# Représentation du jeu Pacman avec NumPy (Mise à jour 3)

Ce document présente la façon d’utiliser un tableau NumPy à canaux multiples pour représenter l’état du jeu Pacman afin de l’utiliser comme entrée dans un réseau de neurones (par exemple un DQN). Cette version inclut la gestion du retour au spawn des fantômes (returnToSpawn) et ne fait plus usage d’un canal global pour le mode frightened.

## 1. Pourquoi utiliser NumPy pour coder l’état ?

NumPy permet de manipuler efficacement des tableaux multidimensionnels en Python. Dans le cadre d’un Deep Q-Network, il est courant de représenter l’état du jeu sous la forme d’un tensor à plusieurs canaux, chaque canal codant une information spécifique (murs, pac-gommes, bonus, positions des entités, téléporteurs, portails, etc.).  
  
Les avantages principaux sont :  
- Rapidité de calcul grâce à des opérations vectorisées.  
- Structure fixe et connue (dimensions du tableau).  
- Facilité d’intégration avec les bibliothèques de deep learning (PyTorch, TensorFlow) qui acceptent directement des tensors.  
- Clarification et séparation des différentes informations du jeu dans des canaux dédiés.

## 2. Structure du tableau NumPy à canaux multiples

On construit un tableau de forme `(n\_canaux, hauteur, largeur)`.  
Pour Pacman, on définit désormais les canaux suivants :  
- \*\*Canal 0 : Murs\*\* (masque binaire indiquant la présence d’un mur à chaque position '#').  
- \*\*Canal 1 : Pac-gommes\*\* (masque binaire indiquant la présence de petites pastilles '.').  
- \*\*Canal 2 : Bonus\*\* (masque binaire pour les gros points 'o').  
- \*\*Canal 3 : Téléporteurs\*\* (masque binaire pour les caractères 'T').  
- \*\*Canal 4 : Portails de fantômes\*\* (masque binaire pour les caractères '-').  
- \*\*Canal 5 : Position de Pacman\*\* (1 uniquement à la case où se trouve Pacman).  
- \*\*Canal 6 : Fantômes normaux\*\* (1 aux cases occupées par des fantômes non effrayés).  
- \*\*Canal 7 : Fantômes effrayés\*\* (1 aux cases occupées par des fantômes en mode ‘frightened’).  
  
Ainsi, si la grille fait `Hauteur × Largeur` (ex. : 31 × 29), alors le tableau aura la forme `(8, 31, 29)`.

## 3. Détail de chaque canal

Voici en détail ce que représente chaque canal :  
- \*\*Canal 0 : Murs '#'.\*\*  
 - À la position `(y, x)`, si le caractère est `#`, on met `1.0`, sinon `0.0`.  
  
- \*\*Canal 1 : Pac-gommes '.'.\*\*  
 - Si le caractère est `.` (petite pastille), on met `1.0` à `(y, x)`, sinon `0.0`.  
  
- \*\*Canal 2 : Bonus 'o'.\*\*  
 - Si le caractère est `o`, on met `1.0` à `(y, x)`, sinon `0.0`.  
  
- \*\*Canal 3 : Téléporteurs 'T'.\*\*  
 - Si le caractère est `T`, on met `1.0` à `(y, x)`, sinon `0.0`.  
  
- \*\*Canal 4 : Portails de fantômes '-'.\*\*  
 - Si le caractère est `-`, on met `1.0` à `(y, x)`, sinon `0.0`.  
  
- \*\*Canal 5 : Position de Pacman.\*\*  
 - À la position en cases `{ 'x': px, 'y': py }`, on met `1.0` à `(py, px)`.  
  
- \*\*Canal 6 : Fantômes normaux.\*\*  
 - Pour chaque fantôme, si `returnToSpawn` est `False` et `isFrightened` est `False`, on met `1.0` à `(gy, gx)`.  
  
- \*\*Canal 7 : Fantômes effrayés.\*\*  
 - Pour chaque fantôme, si `returnToSpawn` est `False` et `isFrightened` est `True`, on met `1.0` à `(gy, gx)`.

