リソース管理を最適化するためのサウンドシステム

– サウンドマネージャの役割の明確化とシステムの連携 –

2014 年 2 月 24 日　初稿

板垣 衛

■ 改訂履歴

稿	改訂日	改訂者	改訂内容
初稿	2014 年 2 月 24 日	板垣 衛	（初稿）

■ 目次

■ 概略

何かと扱いにくい面のあるサウンドリソースに対して、最適な管理を行うためのサウンドマネージャを設計する。

シーンマネージャ、リソースマネージャと組み合わせて効果的な管理を行う。

■ 目的

本書は、サウンドリソースの管理手法を確立し、ゲームシステムの安全性をより高めることを目的とする。
また、「どこまでがサウンドマネージャ側の担当処理か？」という切り分けを明確にすることも目的とする。

別紙の「開発の効率化と安全性のためのリソース管理」および「ゲーム全体を円滑に制御するためのシーン管理」と密接に関連した内容となる。

■ サウンドマネージャの設計指針

「サウンドシステム」というと、ゲームシステムの中でもとりわけ独立性が高いものとして扱うことが多そうだが、本書においては、共通のリソース管理システムに依存することで、効率よくリソースを共有し、リソースのライフサイクルを安全に管理するものとする。

▼ 設計の前提

本書の設計は、例えば PS2 のような、サウンド専用メモリを扱う必要のあるハードウェアには対応しない。他のリソースと同様に、メインメモリ上のサウンドリソースを扱うことを前提とする。

▼ 設計の範囲

本書はサウンドマネージャ全体を設計するものではなく、リソースの扱いに関してのみを規定する。

実際のサウンド制御には優秀なサウンドエンジン（ミドルウェア）などを用いるものとし、サウンドエンジンとゲームとの間に入る中間レイヤーとしてのサウンドマネージャとする。

▼ 設計の基本方針

ゲーム側から見たサウンドの制御は、「シーンに音が登場するかどうか」を基本とし、シーンに対する処理でサウンドを扱う。サウンドエンジン（ミドルウェア）どころかサウンドマネージャの存在もほとんど意識する必要がないものとする。

本書は、サウンドをそのように扱うためのサウンドマネージャを設計すると共に、シーン（シーンマネージャ）側の処理も規定する。

本書は、別紙の「ゲーム全体を円滑に制御するためのシーン管理」の延長にあるものであり、その内容に基づいて構成する。

■ 要件定義

▼ 基本要件

本書が扱うシステムの基本要件は下記のとおり。

- サウンドリソースは、BGM、ボイスなどのストリーミングを除いて、すべてリソースマネージャによって管理するものとする。
  - リソースマネージャの強力なリソース共有機能を活用する。
- キャラのモーションに連動した SE や爆発音などのエフェクト用 SE、マップ上の固定発音体などといった SE のリソースは、キャラモデルやモーションデータ、マップ上の発音体設定などと共に、個別に扱うものとする。
  - SE を使用する元になるリソース（キャラモデル、モーション、エフェクト、マップなど）と SE のリソースを関連づけ、リソースをいっしょに読み込む。
  - 「キャラ A とキャラ B では、モデルは異なるが SE は同じ」というような場合、リソースマネージャの機能により、一つの SE リソースを共有し、無駄な読み込みを行わない。
  - 通常、SE ファイルは多数の SE がひとまとめになっている。例えば、キャラの基本モーション全部をまとめた SE ファイル、マップ上で使用する全ての発音体をまとめた

SE ファイルなど。SE リソースはリソースマネージャによって共有されるため、多少大きくまとまった SE ファイルとなっても問題ない。

- サウンドエンジン（ミドルウェア）に登録可能なリソース数にも限界があるので、ある程度大きくまとめる必要がある。
- あまり大きくまとめすぎると、通常使わない SE がメモリを圧迫するので、適度なバランスが必要。

・ 発音中の SE は、一つ一つ「シーンオブジェクト」として扱うものとする。
  - シーンマネージャでは、シーンに登場するキャラやエフェクトなど一つ一つをシーンオブジェクトとして扱っており、SE も独立したシーンオブジェクトとして扱う。
  - シーンオブジェクトとして扱うことで、SE のサラウンドやループ再生、途中の打ち切り、線音源、麺音源などの様々な要件に対して、統一的な手法で柔軟に制御できるものとする。

