オープンワールドのためのレベル管理

– シーン管理のサブシステムとしてのレベルコントローラ –

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

シーンマネージャ内の一機能として、オープンワールドを実現するためのレベルコントローラを設計する。

■ 目的

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」に基づき、シーン管理システムの中のサブシステム「シーンコントローラ」が担う処理の一つを明確にすることを目的とする。

とりわけ、「オープンワールド」を実現するための「レベルコントローラ」の処理構造を確立する。

■ 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。

- 「レベルコントローラ」は、「シーンコントローラ」の一つとして、ゲームの「レベル」を管理するものとする。
  - 「レベル」の構成要素として扱うものは、主に下記のとおり。
    - 読み替え領域データ
      - マップ（地形データ）の読み替え領域区分、経路探索パスの読み替え領域区分、その他の配置データの読み替え領域区分を扱う。
    - 地形データ
      - モデル、テクスチャ、コリジョン、ライティング、エフェクト、発音体、カメラ、など
    - 経路探索パス
    - トリガーポイント（イベント発生領域、別レベルへの移動領域、ギミック発動領域など）
    - NPC 配置・行動データ
    - 敵配置・行動・戦闘データ
    - その他、宝箱、採取、セーブポイントなどの諸々の配置データ

・ レベルコントローラは、直接上記の構成要素を制御するのではなく、それぞれ専用の「サブコントローラ」を統轄するものとする。
  ➢ 上記の各構成要素は、それぞれ別個のサブコントローラが管理する。

・ レベルコントローラが直接扱うのは、基本的に「読み替え領域データ」のみとし、読み替え領域の区分への進入・退出を各サブコントローラに通知することを主な役割とする。
  ➢ 各サブコントローラを初期化してレベルの初期状態を構築することや、レベルから抜ける時の終了処理も行う。

・ レベルコントローラはタイトル固有の実装として作成するものとする。
  ➢ 「シーンマネージャ」は標準的な汎用システムだが、その中のシーンコントローラは、基本的にタイトル固有の実装になる。

・ レベルコントローラおよびそのサブコントローラは、複数のインスタンスが同時に作られることがある点に配慮して作らなければならない。
  ➢ レベルコントローラは「シーン」の構成要素の一つであり、シーンは複数同時に存在することがある。
  ➢ 例えば、別レベルへの移動ポイント付近に使づくと、移動先のレベルのシーンの準備が始まり、レベルコントローラも作られる。
    - この時、ある程度のデータ構築も行う。
    - 初期出現位置に必要なリソースを構築する一歩手前まで構築を進め、現在のレベルと共有可能なリソースのハンドルだけつかんでおく。（参照カウンタをカウントアップし、レベル移動時に共有リソースが破棄されず維持されるようにする）
  ➢ 「シーン」自体がタイトル固有の実装となる。

▼ 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。

・ 「レベルコントローラ」は、「読み替え領域データ」に基づき、幾つかの範囲の読み替え領域を扱う。
  ➢ 【読み替え範囲：大】
    - 経路探索パス
  ➢ 【読み替え範囲：中】
    - プランナー系の設定データ
      ・ NPC 配置・行動データ

