Cadres d'application
(Object-Oriented Application Frameworks)

Gerson Sunyé

Université de Nantes

http://sunye.free.fr

Plan

- Introduction.
- Exemple fictif.
- Contre-exemples.
- Exemples réels.
- Conclusion.

Introduction

Introduction

- Définition
- Propriétés de base.
- Un exemple fictif.
- Des exemples réels.
- Contre-exemples.

Définition [Deutch, 1983]

(...) une collection de classes abstraites et leurs algorithmes constituent une sorte de cadre dans lequel certaines applications peuvent insérer leur code spécialisé, en construisant des sous-classes concrètes qui collaborent entre elles.

Le cadre d'applications est constitué des classes abstraites, leurs opérations et l'attente placée sur les sous-classes concrètes.

Définition [Johnson et Foote, 1988]

Un cadre d'applications est une conception abstraite pour des sortes particulières d'applications. Il consiste, habituellement, d'un certain nombre de classes.
Ces classes peuvent appartenir à une bibliothèque, ou peuvent être spécifiques à une application.

Définition [Gamma et al., 1995]

(...)un ensemble de classes coopérantes, qui créent une conception réutilisable pour un domaine logiciel spécifique. Un cadre guide l'architecture en divisant la conception en classes abstraites et en définissant leurs responsabilités et collaborations.

Le développeur personnalise le cadre pour une application particulière en créant des sous-classes et en composant des instances des classes du cadre.

Objectifs

- Réutilisation.
- Simplifier le développement d'applications.
- Réduire, autant que possible, l'écriture de code.
- Permettre aux débutants d'écrire des bons programmes.
- Extraire l'expérience des informaticiens experts d'un domaine.

En résumé

- Un cadre s'adresse à un domaine ou à une famille de produits.
- Il prescrit la décomposition d'un problème.
- C'est la conception d'une application ou d'un sous-système.
- Un ensemble de classes et leurs collaborations:
  - invariants partagés d'objets et leur maintenance.
  - conformité à un modèle de concepts et de collaborations.
- En d'autres termes, un cadre est représenté par son code.

En résumé (suite)

- Une application incomplète, qui peut devenir une vrai applications par:
  - spécialisation — création de nouvelles sous-classes (boîte blanche).
  - configuration d'objets (boîte noire).
  - la modification d'exemples (qui marchent).

Propriétés de base

- Modularité
- Réutilisabilité
- Extensibilité
- Inversion de contrôle

Modularité

- Les classes abstraites implémentent une interface stable, qui encapsule et limite les changements dans des «trous» ou points de «variabilité planifiée».

Réutilisablilité

- Analyse,
- Conception et
- Code

Extensibilité

- En fournissant des trous explicites ou des «crochets» (hooks) pour la variabilité planifiée

Inversion de contrôle

- Le principe de Hollywood:

- « — ne nous appelez pas, nous vous appellerons. »

Hook & Template

- Les briques de base de la variabilité dans le code.

- Une méthode *template* fournit un algorithme générique ou les étapes d'une tâche. Elle appelle une ou plusieurs méthodes *hook* (crochet).

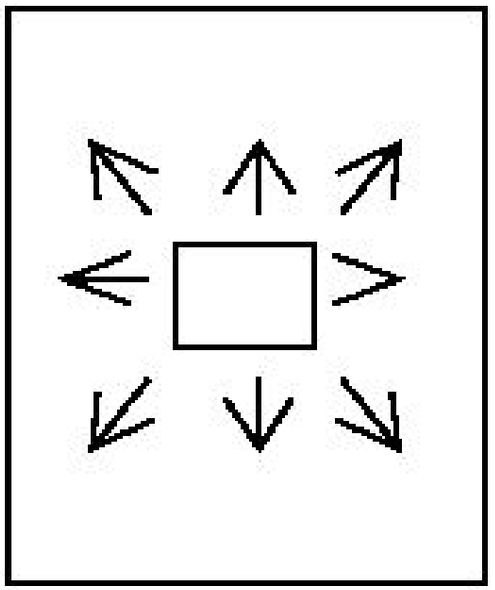
- Chaque méthode crochet représente un point de variabilité, en fournissant une interface d'appel pour les tâches variables.

- Chaque implémentation d'une méthode crochet fournit une variante de cette tâche.