・ SE 再生の際は、リソースマネージャが管理するリソースハンドルと SE 番号を指定して再生するものとする。
  - 表向き SE はリソースハンドルで扱うものとするが、サウンドマネージャの内部では、リソースハンドルをサウンドエンジンのハンドルに変換して扱う。

・ SE 再生終了に合わせた、シーンオブジェクトの自動消滅に対応するものとする。

・ シーンオブジェクトの消滅に合わせた、SE の自動終了に対応するものとする。
  - シーンオブジェクトは、「固定発音体」や「移動音」のようなループ音にも対応する。

・ 同時発音数の調整はサウンドマネージャが行うものとする。
  - 例えば、極端に遠くに離れた発音体の SE であっても、シーンオブジェクトが存在するなら、そのままサウンドマネージャに登録する。実際に発音するかどうかはサウンドマネージャが決定する。
  - もう少し厳密には、距離に応じた発音体の音量と再生状態の制御をしっかりと行う。
    - 非ループ音なら、可聴範囲から外れたところでも再生を始め、近づいた時に途中から音が聞こえる必要があるため、音量のみで制御する。
    - ループ音なら、音量制御も当然行うとしても、可聴範囲に入ったところで初めて再生を開始する。

・ シーンオブジェクトのアップデート処理により、音量変更／フェードイン・アウト、ステレオ／サラウンドのパンニング、リバーブ ON/OFF、ドップラー効果有無などの状態を毎フレーム更新する。
  - 「線音源」や「面音源」といった、広域的な発音体の発音点の決定も行う。このような処理では、「線データ」や「面データ」といった、サウンドリソース以外のデータも扱

う。

▼ 要求仕様／要件定義

以下、本書が扱うシステムの要件を定義する。なお、要件として不確定の要求仕様も併記する。

また、他のシステムと絡んでやや要件が複雑なため、一部処理仕様レベルの内容も織り交ぜて記載する。

- ゲームシーンに登場する全てのサウンドは、「シーンマネージャ」によって管理されるものとする。
- 実際の発音はサウンドエンジン（ミドルウェア）によって制御されるものとする。
- シーン上のサウンドオブジェクトと実際の発音との関連を「サウンドマネージャ」が管理するものとする。

● システム間の依存関係

サウンドマネージャとリソースマネージャ、シーンマネージャは密接に連携するが、その依存関係は、サウンドマネージャ⇒リソースマネージャ、シーンマネージャ⇒サウンドマネージャのそれぞれ一方向である。

● リソース生成に関する要件

サウンドリソースの生成にはサウンドマネージャは関与しない。

- キャラモデルなどと同じく、シーンマネージャとリソースマネージャ間の処理でリソースを生成し、また、共有する。
- サウンドリソースも、このような一般的なリソース管理の仕組みで管理することで、効率的なリソース共有、リソース再利用、スレッドセーフの恩恵を教授するものとする。
- サウンドマネージャは、SE 再生時にリソースを参照するが、参照先のリソースがない場合は再生に失敗する。リソースの生成を要求することはない。

● リソース消滅に関する要件

サウンドリソースの消滅も、同じくサウンドマネージャは関与しない。

- ただし、SE 再生中はサウンドマネージャがリソースのリードロックを取得し、かつ、参照カウンタをカウントアップする。これにより、シーンが放棄したリソースであっても、再生中に消滅することはない。

● リソースの扱いに関する要件

サウンドマネージャは、リソースマネージャのリソースハンドルを保持してリソースにアクセスする。

- サウンドマネージャでは、内部でリソースマネージャのハンドルとサウンドエンジンの管理ハンドルを関連づけて扱う。
- サウンドエンジンへのリソースの登録は、初めてそのリソースを使用する時に自動的に登録する。
  - サウンドマネージャ内で、リソースハンドルに対する管理ハンドルの関連づけの記録が存在していないことで判断する。
  - サウンドエンジンにリソースを登録した時点で、参照カウンタをカウントアップし、リソースのリードロックを取得して、メモリ再配置や削除を防ぐ。
- SE 再生中はリソースのリードロックを取得し、かつ、参照カウンタをカウントアップする。
- リソースの参照カウンタとは別に、リソースに対する再生中の数をカウントし、再生中の SE がないリソースはサウンドエンジンから解放する。（サウンドエンジンの管理ハンドルの解放）
  - 【要検討】サウンドエンジンへのリソース登録／解放の負担が極小であるなら、小まめな解放を行い、リソースのメモリ再配置が可能な状態を作る。
  - 【要検討】サウンドエンジンへのリソース登録／解放の負担が大きいなら、毎フレームリソースの参照カウンタを監視し、参照カウンタが 0 のリソースを解放する。
    - 再生中の SE が 1 のリソースのみ監視する。
    - サウンドエンジンにリソースを登録している間は参照カウンタをカウントアップしているので、シーンオブジェクトがリソースを解放していても、削除されない。
  - サウンドエンジンのリソースを解放する際に、リソースの参照カウンタをカウントダウンし、リードロックを開放する。

