ゲーム全体を円滑に制御するためのシーン管理

– ゲームシステムの根幹 –

2014 年 2 月 24 日 初稿

板垣 衛

■ 改訂履歴

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

■ 目次

■ 概略

シーン上に登場する全てのオブジェクトとその依存関係を管理し、不整合を起こさず最適なパフォーマンスを得るためのシーン管理システムを設計する。

また、リソースマネージャと密接に連携することにより、シーン間の効率的なリソース共有とマルチスレッドの最適化を実現し、無駄のない高速なシーン切り替えに対応する。

■ 目的

本書は、ゲーム上のオブジェクトを例外なく集中的に管理可能な基本システムを構築することにより、各ゲームシステムの開発の負担を軽減し、高品質なゲームを構成できるようにすることを目的とする。

■ 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。

- シーン上に登場する座標を持ったあらゆるオブジェクトを管理するものとする。
- 管理対象のオブジェクトは、不可視オブジェクトも含むものとする。
  - カメラ、SE、ボイス、ライト、トリガーポイントなど。
- 「処理のためのシーングラフ」を構成し、オブジェクト間の依存関係を階層構造で管理するものとする。
- マルチスレッドに最適化し、各オブジェクトは可能な限り並行処理するものとする。
- 大量配置オブジェクトに対応するものとする。
  - 一つのオブジェクトで複数の配置情報を扱う、個別のアニメーション情報（姿勢データ）を持たない、描画時アニメーションに対応する、など。
- シーングラフは、(大枠としての)「レンダーターゲット」との関連付けを行い、マルチスクリーン要件に対応するものとする。
  - 「二画面」や「鏡」、「ワイプ」など。

・ 柔軟な「経過時間」と「ポーズ」の要件に対応するものとする。
  - スローモーション、全体のポーズ、オブジェクト個別のポーズなど。

・ シーングラフ以外のシーンの構成要素も合わせて管理するものとする。
  - 「レベルコントローラ」（敵、NPC 配置など）、「BGM」、「環境音」など。
  - 要は、そのシーンで必要な「処理」を管理する。

・ シーン切り替えを強力にサポートするものとする。
  - 「移動シーンから戦闘シーン」、「移動シーンや戦闘シーンからイベントシーン」、「戦闘シーンからゲームオーバーシーン」、「ゲームオーバーシーンからタイトルシーン」、「移動シーンからメインメニューシーンやショップシーン」、「移動シーンから別区画の移動シーン」などを扱う。
  - シーン間でのリソース共有、切り替え前のシーンオブジェクトの先行構築、シーン（リソース）の一時破棄と復元、シーン固有の切り替え演出（暗転）、シーン切り替え要求の競合の解消など。

▼ 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、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オブジェクト、および、シーンを制御するためのすべての処理は、「シーンマネージャ」によって管理されるものとする。

・ シーンマネージャには、タイトル固有の「シーン」（シーン編成）を複数登録して扱えるものとする。
  - 「タイトルシーン」「移動シーン」「戦闘シーン」「イベントシーン」「メインメニューシーン」「ショップシーン」「ゲームオーバーシーン」といった様々なシーンを、ゲーム側から任意に設定できる。
  - それぞれが個別にシーングラフを持ち、それぞれが個別の処理を行うことができる。
  - 複数のシーンが同時にアクティブになることもできる。

・ 多少メモリの扱いが冗長になっても、高速なシーン切り替えの実現を重視するものとする。
  - シーン切り替えの際は二つのシーンをオーバーラップさせることで、最大限にリソースの共有と事前準備を行う。
  - このため、プログラムモジュールのオーバーレイは考慮しないものとする。

● システム間の依存関係

シーンマネージャとリソースマネージャは密接に連携するが、その依存関係は、シーンマネージャ⇒リソースマネージャの一方向である。

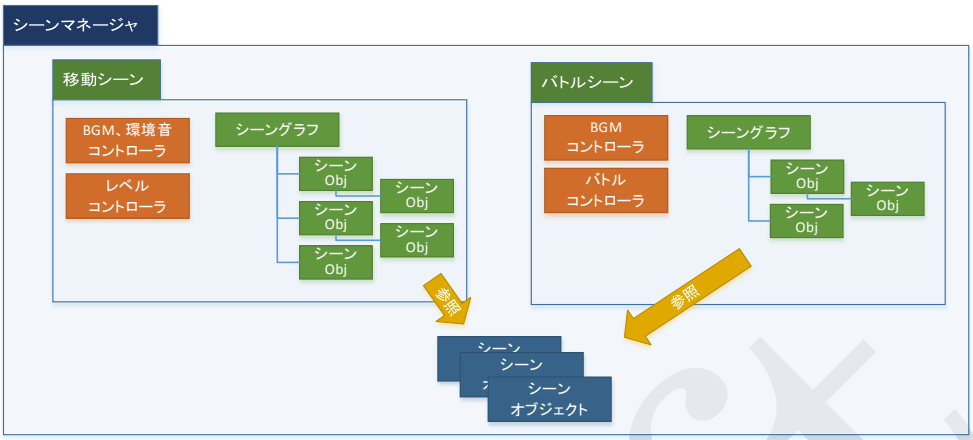
- シーンマネージャは直接ファイルを読み込んだり、リソースのメモリを管理したりせず、そうしたリソースの管理は全てリソースマネージャに任せる。リソースマネージャが、リソースの共有やマルチスレッドでの同時アクセスの保護を行う。
- シーンオブジェクトごとに、そのオブジェクトが必要とするリソースのハンドルを保持して扱う。一つのオブジェクトが、モデル、テクスチャ、モーション、SE といった、多数のリソースを扱う。
- リソースマネージャについては、別紙の「開発の効率化と安全性のためのリソース管理」を参照。

● シーンに関する要件

シーンマネージャは、多数のシーンを管理し、シーンごとにシーングラフとシーン固有の処理を管理する。

- シーンに登場するキャラクターなどの「シーンオブジェクト」(「シーン ID」で一意に識別するオブジェクト）は、全シーンで共通利用するが、その関係を構成する「シーングラフ」はシーンごとに扱う。
- シーン固有の処理は、多数の「シーンコントローラ」によって扱う。
  - レベル（マップ／ステージ）制御や、BGM・環境音の制御などは、それぞれ専用のシーンコントローラによって処理する。
  - 一つのシーンは、多数のシーンコントローラを持つ。
- シーンのインスタンスは複数同時に存在し、別シーンに切り替えをする際には、二つのシーンがオーバーラップすることで高速切り替えを行う。
  - 例えば、イベントシーンに切り替える前に、イベントシーンの構築が始まり、ポイント到達と同時にイベントシーンが再生される。
  - また、例えば、マップ境界に近づいたところで、次のマップの移動シーンの構築が始まり、境界到達から短時間で次のマップにつなぐ。

シーンとシーングラフ、シーンコントローラ、シーンオブジェクトの関係：



● シーンオブジェクトに関する要件

シーンに登場する個々のオブジェクトの姿形、状態、処理を扱うために、「シーンオブジェクト」を用いる。

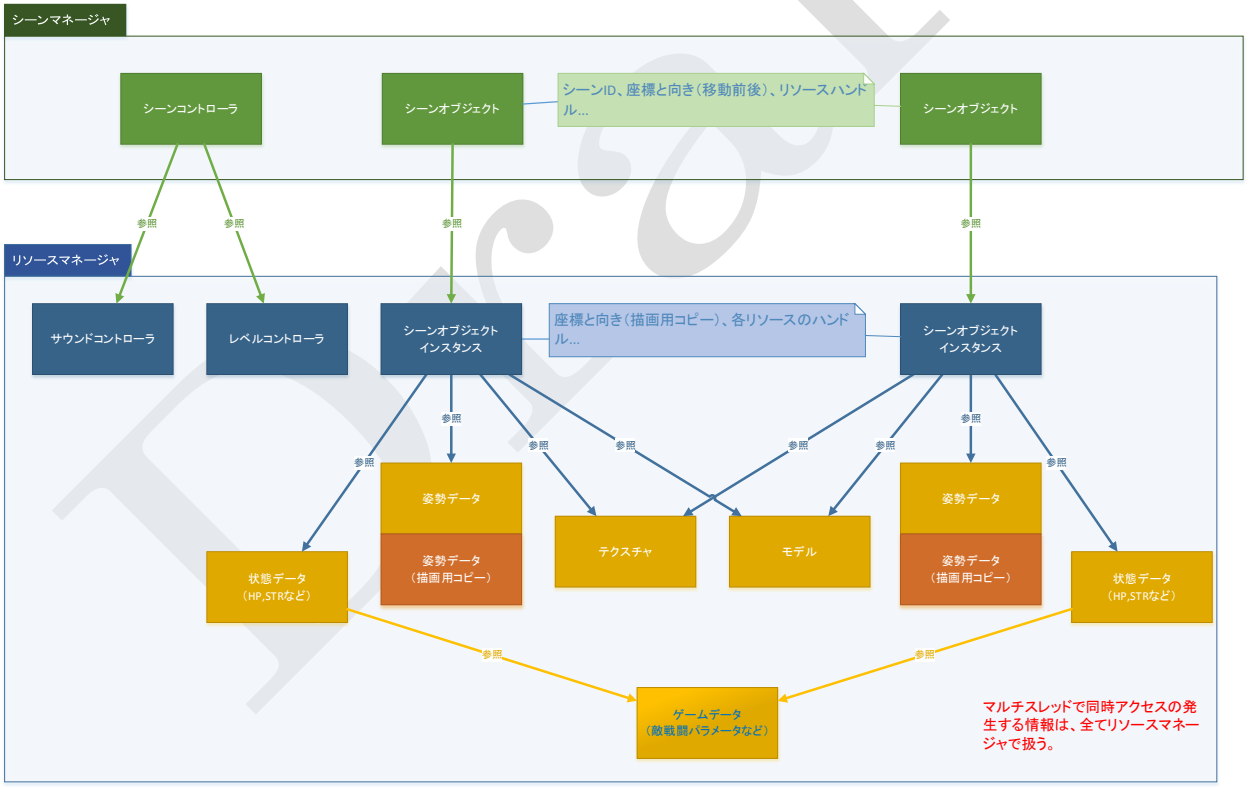
- シーンオブジェクトは、「キャラ」（モデル）、「マップ」、「エフェクト」、「光源」、「発音体」（SE）、「カメラ」、「頭上アイコン」、「トリガーポイント」（何らかのイベント発生点）など、座標を持ったあらゆるオブジェクトを扱う。
  - オブジェクトの可視性は問わず扱う。
- シーンオブジェクトは、そのオブジェクトが扱うリソースの全てのハンドルを管理する。
  - 「モデル」、「テクスチャ」、「モーション」、「メニューデータ」、「SE」、「状態データ」（HP や攻撃力など）などのリソースを、リソースマネージャを通して管理し、ハンドルを管理する。
  - シーンオブジェクトが必要とするリソース全てが「構築済み状態」にならないと、そのシーンオブジェクトが有効な状態としてみなされない。
- シーンオブジェクトは、シーンオブジェクトを一意に識別する「シーン ID」によって管理される。
- 個々のシーンオブジェクトが扱う情報（保持するべきリソースハンドルも含む）やアップデート処理、描画処理（描画時アニメーションを要する時のみ）は、タイトル固有の任意の構造・処理を実装することが可能。
  - シーンオブジェクトの抽象クラスに基づいて任意のクラスを作成する。
  - 一般的なモデルや SE など、規定のクラスも多数。
  - シーンオブジェクトのインスタンス自体は、リソースマネージャを通してリソースとして管理する。