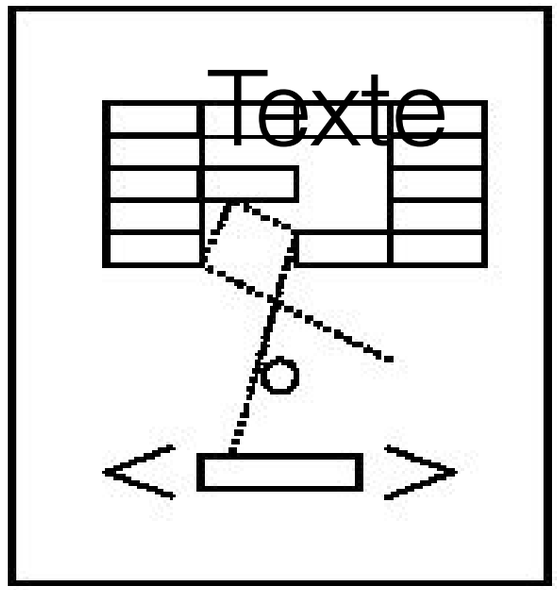
Exemple fictif
[Greg Butler]

Domaine

Les jeux d'objets qui sautent et qui rebondissent:



Bump-Em Car



Brick World

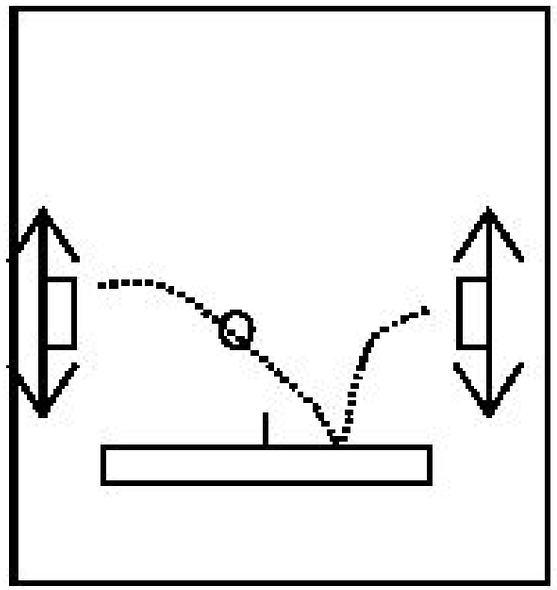
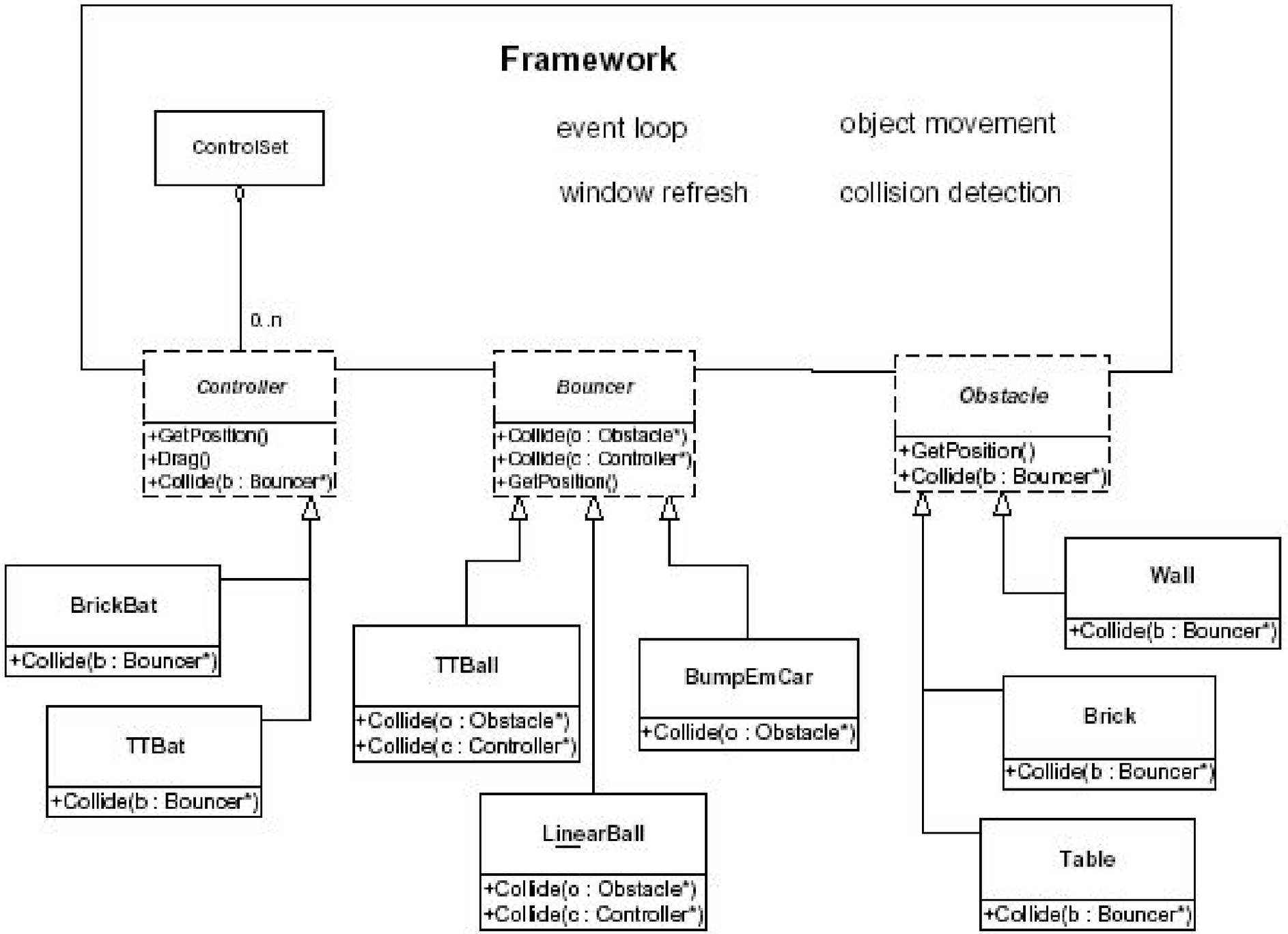


