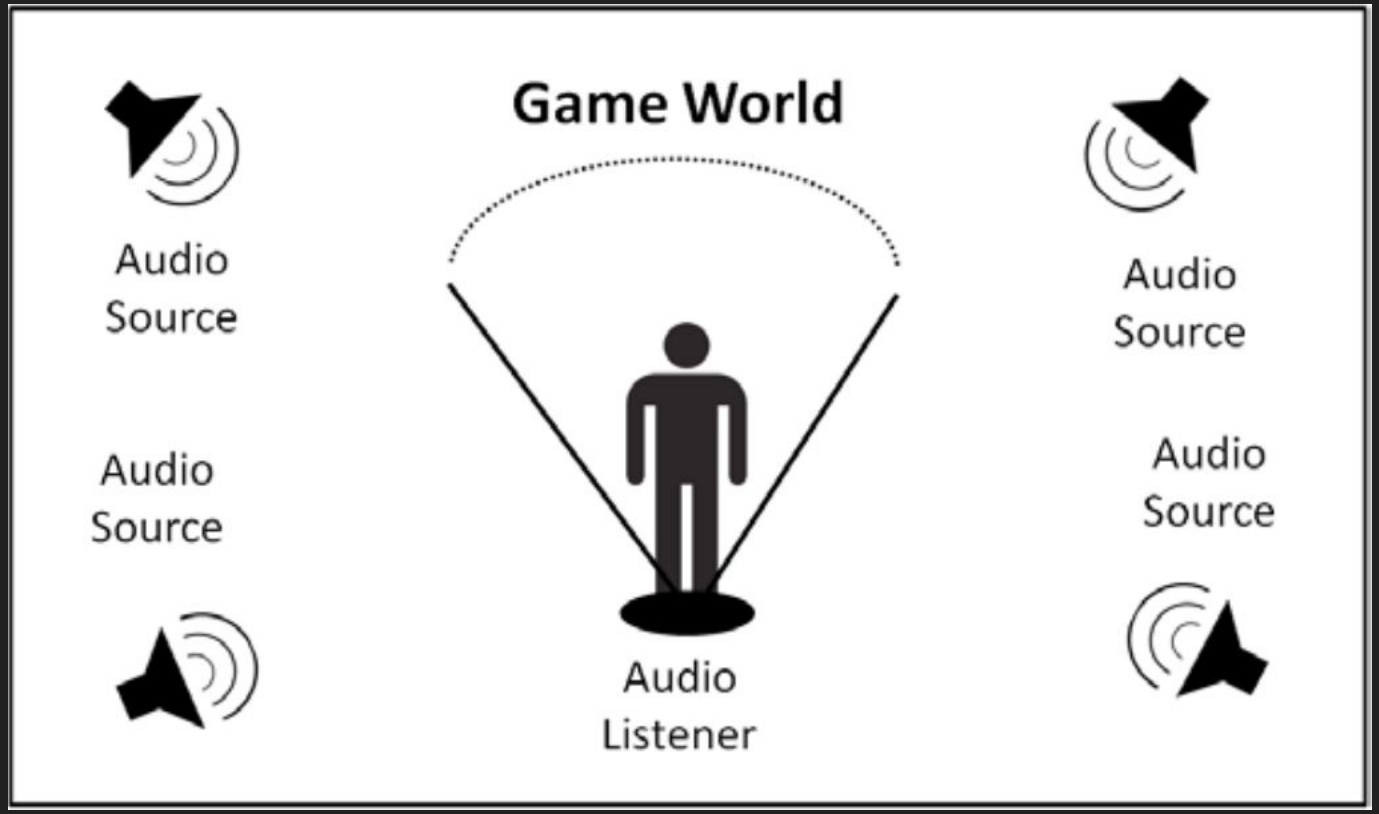
# 4. Áudio

- Música sequencial: MIDI, MOD
  - Contém informação sobre como executar o som, não o que é o som
  - Padrão baseado em eventos, como Note ON/OFF
  - Sequenciador, como partituras
  - MOD = MIDI + instrumentos