- シーンオブジェクトを構成する上で、必須の要素。

➢ シーンマネージャ上で管理するシーンオブジェクトの情報は、「シーン ID」、「基本座標と向き」、「ワールド座標系での座標と向き」（移動前と移動後）、「シーンオブジェクトインスタンス（リソース）ハンドル」、「構築状態」、「ビジビリティ」、「透明度（アルファ）」、「基本アニメーション経過時間」、「ポーズ状態」、「処理時間レート」（スローモーション表現などに使う）といったもので固定する。

- これ以上の情報は、シーンオブジェクトインスタンスを使用し、任意の構造や処理を実装して扱う。
- 「基本座標と向き」は、シーングラフ上の親オブジェクトのローカル座標系で扱う。
- ただし、「Y 軸の向きだけは追従しない（ワールド座標系で扱う）」のような特殊な連結制約を用いることが可能。ビルボード方法の指定にも用いる。

シーンオブジェクトとリソースの関係：



● シーンコントローラに関する要件

シーンの構成要素はシーンオブジェクト以外にもある。

多数の専用処理により、ゲームの場面に応じ敵や NPC の出現（シーンオブジェクト登録処理）、BGM・環境音の制御などを行う。

- シーンを制御するための処理オブジェクトを「シーンコントローラ」と呼ぶ。
- 「移動シーン」や「バトルシーン」など、シーンに応じて任意のシーンコントローラを設定することができる。
- シーンコントローラはシングルトンでは扱わない。シーン切り替えは二つのシーンがオーバーラップすることで高速切り替えを実現するので、シーンコントローラのインスタンスはシーンごとにもつ。
- 各シーンコントローラの処理タイミングは、シーンごとに任意に設定可能。
  - 例えば、「イベントコントローラはシーンオブジェクトのアニメーションアップデート前」、「サウンドコントローラは行動決定処理のあと」、「バトルコントローラは移動前、移動後、行動決定後にそれぞれ」など。
- 主なシーンコントローラ：「レベルコントローラ」
  - マップの読み替えや、イベント発生点の配置、敵・NPC の配置、といったレベルを制御するためのシーンコントローラ。
  - ものによってはオープンワールドの制御も行う。
- 主なシーンコントローラ：「バトルコントローラ」
  - 戦闘を管理するためのシーンコントローラ。
- 主なシーンコントローラ：「イベントシーンコントローラ」
  - イベントシーケンスを再生して、キャラクターの演技やカメラ演出などを行うためのシーンコントローラ。
  - イベントシーン専用ではなく、移動シーンや戦闘シーンの最中でもリアルタイムに呼び出して、「特定のポイントに到達した時のリアルタイム演出」などに用いる。
- 主なシーンコントローラ：「サウンドコントローラ」
  - 主には BGM と環境音を管理するためのシーンコントローラ。（それぞれ専用のコントローラになっても良い）
  - サウンドマネージャのようにサウンド全体を管理するようなものではなく、あくまでも「そのシーンのため」のサウンド制御を行う。
  - 例えば、「戦闘から復帰した時にフィールド BGM を続きから再生」といった制御や、「特定の場所では BGM の音量を半分にする」といった制御を行う。
  - 重要なシーンでは通常の BGM 演出と変えて特殊な BGM 再生状態になり、シーンを切り替えても継続することがある。そうした処理の実現のために、サウンドコントローラの一部の情報はシングルトン（static）で扱い、サウンドコントローラ間で継続的な状態管理が行えるように構成する。

- シーンコントローラとして扱わない：「メニューシステム」
  - メニューシステム（2D 描画システム）は、シーンマネージャやリソースマネージャと同様に、ゲーム中に一つだけ存在する基幹システムの一つとして扱い、シーンに登場するメニューはシーンオブジェクトとして扱う。

● シーングラフに関する要件

シーンオブジェクトどうしの関係を明確にするために、「シーングラフ」を用いる。

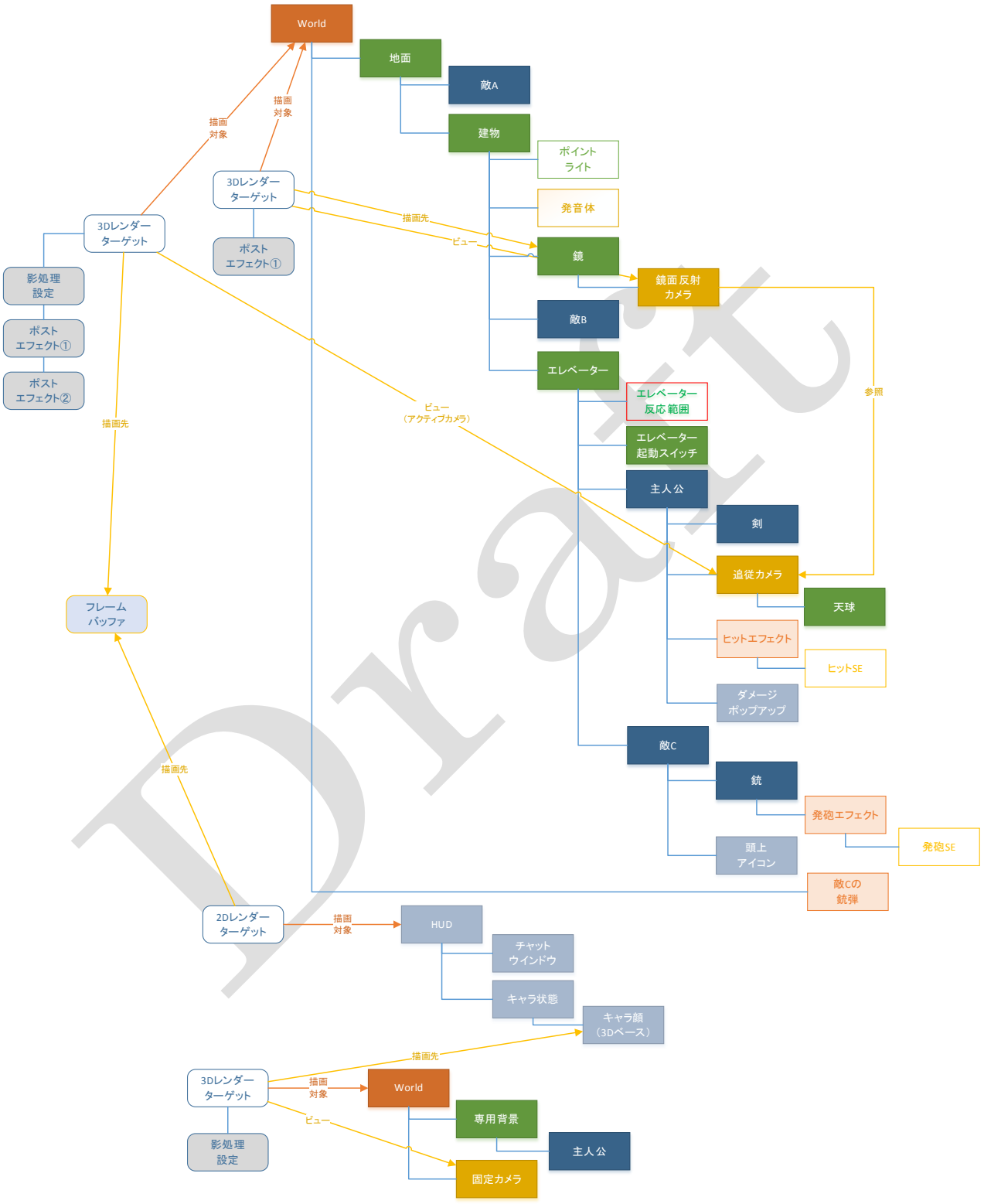
- 例えば、「崩れ落ちる床」や「上昇するエレベーター」、「巨大な振り子」などといった、動く足場があるとし、そこに乗っているキャラはその足場に追従して一緒に移動する。キャラが手に持っている剣はさらにそのキャラに追従する。また、エレベーター内の「起動スイッチ」（決定ボタンに反応するポイント）は、エレベーターの昇降に追従する。
- こうした追従動作を、シーングラフで依存関係（親子関係）を設定することにより、対応する。

● レンダーターゲットに関する要件

シーングラフの描画方法を指定するために「レンダーターゲット」を用いる。

- シーングラフとレンダーターゲットを関係づけることで、描画対象と描画先を設定する。
- 影の描画の有無や使用するポストエフェクトなどは、レンダーターゲットに対して設定する。
- レンダーターゲットは 2D 描画やマルチスクリーンに対応して複数扱うことができ、それに伴い、シーングラフも複数扱うことができる。

