

# Visitor Pattern

Dans l'implémentation des interactions d'un jeu « à la pac-man »

---

Loïc Herman, Samuel Roland, Massimo Stefani, et Timothée Van Hove

12 juin 2024

HEIG-VD — MCR

2024-06-12

## Visitor Pattern

### Visitor Pattern

Dans l'implémentation des interactions d'un jeu « à la pac-man »

---

Loïc Herman, Samuel Roland, Massimo Stefani, et Timothée Van Hove  
12 juin 2024  
HEIG-VD — MCR

- 1. Introduction
- 2. Rappel du pattern
  - 2.1 Principes d'implémentation
  - 2.2 Séquence d'interactions
- 3. Implémentation
  - 3.1 Choix architecturaux
  - 3.2 Quelques interactions en détail
- 4. Conclusion

Sommaire

- 1. Introduction
- 2. Rappel du pattern
  - 2.1 Principes d'implémentation
  - 2.2 Séquence d'interactions
- 3. Implémentation
  - 3.1 Choix architecturaux
  - 3.2 Quelques interactions en détail
- 4. Conclusion

# Introduction

## Éléments visitables :

- Pellets et SuperPellets
- Fantômes (5 spécialisations)
- Pac-man

## Éléments visiteurs :

- Fantômes
- Pac-man

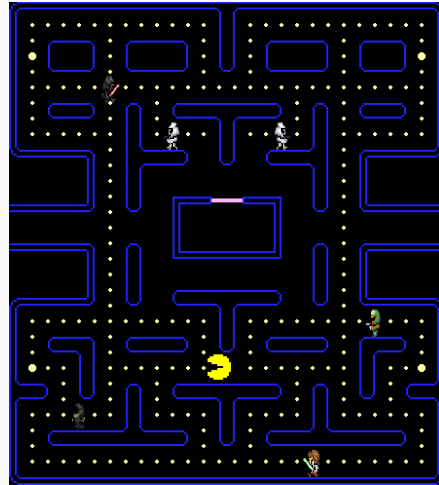


Figure 1 – Exemple de situation du jeu

2024-06-12

## Visitor Pattern └ Introduction

### └ En guise d'introduction

Présenter les éléments visitables en gros (on verra plus loin les sprites). Ils réagissent aux interactions.

Présenter les éléments visiteurs, c'est ceux qui déclenchent les interactions.

Chaque élément visiteur aura donc une ou plusieurs classes de visiteurs associées.

En guise d'introduction

Éléments visitables :

- Pellets et SuperPellets
- Fantômes (5 spécialisations)
- Pac-man

Éléments visiteurs :

- Fantômes
- Pac-man




Figure 1 – Exemple de situation du jeu

Player en mode normal :

- Player → Super Pellet
- Player → Boba Fett

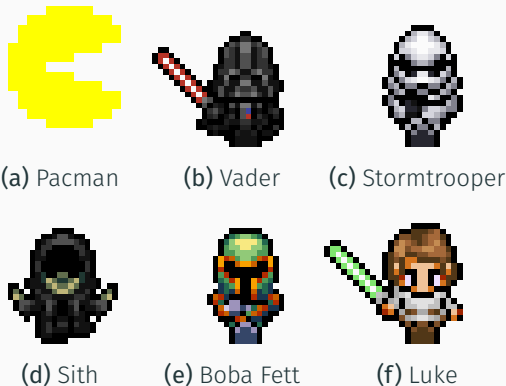


Figure 2 – Sprites de Pacman

2024-06-12

Visitor Pattern  
└ Introduction

└ Interactions dans l'implémentation

**Player → SuperPellet**

Mange et passage en mode invincible

**Player → BobaFett**

Applique un boost de vitesse à boba fett



Player en mode invincible :

- Player → Super Pellet
- Player → Boba Fett
- Player → Luke
- Player → Sith
- Player → Storm Trooper
- Player → Vader