● シーンオブジェクトに関する要件：処理の切り分け

表向き、サウンドを制御するのはシーンマネージャであり、個々の SE はシーンオ

ブジェクトによって扱う。

- シーンオブジェクト生成時に、サウンドマネージャに SE の登録を行い、「SE 再生ハンドル」の発行を受け、そのハンドルを保持する。
  - 登録時点では、（リソースマネージャの）「リソースハンドル」、「SE 番号」、「ループ音指定」、「減衰距離」のみを指定する。
  - 「SE 再生ハンドル」の発行ができず、登録に失敗した場合、シーンオブジェクトは消滅する。
  - この時点ではまだ SE の再生は開始されない。

- シーンオブジェクトは、毎フレームのアップデート処理で、サウンドマネージャに対して「SE 再生ハンドル」と共に「SE 再生状態の更新」を通知する。
  - サウンド用シーンオブジェクトのアップデート処理は、描画（スレッド）との並行処理で行う。
  - 「SE 再生状態の更新」とは、「音量」、「発音位置」、「フェード」（変更する音量と変化時間）、「パンニング指定」、「リバーブ効果指定」、「ドップラー効果有効化指定」、「任意のエフェクト指定」などのこと。
  - 【要検討】「再生ピッチの変更が可能（もしくは許可された）」な SE の場合、アップデート処理の「処理時間レート」に応じたピッチを指定する。

    ・ キャラの行動速度が 2 倍速になったら SE も 2 倍速にする、など。

- サウンドマネージャは、サウンドマネージャ内の状態に応じて、「SE 再生開始」、「SE 再生停止」、「音量変更／フェード」、「パンニング変更」、「リバーブなどの効果変更」といった処理を適切に実行する。

- リスナー位置（マイク位置）も同様にシーンマネージャからサウンドマネージャに渡され、位置関係に応じた音量の計算はサウンドマネージャ側で行う。

- 「線音源」「面音源」といった、発音点が動的に変化する音については、シーンオブジェクト側で発音点を決定する。
  - サウンドマネージャには常に「点」で発音位置を指定する。

- 「ドップラー効果」は、発音位置とリスナー位置のそれぞれのベロシティが必要になるが、その計算はサウンドマネージャ側で行う。
  - 単純に前のフレームからの移動量を見て、そのフレームでのピッチを変化させる。

- 「サラウンド」と「ドップラー効果」はシーンオブジェクトがその機能を使用すると指定した場合のみ有効となる。
  - 毎フレームのアップデート時に、他の効果と共に指定する。
  - 位置関係を無視したパンニングの指定や、プリセットによるパンニング指定も可。
  - プリセットには、「センターのみ」、「ファントムセンター（前面 LR のみでセンターなし）」、

「センターを除く全スピーカー」といったものがある。

- 「狭いトンネルの中に入った」などのリスナー位置の状態に応じたリバーブの適用は、シーンマネージャ側で判定し、各シーンオブジェクトのアップデート時にリバーブを指定する。
- 同様に、例えば「遮蔽物を挟んだこもった音」などを表現したければ、シーンマネージャ側で判断して、シーンオブジェクトごとにアップデート時に「効果」（コンプレッサーなど）を指定する。
  - こうした特殊な「効果」は、サウンドエンジンのエフェクトとして用意したプリセットを使用する。
  - 「効果 ID」や「効果の度合い」を外部指定できるようにし、ゲームタイトルに応じて自由に扱う。
- 「コントローラ（入力デバイス）スピーカーからの再生」といった、出力装置指定の要件に対しては、シーンオブジェクトからの再生指定時に「論理デバイス」を指定する。
  - 「論理デバイス」は、「入力デバイス」や「振動装置」などを示す抽象的な外部装置を指す。「コントローラスピーカー」のような特殊なスピーカーは「論理デバイス」で扱い、再生先のスピーカーの指定として用いる。
  - 「論理デバイス」については、別紙の「反応性と安全性を考慮した入力デバイス管理」を参照。