シーングラフとレンダーターゲットのイメージ：



● シーンカテゴリグラフに関する要件

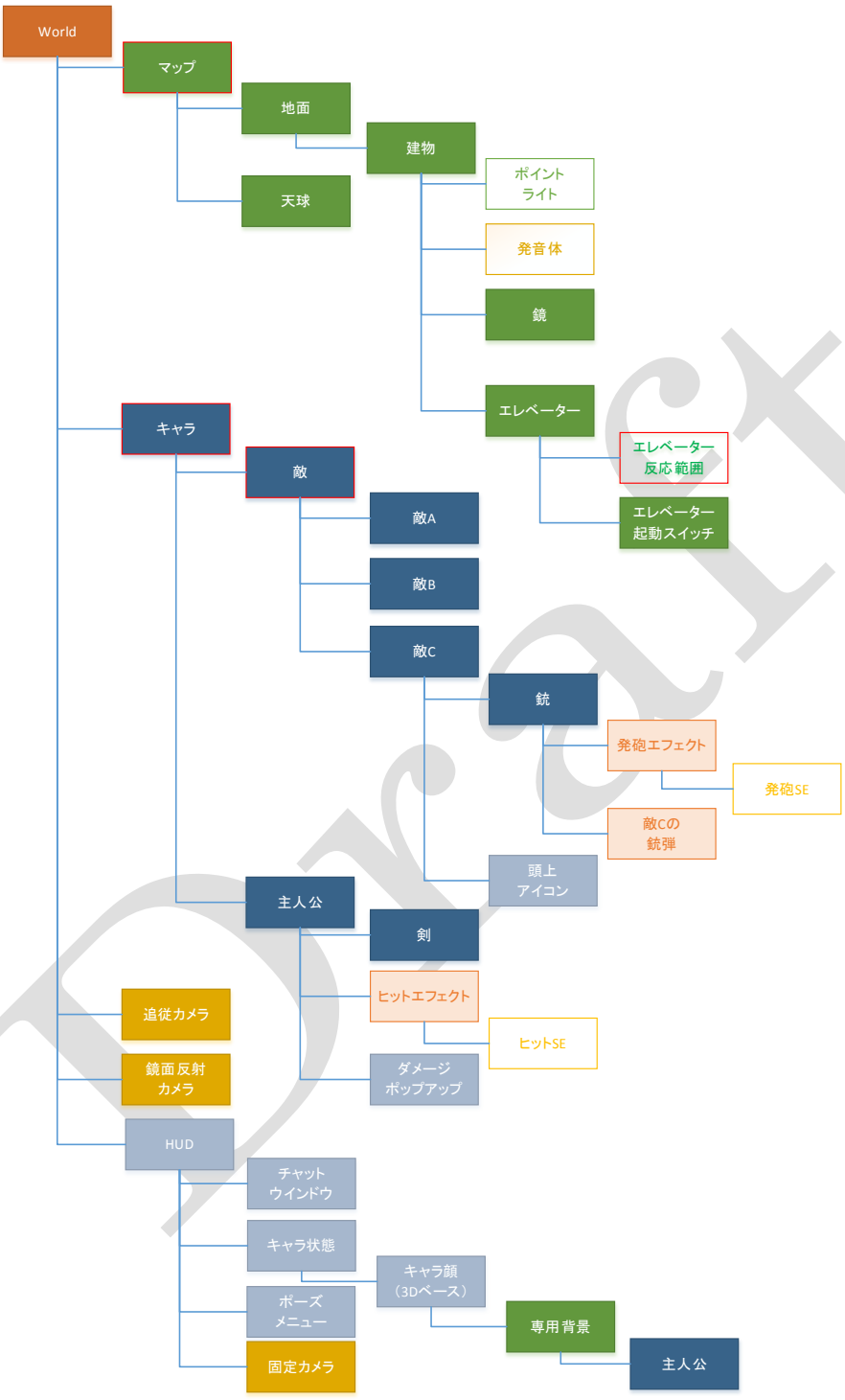
シーンオブジェクトは、物理的な関係を示すシーングラフとは別に、「シーンカテゴリグラフ」を用いて階層的なカテゴリを構成する。これにより、「ポーズ」や「ビジビリティ」、「スローモーション」の一括操作を行う。

- 例えば、「キャラだけ全てポーズ」、「一部のキャラだけスローモーション」、「特定の敵とそれに関するオブジェクト（所持武器、エフェクト、SE）を全て消す」といった、特定のカテゴリに対する一括操作に用いる。

- シーンオブジェクトの「ビジビリティ」、「ポーズ状態」、「処理時間レート」は、「親オブジェクトと同じ」という設定が基本。
  - 階層のトップノードに対してポーズやスローモーション（1.0 未満の処理時間レートを設定）すれば、全体のポーズやスローもションが設定できる。
  - 途中階層のノードを指定して、「マップだけポーズ」や、「NPC の表示を消す」といった制御が可能。デバッグにも活用できる。
  - 子オブジェクトは、親オブジェクトの設定を継承せず、独自の設定を行う事も可能。
  - これを利用し、例えば、「全キャラをポーズ」させた状態で、「主人公と直接の相手だけ動いている（ポーズ解除）」といった動作が可能。
  - また、「全体がポーズしている中、ポーズメニューだけが動作」という状態も、この仕組みを利用して作り出す。

- （物理的な）シーングラフには存在しない、カテゴリ操作のためだけのシーンオブジェクトを扱う事が可能。

シーンカテゴリグラフのイメージ：



- **シーンオブジェクトのアップデート処理に関する要件**

個々のシーンオブジェクトは、数段階のアップデート処理を行う。また、シーングラフに基づいて、可能な限り並行処理を行う。

- アップデート処理①：アニメーションと移動処理。
  - アニメーション（モーション）と移動を行うためのアップデート処理。
  - プレイヤーのコントローラ操作やカメラ操作もこの処理で行う。
    - コントローラ操作は、「前のフレームのカメラ」（つまりユーザーが目で見て操作を判断した画面）に基づいて行う。
    - カリング処理は描画処理の際に行うので、カメラの処理タイミングを無理に最初に持ってくるなどせず、シーングラフに基づいて順当に処理する。
    - ただし、カメラが確定しないとオブジェクトの「アニメーション省略判定」ができないといった問題は残るが、これ以上は無理せず、整合の取れた処理順序を保証することを優先する。
  - シーンマネージャは、親のアニメーションと移動の結果に基づいて、子のワールド座標系座標を更新した後、アップデート処理を呼び出す。
  - 親子関係にあるオブジェクトは並行処理されないが、そうでないものは並行で処理される。
  - この段階では他のオブジェクトの座標を参照してはいけない。
  - 十分に遠いキャラや、（カメラ位置確定後）可視領域から外れることが確定しているオブジェクトは、アニメーション処理を省略し、少しでもパフォーマンスを稼ぐ。
    - 移動→アニメーションの順で処理することで、この判定を確実にする。
    - ダイナミックなアニメーションで突然視界に入ってくるようなものや、オブジェクトを動かすための親オブジェクトになっている場合もあるので、オブジェクトごとに「アニメーション省略禁止」の設定も可能とする。
- アップデート処理②：衝突コールバック。
  - 衝突に基づくアップデート処理。
  - 一通りのシーンオブジェクトの移動を終えた後、衝突判定と物理演算を行う。
  - 一通りのシーンオブジェクトの衝突に基づく座標補正が行われる。
  - 座標補正が終わったのち、その間に溜め込まれた衝突イベント（コールバック）を実行する。
  - このコールバックでは、攻撃のヒット処理や、着地などの衝突エフェクト&SE 処理、（可能なら）足音処理などを行う。
  - 足音などは、アニメーションの状態に応じてイベントが発行されることもあるが、そのようなイベントも含めてこのタイミングでまとめてコールバックする。
  - トリガーポイントへの進入・退出といったイベントにもこのタイミングで反応する。
- アップデート処理③：行動決定（意思決定）。
  - 全てのアニメーションと移動と終えて、そのフレームでの全オブジェクトの姿勢が確定した後に順次呼び出されるアップデート処理。
  - 例えば、「攻撃ボタンを押した」、「メインメニューを開くボタンを押した」という（移動を除

く）操作系の処理や、「AI による攻撃対象の決定」、「NPC の行動決定」といった、移動・衝突以外の処理全般を行う。

- 基本的に、この段階で座標を動かしてはいけない。

➢ アップデート処理④：サウンド。

- サウンドのアップデート処理では、SE の再生開始や SE の位置変更といった制御を行う。
- サウンドのアップデートは他のアップデート処理よりも後に回し、描画スレッドを始動した後に行う。これは、並行処理をより効率化するためである。
- サウンドのアップデート処理は規定の動作に従うのみで、個別に任意の処理を記述することほとんどない。
- 例外として、「線音源」「面音源」のような広域的な発音体の発音点を決定するような処理は、任意に記述する必要がある。（線や面のデータを別途扱う必要があるため、規定のサウンド処理で完結できない）

➢ アップデート処理の呼び出し時は、前のフレームの処理に費やした処理時間が渡される。

- 通常は 30fps なら 1/30 sec が、60 fps なら 1/60 sec が基準。処理落ちしない限りは常にこの時間が渡される。
- 処理落ちした時は通常より長い時間が渡される。（30 fps で 1.5 フレーム分の処理時間がかかったなら、 1.5/30 sec となる）
- この処理時間に応じたアニメーションや移動量を進める。
- さらに、シーンオブジェクトごとの「処理時間レート」が掛けられた値が処理時間としてえ扱われる。
- 毎フレームの画面をキャプチャしてムービー化するような用途のために、実際の処理落ちと無関係に、一定の処理時間が渡される実行モードがある。
- デバッグ用に、通常の数倍の処理時間が渡される早送りモードがある。