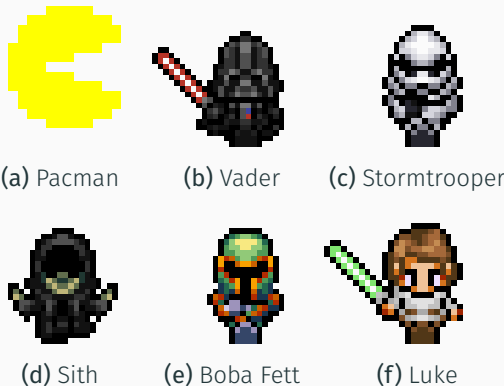


Figure 2 – Sprites de Pacman

2024-06-12

Visitor Pattern  
└ Introduction

└ Interactions dans l'implémentation

**Player → SuperPellet**

Prolonge le timer de mode invincible

**Player → Boba Fett**

Prend la vitesse de boba fett et le renvoie à la case départ

**Player → Luke**

Player essaie de manger Luke, ce qui tue le Player

**Player → Sith**

Le renvoie à la maison

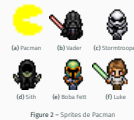
**Player → Storm Trooper**

Le retire du jeu

**Player → Vader**

Renvoi à la maison. Change la couleur du pacman en rouge

- Player en mode invincible :
- Player → Super Pellet
  - Player → Boba Fett
  - Player → Luke
  - Player → Sith
  - Player → Storm Trooper
  - Player → Vader



Côté fantômes :

- Boba Fett → Super Pellet
- Sith → Storm Trooper
- Vader → Luke
- Fantômes → Player



(a) Vader → Luke



(b) Boba Fett → Super Pellet

Figure 2 – Dialogues entre fantômes

2024-06-12

Visitor Pattern  
└ Introduction

└ Interactions dans l'implémentation

**Boba Fett → SuperPellet**

Une pensée apparait en la bouffant.

**Sith → Storm Trooper**

Fait se téléporter le storm trooper à une position aléatoire

**Vader → Luke**

Dialogue entre les deux

**Fantômes → Player**

Si le joueur est en mode normal, il sera tué

Côté fantômes :

- Boba Fett → Super Pellet
- Sith → Storm Trooper
- Vader → Luke
- Fantômes → Player



(a) Vader → Luke



(b) Boba Fett → Super Pellet

Figure 2 – Dialogues entre fantômes

## Rappel du pattern

---

2024-06-12

Visitor Pattern  
└─ Rappel du pattern

Rappel du pattern

---



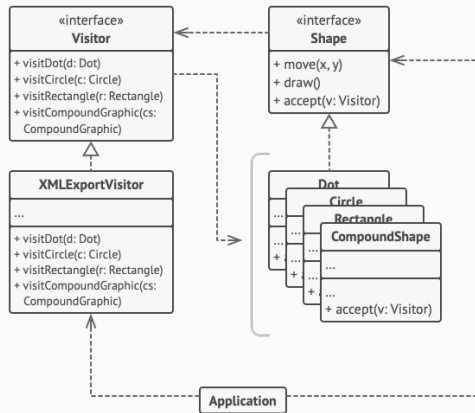


Figure 3 – Modélisation d'un système d'export en plusieurs formats.

Source : refactoring.guru

2024-06-12

- Visitor Pattern
  - Rappel du pattern
  - Principes d'implémentation
  - Implémentation classique

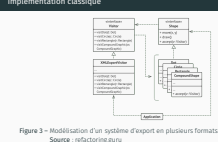


Figure 3 – Modélisation d'un système d'export en plusieurs formats.  
Source : refactoring.guru

Il faut distinguer d'un côté la partie « métier », le **modèle** de données. De l'autre côté, il y a la partie **comportement** avec la définition de comment traiter les données. Un code appelant devra donc créer le visiteur nécessaire et appeler la méthode **accept** de la forme en cours de traitement. Le jour où notre chef se dira que le XML c'est déprécié on pourra rajouter un visiteur JSON qui fera le travail de conversion correspondant.

# Séquence d'interactions

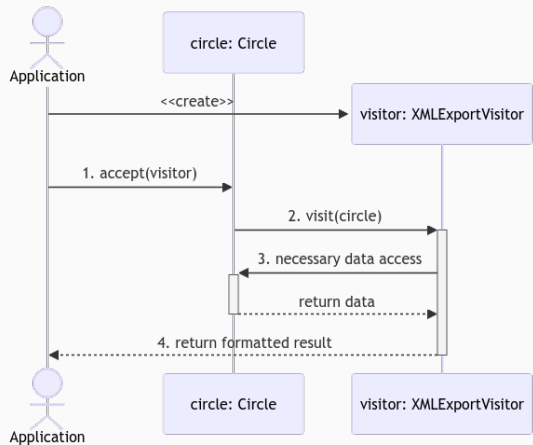
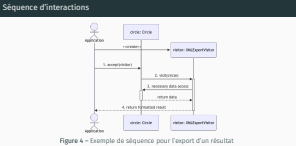