・ 敵配置・行動データ

・ トリガーポイント配置・反応データ

・ 宝箱・採取・セーブポイントなどの配置データ

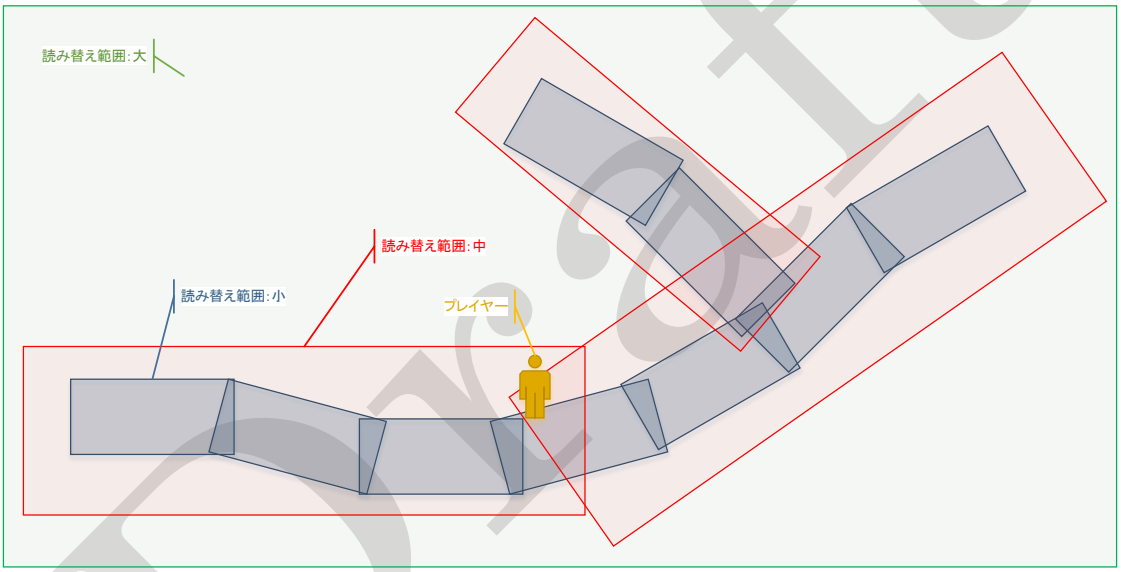
➢ 【読み替え範囲：小】

- デザイン系のデータ

・ マップ（地形データ）：モデル、テクスチャ、アニメーション、コリジョン、オクルーダー、大量配置物、ライティング、配置エフェクト、発音体（点音源、線音源、面音源）、定点カメラ、レールカメラ、など

・ トリガーポイント配置・反応データ（扉稼働やレベル境界の出入り口など、地形関連に限る）

各領域の読み替え範囲のイメージ：



・ 「レベルコントローラ」は、多数の「サブコントローラ」を扱う。

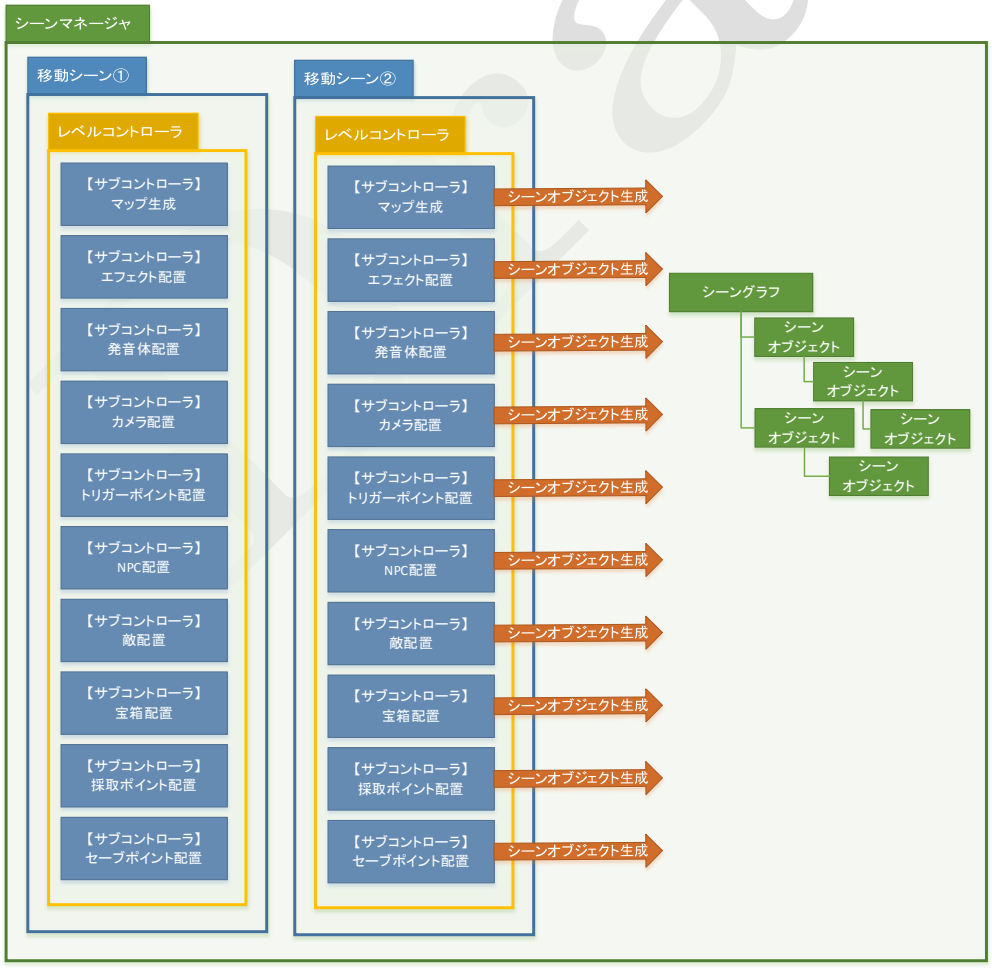
➢ サブコントローラは、完全にレベルコントローラの管理下に置かれ、（処理オブジェクトの）ライフサイクルを共にする。