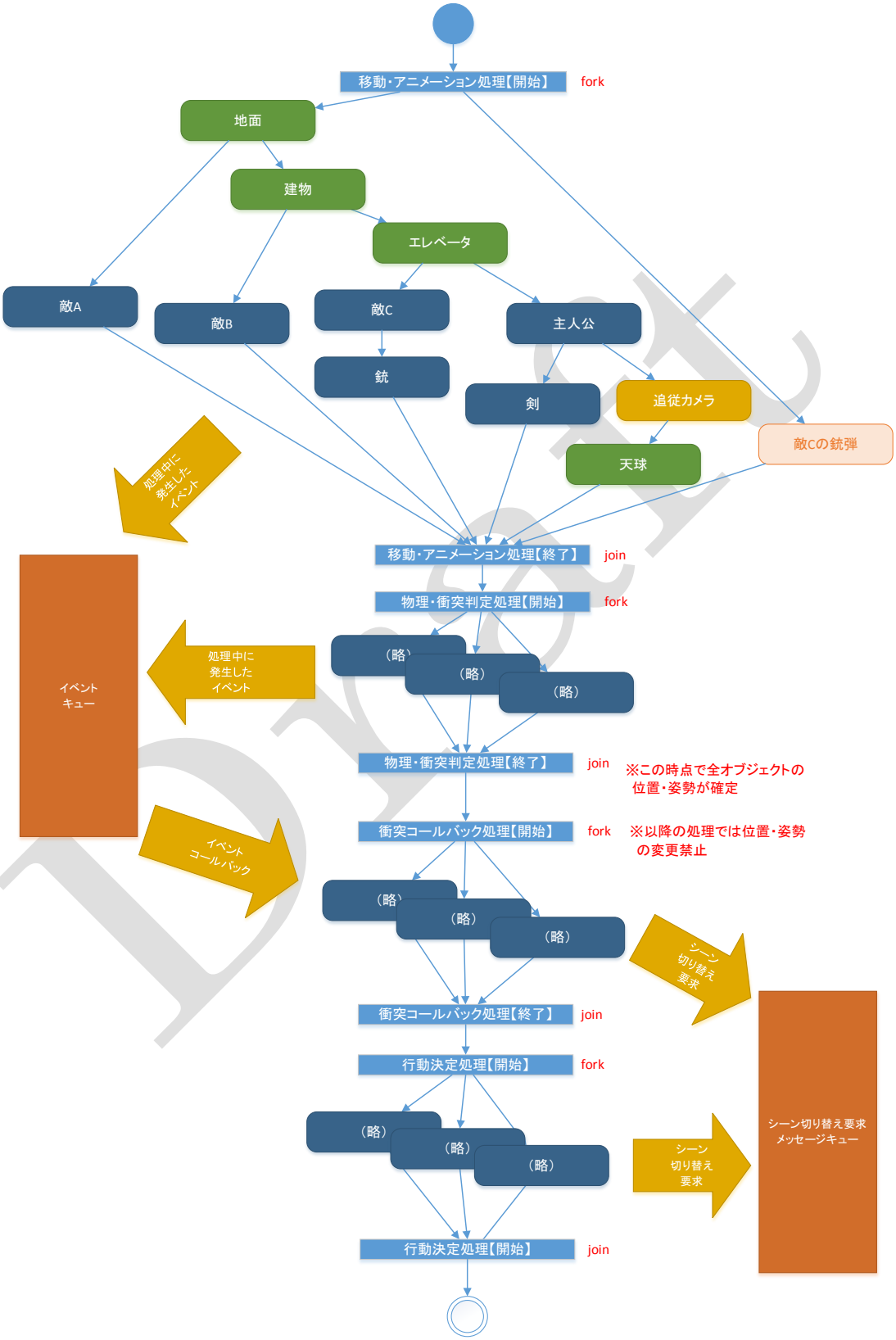
➢ ポーズ状態にあるシーンオブジェクトは、一切のアップデートが呼び出されない。

● シーンオブジェクトの並行処理に関する要件

シーンオブジェクトの並行処理は、「ジョブスケジューラ」を通して実行する。それにより、並列処理性能と依存関係に基づく最適なスケジューリングで処理される。

ジョブスケジューラについては、別紙の「「サービス」によるマルチスレッドの効率化」を参照。

アップデート処理とその並行処理のイメージ：



● シーンオブジェクトの描画処理に関する要件

描画は、描画スレッドで行われる。描画処理を開始する際に投入されたシーン ID のシーンオブジェクトが可能な限り並行で処理される。

- 最初に、描画と並行してアップデート処理が実行されても良いように、情報のコピーを行う。
- 【コピー処理①】リソースマネージャのリソースとして管理される「シーンオブジェクトインスタンス」内の描画用情報に「ワールド座標系での座標と向き」(移動後のみ)をコピーする。
- 【コピー処理②】同様にリソース管理される「姿勢情報」も、描画用にコピーを行う。
  - 姿勢情報はアップデート用と描画用の情報が二重化されている。
  - ものによっては、ここでバウンディングボックスを再計算する。
- 【コピー処理③】描画用のシーングラフのコピーを行う。
  - グラフィカル要素のシーンオブジェクトのみのシーングラフを構成する。
  - 不透明シーングラフ、半透明シーングラフ、影描画シーングラフなどに分けてコピーする。
- 以上のコピー処理が終わるまで、メインスレッド側は、次のフレームのアップデート処理を実行できない。
- 描画処理は、マテリアルおよびシーンオブジェクトの透明度に基づいて、不透明描画フェーズと半透明描画フェーズで処理する。
  - 透明度が設定されたキャラは、影の描画もアルファ付きで個別に行うことを検討したほうがよい。
- 草木のような大量配置物は、一つのシーンオブジェクトが多数の位置情報を配列で持ち、個々の姿勢データを持たない。そのため、アップデート処理でアニメーションできない。アニメーションをさせるには、描画時に姿勢を更新する必要がある。
  - シェーダーで姿勢を更新するのが最も簡単な方法。
  - ほか、シーンオブジェクトインスタンスの描画関数を使用することができる。
- シーンオブジェクトインスタンスは、任意の描画関数を定義し、描画時だけの姿勢更新を行うことができる。
  - ちょっとした回転物などは、モーションを使用せず、この方法でアニメーションさせることができる。
  - 大量配置物の場合、配置物の一つ一つに対して呼び出される。
  - ものによっては同じオブジェクトが不透明フェーズと半透明フェーズの二回呼び出されるので、どちらで呼び出されても結果が変わらない方法でアニメーションさせなければならない。

乱数の使用は禁止。

● シーン ID に関する要件

シーン ID は、シーンオブジェクトを一意に識別する ID。

- シーン ID は 24bit の整数値とする。
  - 32bit にしないのは、64ibt のリソース ID を扱う際、シーン ID（24bit）＋リソースファイルの CRC（32bit）＋任意のデータ種別（8bit）で扱えるようにするため。
- 24bit 中の下位 16bit は単純な連番であり、シーンオブジェクトの配列要素番号＋１を表す。
  - 0 は無効な ID を示す。
  - 発番のたびに、1, 2, 3, … と順に割り振られる。
  - 仮にシーオブジェクトの配列の要素数が 8192 なら、値の上限は 8192 となる。
  - これにより、シーン ID からシーンオブジェクトの実体（配列要素番号）の場所を素早く算出する。
  - 上限を超えたらまた配列要素 0（+1）番 から発番されるが、使用中のものはスキップする。
  - 例えば、ID = 8192 のシーン ID が発番された後、次のシーン ID を発番する際、配列要素 0 番 のシーンオブジェクトが使用中なら、それをスキップして 1 番の割り当てを試みる。
- 24bit 中の上位 8bit は、一度使用された配列番号を再利用するたびにカウントアップする。
  - 一度使用を終えて未使用状態になったシーンオブジェクトには、前回のシーン ID が残っている。それを参考に、シーンオブジェクトを再利用する際は、前回の ID との重複を避けて発番する。
  - 例えば、配列要素 0 番のシーンオブジェクトが「シーン ID＝1」（0x000001）で発番されたことがある状態なら、次は「シーン ID =65537」（0x010001）として発番する。0xff0001 の次は、0x000001 に戻る。
  - これにより、同じシーン ID が再利用されるまでのスパンを長くし、誤ったシーンオブジェクトへのアクセスを防ぐ。
    - 例えば、「シーン ID=1」のシーンオブジェクトにアクセスしようとしても、配列要素 0 番のオブジェクトが「シーン ID=65537」として再割り当て済みなら、「シーン ID=1」のシーンオブジェクトは既に存在しないものとして、アクセスに失敗する。
- 任意の「固定シーン ID」が使用可能。
  - 自動発番対象の ID の範囲（下位 16bit の最小値と最大値）を設定可能とし、その範囲外の ID を固定シーン ID として任意に使用できる。

- 例えば、「シーン ID = 1 を主人公に固定する」ということにし、「シーン ID の自動発番は 2 番から」という設定が可能。
- 固定シーン ID は、上位 8bit が 0 で、使用可能なシーンオブジェクトの配列要素でなければならない。
- 通常、256 ぐらいは自動発番の対象から外し、主人公や仲間キャラ、メニューオブジェクトなどの固定用途に割り当てて使用する。

➢ シーン ID は、全シーンを通してユニークであり、かつ、共通利用する。
- 例えば、移動シーンと戦闘シーンの両方で「ID=1」の主人公キャラを扱うなら、どちらも同じシーンオブジェクトである。
- もし、戦闘シーンでの操作を移動シーンに全く影響を与えず、移動シーンの状態を維持したい場合、それぞれシーン ID の異なるキャラとして扱うとよい。
- イベントシーン再生時に既存のオブジェクトを使用しつつ、完全に元の状態に復帰したい場合、シーンオブジェクトを複製して別のオブジェクトとして扱う。

● トリガーポイントに関する要件

「トリガーポイント」は、シーオブジェクトの一種で、シーン（マネージャ）が直接制御するオブジェクト。

➢ 不可視オブジェクトで、キューブ、シリンダー、スフィアといった形状がある。

➢ 「進入」「退出」「進入中毎フレーム」のイベントトリガーがあり、トリガーポイント毎に任意の処理を定義できる。

➢ 主には、「操作キャラがトリガーポイントに到達したらイベント発動」「別のマップに移動」といった操作に使用する。

➢ 任意の処理が記述可能なため、「フラグ更新」や「ライティング変更」などの処理も行える。

➢ 操作キャラに反応するだけではなく、全てのキャラに反応して、シーングラフのつなぎ替えを行うようなようとにも用いる。
- 例えば、主人公や敵キャラがエレベーターに乗ったら、シーングラフ上はエレベーターの子オブジェクトとなり、エレベーターの動きに合わせていっしょに動くようになる。
- 退出トリガーによって、エレベーターの親オブジェクトの子に連結され直す。
- このような、シーングラフつなぎ替えトリガーは、一つ一つ処理を記述せずとも、トリガー種別として設定するだけで機能するようになる。

- イベント発動や別マップへの移動といった「シーン切り替えのためのトリガー」は、「二重トリガー」として構成し、事前のシーン構築を行う。
  - トリガーの範囲の「1.5 倍」といった設定で、「事前シーン構築」が反応する。
  - トリガー範囲の「1.75 倍」といった設定で、その範囲から外に出た時に「事前構築シーン」を破棄する。
  - シーン切り替えのためのトリガーは、一つ一つ処理を記述するのではなく、「切り替え対象のシーン」、「イベント ID やマップ ID などのパラメータ」、「繰り返し起動防止用のフラグ」といった設定をおこなうこと、「二重トリガー」が自動的に機能するようにする。
- トリガーの実行は、「アップデート処理②：衝突コールバック」と同様に、衝突検出の一環で実行判定を行う。
- 全般的に退出トリガーを優先実行する。
  - 密着したトリガーポイントのある場面では、「A からの退出」と「B への進入」が同時に発生するが、必ず退出トリガーが先に反応することを保証する。