# 4. Áudio

- Áudio espacial
  - Baseado em um Ouvinte e Fontes
  - Panning: percepção de onde vem o som
    - Muito importante em alguns jogos, como FPS
  - Atenuação de volume baseado na distância entre ouvinte e fonte (rolloff)

# Áudio Espacial



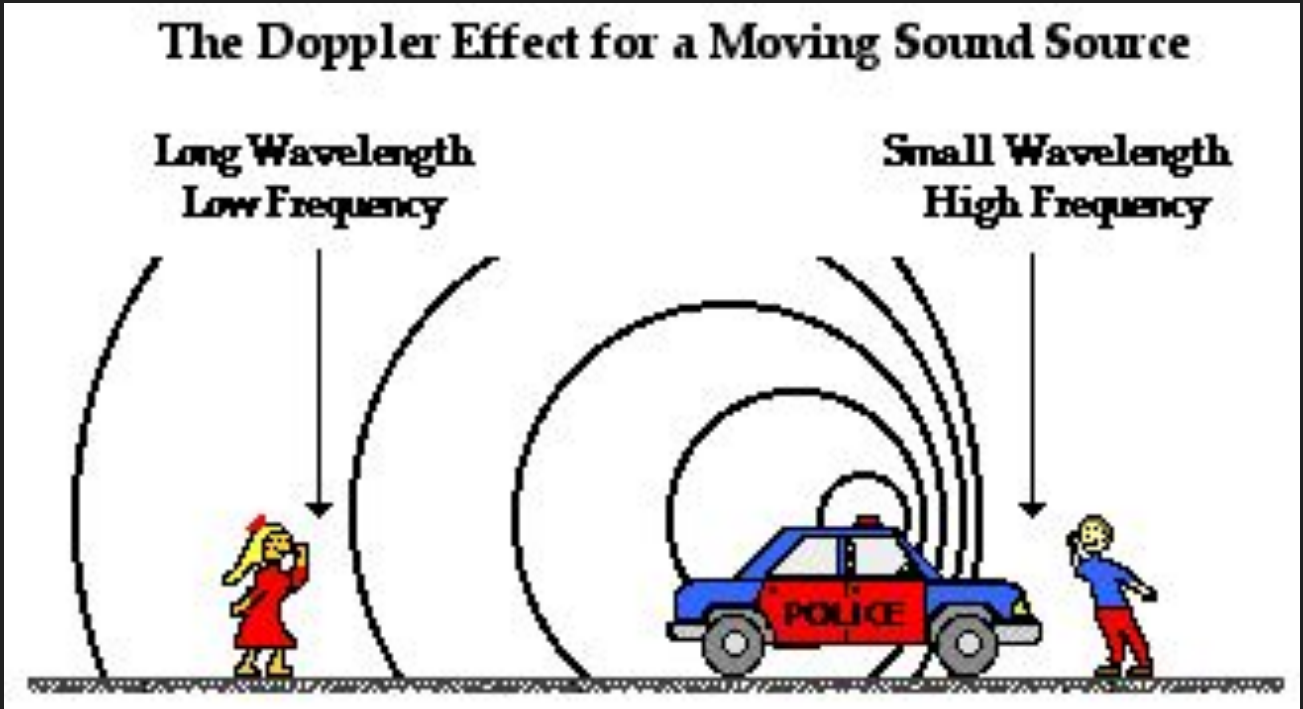

Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# 4. Áudio

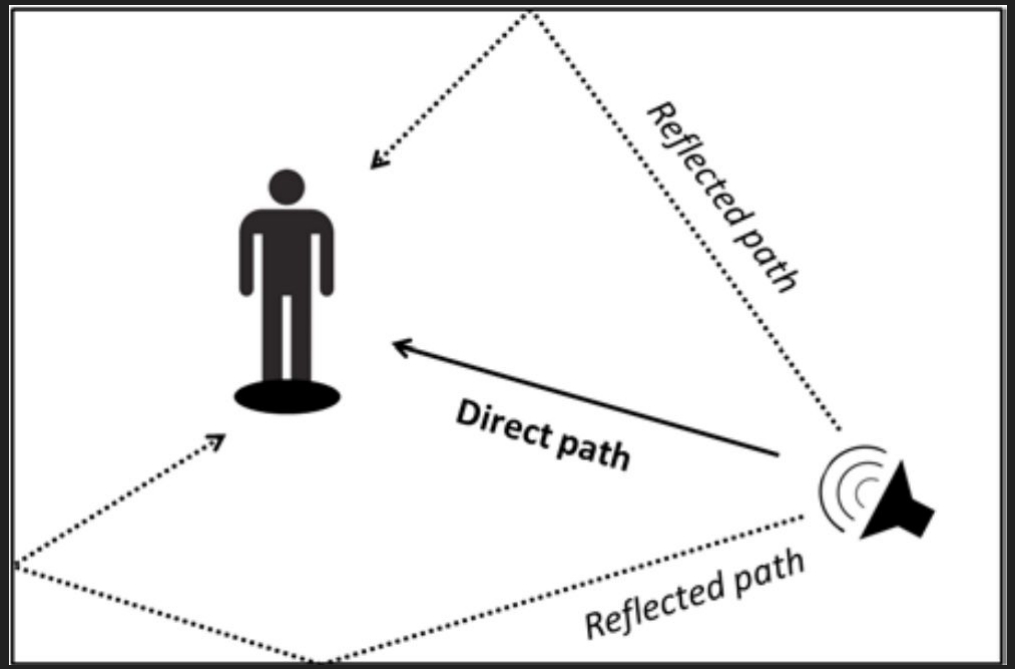
➔ Efeitos físicos do som

- Doppler: variação na altura (*pitch*) de um som baseado na velocidade relativa entre ouvinte e fonte
- Reverberação e Eco: quando ouvimos a persistência de um som logo após ser extinta sua emissão
  - Eco > 50~100ms

# Efeito Doppler



# Reverberação




Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# 4. Áudio

- Ouvinte (Listener)
  - Posição: panning, distância
  - Velocidade: efeito Doppler
  - Vetor cima e frente: panning