● SE 再生ハンドルに関する要件

再生が要求された SE は、サウンドマネージャが「SE 再生ハンドル」を発行して管理する。

- 一つの再生要求に対して一つのハンドルを発行する。
- 再生を終えたら、もしくは、再生の必要がなくなったら、ハンドルを開放する。
- 基本的に、サウンド用のシーンオブジェクトと 1 対 1 で関連づく。
- サウンドマネージャは、サウンドエンジンの発音数限界と無関係に、多数の再生要求を受け付ける。

● SE の再生開始に関する要件

サウンド用シーンオブジェクトの毎フレームのアップデート処理により、サウンドマネージャに渡される要求に応じて、適宜 SE の再生を開始する。

- サウンド用のシーンオブジェクトは、シーンオブジェクト生成時にサウンドマネージャに SE 再生を登録し、「SE 再生ハンドル」を発行する。
  - 登録時点では、（リソースマネージャの）「リソースハンドル」、「SE 番号」、「ループ音指定」、「減衰距離」のみを指定する。
  - シーンオブジェクトは、発行された「SE 再生ハンドル」を保持する。
  - このタイミングではまだ再生を開始しない。
  - 次のアップデート処理で、「可能なら」再生が開始される。
    - 発音位置の指定やエフェクトの指定などはアップデート時に行うため、最初の登録段階では「発音状態」が確定せず、再生できない。
  - 「距離が遠く離れたループ SE の発音体」のように、その時発音できないオブジェクトであってもハンドルの発行は行う。
  - メモリ不足などの理由でハンドルが発行できなかった場合、その SE の再生は失敗し、シーンオブジェクトが消滅する。
    - サウンドマネージャは、このような問題が起こらないように、十分な数のハンドル管理ができなければならない。
- 再生要求された SE に応じて、サウンドエンジンにリソースを登録する。
  - 初めて再生するリソースの SE である場合に限り、サウンドエンジンへの登録が必要になる。
  - リソースハンドルでリソースを参照してサウンドエンジンに登録する。
    - サウンドエンジン側でのファイル読み込みやリソース本体のメモリ確保は行わず、バッファを受け渡すだけにする。
    - サウンドエンジン側でメモリを確保してインデックス情報を構築するようなことはあるので、必要最低限以外のことはさせないようにする。
  - この時、リソースが「構築済み状態」じゃなければ登録に失敗し、シーンオブジェクトを破棄する。
    - そもそもシーンオブジェクトが有効化された時点で、リソースの構築済み状態は保障されている。
- 再生を開始したら、リソースのリードロックを取得し、参照カウンタをカウントアップする。かつ、サウンドマネージャ内でリソースに対する SE 再生数をカウントアップする。
  - 再生中のリソースの破棄やメモリ再配置を防ぐ。

- 新規 SE 再生時に、音量指定や距離減衰の計算の結果、「音量が 0」と判断された場合、SE の種類によって対応が変わる。
  - 【非ループ SE の場合】
    - 音量が 0 のまま再生を開始する。
    - 再生途中で可聴範囲に入った時に聴こえるようにするため。
    - ただし、途中で可聴範囲に入る可能性がないと判断できるものは再生しない。（例えば、減衰距離の 2 倍以上離れている、再生を無視する距離を別途指定する、といった方法で基準を設ける）
    - 「再生しない」と判断したら、すぐにシーンオブジェクトは消滅する。
  - 【ループ SE の場合】
    - 再生しない。
    - ただし、シーンオブジェクトは消滅しない。
    - やがて可聴範囲内に入った時に再生が行われる。
  - 【モーションと完全に連動したループ SE の場合】
    - 大きくゆっくり回転する物体などに適用することを想定。
    - ループ SE とせず、モーションのタイミングに合わせて毎回再生する。

● SE の再生停止に関する要件

SE の再生停止は幾つかの判断要件がある。

- 非ループ SE が終端まで再生を終えたら、再生停止とみなす。
  - 再生を終えた SE は、シーンオブジェクトのアップデート処理が失敗するので、SE 再生ハンドルを開放してシーンオブジェクトを消滅させる。
- SE 再生中にシーンオブジェクトが消滅したら、再生停止とみなす。
  - 再生途中であっても、シーンオブジェクト消滅時には SE 再生ハンドルを解放する。
  - SE 再生ハンドルが解放されても、その時点で音をぶつ切りにせず、0.1 秒程度の急速なフェードアウトで終了する。（ノイズになる事を防ぐ）
- 再生を停止したら、リソースのリードロックを開放し、参照カウンタをカウントダウンする。かつ、サウンドマネージャ内でリソースに対する SE 再生数をカウントダウンする。
  - SE 再生数が 0 になったリソースは、「リソースの扱いに関する要件」に基づいて、サウンドエンジンからリソースを解放する。
  - 先にシーンオブジェクトが消滅しても、リソースのロックを取得しているため、リソースが自