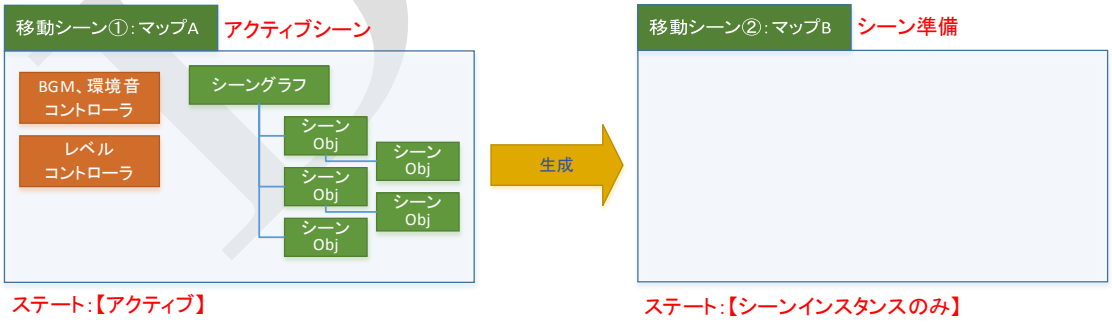
● シーン遷移に関する要件①：シーンの移行

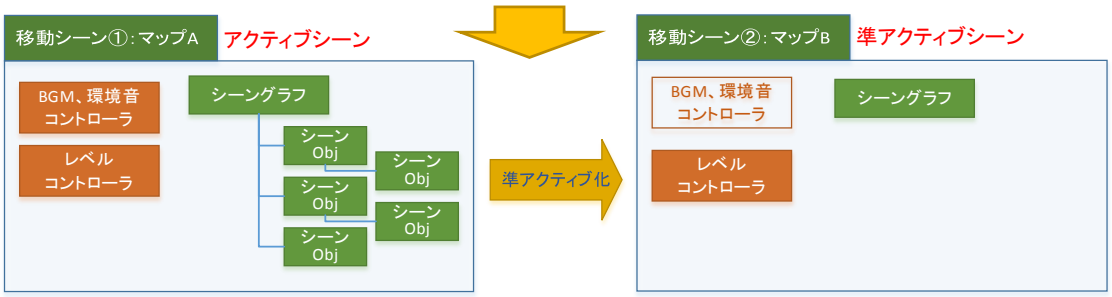
シーンから別シーンに遷移し、元のシーンが消滅するパターンに対応する。

主なシーン遷移のパターン：　「別マップへの移動」、「現在のシーンと無関係のイベントシーン（回想シーンや別の場所のイベントなど）」

【ケース】トリガーポイントへの進入により、「マップA」から「マップB」にマップジャンプする場合

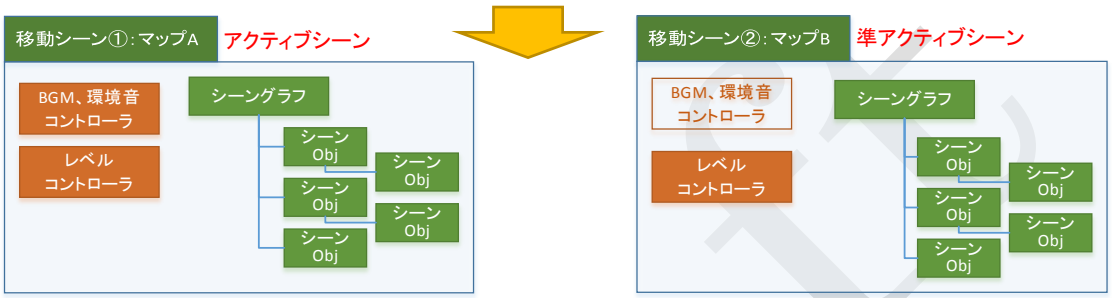
「マップジャンプ」トリガーポイントの、「事前シーン構築」の範囲に進入





ステート：【アクティブ】

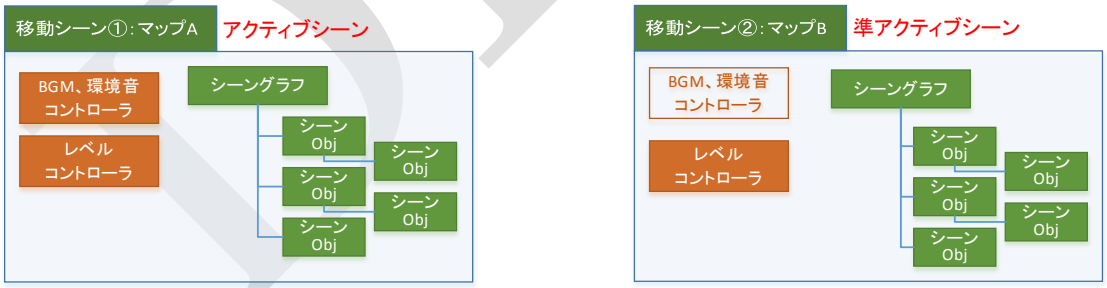
ステート：【シーン初期化】
・シーンコントローラ生成
・レベルコントローラ起動



ステート：【アクティブ】

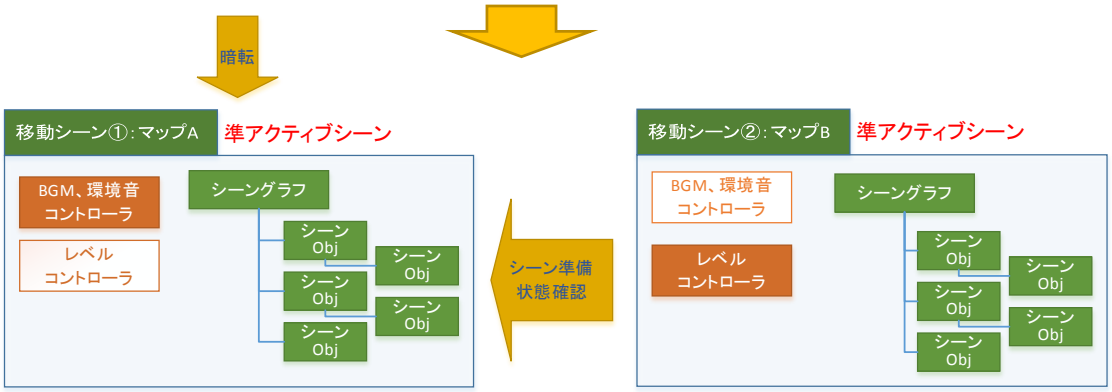
ステート：【シーンコントローラ初期化】
・レベルコントローラによるシーンオブジェクト生成
※リソースは「サスペンド状態」で構築する。
※実際にリソースの構築は行わないが、「レベル設定」などの読み込みは済ませ、「構築すべきマップやキャラ」の情報は読み込み、シーングラフは構成する。
※シーンに跨がるリソースの参照カウンタをカウントアップさせておき、元のシーンが消えてもリソースが削除されない状態を作る。

「マップジャンプ」トリガーポイントの、「トリガー実行」の範囲に進入



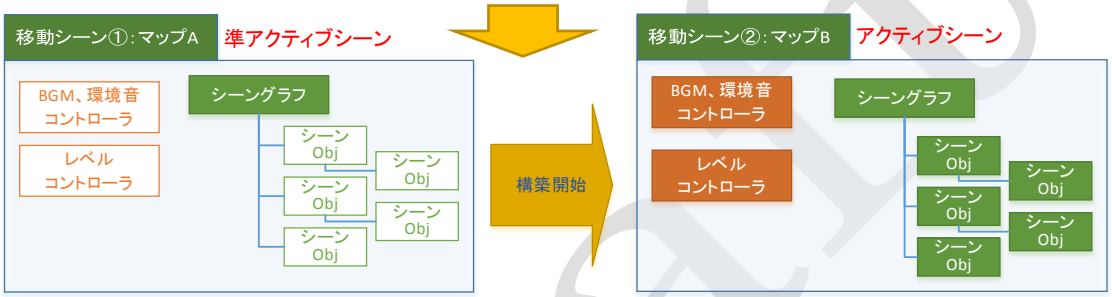
ステート：【アクティブ】

ステート：【シーン準備中】
※まだリソース構築は行っていない。



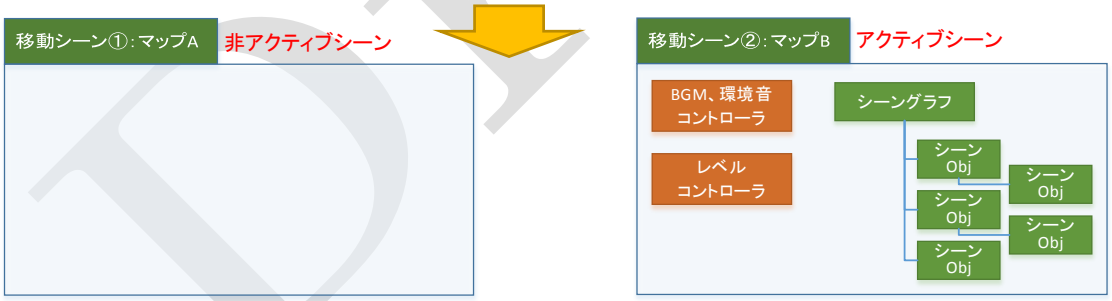
ステート：【終了準備中】
※レベルコントローラ活動停止

ステート：【シーン準備完了】
※まだリソース構築は行っていない。



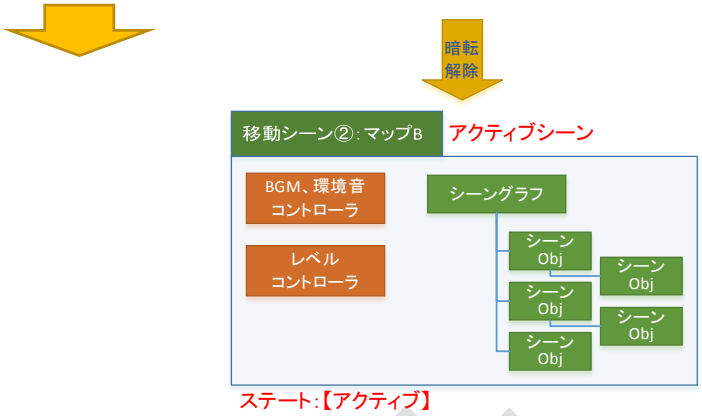
ステート：【終了中】
※シーンオブジェクト削除。
※これにより、リソースの参照カウンタもカウントダウンし、参照の無くなったリソースは自動的に削除される。

ステート：【シーン構築中】
※リソース構築開始。（リソース構築のサスペンド状態を解除）



ステート：【終了】

ステート：【シーン開始待ち】
※リソース構築完了確認。
※シーンオブジェクトの全ての構築待ちをするわけではなく、可視範囲内のキャラが構築されればOkとする。リソース構築開始時にマーキングしておき、マーキングされたものの構築が済んだら完了。その後もマーキングされていないものの構築が進む可能性はある。