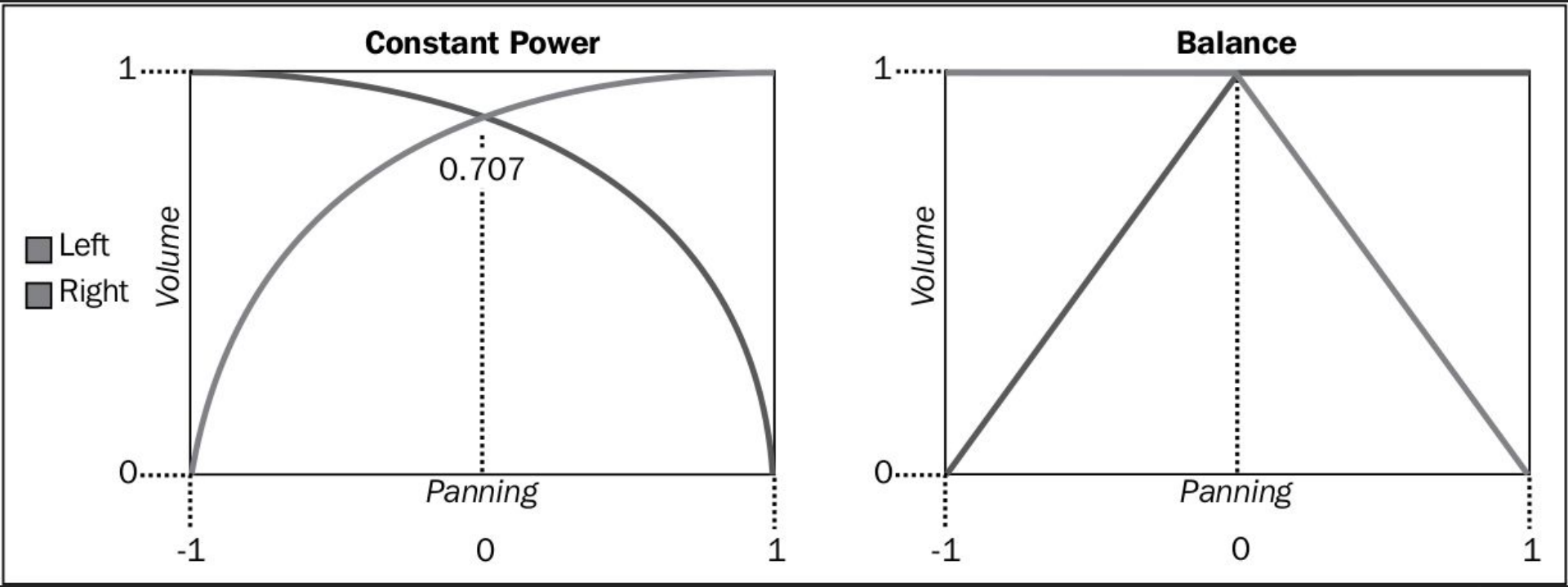
# 4. Áudio

- Fonte (Source)
  - Posição: panning, distância
  - Velocidade: efeito Doppler
  - Intervalo de distâncias onde o som se atenua (rolloff)
  - Direção: pode ser omnidirecional ou direcional (cone)
    - Se direcional: ângulos interno e externo do cone

# 4. Áudio

- Áudio 2D
  - Mono: interpolação usando curva de potência constante onde o centro é ~71%
  - Stereo: interpolação linear, sendo o centro 100%