## 4. Exemple de code Python pour l’encodage

Le code suivant montre la fonction `encode\_state` mise à jour qui prend un dictionnaire `parsed\_state` et renvoie un tableau NumPy de forme `(8, hauteur, largeur)` :

import numpy as np  
  
def encode\_state(parsed\_state):  
 """  
 Encode l’état du jeu Pacman dans un tableau NumPy à 8 canaux :  
 0: Murs (#)  
 1: Pac-gommes (.)  
 2: Bonus (o)  
 3: Téléporteurs (T)  
 4: Portails de fantômes (-)  
 5: Pacman  
 6: Fantômes normaux  
 7: Fantômes effrayés  
 """  
  
 grid\_lines = parsed\_state['map']  
 height = len(grid\_lines)  
 width = len(grid\_lines[0]) if height > 0 else 0  
  
 state\_array = np.zeros((8, height, width), dtype=np.float32)  
  
 # Remplir canaux 0 à 4 selon les caractères de la grille  
 for y, line in enumerate(grid\_lines):  
 for x, char in enumerate(line):  
 if char == '#':  
 state\_array[0, y, x] = 1.0  
 elif char == '.':  
 state\_array[1, y, x] = 1.0  
 elif char == 'o':  
 state\_array[2, y, x] = 1.0  
 elif char == 'T':  
 state\_array[3, y, x] = 1.0  
 elif char == '-':  
 state\_array[4, y, x] = 1.0  
  
 # Position de Pacman  
 px, py = parsed\_state['pacman']['positionGrid']['x'], parsed\_state['pacman']['positionGrid']['y']  
 if 0 <= py < height and 0 <= px < width:  
 state\_array[5, py, px] = 1.0  
  
 # Positions des fantômes (normaux vs effrayés)  
 for ghost in parsed\_state['ghosts']:  
 gx, gy = ghost['positionGrid']['x'], ghost['positionGrid']['y']  
 is\_frightened = ghost.get('isFrightened', False)  
 return\_to\_spawn = ghost.get('returnToSpawn', False)  
  
 # Ne pas encoder les fantômes en cours de retour au spawn  
 if return\_to\_spawn:  
 continue  
  
 if 0 <= gy < height and 0 <= gx < width:  
 if is\_frightened is False:  
 state\_array[6, gy, gx] = 1.0  
 else:  
 state\_array[7, gy, gx] = 1.0  
  
 return state\_array

## 5. Avantages détaillés de cette représentation

- \*\*Séparation claire des informations\*\* : chaque canal ne contient qu’un seul type d’information, ce qui facilite la tâche du réseau pour extraire des motifs.  
- \*\*Gestion explicite du retour au spawn des fantômes\*\* : les fantômes en `returnToSpawn` ne sont pas encodés, ce qui reflète correctement leur absence de la grille.  
- \*\*Dimension fixe et compacte\*\* : le tableau a toujours la même forme `(8, H, W)`, quel que soit l’état du jeu, ce qui simplifie la gestion des batches.  
- \*\*Compatibilité directe avec les CNN\*\* : les convolutions peuvent apprendre des filtres spatiaux sur plusieurs canaux simultanément (murs, pac-gommes, téléporteurs, portails, etc.).  
- \*\*Légèreté\*\* : uniquement des valeurs 0/1, pas de données inutiles comme des couleurs.  
- \*\*Extensible\*\* : si de nouveaux éléments de jeu apparaissent, il suffit d’ajouter un canal.  
- \*\*Facilité de debugging\*\* : on peut visualiser chaque canal individuellement pour vérifier la présence de chaque élément.

## 6. Conclusion

Cette mise à jour de la représentation de l’état du jeu Pacman par un tableau NumPy à 8 canaux inclut la gestion du retour au spawn des fantômes (returnToSpawn) sans avoir de canal global pour le mode frightened. Cette approche reste simple, efficace et évolutive, et fournit au réseau DQN toutes les informations nécessaires de manière structurée.