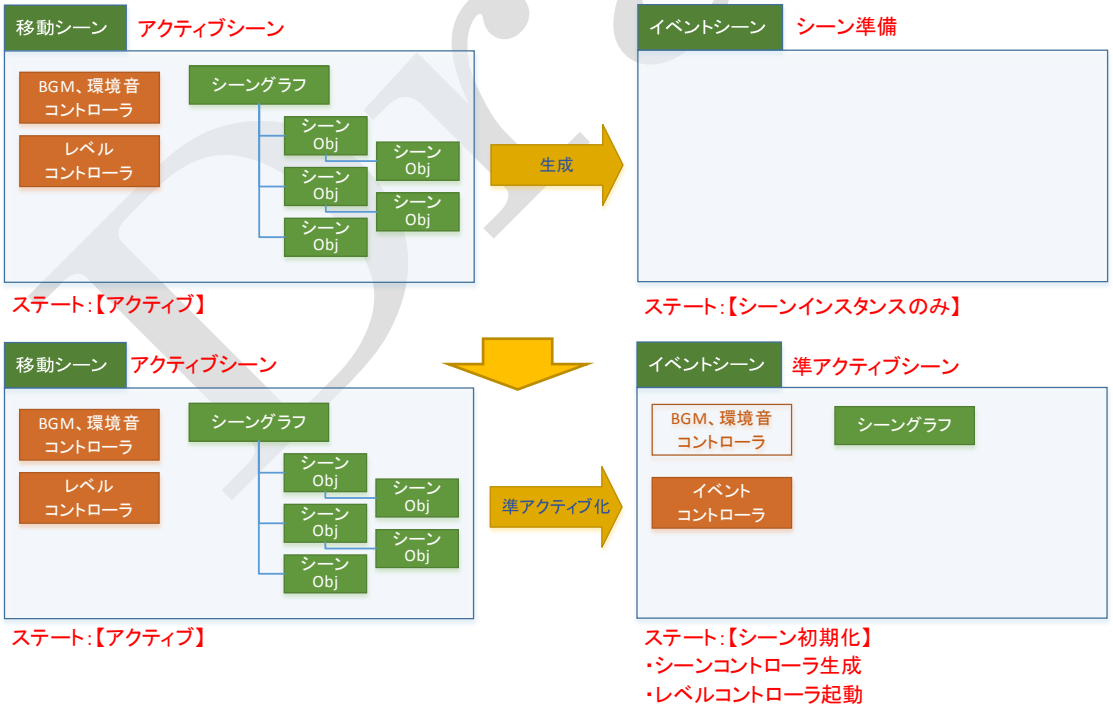
● シーン遷移に関する要件②：シーンの複製

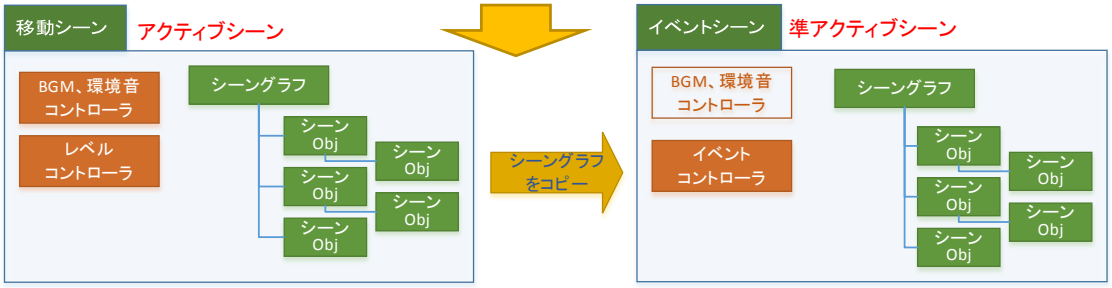
シーンを複製して遷移し、元のシーンが消滅するパターンに対応する。

主なシーン遷移のパターン：　「現在のシーンからつながるイベントシーンで、その後別のシーンにつながる（戦闘⇒イベント⇒移動など）」

【ケース】トリガーポイントへの進入により、「イベントシーン」が起動する場合

「イベントシーン」トリガーポイントの、「事前シーン構築」の範囲に進入

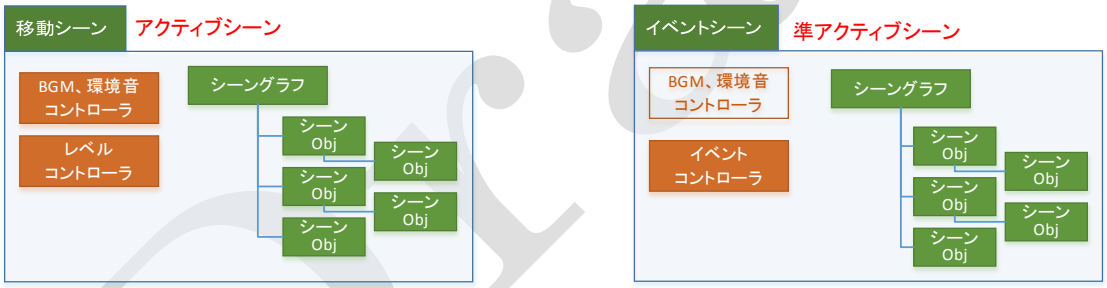




ステート:【アクティブ】

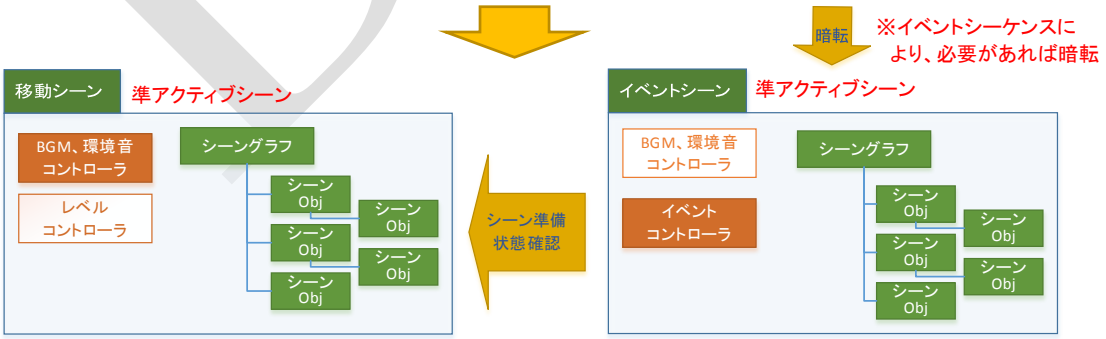
ステート:【シーンコントローラ初期化】
・まずはシーングラフをコピーしてシーンオブジェクトを共有。(グラフィカルオブジェクト以外はコピーしない)
・イベントコントローラにより、イベントシーケンスの読み込みと実行を開始し、1カット目のシーンオブジェクトを準備する。
※新たに必要なリソースは「サスペンド状態」で構築する。
※実際にリソースの構築は行わないが、シーングラフは構成する。
※シーンに跨がるリソースの参照カウンタをカウントアップさせておき、元のシーンが消えてもリソースが削除されない状態を作る。

「イベントシーン」トリガーポイントの、「トリガー実行」の範囲に進入



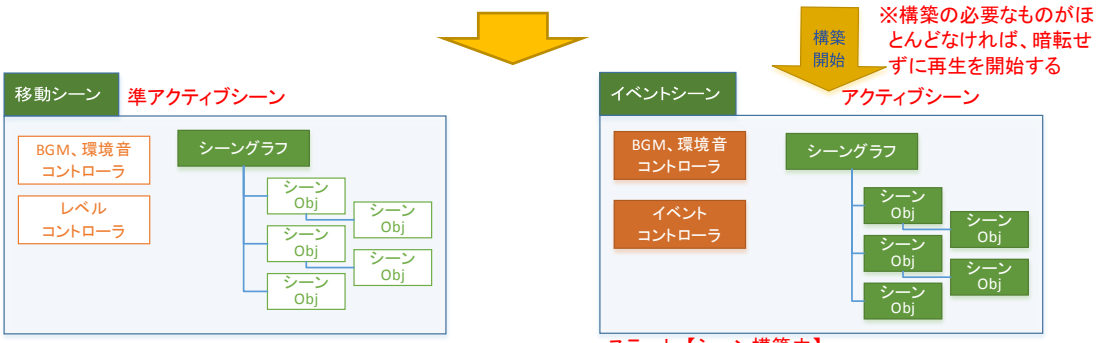
ステート:【アクティブ】

ステート:【シーン準備中】
※まだリソース構築は行っていない。



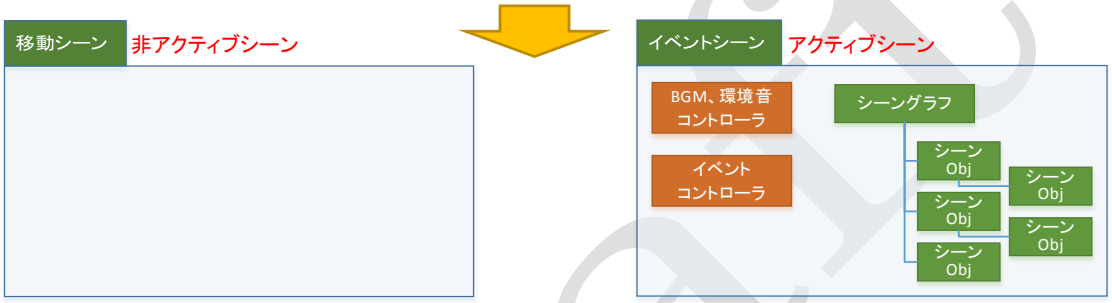
ステート:【終了準備中】
※レベルコントローラ活動停止

ステート:【シーン準備完了】
※まだリソース構築は行っていない。



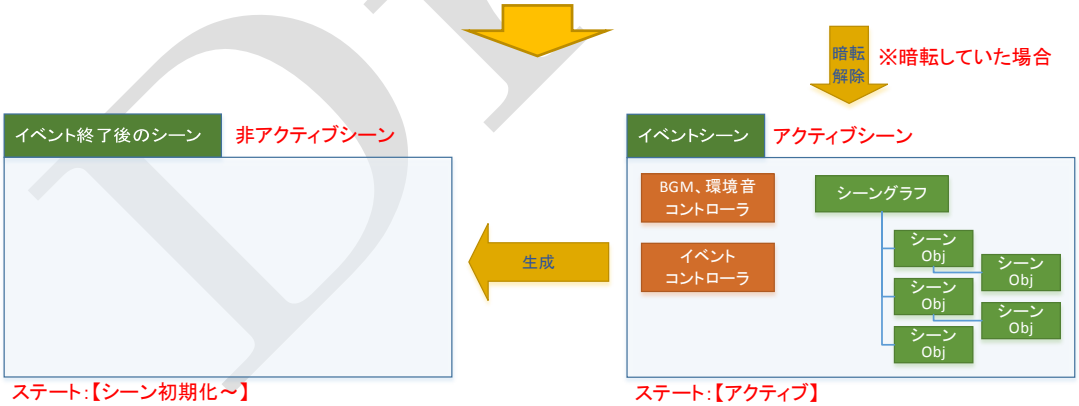
ステート:【終了中】
※シーンオブジェクト削除。
※これにより、リソースの参照カウンタもカウントダウンし、参照の無くなったリソースは自動的に削除される。

ステート:【シーン構築中】
※リソース構築開始。(リソース構築のサスペンド状態を解除)
※必要に応じて、移動シーン専用のモーションを解放し、メモリを空ける。



ステート:【終了】

ステート:【シーン開始待ち】
※リソース構築完了確認。
※シーンオブジェクトの全ての構築待ちをするわけではなく、1カット目に必要なオブジェクトのみマーキングしておき、マーキングされたものの構築が済んだら完了。



ステート:【シーン初期化～】

ステート:【アクティブ】
※イベント終了前から(最終カットの時点から)、次のシーンの構築を始める。
※イベント終了後のシーンは、イベントシーケンスと一組の専用設定を用いて定義し、イベントシーケンスと共に読み込んでおく。
※他のマップの移動シーンとその立ち位置や、戦闘シーン、次のイベントシーンなどを扱う。
※イベント再生のバックグラウンドでシーンの構築を進める。

