Common C++ Design Pattern Combinations for All Projects

Singleton + Factory Method

Global access point for creating objects in a controlled way.

Example: LoggerFactory::Instance()->CreateLogger("file");

Factory Method + Strategy

Choose algorithmic behavior at runtime.

Example: RendererFactory creates different RenderStrategy based on hardware.

Observer + State

Notify subscribers when an object's internal state changes.

Example: UI elements update when the game state changes (e.g., health bar).

Command + Memento + Invoker

Build undo/redo systems that save and restore state.

Example: A code editor where user actions are commands and state is saved/restored.

Composite + Decorator

Handle hierarchical structures where components can be nested and enhanced.

Example: GUI system or file system with extra behavior like permissions or styles.

Adapter + Bridge

Abstract and adapt multiple implementations to a common interface.

Example: Cross-platform I/O or rendering backend abstraction.

Template Method + Strategy

Enforce a process flow while customizing steps.

Example: A game engine base class defines loop but subclasses implement AI or physics steps.

Facade + Mediator

Simplify complex subsystems and centralize communication.

Example: GameFacade talks to Mediator for input, audio, and rendering subsystems.

Builder + Singleton + Director

Centralized and reusable construction logic.

Example: Configuration system builds settings objects via a global director.

Observer + Command + Strategy

Flexible event-driven systems with modular responses.

Example: Game input system where input triggers actions using control logic strategy.