Figure 4 – Exemple de séquence pour l'export d'un résultat

2024-06-12

- Visitor Pattern
  - Rappel du pattern
    - Séquence d'interactions
      - Séquence d'interactions



## Implémentation

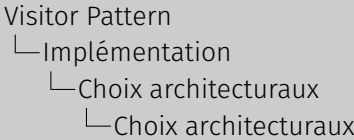
En termes de pattern :

- Quand une entité veut interagir avec une autre, elle envoie à sa méthode ***accept*** son propre visiteur,
- L'entité cible appellera ensuite la bonne méthode ***visit*** du visiteur envoyé (double-dispatch).

En termes de comportement :

- Comme les collisions sont bidirectionnelles, c'est toujours l'entité « attaquante » qui définit l'interaction.
- Le visiteur envoyé lors d'une action par le joueur changera dynamiquement en fonction de son état.

2024-06-12



Choix architecturaux

En termes de pattern :

- Quand une entité veut interagir avec une autre, elle envoie à sa méthode **accept** son propre visiteur.
- L'entité cible appellera ensuite la bonne méthode **visit** du visiteur envoyé (double-dispatch).

En termes de comportement :

- Comme les collisions sont bidirectionnelles, c'est toujours l'entité « attaquante » qui définit l'interaction.
- Le visiteur envoyé lors d'une action par le joueur changera dynamiquement en fonction de son état.