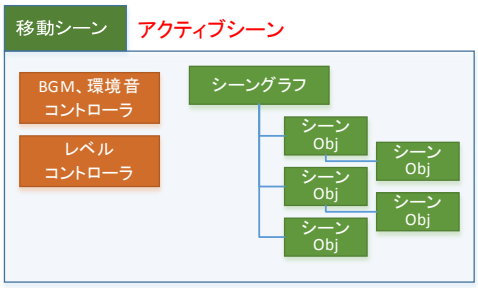
● シーン遷移に関する要件③：シーンのスタックと復元

シーンの状態を維持したまま次のシーンに遷移した後、また元のシーンを復元するパターンに対応する。

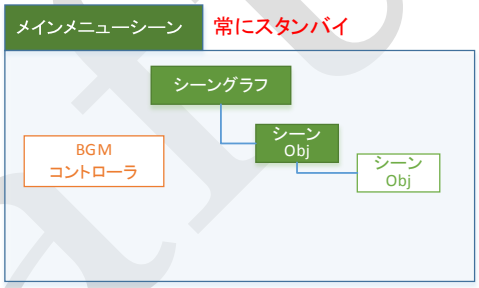
主なシーン遷移のパターン：　「移動シーン⇒メインメニューシーン⇒移動シーン」、「移動シーン⇒戦闘シーン⇒移動シーン」、「ミニゲーム」、「現在のシーンからつながるイベントシーンで、また元のシーンに復帰する」

【ケース】メインメニューを開く場合、さらに、その中でリッチなコンテンツのサブメニューを開く場合

普段の状態

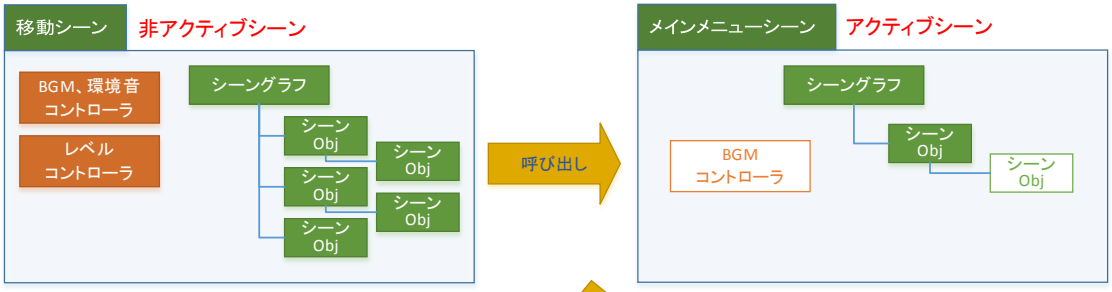


ステート:【アクティブ】



ステート:【シーン準備完了】
※ただし、一部のメニュー要素は未構築状態。
※シーンオブジェクトがリソースは掴んでいるが、リソースはサスペンド状態で実体がない状態。

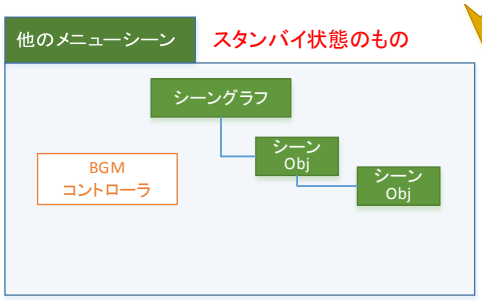
メインメニューボタンを押下



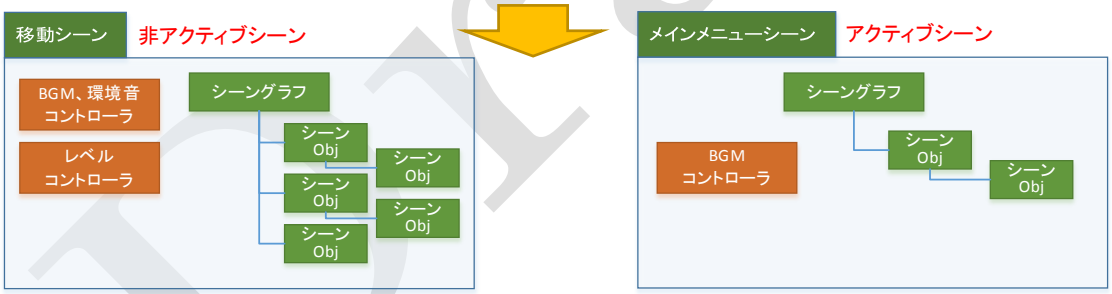
ステート:【一時停止】
※メインメニュー呼び出しと同時に即停止。

ステート:【シーン構築中】
※この時点でメニューの表示を開始する。
※操作はまだできない。
※他にスタンバイ中のメニューシーン(ショップメニューシーンなど)があれば、リソースを削除するように要求。
※未構築分のリソースの構築を開始。
※専用のBGMを扱うなら、BGMコントローラもアクティブ化。

リソース削除要求



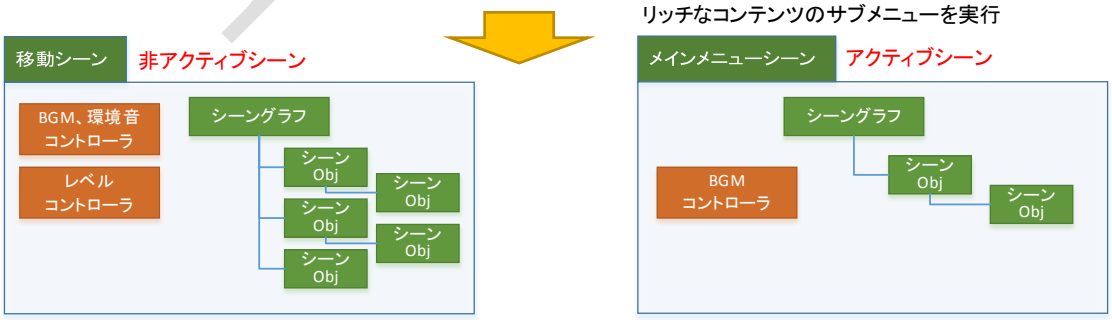
ステート:【シーン開始待ち】
※リソースを削除したら、ステートは「シーン構築中」になる。



ステート:【一時停止】
※とくにリソースの削除などは行っていない状態。

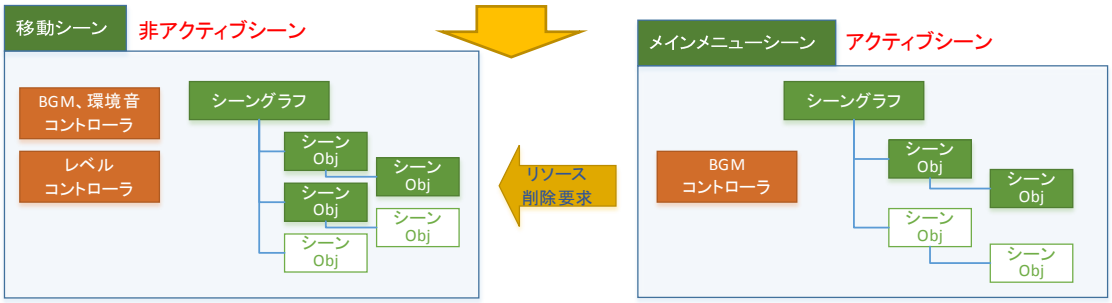
ステート:【アクティブ】
※リソースが構築され次第アクティブ状態となり、メインメニューが操作可能になる。

リッチなコンテンツのサブメニューを実行



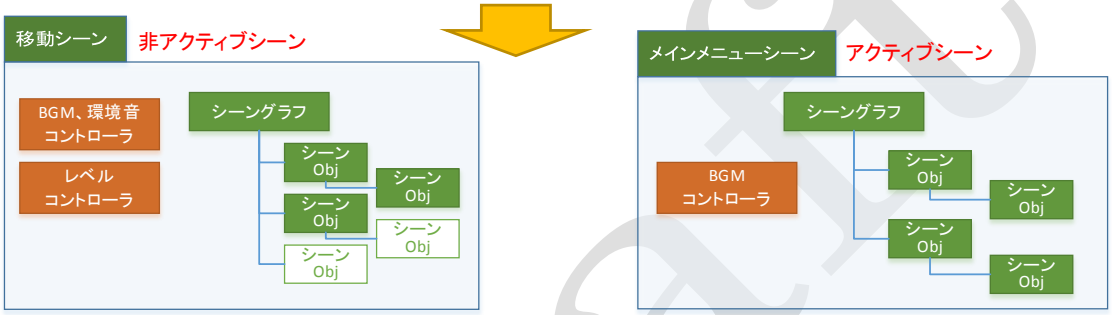
ステート:【一時停止】
※とくにリソースの削除などは行っていない状態。

ステート:【アクティブ】※メニュー遷移中
※メニュー遷移中は、メニュー内の制御により、操作ができない状態。



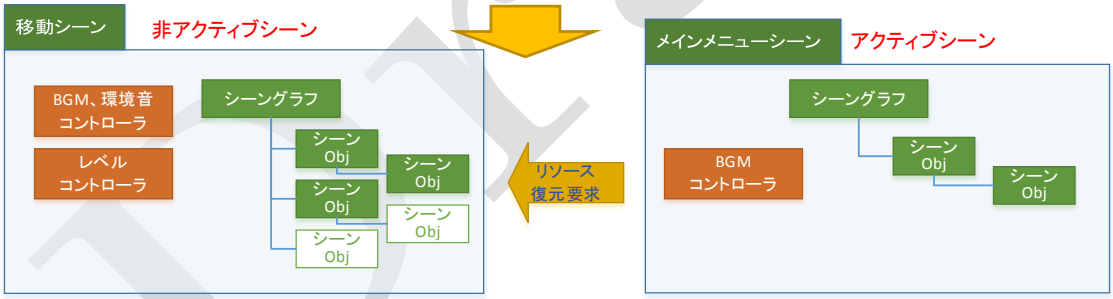
ステート:【一時停止】
※リソース削除要求を受けて、リソースの削除を行う。シーンオブジェクトは消さずに、リソースをサスペンド状態にして実体だけ削除する。また、「NPCだけ削除」など、条件付きで必要最小限の削除を行う。

ステート:【アクティブ】※メニュー遷移中
※コンテンツ用のシーンオブジェクトを追加。
※メモリが不十分なので、非アクティブ状態のシーンに、リソースの削除を要求。



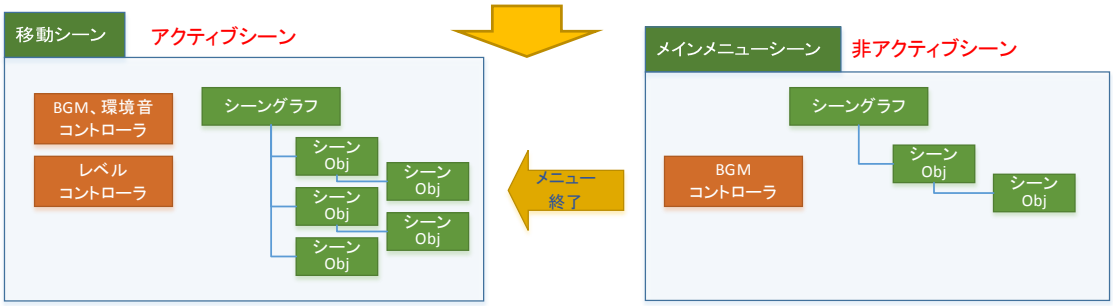
ステート:【一時停止】

ステート:【アクティブ】
※リッチなコンテンツのサブメニューに遷移完了。



ステート:【一時停止】
※リソース復元要求を受けて、リソースのサスペンドを解除する。サスペンドを解除されたリソースは、本来の状態に戻ろうとして、自動的にロードと構築を行う。

