Documentação

Documentação de Implementação do Sistema de Som

1. Visão Geral

Este sistema de som permite gerenciar e tocar músicas de fundo (OST) e efeitos sonoros (SFX) em seu jogo. Ele utiliza uma abordagem baseada em eventos para desacoplar as solicitações de som da sua execução e gerencia um pool de AudioSources para SFX para otimizar o desempenho.

Os componentes principais são:

- `SoundManagerSO` **(ScriptableObject - não fornecido, mas referenciado):** Contém as configurações centrais do sistema de som, como caminhos para os clipes de áudio, prefab para SFX, configurações do mixer e tamanho do pool.
- `SoundManager`**:** Classe estática responsável por carregar os clipes de áudio, gerenciar o pool de SFX, interagir com o AudioMixer e tocar os sons.
- `SoundEvent`**:** Classe estática que define os argumentos de evento (`SoundEventArgs`) e gerencia o evento `OnSoundRequested` para solicitar a reprodução de sons.
- `SoundController`**:** Classe estática que escuta o evento `SoundEvent.OnSoundRequested` e direciona as solicitações para o `SoundManager`.

2. Configuração Necessária

Para que o sistema funcione corretamente, algumas configurações são essenciais:

2.1. `SoundManagerSO` Asset

- **Criação:** Você precisará criar um ScriptableObject do tipo `SoundManagerSO` (o script `SoundManagerSO.cs` não foi fornecido, mas é referenciado pelo `SoundManager`).
- **Localização:** Coloque a instância criada deste asset na pasta `Assets/Resources/Database/` e nomeie o arquivo como `SoundManagerSO.asset`.
- **Campos a Configurar (no asset `SoundManagerSO`):**

- `musicFolderPath`: Caminho (dentro de uma pasta `Resources`) onde seus clipes de música estão localizados (ex: `Audio/Music`).
- `sfxFolderPath`: Caminho (dentro de uma pasta `Resources`) onde seus clipes de SFX estão localizados (ex: `Audio/SFX`).
- `SFXPrefab`: Um Prefab que será usado para tocar SFXs. Este prefab deve conter um componente `AudioSource` e um script `PooledAudioSource` (veja item 2.3).
- `sfxPoolSize`: O número inicial de instâncias do `SFXPrefab` a serem criadas no pool.
- `audioMixer`: Referência ao seu Unity AudioMixer.
- `masterVolumeParam`, `musicVolumeParam`, `sfxVolumeParam`: Os nomes exatos dos parâmetros de volume expostos no seu `audioMixer` (ex: "MasterVolume", "MusicVolume", "SFXVolume").

2.2. Nomenclatura dos Clipes de Áudio

Os clipes de áudio devem seguir uma convenção de nomenclatura para serem carregados corretamente:

- **Músicas (OST):** Devem começar com o prefixo `ost_`. O `AudioID` usado para tocar será a parte do nome do arquivo após o prefixo, em minúsculas.
  - Exemplo: `Assets/Resources/Audio/Music/ost_MainTheme.mp3` → `AudioID`: `"maintheme"`
- **Efeitos Sonoros (SFX):** Devem começar com o prefixo `sfx_`. O `AudioID` usado para tocar será a parte do nome do arquivo após o prefixo, em minúsculas.
  - Exemplo: `Assets/Resources/Audio/SFX/sfx_PlayerJump.wav` → `AudioID`: `"playerjump"`

2.3. Prefab para SFX (`SFXPrefab`)

- Crie um Prefab (por exemplo, um GameObject vazio).
- Adicione um componente `AudioSource` a este Prefab.
- Adicione um script chamado `PooledAudioSource` a este Prefab. Este script (não fornecido nos arquivos) é crucial e o `SoundManager` espera que ele tenha:
  - Um método `SetPool(Queue<GameObject> pool, Transform audioHostTransform)` para que o `SoundManager` possa injetar a referência do pool.
  - Um método `Play(AudioClip clip, float volumeScale)` para tocar o clipe e, idealmente, desativar o GameObject e retorná-lo ao pool quando o som terminar.

