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

Sommaire

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 :

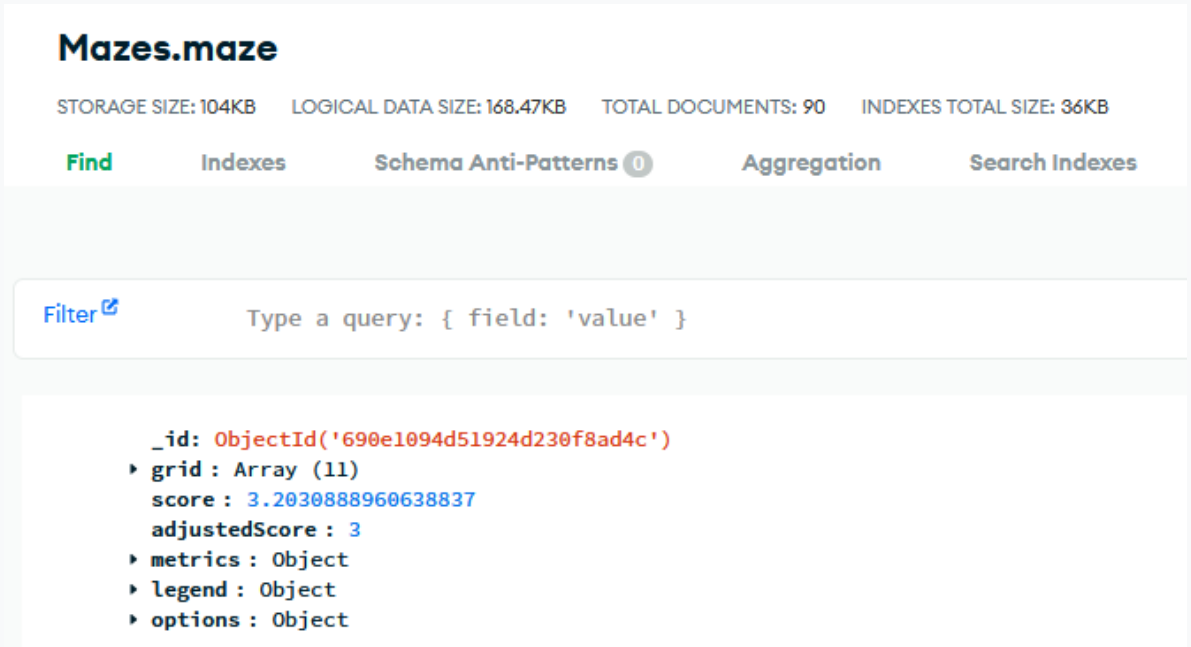

```
{
    "_id": "None",
    "grid": [[0,0,1,...],[...],...],
    "legend": {"0":"path","1":"wall","2":"phantom","3":"portal"},
    "metrics": {"Crossroads%":10.26, "Dead-Ends%":1.28, ...},
    "options": {"xSize":30,"ySize":30,"nStep":2000,...},
    "score": 2.65
}
```

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)
```



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

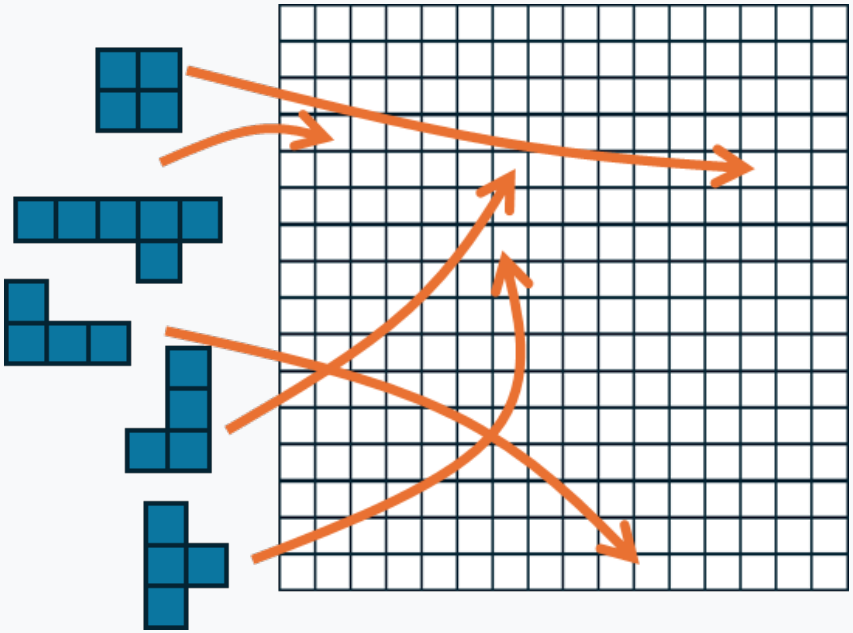


Figure: Algorithme Tetris Modifié

Métriques calculés:

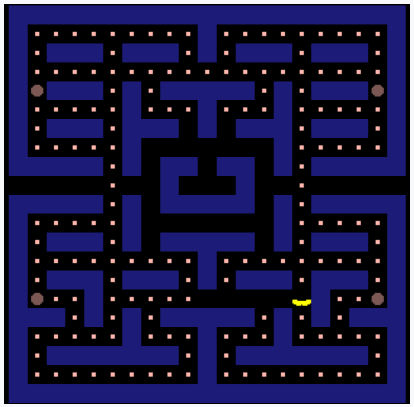
