

# **Génération de labyrinthes PACMAN**

## TER S5, M1 Informatique

**Lesage Arno, Razafindrabe Keryann, Viale  
Jean-Jacques**

EUR DS4H - Université Côte d'Azur

 [github.com/KeryannR/TER\\_S1\\_F](https://github.com/KeryannR/TER_S1_F)

# Sommaire

1. API

2. Tests

3. Évaluation et métriques

4. PACMAN

5. GitHub et collaboration

## **1. API**

# Endpoint /

**Description :** Vérifie que l'API est en fonctionnement.

**Méthode :** GET

**Paramètres :** Aucun

**Retour :** JSON avec un message de confirmation

**Exemple :**

GET /

{"message": "Maze Generator API is running!"}

# Endpoint /generate

**Description :** Génère dynamiquement un labyrinthe PACMAN selon les paramètres fournis.

**Méthode :** GET

**Paramètres :**

- xSize, ySize : dimensions du labyrinthe (default 15)
- minScore : score minimal pour le labyrinthe généré
- nPortal : nombre de portails (default 1)
- seed : graine pour reproductibilité
- nStep : nombre d'itérations (default 20000)
- maxBorderSpikeSize : taille max des prolongements de bord
- includeTile : tuiles à inclure

# Format du résultat de /generate

Exemple d'utilisation de la route :

```
GET /generate?xSize=30&ySize=30&nStep=20000
```

Le résultat renvoyé par l'API est un JSON structuré :

```
{
  "_id": "None",
  "grid": [[0,0,1,...],[...],...],
  "legend": {"0":"path","1":"wall","2":"phantom","3":"portal"},
  "metrics": {"Crossroads%":10.26, "Dead-Ends%":1.28, ...},
  "options": {"xSize":30,"ySize":30,"nStep":2000,...},
  "score": 2.65
}
```

# Insertion dans MongoDB

Lorsqu'un labyrinthe est inséré dans MongoDB, la base lui attribue automatiquement un `_id` unique.

Cet `_id` est récupéré avec le code suivant pour ensuite être ajouté au JSON renvoyé par l'API :

```
inserted_id = collection.insert_one(json_data).inserted_id
json_data["_id"] = str(inserted_id)
```

**Mazes.maze**

STORAGE SIZE: 104KB LOGICAL DATA SIZE: 168.47KB TOTAL DOCUMENTS: 90 INDEXES TOTAL SIZE: 36KB

[Find](#) [Indexes](#) [Schema Anti-Patterns](#) [Aggregation](#) [Search Indexes](#)

**Filter** Type a query: { field: 'value' }

```
_id: ObjectId('690e1094d51924d230f8ad4c')
grid: Array (11)
score: 3.203888960638837
adjustedScore: 3
metrics: Object
legend: Object
options: Object
```

# Endpoint /get

**Description :** Récupère un ou plusieurs labyrinthes depuis la base selon des critères.

**Méthode :** GET

**Paramètres :** id, score, xSize, ySize, seed, nStep, nPortal, maxBorderSpikeSize, includeTile, limit

**Retour :** JSON avec un ou plusieurs labyrinthes correspondant aux filtres.

**Exemple :**

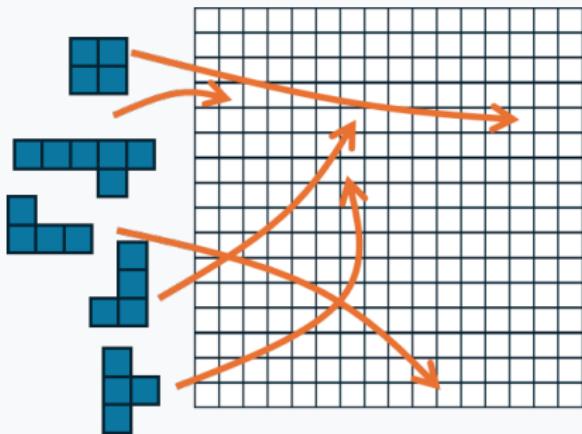
```
GET /get?xSize=30&ySize=30&limit=5
=> [{ maze1 }, { maze2 }, { maze3 }, { maze4 }, { maze5 }]
```