2024-06-12

# Visitor Pattern

- Implémentation
- Choix architecturaux



Figure 5 – UML simplifié du jeu

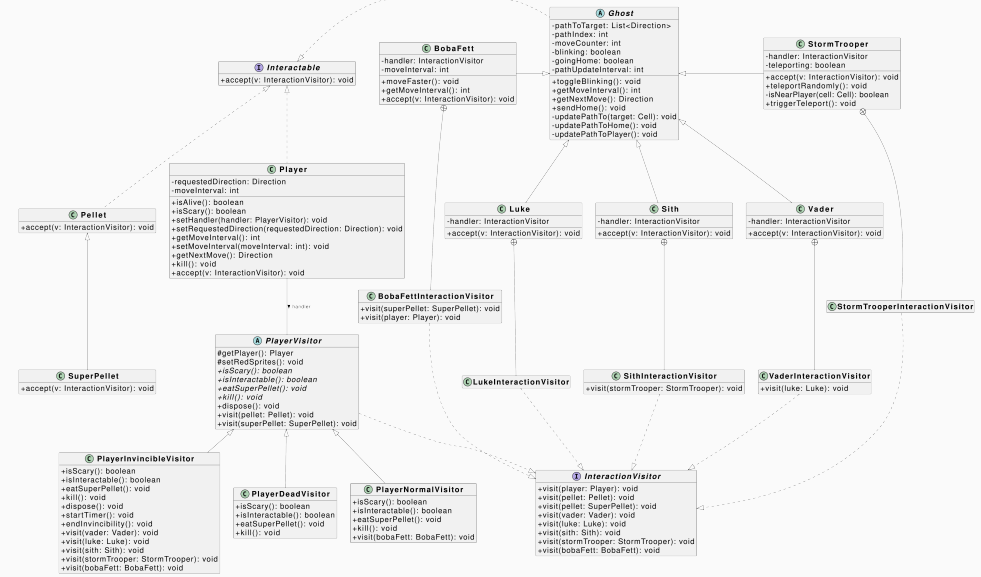


Figure 5 – UML simplifié du jeu

# Player → SuperPellet

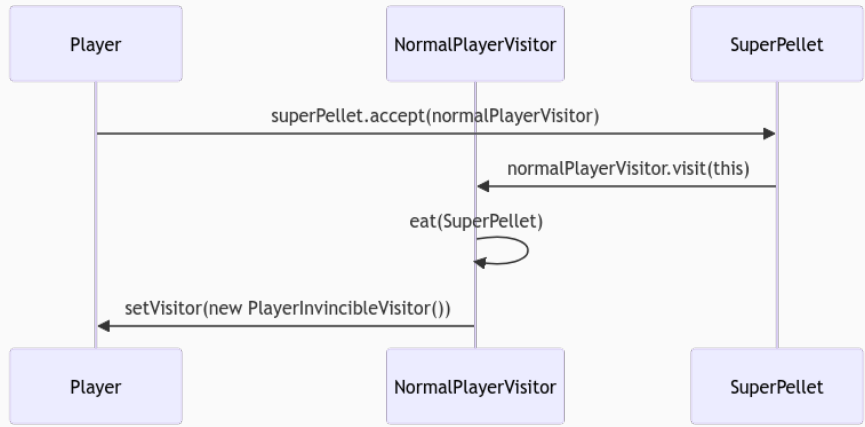


Figure 6 – Séquence d'interaction entre pac-man et les super pellets

2024-06-12

## Visitor Pattern

- Implémentation
  - Quelques interactions en détail
    - Player → SuperPellet

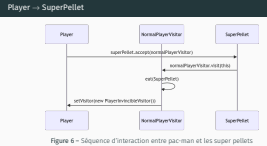


Figure 6 – Séquence d'interaction entre pac-man et les super pellets

- Choix de l'Ustensile (Visitor)** : Le joueur choisit un ustensile (normalVisitor) pour interagir avec la superPellet.
- Utilisation de l'Ustensile** : La SupetPellet utilise cet ustensile (normalVisitor) pour s'envoyer vers l'assiette où il sera mangé.
- Transformation de l'Ustensile** : En mangeant la superPellet, l'ustensile (normalVisitor) se transforme en un ustensile plus puissant (SuperVisitor).
- Effet du Nouvel Ustensile** : Le nouvel ustensile (SuperVisitor) offre au joueur de nouvelles capacités pour interagir avec les autres éléments du jeu. Par la suite, le SuperVisitor c'est l'utensile qui sera utiliser pour manger les autres elements.

“Toutefois, Pac-Man, dans sa quête insatiable, se précipite vers Luke pour l’engloutir. Cependant, il se heurte à une force inattendue : la défense de Luke, alimentée par la Force, est impénétrable. Chaque tentative de Pac-Man d’avaler Luke est vaine, même si ce-dernier possède un SuperVisitor”

2024-06-12

- Visitor Pattern
  - └ Implémentation
    - └ Quelques interactions en détail
      - └ Luke ↔ Pac-man

“Toutefois, Pac-Man, dans sa quête insatiable, se précipite vers Luke pour l’engloutir. Cependant, il se heurte à une force inattendue : la défense de Luke, alimentée par la Force, est impénétrable. Chaque tentative de Pac-Man d’avaler Luke est vaine, même si ce-dernier possède un SuperVisitor”

# Démonstration

2024-06-12

- Visitor Pattern
  - Implémentation
    - Quelques interactions en détail

Démonstration



## Conclusion

---

- **Avantages** : Réutilisation du code commun des interactions, et ajout simple de comportements très divers
- **Désavantages** : Implémentation d'une classe abstraite pour éviter de redéfinir la méthode Visit dans tous les visiteurs concrets
- **Ajout d'autres patterns** : State pour gérer l'état du joueur, Builder pour construire la Map, Observer pour gérer win/lose

2024-06-12

Visitor Pattern  
└─ Conclusion

└─ Conclusion

- **Avantages du pattern dans notre jeu** : Réutilisation du code commun des interactions, et ajout simple de comportements très divers
- **Désavantages du pattern** : Implémentation d'une classe abstraite pour éviter de redéfinir la méthode Visit dans tous les visiteurs concrets
- **Ajout d'autres patterns** : State pour gérer l'état du joueur, Builder pour construire la Map, Observer pour Gérer win/lose

Conclusion

- **Avantages** : Réutilisation du code commun des interactions, et ajout simple de comportements très divers
- **Désavantages** : Implémentation d'une classe abstraite pour éviter de redéfinir la méthode Visit dans tous les visiteurs concrets
- **Ajout d'autres patterns** : State pour gérer l'état du joueur, Builder pour construire la Map, Observer pour gérer win/lose

Questions?

2024-06-12

Visitor Pattern  
└ Conclusion

Questions?