- 「移動シーン」への切り替え要求が発生すると、「シーンマネージャ」はまず、あらたな「シーン」を生成する。
- 「移動シーン」は、自身が管理するシーンコントローラの一つとして、「レベルコントローラ」を生成する。
- レベルコントローラは、自身が管理する多数の「サブコントローラ」を生成する。
- シーンから直接見えるのはレベルコントローラのみであり、シーンが消滅する際には、レベルコントローラはサブコントローラを適切に終了させなければならない。

➢ レベルコントローラのアップデート処理内で、各サブコントローラのアップデート処理を呼び出し、処理を進める。

- サブコントローラの主な処理は、読み替え領域に対する進入・退出の通知を受けて、その領域用の設定データを読み込み、シーンオブジェクトを生成もしくは削除すること。
  - 各オブジェクト固有の処理は、サブコントローラではなく、シーンオブジェクトの処理で行う。
  - 例えば、トリガーポイントの進入判定とトリガーの実行、NPC の個々の行動（AI）、ボタンに反応する宝箱といった処理は、サブコントローラの処理ではなく、シーンオブジェクトの処理である。
  - ただし、「状況によって出現する NPC を変更する」など、サブコントローラが状況判断して制御するような処理もある。それでも、基本はシーンオブジェクトの生成と削除に関する処理である。
  - ものによっては、シーンオブジェクトの生成もシーンオブジェクトに任せる。例えば「敵の出現」のような処理は、サブコントローラによる処理よりも、サブコントローラは「出現ポイント」のシーンオブジェクトを配置するのみとし、「出現ポイント」がタイミングを見計らって「敵」を出現させる。（「敵」のシーンオブジェクトを生成する）