# *Panning* em 2D com potência constante

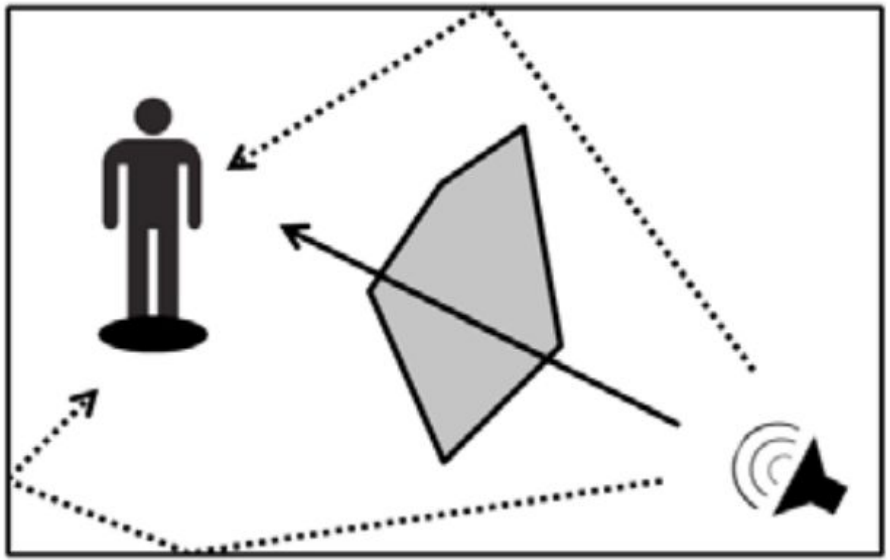


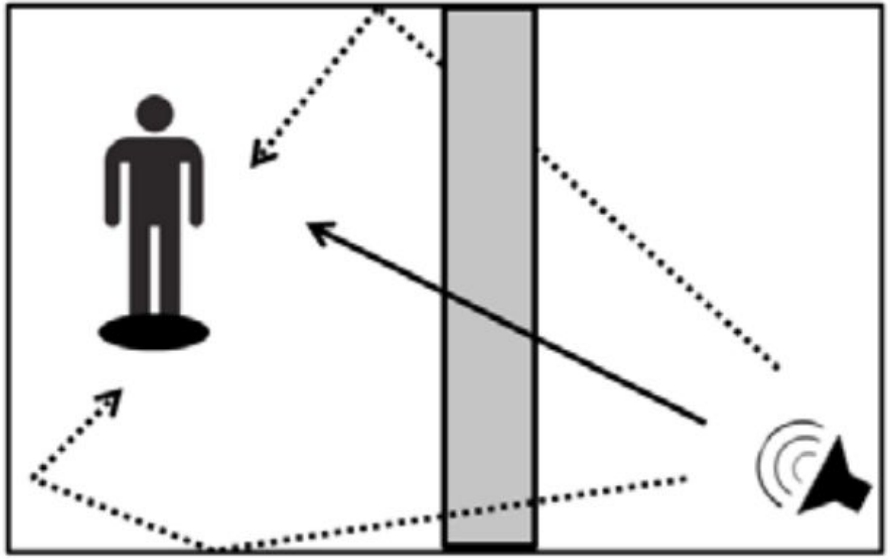

Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# 4. Áudio

- Áudio 3D
  - Obstrução: parede fina, que atenua o som
  - Oclusão: parede de concreto, que bloqueia som
  - Podem ser calculados em runtime usando a geometria do mundo
    - Pesado para calcular
    - Atenuação mais um filtro passa baixa podem dar conta

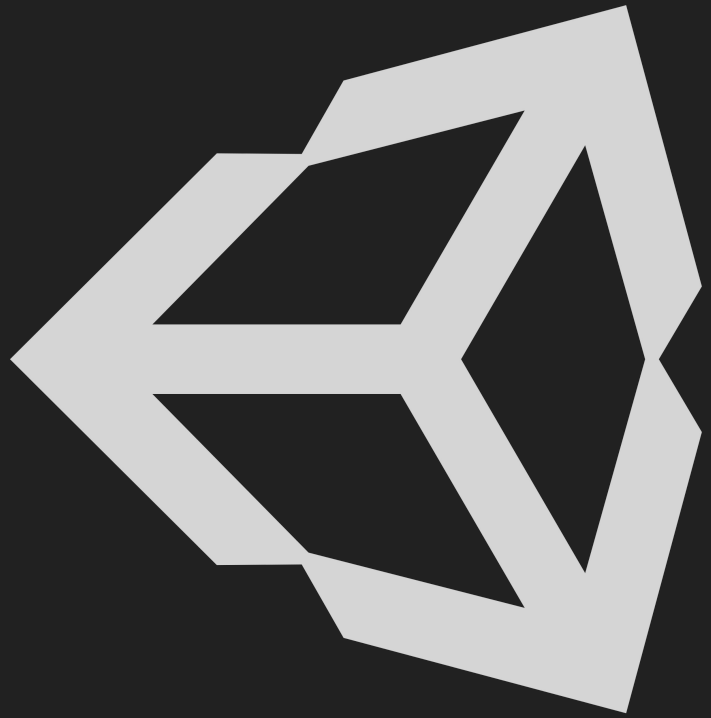
# Obstrução e Oclusão




Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# UNITY TIME !!!! - Audio Source



# 4. Áudio

- Mixer
  - Agrupa áudios em canais
  - Cada canal tem seu próprio volume, efeitos, panning...
  - Canais podem ser conectados a outros canais, formando um pipeline de áudio

# 4. Áudio

- Transições
  - Importante para uma experiência fluida (Game Design)
  - Exemplo: efeito de transição entre cenas, ou entre momentos calmos e tensos
  - Interpolação entre músicas, múltiplos parâmetros, controladores...