Table Tennis

Diagramme de classes



Méthodes template

```
Game::makeWorld() {
    makeBouncer();
    makeControllers();
    makeObstacles();
    makeEventHandlerTable();
}
Game::run(){
    loop over event e in eventQueue {
        ehTable[e]-> handleEvent(e);
        refreshDisplay(); }
}
```

Contre-exemples

Cadres et Bibliothèques

- Cas d'utilisation d'une bibliothèque:

- Le développeur écrit un programme principal, détermine le flot de contrôle et la décomposition du problème.

- Le code spécifique appelle le code de la bibliothèque.

Cadres et Bibliothèques

- Cas d'utilisation d'un cadre:

- Le développeur écrit une sous-classe. Le cadre d'applications a déjà défini le flot de contrôle et la décomposition du problème.

- Le code du cadre appelle le code spécifique.

Cadres et Patrons

- Les patrons de conception:
  - décrivent des micro-architectures.
  - sont abstraits.
- Les cadres d'application:
  - ont une architecture concrète.
  - peuvent incorporer des design patterns dans le niveau des micro-architectures.

Cadres et Patrons

La flexibilité de la spécialisation d'un cadre est souvent fournie par un patron de conception.

Cadres et Architectures

- Les architectures:
  - sont des éléments de conception.
  - peuvent être utilisées pour une seule application, ou pour une ligne de produits.
  - sont conçues pour respecter certaines mesures de qualité.

Cadres et Architectures

- Les cadres sont des implémentations concrètes d'architectures semi-complètes.

- Ils sont conçus pour être:

  - réutilisables.

  - spécialisés.

Exemples réels

Quelques exemples

- MVC (Modèle Vue Contrôleur) — Smalltalk [1980].
- MacApp/ACS — Objective Pascal, C++ [1986]
- NeXTStep/OpenStep/Cocoa/GNUStep — Objective C, Java.
- OpenClass (Taligent/IBM). — VisualAge C++.
- MFC (Microsoft), OWL (Borland), Java Swing.
- HotDraw(Smalltalk, Java), ET++(C++) - Editeurs graphiques.
- MET++ (Phillipe Ackermann et al). Multimedia

Quelques exemples (suite)

- SAP (Baan) — Java.
- Zope (Zope Corporation) — Python.
- Apache Struts — Java.
- J2EE — Java
- San Francisco Business Objects (Taligent/IBM).
- Adaptive Communication Environment (Doug Schmidt et al).
- Systèmes de guerre navale (CelsiusTech/SaabTech Vectronics).

Conclusion

Caractéristiques d'un cadre

- Trous (*Hotspots*): les points d'extension pré-définis.

- Contrôle inversé: le cadre contrôle l'application et non le contraire.

Problèmes majeurs

- Courbe d'apprentissage.
- Investissement important
- Expertise du domaine requise par les développeurs de cadre.
- Evolution du cadre.
- Abstraction.

Bénéfices

- Réutilisation de la conception.

- Changement de perspective: développement pour la réutilisation. Le programmeur est contraint à écrire des logiciels réutilisables.

- Amélioration de la qualité des applications et de la productivité.

Cadres d’application

Object-Oriented Application Framework