Gang of four

Stratégie

pros

- Rend le code plus flexible
- Permet d'avoir des objets divers ; à la fois très semblable, mais avec des différences

cons

- Beaucoup de changements dans la structure (the core) du code si l'héritage a été utilisé

Façade

pros

- Cache la complexité du code
- Une classe où tout est réalisé

cons

- Peut rendre le code restrictif

State

pros

- Permet de déterminer dans quelle état le système se trouve actuellement
- Permet de changer d'état facilement
- Permet de d'agir en conséquence d'un certain état

Cons

Prototype

pros

- Facilite la création d'objet
- Clone des objets d'abord, ajoute leurs différences ensuite
- Peut-être utilise pour la classe ''joueur''

Cons

Proxy

pros

- Facilite les testes
- Permet de tester dans l'usage des interfaces d'entré et de sortie

Cons

Les patrons GoF les plus utilitsés

- *Création*
  - *Builder (intéressant, mais je ne vois pas trop l'utilité dans notre cas. Facile la création d'objet divers)*
  - *Factory Method (inutile, nous n'utilisons pas l'abstraction)*
  - *Protoype*
  - *Singleton (mauvais pour un board ou un controleur)*
- *Structure*
  - *Adapter (pas utile dans notre cas)*
  - *Composite (pas utile dans notre cas)*
  - *Decorator (pas utile dans notre cas)*
  - *Facade*
  - *Proxy*
- *Comportement*
  - *Observer (pas utile dans notre cas)*
  - *State*
  - *Strategy*
  - *Template Method (seulement s'il on utilise le polymorphisme)*
  - *Visitor (nécessite Composite)*