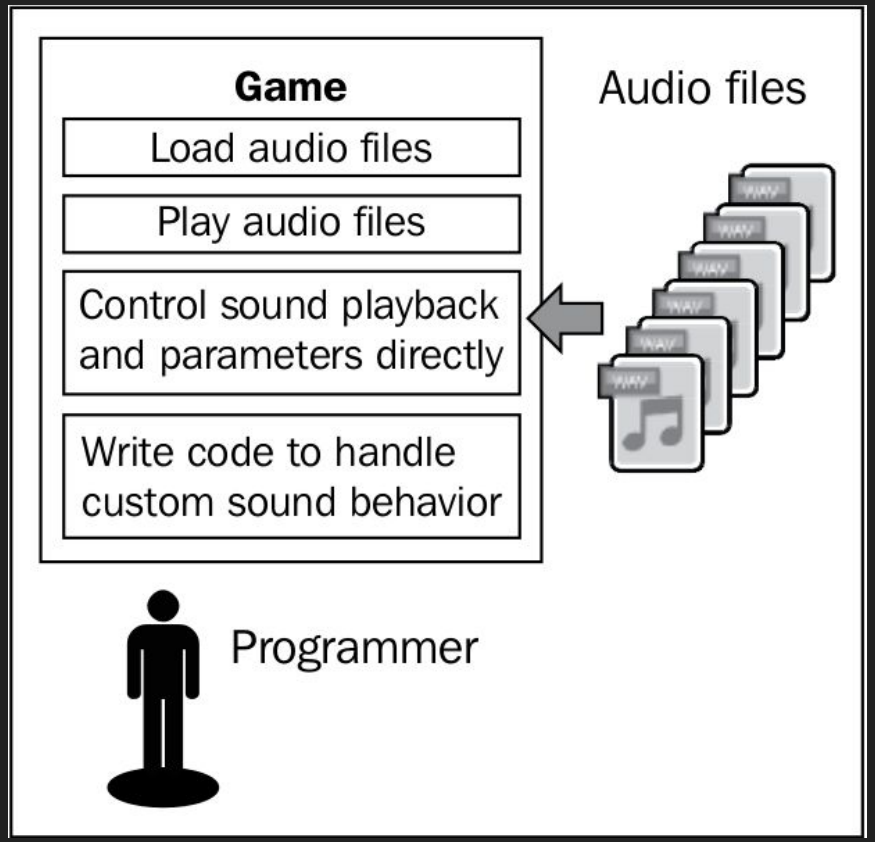
# 4. Áudio

- ➔ Áudio inteligente
  - ◆ Sistema de eventos e callbacks
  - ◆ Música interativa
    - ● Pontos de sincronização, transição, superposição