レベルコントローラとサブコントローラの関係のイメージ：

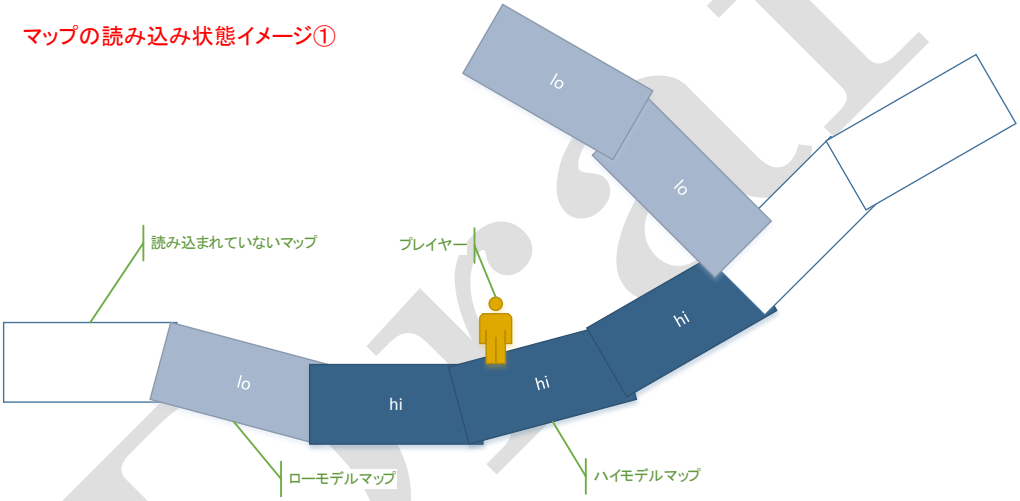


・ レベルコントローラやサブコントローラが扱う各種設定データのインスタンスは、リソースマネージャを通して、リソースとして管理する。
  - 「各種設定データ」とは、「読み替え領域の座標範囲データ」や「NPC の配置設定データ」などのこと。
  - 例えば、プレイヤーが「読み替え領域 A」に入ると、「NPC 配置」サブコントローラは、「読み替え領域 A に進入」の通知を受けて、「領域 A」用の「NPC 配置設定」を、リソースマネージャを通して構築する。
    - 「NPC 配置」サブコントローラは、「領域 A 用の NPC 配置設定ファイルパス」をリソースマネージャに渡し、リソースの構築を要求する。
      - 同時にリソースハンドルを受け取る。
      - ファイルが存在しない場合もハンドルが返るので、ハンドルを通して構築状態を調べた際にエラーだったら「その領域のせっていはなし」という扱いにする。（エラー時もハンドルは明示的に開放する必要がある）
    - リソースマネージャは、要求に応じてファイル読み込みを行い、そのまま、「リソースビルダー」（あらかじめ登録しておくリソース構築用の処理）を使用してリソースを構築する。
    - リソース構築成功後は、「NPC 配置」サブコントローラは、「領域」とそれに対応するリソース（NPC 配置設定データのリソース）のハンドルを保持する。
    - プレイヤーが読み替え領域の境界に位置すると、複数のリソースを同時に扱う場合がある。
    - 「NPC 配置」サブコントローラは、全てのリソースを使用して、NPC 配置処理を行う。
    - 領域から退出したら、リソースのハンドルを開放する。
    - リソースのハンドルが解放されると、リソースマネージャはリソースの共有状態（参照カウンタ）に基づいてリソースの削除を行う。
    - 以上のように、設定データの読み込みと構築も、リソースマネージャを通して簡単かつ安全に管理できる。
  - 「マップ（地形データ）生成サブコントローラ」は、プレイヤーが進入した領域だけに限らず、隣接する領域や遠景として見える領域など、広域のデータを構築する。
    - 領域ごとに、「どのマップ（各領域の地形データ）をいっしょに構築するか」という設定を設ける。
    - このため、隣の領域に移動した際に、すでに構築済みのマップの構築も多く含むことになるが、状態を気にせず、単純に構築を行ってよい。リソースマネージャのリソース共有の仕組みにより、無駄な二重構築は行われず、適切な共有が行われる。
    - 領域の移動に伴って、視界から外れた（構築対象から外れた）リソースの破棄は、単純にリソースハンドルを解放するのみでよい。リソースマネージャが、リソースの参照カウンタに基づいて、適切に削除を実施する。