動的に破棄されることはない。

● SE の状態更新に関する要件

サウンド用シーンオブジェクトの毎フレームのアップデート処理で「SE 再生状態」を更新する。

- 「SE 再生状態」とは、「音量」、「発音位置」、「フェード」（変更する音量と変化時間）、「パンニング指定」、「リバーブ効果指定」、「ドップラー効果有効化指定」、「任意のエフェクト指定」、「再生ピッチ」など。
- 「フェード」は、例えば「音量 1.0 から 0.5 まで 1.0 秒で変更」や「音量 1.0 から 0.0 まで 0.5 秒で更新し、完了時に自動消滅」といった指定を行う。
  - フェードの指定は毎フレーム同じ内容が指定されることにより、フェードによる音量更新が継続する。
  - 途中で指定内容が変わると、新たなフェードに切り替わる。
  - フェード時は開始音量も指定するが、あくまでも変化量を計算するための指定であり、フェードはその時点の音量を基準に行う。そのため、フェード完了までにかかる実際の時間が指定時間と前後することがある。
  - フェードが完了しているかどうかは、サウンドマネージャからのレスポンスで判別できる。
- SE 再生時に減衰距離が設定されている SE は、リスナー位置との距離に基づいて、指定の「音量」を減衰させて音量を決定する。
  - この計算は、サウンドマネージャが行う。
  - 実際には、更にゲームのユーザー設定による音量係数を掛ける。
- 「パンニング指定」で「自動計算」が指定された場合、リスナー位置との位置関係に基づいて、ステレオもしくはサラウンドのパンニングが決定する。
  - この計算は、サウンドマネージャが行う。
- 【要検討】「ドップラー効果有効化指定」により、SE にドップラー効果をかける。
  - サウンドエンジンが対応していれば対応する。
  - リスナー位置の前のフレームからの移動量（ベロシティ）と、発音位置の前のフレームからの移動量（ベロシティ）に応じて、再生ピッチを決定する。
  - この計算は、サウンドマネージャが行う。
- SE の再生が終了しているものは、サウンドマネージャに対するアップデート処理に失敗するので、SE 再生ハンドルを開放してシーンオブジェクトを消滅させる。
  - ループ SE の場合、シーンオブジェクト側から止めない限り、SE 再生ハンドルが自動的に開放されることはない。

● SE のポーズに関する要件

SE のポーズは、シーンオブジェクトのポーズ状態に連動する。

- ポーズ状態のシーンオブジェクトは、サウンドマネージャにポーズを要求する。
  - 通常のシーンオブジェクトは、「ポーズ状態」になるとアップデート処理が呼び出されなくなる。
  - しかし、サウンド用のシーンオブジェクトに関しては、ポーズ状態になってもアップデート処理が行われ、サウンドマネージャに「ポーズ要求」を行う。
  - この時、諸々の「SE 再生状態」は特に扱われず、ポーズ要求を送るだけの処理となる。
- ポーズが解除されると、シーンオブジェクトは通常のアップデート処理に戻る。
  - サウンドマネージャは、明示的なポーズ解除要求がなくても、通常アップデート処理が呼び出されることでポーズを解除する。

● リスナー位置（マイク位置）に関する要件

サウンドマネージャ内で減衰やパンニングの計算を行うために、リスナー位置も通知する必要がある。

- サウンド用シーンオブジェクトのアップデート処理では、最初にリスナーの位置と向きの更新を行う。
  - リスナー位置をどこにするのが適切かは、ゲームによって異なる。

● マルチスレッドに関する要件

【要検討】サウンドマネージャの処理は、とくにスレッド化しない。

- サウンドエンジンの処理が通常スレッド化されているため、サウンドマネージャまでスレッド化する必要はない。
  - サウンドマネージャはあくまでも中継役である。
- サウンド用シーンオブジェクトのアップデート処理は、他のシーンオブジェクトの処理と異なり、ジョブ投入せずにメインスレッドで全て行う。
  - この間、並行処理で描画処理がジョブ投入を行っているので、そこに割り込まず、メインスレッドで一通りの処理を済ませる。
- 【要検討】ストリーミングの再生遅延の対策（バッファリングでき次第再生する）や、SE 再生時のディレイ指定（例えば「指定の SE を 0.2 秒後に再生開始」といった指定）を必要とする場合、サウンドマネージャもスレッド化する意義が十分にある。