# FMOD Studio

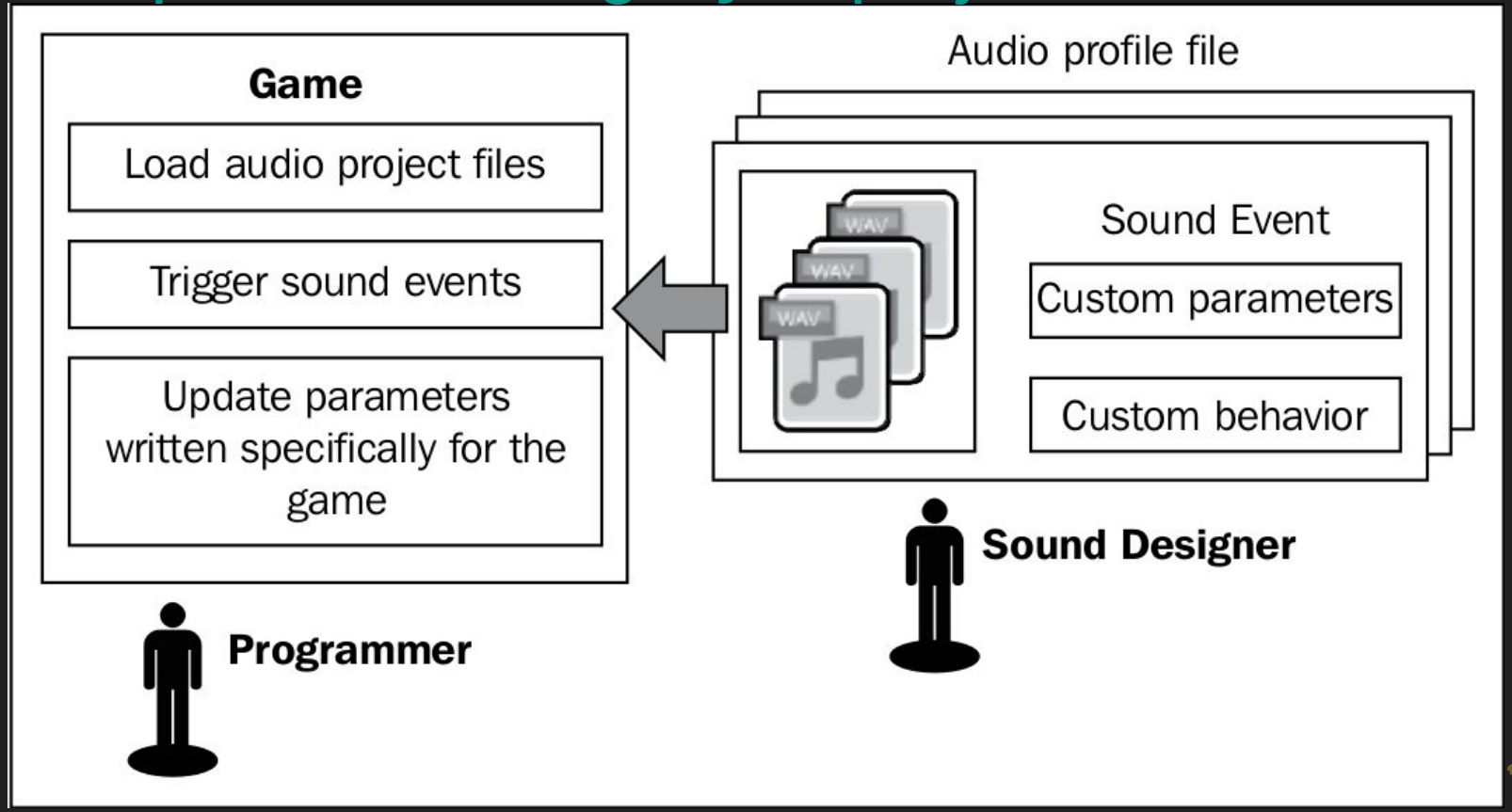

Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# Pipeline de integração áudio




Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# Pipeline de integração projetos de áudio

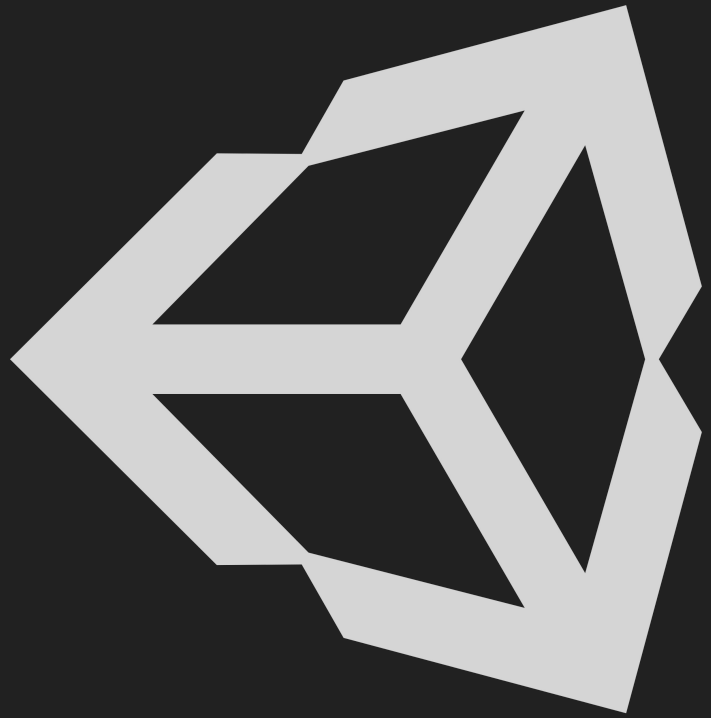



Gouveia, David. Getting Started with C++ Audio Programming for Game Development

# 4. Áudio

- Efeitos
  - Digital Signal Processing (DSP)
  - Transformações do PCM
  - Equalizador
  - Distorção
  - Filtro passa-baixa (low-pass filter)
  - Filtro passa-alta (high-pass filter)

# UNITY TIME !!!! - Mixer



# Dúvidas?

# Referências

# Referências


[1] Game Coding Complete, Fourth Edition (2012) - Mike McShaffry, David Graham
[2] Gouveia, David. Getting Started with C++ Audio Programming for Game Development
[3] https://en.wikipedia.org/wiki/Color_quantization
[4] https://en.wikipedia.org/wiki/Mipmap
[5] https://en.wikipedia.org/wiki/Doppler_effect
[6] http://www.physicsclassroom.com/class/waves/Lesson-3/The-Doppler-Effect
[7] http://gameprogrammingpatterns.com/
[8] http://www.arcsoft.com/topics/photostudio-darkroom/what-is-color-space.html