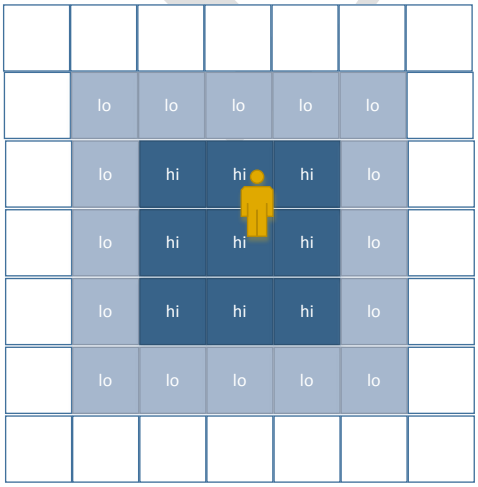
- マップ（地形データ）用のリソースは、少しでも広域を扱えるように、ハイモデルとローモデルを扱う。
  - ハイモデルとローモデルは、リソースとしては完全に別物である。
  - 一つのシーンオブジェクトが、ハイモデルとローモデルのそれぞれのリソースを扱う。
    - 通常は、ハイモデルかローモデルのどちらかのリソースをのみが存在する状態だが、読み替えの境界では、両方のリソースが読み込まれた状態になる。
    - 両方のリソースが読み込まれている場合、ハイモデルを優先で描画する。
- 読み替え範囲は、「進入」（読み込み発生）よりも「退出」（読み捨て発生）の範囲が一回り大きく扱われるようにし、読み替え範囲の境界付近でうろうろしても読み込みと読み捨てが頻繁に繰り返されることが無いようにする。
  - 境界付近でリソース量が増大するので、メモリ状態には注意が必要。