- Proportion de **carrefour**
- Proportion de **jonction**
- Proportion de **virage**
- Proportion de **ligne droite**
- Proportion de **cul-de-sac**
- Proportion de **murs**
- Proportion de **chemin**

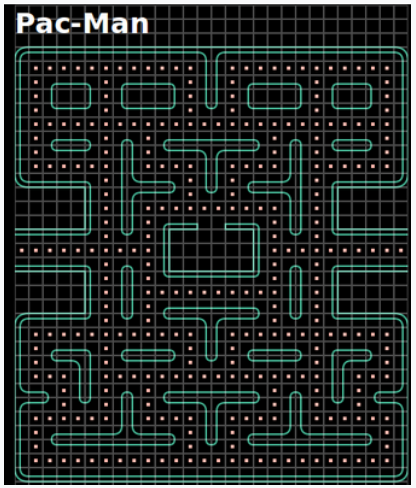
Question : Comment savoir si un labyrinthe est bon ?

1. À la main ? $\longrightarrow$ trop contraignant…
2. Automatiquement ? $\longrightarrow$ oui, mais comment ?

Mesure d'évaluation

Comparaisons





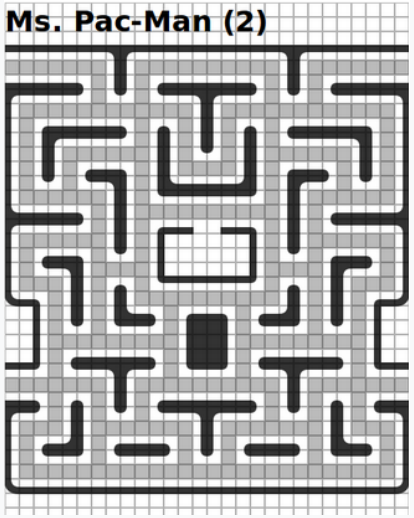


Figure: Quelques "vrai" PACMAN

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 μ le vecteur de la moyenne des mesures sur les trois labyrinthes et x les mesures sur le labyrinthe cible.

