REFACTOR PROPOSAL

1. Overview

The SoundManager is a comprehensive Unity audio management system implementing the Singleton pattern. It handles background music, sound effects, and UI sounds with features like crossfading, volume control, and addressable asset loading.

2. Core Feature

a. Audio Source Management

- Multiple dedicated audio sources:
  + 3 music sources for crossfading
  + Effects source for SFX
  + UI effects source
- Smart source allocation system for managing multiple simultaneous audio streams

b. Music System

- Background music playlist with shuffle capability
- Crossfading between tracks
- Victory music system with automatic background music pause/resume
- Support for looping and non-looping playback
- Fresh background music system for restarting playlists

c. Sound Effects

- Separate systems for general SFX and UI sounds
- Volume control independent from music

- Support for one-shot sound effects
- Match entry sound effect sequence system

d. Volume Management

- Separate volume controls for music and effects
- Persistent volume settings using PlayerPrefs
- Real-time volume updates

e. Addressable Asset System

- Async loading of audio clips
- Mapping system between SoundType enum and addressable addresses
- Error handling for failed asset loads
- Reverse lookup capability for currently playing sounds

3. Current Limitations

a. Resource Management

- No proper memory management
- Limited audio source pooling
- Potential memory leaks
- No cleanup mechanisms

b. Architecture

- Monolithic design
- Tight coupling between components
- Limited extensibility
- Hard to test individual components

c. Performance

- Inefficient crossfading
- No background loading
- Potential main thread blocking
- No resource prioritization

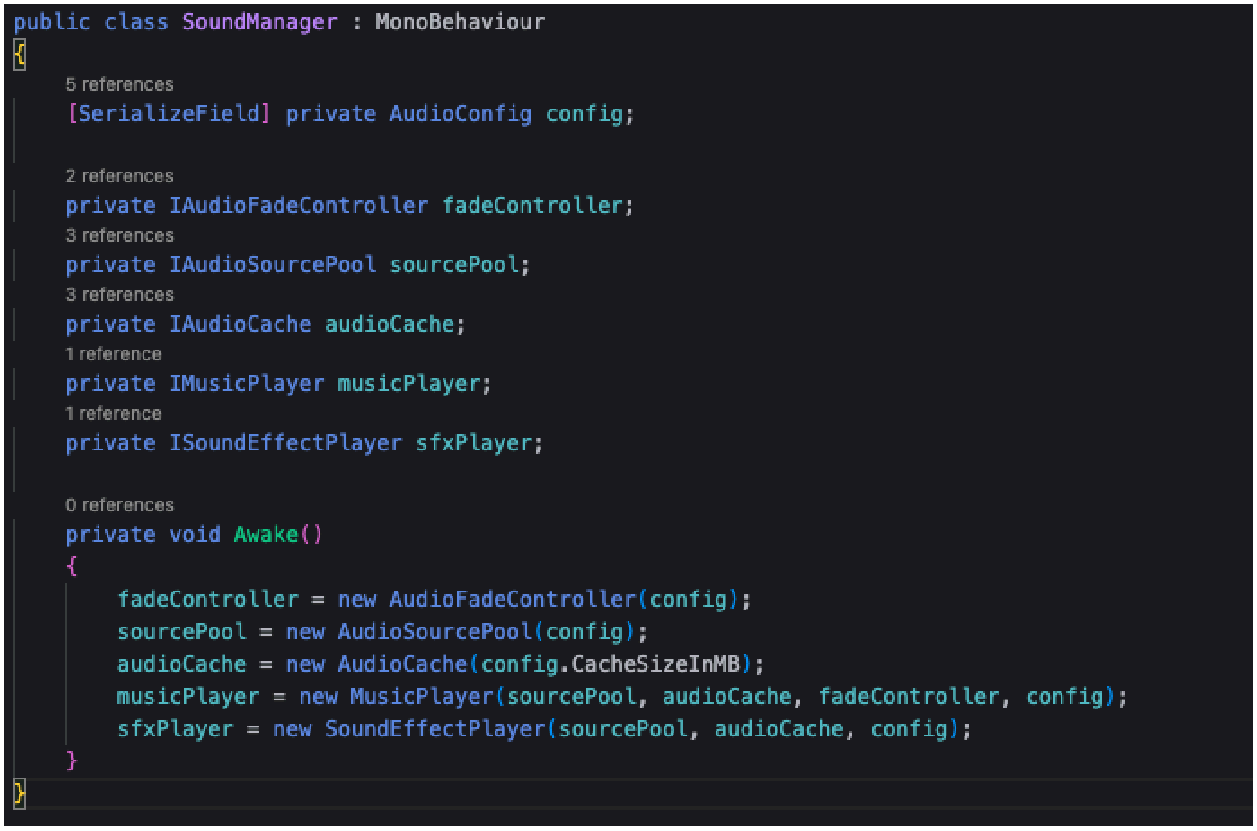
4. Proposed Solution

a. Core Architecture Separation

Split the monolithic SoundManager into focused components:

- AudioConfig: Centralized configuration
- AudioSourcePool: Manages audio source lifecycle
- AudioCache: Handles clip memory management
- MusicPlayer: Controls background music & playlists
- SoundEffectPlayer: Manages SFX and UI sounds
- AudioFadeController: Handle volume transitions

Example Integration:



```
public class SoundManager : MonoBehaviour
{
    5 references
    [SerializeField] private AudioConfig config;

    2 references
    private IAudioFadeController fadeController;
    3 references
    private IAudioSourcePool sourcePool;
    3 references
    private IAudioCache audioCache;
    1 reference
    private IMusicPlayer musicPlayer;
    1 reference
    private ISoundEffectPlayer sfxPlayer;

    0 references
    private void Awake()
    {
        fadeController = new AudioFadeController(config);
        sourcePool = new AudioSourcePool(config);
        audioCache = new AudioCache(config.CacheSizeInMB);
        musicPlayer = new MusicPlayer(sourcePool, audioCache, fadeController, config);
        sfxPlayer = new SoundEffectPlayer(sourcePool, audioCache, config);
    }
}
```

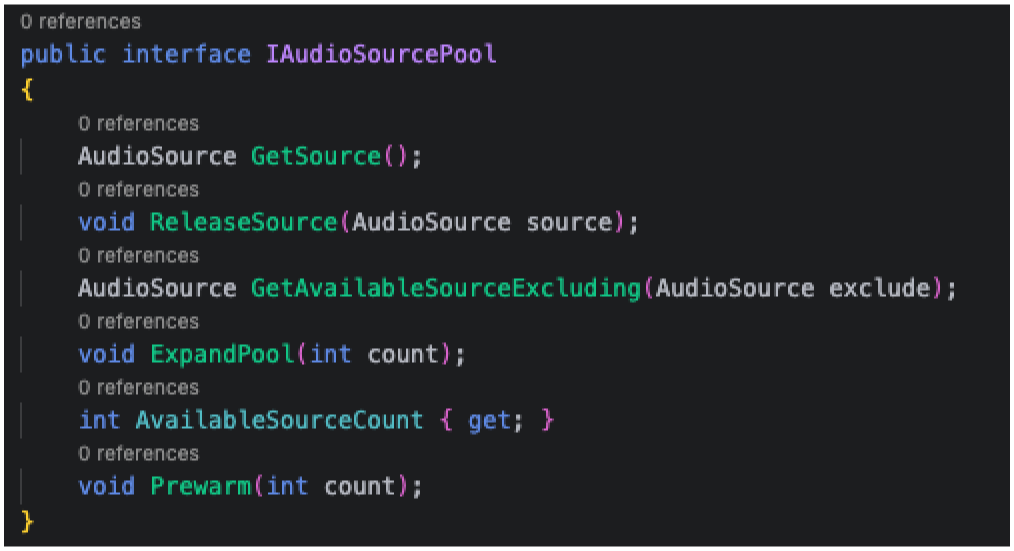
b. Component Responsibilities

AudioConfig:

- Centralizes all audio-related settings
- Provides runtime configuration
- Supports serialization for Unity Inspector
- Enables easy project-wide audio adjustments

AudioSourcePool:

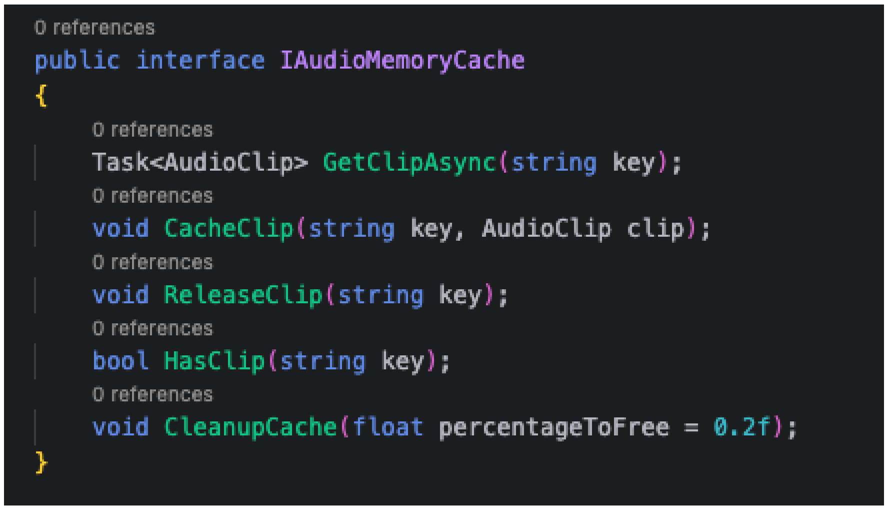
- Manages AudioSource component lifecycle
- Provides efficient source allocation
- Handles dynamic pool sizing



```
0 references
public interface IAudioSourcePool
{
    0 references
    AudioSource GetSource();
    0 references
    void ReleaseSource(AudioSource source);
    0 references
    AudioSource GetAvailableSourceExcluding(AudioSource exclude);
    0 references
    void ExpandPool(int count);
    0 references
    int AvailableSourceCount { get; }
    0 references
    void Prewarm(int count);
}
```