マップのハイモデルとローモデルの読み込み状態のイメージ：

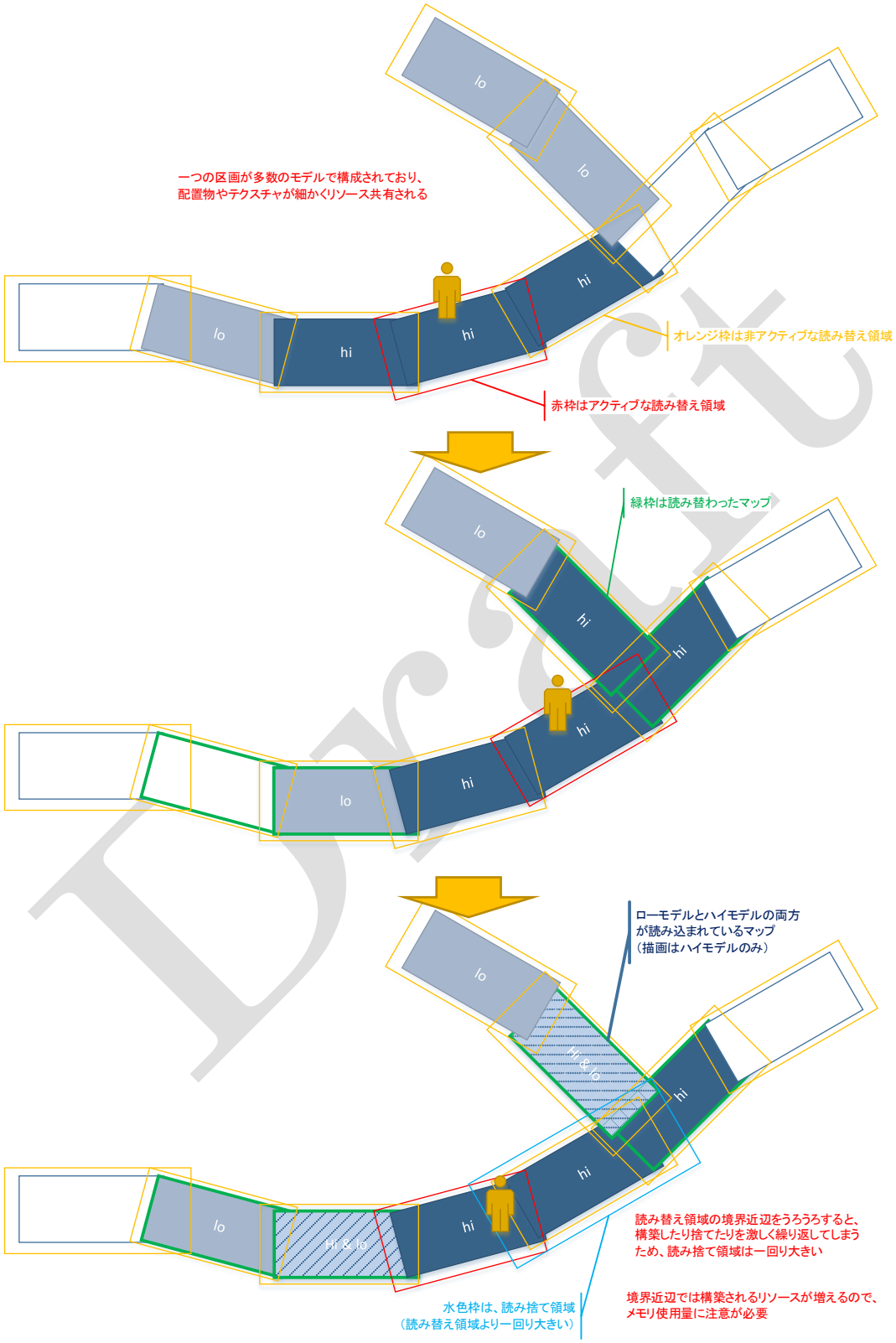
マップの読み込み状態イメージ①



マップの読み込み状態イメージ②



マップの読み替えのイメージ：

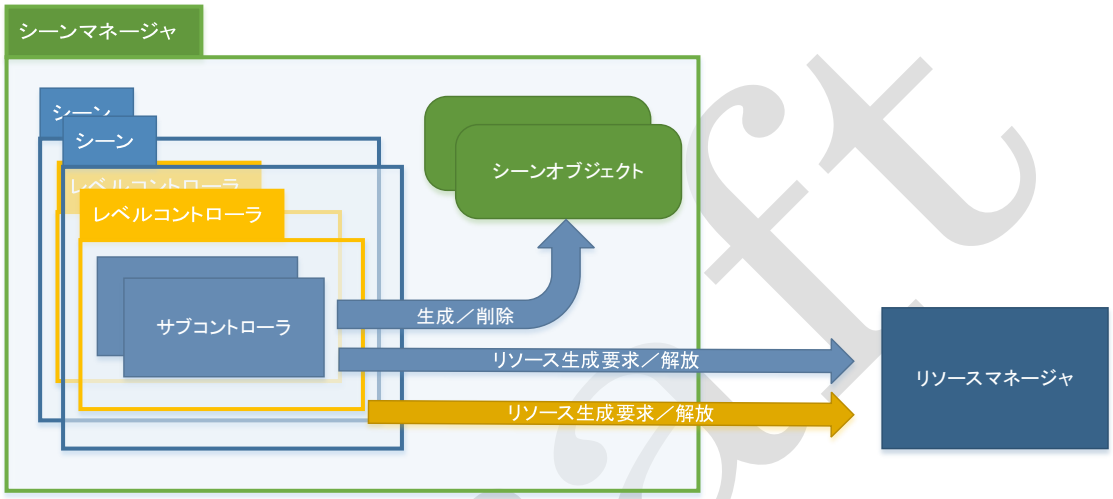


■ 仕様概要

▼ システム構成図

要件に基づくシステム構成図を示す。

レベルコントローラのシステム構成図：



■■以上■■

■ 索引

索引項目が見つかりません。

オープンワールドのためのレベル管理

以　上