$$\mu = \left(\mu_{\text{cross}}, \mu_{\text{junc}}, \mu_{\text{turn}}, \mu_{\text{straight}}, \mu_{\text{dead}}, \mu_{\text{wall}}, \mu_{\text{path}}\right)$$
$$x = \left(x_{\text{cross}}, x_{\text{junc}}, x_{\text{turn}}, x_{\text{straight}}, x_{\text{dead}}, x_{\text{wall}}, x_{\text{path}}\right)$$

Nous définissons **score** (μ, x) comme :

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

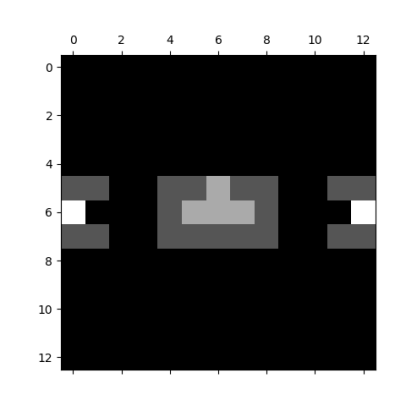
$$\mu_{\text{wall}} > 0.8 \Rightarrow \textbf{score}\,(\mu, x) = 0$$
$$\mu_{\text{path}} > 0.8 \Rightarrow \textbf{score}\,(\mu, x) = 0$$

Interprétation :

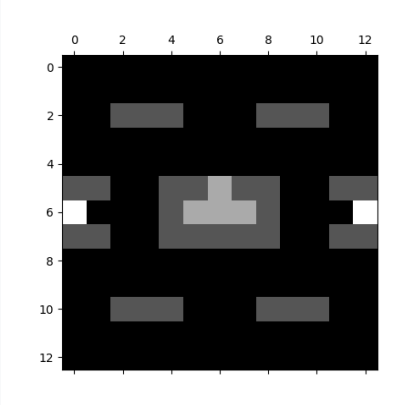
Médiocre : **score** $(\mu, x) \in [0, 2[$
Très mauvais : **score** $(\mu, x) \in [2, 2.5[$
Mauvais : **score** $(\mu, x) \in [2.5, 3[$
Moyen : **score** $(\mu, x) \in [3, 3.5[$
Bien : **score** $(\mu, x) \in [3.5, 4[$
Très bien : **score** $(\mu, x) \geqslant 4$

Mesure d'évaluation

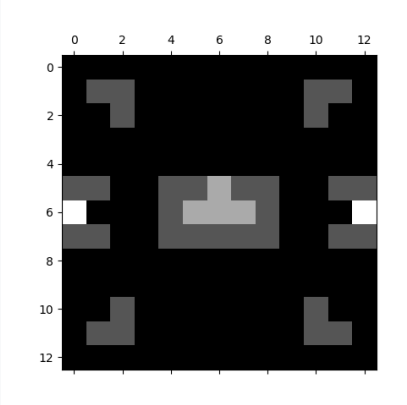
Images



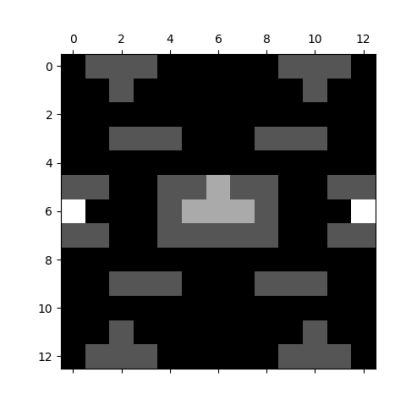
(a) Score: 0 [0]



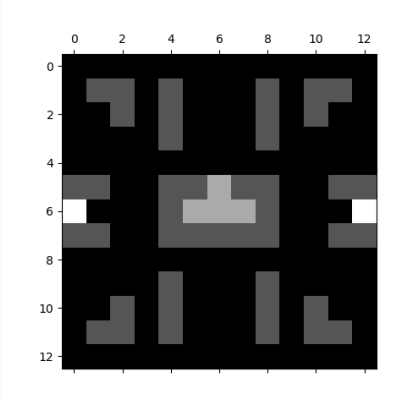