## **2. TESTS**

### **3. ÉVALUATION ET MÉTRIQUES**

# Génération du labyrinthe et métriques

Rappel



**Métriques calculés:**

- Proportion de **carrefour**
- Proportion de **jonction**
- Proportion de **virage**
- Proportion de **ligne droite**
- Proportion de **cul-de-sac**
- Proportion de **murs**
- Proportion de **chemin**

**Figure:** Algorithme Tetris Modifié

**Question :** Comment savoir si un labyrinthe est bon ?

1. À la main ? → trop contraignant...
2. Automatiquement ? → oui, mais comment ?

# Mesure d'évaluation

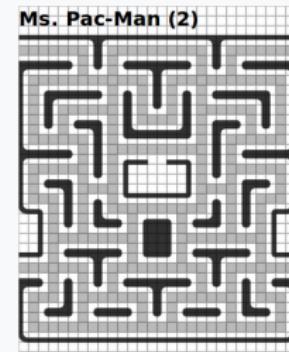
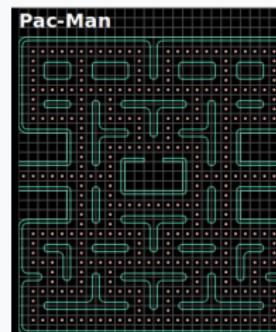
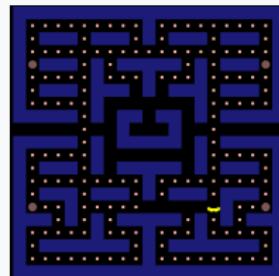


Figure: Quelques "vrai" PACMAN

En moyenne :

- **Carrefours** : 2.034% (↑)
- **Jonctions** : 11.863% (↑)
- **Virages** : 12.213%
- **Lignes droites** : 72.923% (↓)
- **Cul de sac** : 0.967%
- **Murs** : 50.766% (↓)
- **Chemins** : 46.647% (↑)

# Mesure d'évaluation

Formule

Soit  $\mu$  le vecteur de la moyenne des mesures sur les trois labyrinthes et  $x$  les mesures sur le labyrinthe cible.

$$\begin{aligned}\mu &= (\mu_{\text{cross}}, \mu_{\text{junc}}, \mu_{\text{turn}}, \mu_{\text{straight}}, \mu_{\text{dead}}, \mu_{\text{wall}}, \mu_{\text{path}}) \\ x &= (x_{\text{cross}}, x_{\text{junc}}, x_{\text{turn}}, x_{\text{straight}}, x_{\text{dead}}, x_{\text{wall}}, x_{\text{path}})\end{aligned}$$

Nous définissons **score** ( $\mu, x$ ) comme :

$$\text{score}(\mu, x) = 5 \times \frac{x \cdot \mu}{\|x\| \|\mu\|}$$

$$\mu_{\text{wall}} > 0.8 \Rightarrow \text{score}(\mu, x) = 0$$

$$\mu_{\text{path}} > 0.8 \Rightarrow \text{score}(\mu, x) = 0$$

$$\mu_{\text{cross}} > 0.15 \Rightarrow \text{score}(\mu, x) = 0$$

## Interprétation :

Médiocre : **score** ( $\mu, x$ )  $\in [0, 2[$

Très mauvais : **score** ( $\mu, x$ )  $\in [2, 2.5[$

Mauvais : **score** ( $\mu, x$ )  $\in [2.5, 3[$

Moyen : **score** ( $\mu, x$ )  $\in [3, 3.5[$

Bon : **score** ( $\mu, x$ )  $\in [3.5, 4[$

Très bon : **score** ( $\mu, x$ )  $\geqslant 4$

1. API



2. TESTS



3. ÉVALUATION ET MÉTRIQUES



4. PACMAN



5. GITHUB ET COLLABORATION

# Mesure d'évaluation

Images

## **4. PACMAN**

## **5. GITHUB ET COLLABORATION**

**MERCI DE VOTRE ATTENTION !**