カメラシステム

– シンプルな処理を組み合わせて効率化 –

2014 年 1 月 20 日 初版

板垣 衛

■ 改訂履歴

版	リリース	担当	改訂内容
初版	2014 年 1 月 20 日	板垣 衛	（初版）

■ 目次

■ 概略

カメラ処理の無駄を無くすためのごく簡単なプログラム方法論の解説。

■ 目的

本書は、何かと処理に無駄が生じがちなカメラ処理の構造を見直し、シンプルな処理に切り分けることを目的とする。

なお、ここで示すようなプログラムの最適化手法は、カメラ処理に限ったことではない。

■ 対処すべき問題点

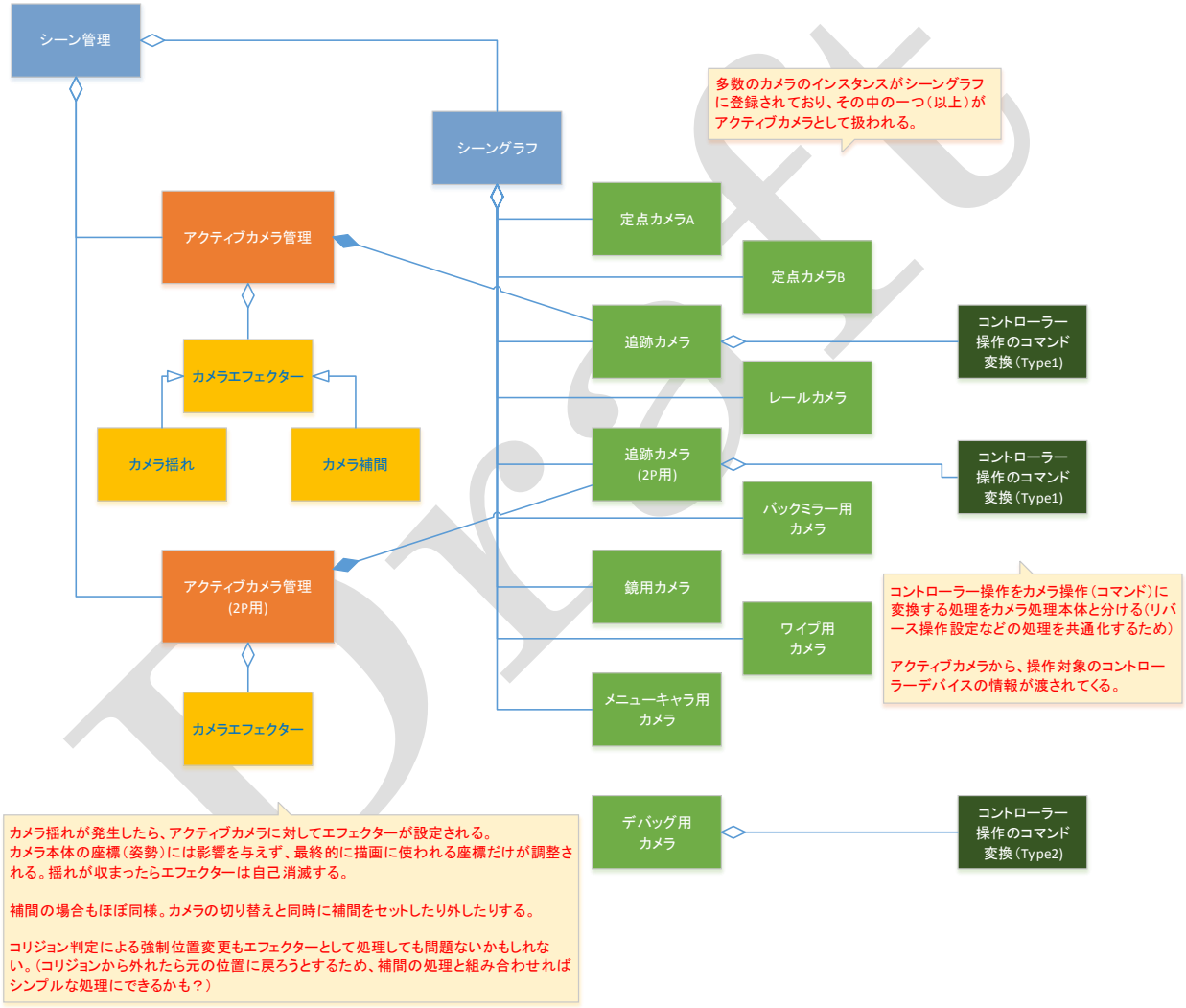
一つのゲームの中では、多数のカメラ処理を切り替えて使用する。プレイヤー追跡型のカメラやレールカメラ、定点カメラ、イベントシーン用カメラなど様々である。問題となるのは、このような処理の対応として、ごり押しのコピペや巨大なカメラクラス、無造作な継承などが行われることである。具体的に説明する。

- 【問題】一つの巨大なカメラ処理（クラス）で構成し、内部で状況に応じて挙動を変えている。
  - 無駄なコピペコードの発生を抑えて共通化することはできるが、コードが複雑になりがち。新しい処理を加える際に、他のカメラに影響を及ぼし易い。
- 【問題】カメラ揺れやコントローラー操作の処理を多数のカメラにコピペして使用している。
  - コピペが抜けていると、一部のカメラの時だけカメラが揺れないということもある。また、バグ修正や挙動の修正が必要になった時、全部を直さなければならず、チェック作業も含めて非効率。
- 【問題】カメラ揺れや補間などの基本処理を親クラスの処理として記述して継承する。
  - カメラ揺れの影響を受けたくないカメラや、補間をしたくないカメラの制御に難があったり、親クラスの処理が座標を書き換えてしまうために処理を作りにくかったりといった事がある。

■ 対策

コントローラー操作、カメラ揺れ、補間といった処理は、なるべくカメラ本体の処理から切り離して扱う。

考え方としては、デザインパターンの Decorator パターンに近いが、クラスの継承ではなく、処理をリレーすることで一時的に処理を拡張する。デザインパターンの Chain of Responsibility パターンとよく似た構造（目的は異なるが）で実現する。



■■以上■■

■ 索引

索引項目が見つかりません。

カメラシステム

以　上