2.4. Unity Audio Mixer

- Crie e configure um `AudioMixer` no Unity (Window → Audio → Audio Mixer).
- Crie grupos dentro do mixer, por exemplo, "Master", "Music", e "SFX".
- Para cada grupo cujo volume você deseja controlar (Master, Music, SFX), clique com o botão direito no parâmetro "Volume" do inspetor do grupo e selecione "Expose 'Volume (of ...)' to script".
- No painel "Exposed Parameters" do Audio Mixer, renomeie esses parâmetros expostos para que correspondam exatamente aos nomes que você colocará nos campos `masterVolumeParam`, `musicVolumeParam`, e `sfxVolumeParam` do seu asset `SoundManagerSO`.

3. Tocando Sons

A reprodução de sons é feita através do sistema de eventos. Você não chama diretamente os métodos `Play` do `SoundManager` de fora do sistema de áudio.

3.1. Fluxo de Solicitação

1. Seu código de jogo cria `SoundEventArgs`.
2. Seu código chama `SoundEvent.RequestSound(args)`.
3. `SoundController` recebe o evento e chama o método apropriado no `SoundManager`.
4. `SoundManager` toca o som.

3.2. Como Usar `SoundEvent.RequestSound()`

Em qualquer script onde você precise tocar um som:

```
// Exemplo para tocar um Efeito Sonoro (SFX) em uma posição:
SoundEventArgs sfxArgs = new SoundEventArgs
{
    Category = SoundEventArgs.SoundCategory.SFX,
    AudioID = "nomedoseuaudio", // O ID do seu SFX (sem "sfx_" e em minúscu
    Position = transform.position, // Posição para o som 3D
    VolumeScale = 1.0f // Escala de volume (opcional, padrão é 1f)
};
SoundEvent.RequestSound(sfxArgs);

// Exemplo para tocar um SFX atrelado a um Transform:
// public Transform objetoSonoro; // Referência ao transform
```

```
SoundEventArgs sfxAttachedArgs = new SoundEventArgs
{
    Category = SoundEventArgs.SoundCategory.SFX,
    AudioID = "outroaudio",
    TargetTransform = objetoSonoro,
    VolumeScale = 0.8f
};
SoundEvent.RequestSound(sfxAttachedArgs);

// Exemplo para tocar uma Música (OST):
SoundEventArgs musicArgs = new SoundEventArgs
{
    Category = SoundEventArgs.SoundCategory.Music,
    AudioID = "temaprincipal" // O ID da sua música (sem "ost_" e em minúscul
};
SoundEvent.RequestSound(musicArgs);
```

4. Controles Adicionais

O `SoundManager` oferece alguns métodos estáticos para controle:

4.1. Controle de Volume

Estes métodos ajustam o volume no AudioMixer e salvam a preferência em `PlayerPrefs`. O valor de `volume` deve ser linear (0.0 a 1.0).

- `SoundManager.SetMasterVolume(float volume)`
- `SoundManager.SetMusicVolume(float volume)`
- `SoundManager.SetSFXVolume(float volume)`

4.2. Controle de Música (OST)

- `SoundManager.PlayOST(string id)`: Pode ser usado para tocar uma música diretamente, mas o método preferencial é via `SoundEvent.RequestSound`.
- `SoundManager.StopMusic()`
- `SoundManager.PauseMusic()`
- `SoundManager.ResumeMusic()`

4.3. Eventos do SoundManager (Callbacks)

Você pode se inscrever nestes eventos para ser notificado quando um som começar a tocar:

- `SoundManager.OnMusicPlay += SeuMetodoParaMusica;` (SeuMetodoParaMusica recebe `string audioID`)
- `SoundManager.OnSFXPlay += SeuMetodoParaSFX;` (SeuMetodoParaSFX recebe `string audioID`)

5. Inicialização

O sistema (`SoundManager` e `SoundController`) é projetado para se inicializar automaticamente quando o jogo carrega, utilizando `[RuntimeInitializeOnLoadMethod]`. Você não precisa chamar nenhum método de inicialização manualmente a partir do seu código de jogo. Apenas certifique-se de que todas as configurações (especialmente o `SoundManagerSO` e os caminhos/nomes dos arquivos) estejam corretas.