ステート:【アクティブ】
※サブメニューから抜ける際、コンテンツ用のシーンオブジェクトを削除し、非アクティブ状態のシーンに、リソースの復元を要求。
※とくにリソースの復元を待たずに操作を継続可能。
※再びサブメニューを開く時、リソースやファイルのキャッシュが残っていると、素早く開ける。



ステート:【アクティブ】
※リソース復元要求を受けて、リソースのサスペンドを解除する。サスペンドを解除されたリソースは、本来の状態に戻ろうとして、自動的にロードと構築を行う。

ステート:【シーン開始待ち】
※メインメニュー終了時は一気に非アクティブになる。
※シーンオブジェクトは破棄しない。
※他のメニューのためにメモリを空ける必要が生じるまで、シーンオブジェクトは構築状態を維持する。(再度メインメニューを開き直す時に高速に開ける)

基本的には、シーンをスタックするので、シーンを終えれば、(元のシーンが何であるかに依らず) 元のシーンに復帰できる。

また、基本的には、複数のシーンをスタックすることも可能。「リソースの破棄要求」などは、スタック上の全非アクティブシーンに対して要求される。

● シーン遷移に関する要件④：リソースのメモリを一時的大きく空ける

ムービー再生のために大きな連続領域が必要となった場合、リソースマネージャに対して、特定のメモリブロックのリソースを全て削除するように呼びかける。

主なシーン遷移のパターン： 「イベントシーン中のムービー再生」

- この時、リソースは「サスペンド状態」となる。
- 「サスペンド状態」になると、リソースは管理情報も参照カウンタもある状態だが、リソースの実体だけが削除された状態となる。
- また、サスペンド状態が解除されるまでは、再ロード～構築が行われない。
- ムービー終了後はサスペンド状態を解除する。それにより、自動的に再構築が始まる。
- ただし、シーンからの要求でサスペンド状態になっているリソースは、サスペンド状態が維持されたままとなる。
- 【課題】ムービーは事前構築ができないのが大きな難点。シーン切り替えのフェードアウト中に、他のファイル読み込みを禁止して、ファイル読み込みバッファにムービーのキャッシュを行う、といった工夫が出来ると良い。

● シーン遷移に関する要件⑤：シーンの重ね合わせ

シーンがアクティブなまま、もう一つのシーンをアクティブにするパターンに対応する。

主なシーン遷移のパターン：「ゲームオーバーシーン」、「現在のシーンをそのまま制御するイベントシーン」

- 単純に複数のシーンが同時にアクティブになることができる。
- 例えば、ゲームオーバーシーンなら、「主人公が倒れ、操作不能。敵キャラが動き回っている。カメラは動かせる。メインメニューは開けない。ゲームオーバーメニューが表示され、操作可能。メニュー選択と決定ボタンでタイトルシーンへ遷移するか、コンティニューとして移動シーンに復帰」といった状態になる。
- 複数のシーンが同時にアクティブになる場合は、一方のシーングラフのみを使用する。
  - 複数のシーンのシーングラフを同時に使うことはできない。
  - 例えば、ゲームオーバーシーンでは、元のバトルシーンに「ゲームオーバーメニュー」のシーンオブジェクトを追加登録して扱う。
  - シーンコントローラは追加シーン側のものが優先的に実行される。
  - 例えば、ゲームオーバーシーンの BGM コントローラでゲームオーバーの BMG を再生することや、イベントシーンのイベントコントローラがイベントシーケンスを再生して元の移動シーンのキャラを操作するなど。

● シーン切り替え要求に関する要件

シーンの切り替え要求は、トリガーポイントに進入した時や、メニューのボタンを押した時、主人公が戦闘で死亡した時などに発生する。

- シーン切り替え要求はメッセージキューを通し、フレームの処理の最後にまとめて判定する。
  - シーン切り替え要求を出す各処理は、状況を気にせずに要求して良い。
  - それが実行可能な要求であるかどうかは集中的に判定する。
  - 結果はとくに返さないので、要求を出した側の処理は、要求を出すだけで完結する。
- シーンの優先度に基づいて、優先度の高いシーンが一つだけ採択される。
  - 例えば、優先度の高い順に、ゲームオーバーシーン、イベントシーン、メインメニューシーンとなり、続く戦闘シーンと移動シーンは同じ優先度、のように設定する。
  - 優先度が同じものは先着優先となる。

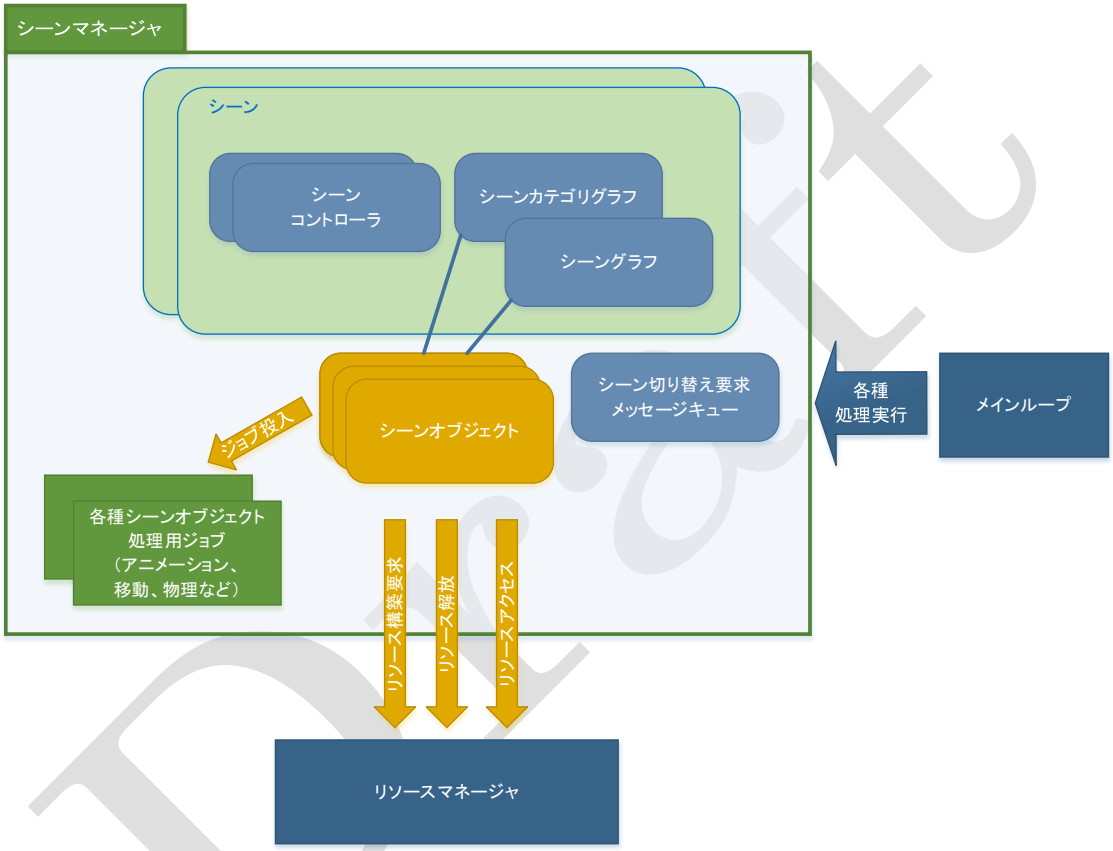
- シーンの優先度判定は、「メッセージキューの要求どうし」でのみ比較されるのではなく、「現在のシーン」、「現在切り替えようとしているシーン」とも比較される。
  - 例えば、「イベントシーン」に切り替わろうとしている状況では、戦闘シーンやメインメニューシーンへの切り替え要求は棄却される。
  - 例えば、「イベントシーンの事前構築中」のように、シーンの構築は行われているが、明確にアクティブ化が確定していないシーンは、優先度判定の対象にならない。
  - 例えば、「イベントシーンが次の移動シーンの構築を始める」といった処理は、直接のシーン制御処理であるため、シーンの優先度とは無関係に処理される。
- シーン切り替えの途中、まだ構築が進んでおらず、シーン切り替えを取り消すことが可能な状態であれば、より優先度の高いシーンの要求を受け入れ直すことができる。
  - シーン切り替えが十分に進んだ状態では、シーン切り替え要求が取り消される。

■ 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。

シーンマネージャのシステム構成図：

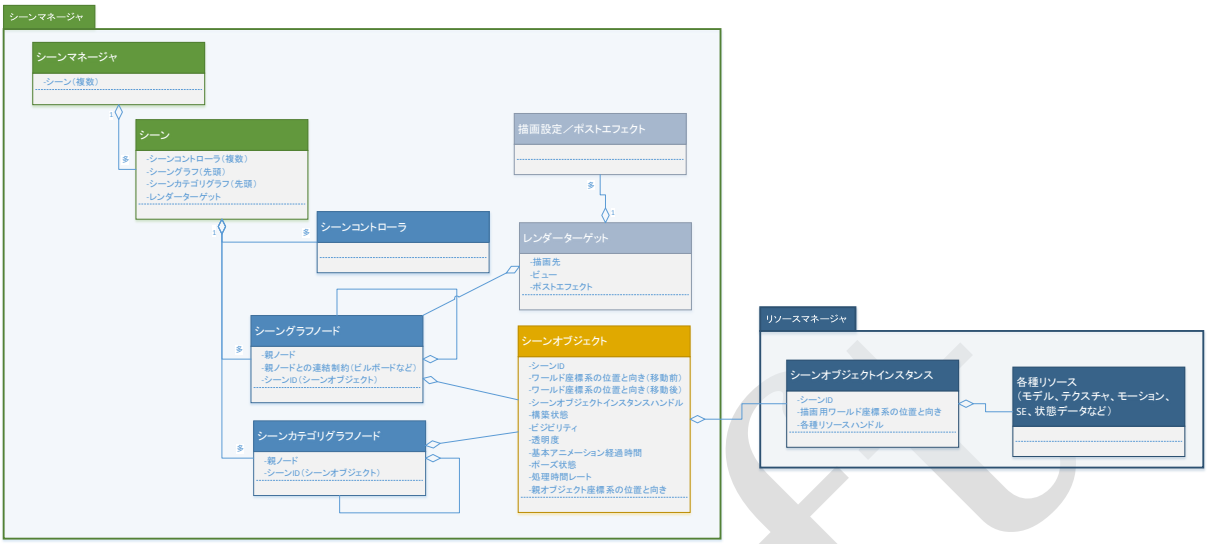


■ データ仕様

▼ データ関連図

シーンマネージャを構成するデータの関連図を示す。

シーンデータ関連図：



■■以上■■

■ 索引

索引項目が見つかりません。

ゲーム全体を円滑に制御するためのシーン管理

以　上