AudioCache:

- Implements Least Recently Used (LRU) caching strategy
- Uses explicit memory size tracking
- Provides efficient clip loading and unloading
- Manages Addressables lifecycle

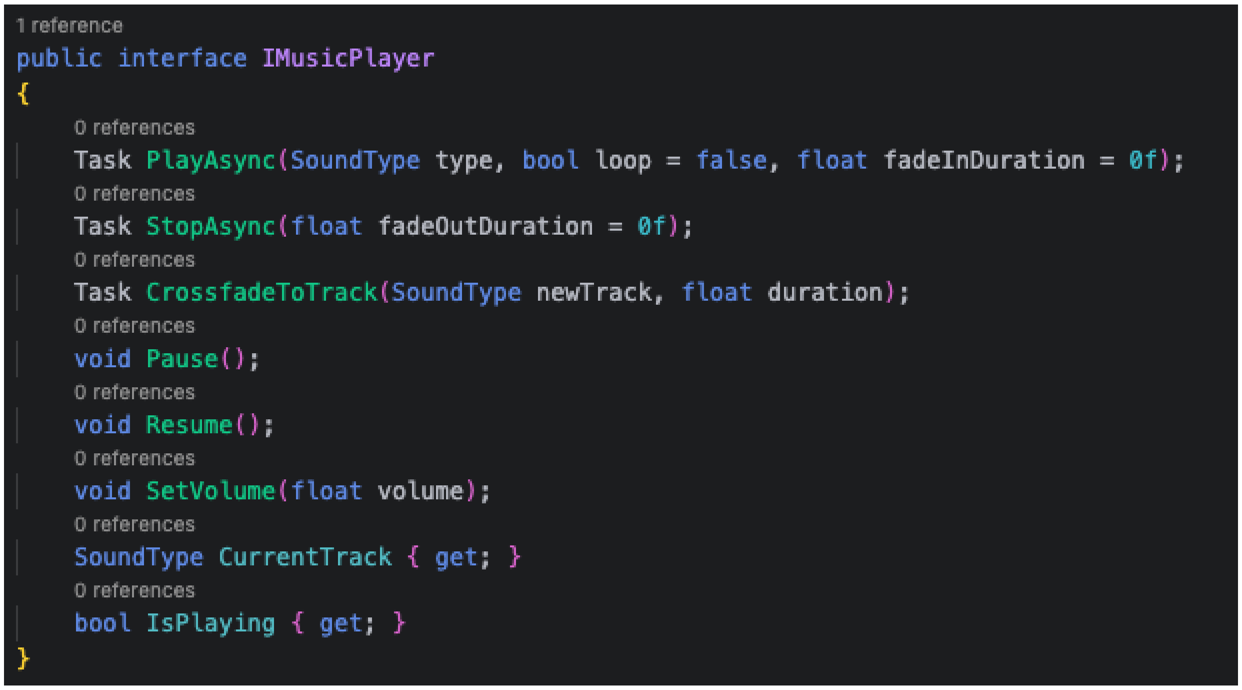


```
0 references
public interface IAudioMemoryCache
{
    0 references
    Task<AudioClip> GetClipAsync(string key);
    0 references
    void CacheClip(string key, AudioClip clip);
    0 references
    void ReleaseClip(string key);
    0 references
    bool HasClip(string key);
    0 references
    void CleanupCache(float percentageToFree = 0.2f);
}
```

MusicPlayer:

- Handles background music playback
- Manages playlists and transitions

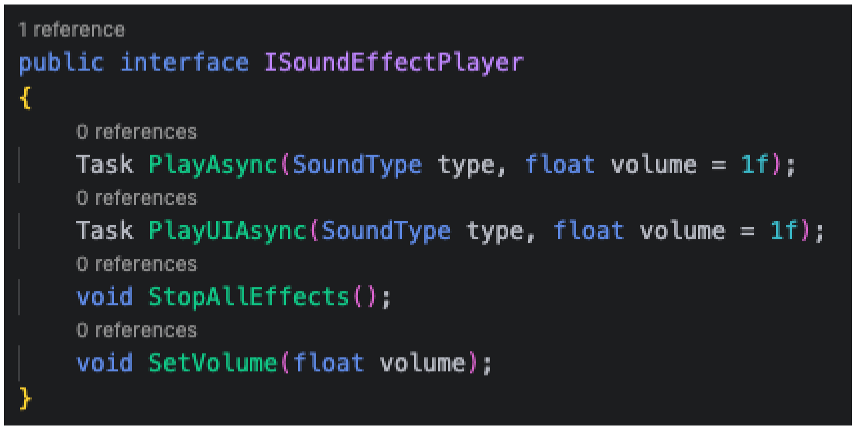
- Controls music-specific features
- Integrates with fade controller



```
1 reference
public interface IMusicPlayer
{
    0 references
    Task PlayAsync(SoundType type, bool loop = false, float fadeInDuration = 0f);
    0 references
    Task StopAsync(float fadeOutDuration = 0f);
    0 references
    Task CrossfadeToTrack(SoundType newTrack, float duration);
    0 references
    void Pause();
    0 references
    void Resume();
    0 references
    void SetVolume(float volume);
    0 references
    SoundType CurrentTrack { get; }
    0 references
    bool IsPlaying { get; }
}
```

SoundEffectPlayer:

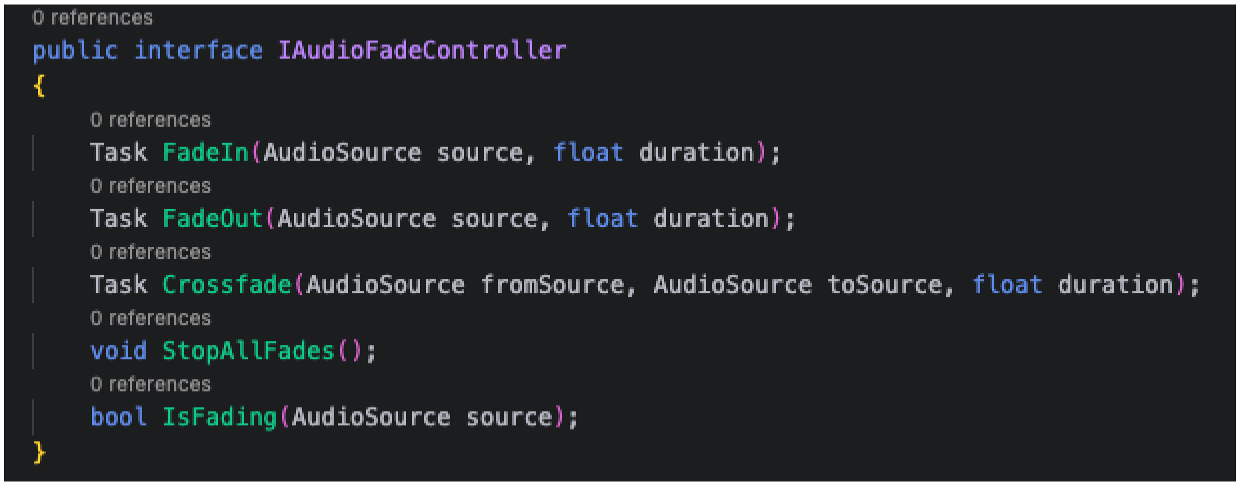
- Manages sound effects and UI sounds
- Handles one-shot audio playback
- Controls effect-specific features



```
1 reference
public interface ISoundEffectPlayer
{
    0 references
    Task PlayAsync(SoundType type, float volume = 1f);
    0 references
    Task PlayUIAsync(SoundType type, float volume = 1f);
    0 references
    void StopAllEffects();
    0 references
    void SetVolume(float volume);
}
```

AudioFadeController:

- Manages all volume transitions
- Handles crossfading between tracks
- Ensures thread-safe fade operations
- Provides cancellation support
- Maintains smooth transitions



```
0 references
public interface IAudioFadeController
{
    0 references
    Task FadeIn(AudioSource source, float duration);
    0 references
    Task FadeOut(AudioSource source, float duration);
    0 references
    Task Crossfade(AudioSource fromSource, AudioSource toSource, float duration);
    0 references
    void StopAllFades();
    0 references
    bool IsFading(AudioSource source);
}
```