(b) Score: 2.50 [1]



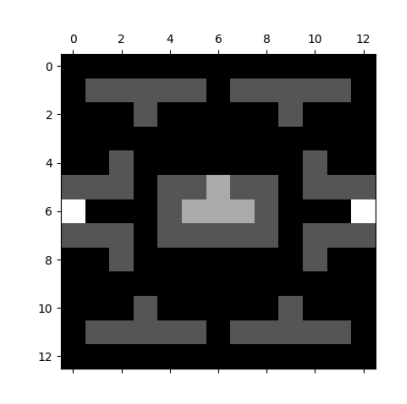
(c) Score: 2.94 [2]



(d) Score: 3.08 [3]



(e) Score: 3.65 [4]



(f) Score: 4.05 [5]

4. PACMAN

5. GitHub et collaboration

Organisation

Gantt

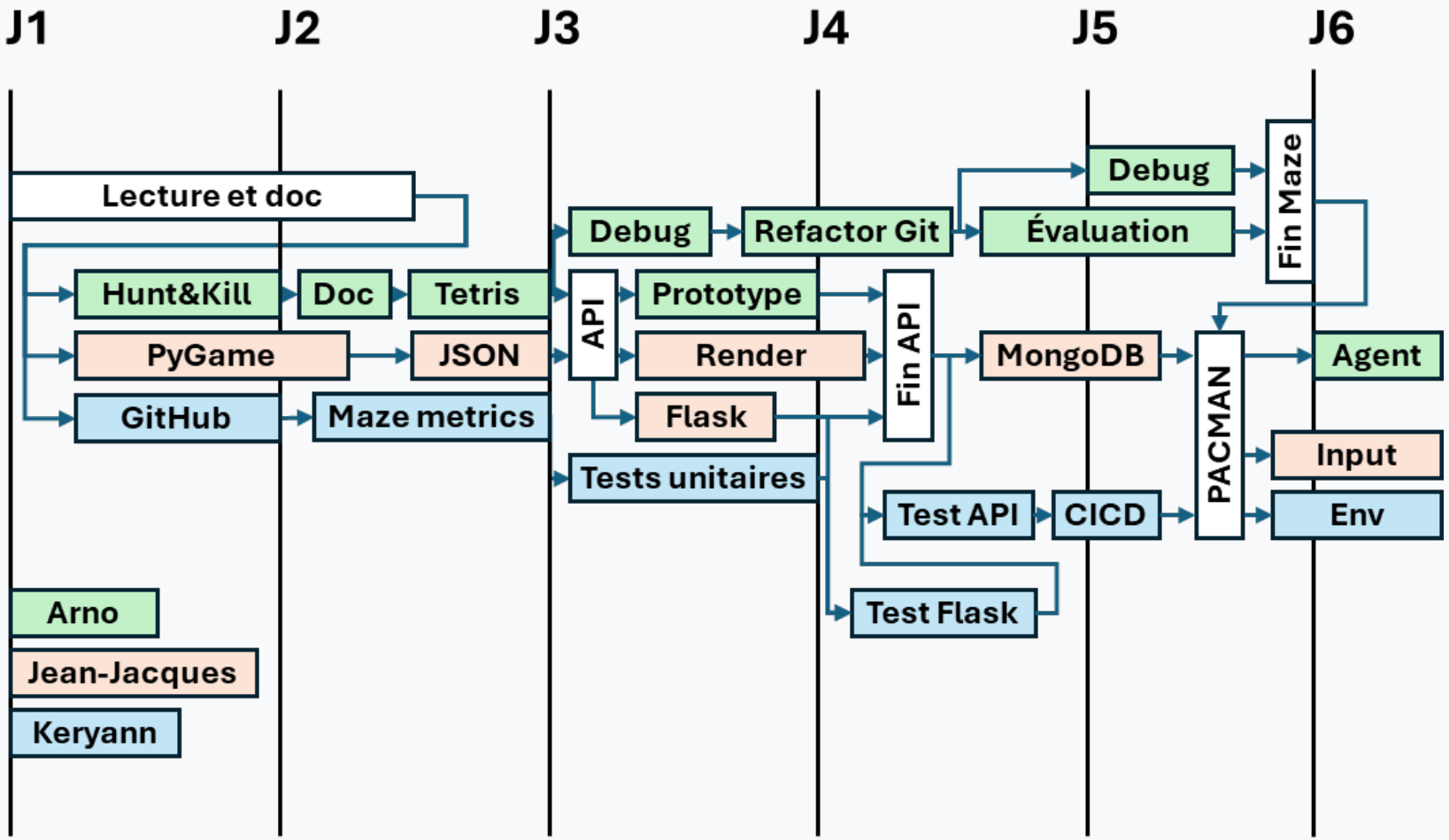


Figure: Diagramme de Gantt jusqu'à aujourd'hui

GitHub

Network Graph

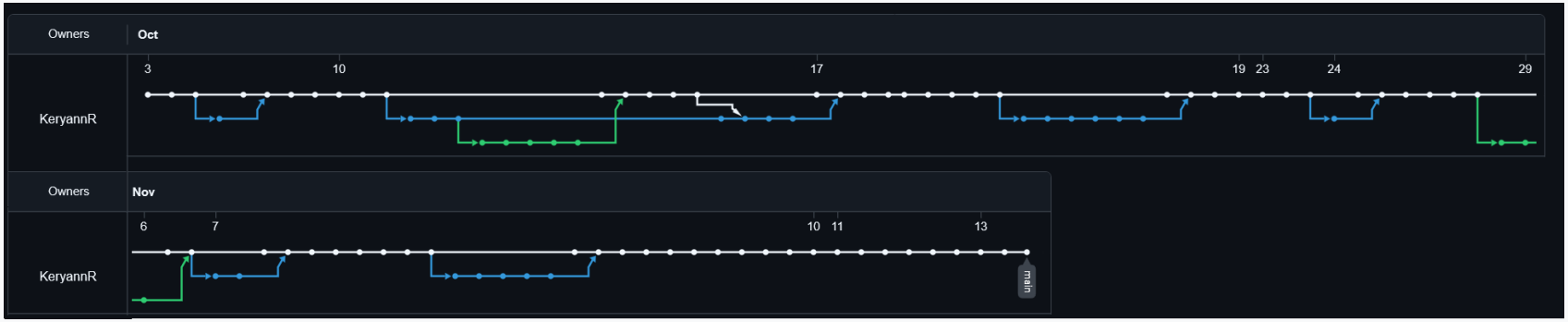


Figure: Network Graph jusqu'à aujourd'hui

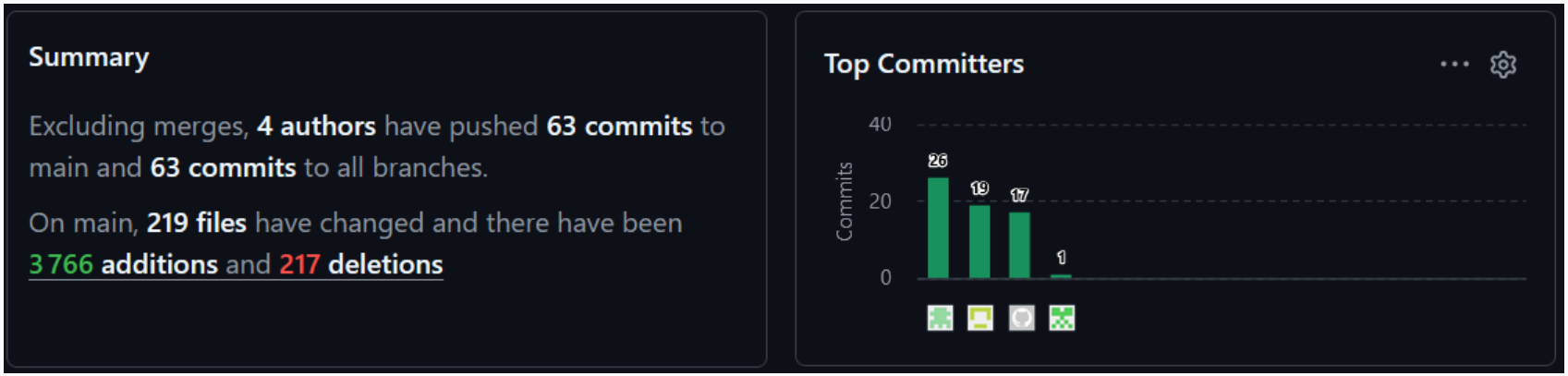


Figure: Commits sur le mois dernier

Merci de votre attention !