● 【追加要件】ボイス再生に関する要件

可能なら、「ボイス」は特殊な扱いを検討すべき。

- 「ボイス」は「ストリーミング再生」で扱うのが通常だが、それとは別に、掛け声やダメージ時の音声などは「SE」で扱うことが多い。
- 「戦闘ボイス」（状況セリフ）などを扱っていると、両者が混ざって再生される事もよくある。これを許容したくない場合は、ボイス再生の仕組みを専用に設けたほうが良い。
- まず、「ボイス」をストリームか SE かに区別せず、ユニークな「ボイス番号」で再生可能なものとし、シーン切り替えなどのタイミングでサウンドマネージャに読み込むものとする。
- それを前提に、ボイス再生時は専用のインターフェースを使用し、ボイス番号を指定して再生する。
  - サウンドマネージャは、指定のボイス番号に応じて、ストリーミングか SE のどちらかで再生を行う。
- さらに、ボイスには優先度を設け、同一キャラのボイスが複数重なって再生されないように制御する。
  - 優先度が高い音声の再生中に、優先度が低い音声の再生要求があった場合、後発の要求はキャンセルする。（例：戦闘ボイス再生中は、攻撃時の掛け声が出ない）
  - 優先度が低い音声の再生中に、優先度が高い音声の再生要求があった場合、再生中の音声を終了させて（0.1 秒などの短いフェードアウト）、後発の音声を再生する。（例：攻撃の掛け声の最中にダメージを受けてノックバックしたら、掛け声をとめてダメージ反応音声を再生する）

● 【追加要件】サラウンド注意点

サラウンドの 3D パンニングを単純に処理すると不自然になることがあるので注意が必要。

- 例えば、三人称視点のゲームで、リスナー位置を主人公位置にして単純に処理すると、主人公とカメラ視点の間（画面には映っている）で発音があると、リアスピーカーから聴こえてしまい、不自然となる。
  - この対策として、リスナー位置を少しカメラ視点よりに動かすと緩和する。
  - しかし、今度は音量のバランスが不自然になる。主人公から少し離れた場所の方が大きな音量になる。（プレイヤーの直感として、主人公により近いものが目立ってほしい）
  - 以上の対策として、サラウンドのパンニング計算は視点に近い位置で、音量の計算は主人公位

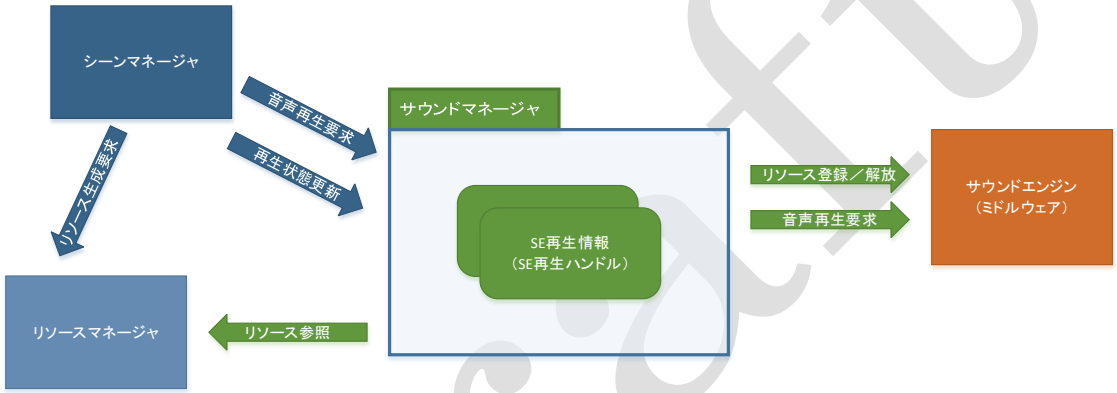
置に近い位置で処理すると、だいぶ自然な感じになる。

■ 仕様概要

▼ システム構成図

要件に基づくシステム構成図を示す。

サウンドマネージャのシステム構成図：



■■以上■■

■ 索引

索引項目が見つかりません。

リソース管理を最適化するためのサウンドシステム

以　上