Présentation TERD

That Time the Hero saved the Village.

Ben Amara Adam
Fissore Davide
Garnier Quentin
Venturelli Antoine

Organisation du projet

Gestion du travail

GitHub

→ Programmation agile

→ Gestion du “fait” et “à faire” (Kanban)

→ *Versioning*

→ Documentation

Discord

→ Communication

→ Réservations de classes

→ Idées

Google drive

→ rédaction du wiki en groupe sur fichier partagé

→ création de la présentation de la soutenance

Partie 1
La map

Infos principales - La map

Les classes principales et leurs champs

Position =
| int x, y

WorldMap =
| List<Corridor> corridors
| List<Room> rooms

Room =
| Position topLeft, bottomRight
| int id, lowestNeighbor
| List<Position> doors
| List<Monster> monsters
| List<AbstractItem> items

Corridor =
| int id
| List<Position> positionList

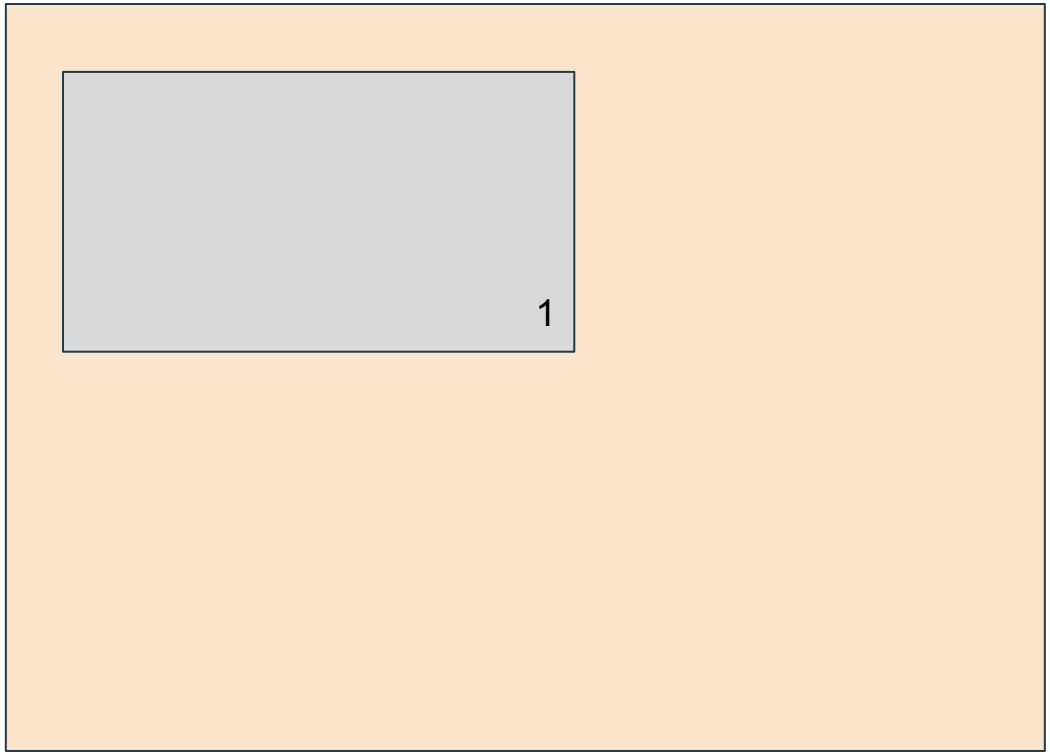
Génération des salles

Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =

Génération des salles

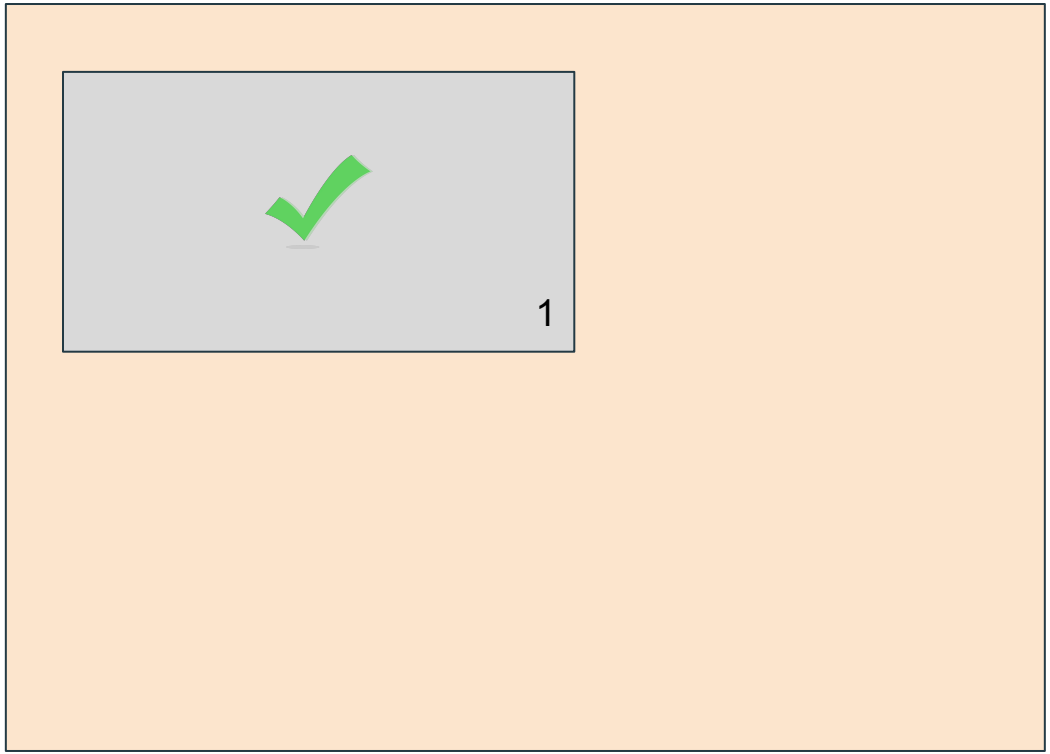


Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =

Génération des salles

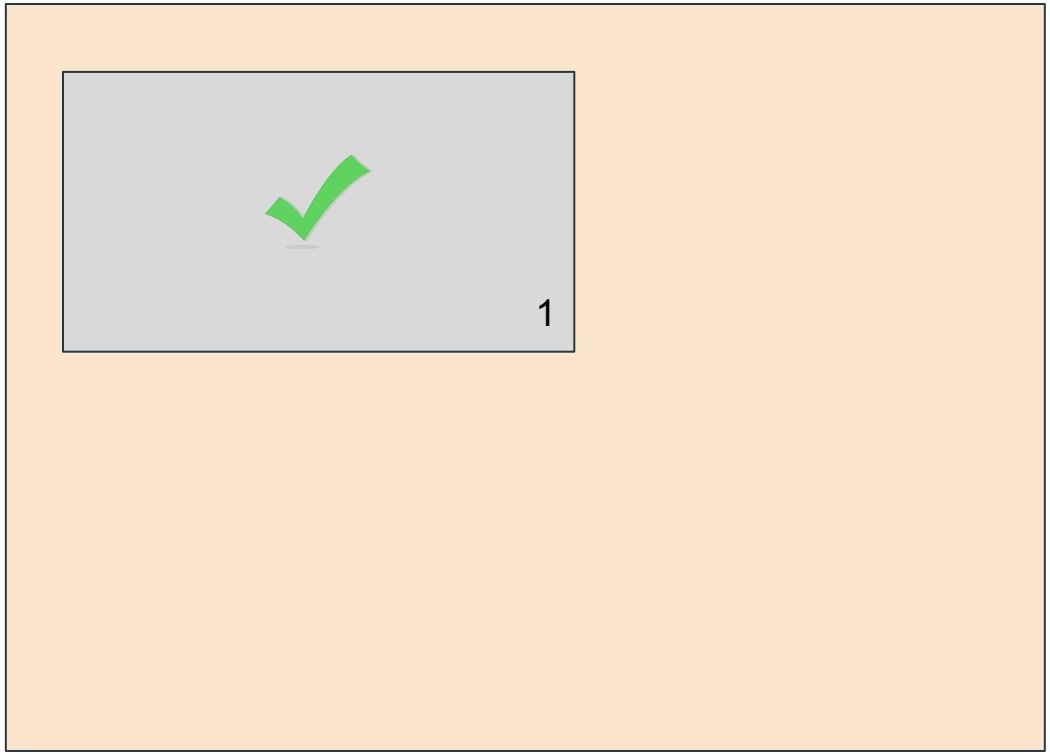


Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =

Génération des salles



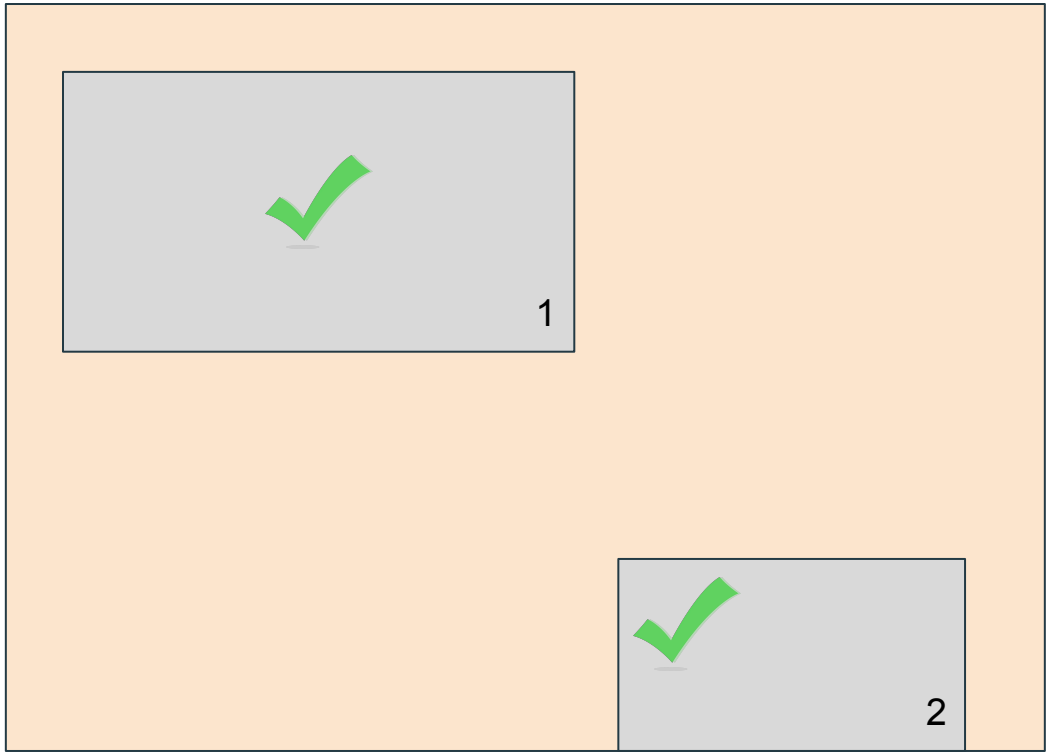
Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =

1

Génération des salles

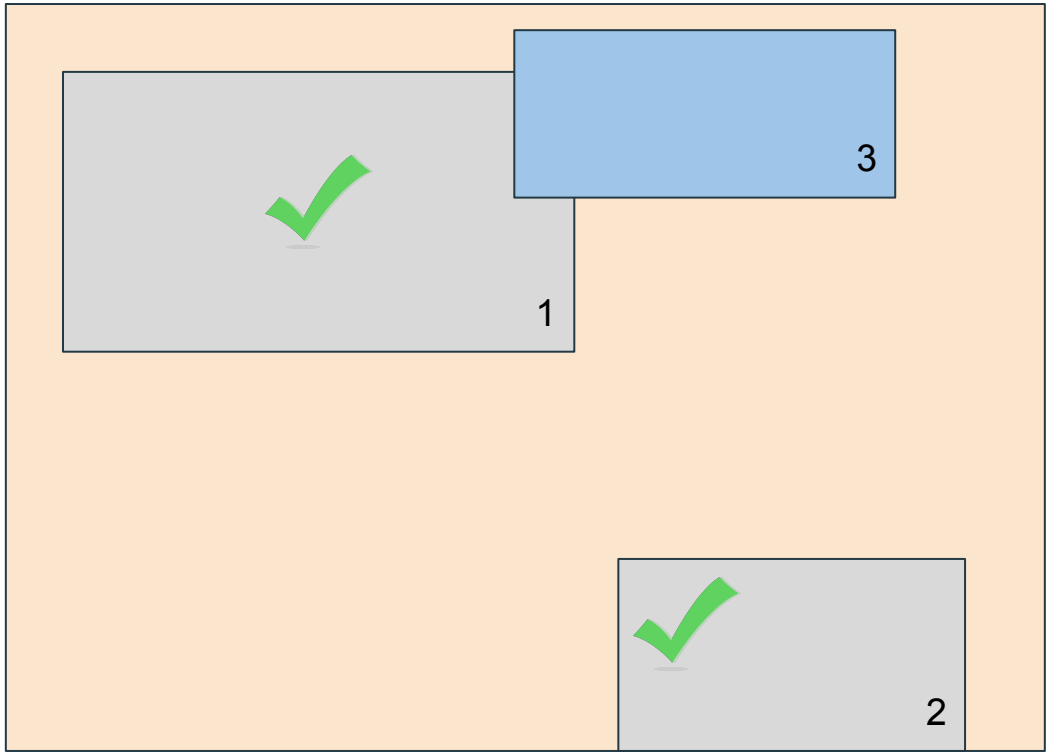


Exemple avec 6 itérations :

2. new Room()

WorldMap.rooms =

Génération des salles

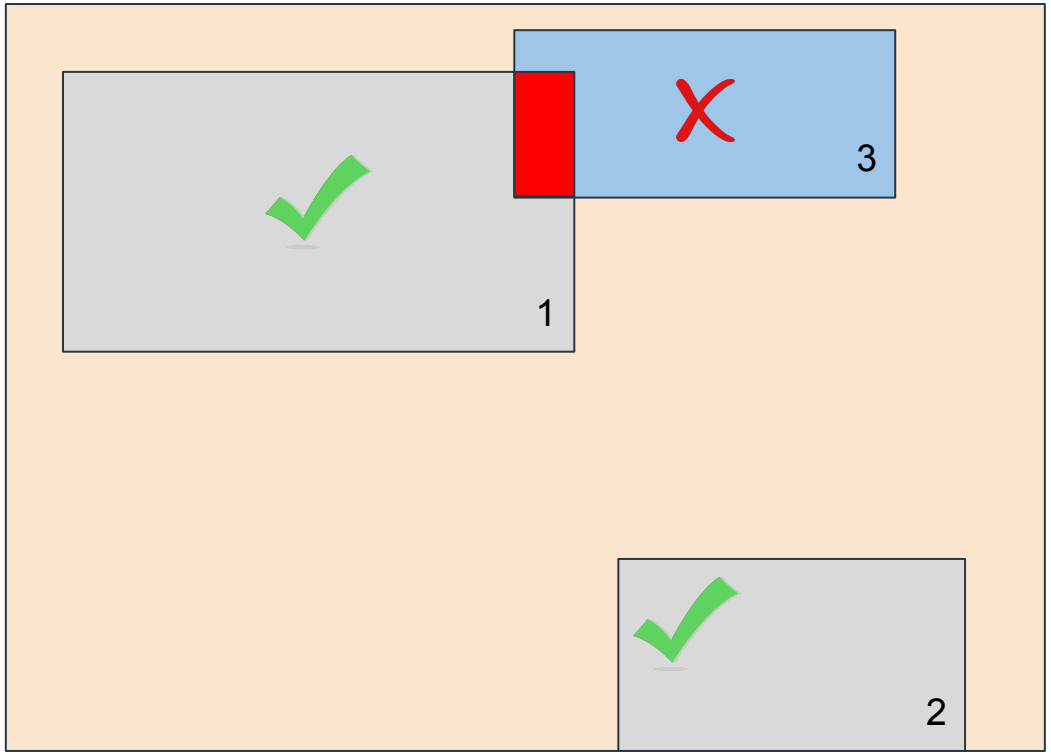


Exemple avec 6 itérations :

3. new Room()

WorldMap.rooms =

Génération des salles

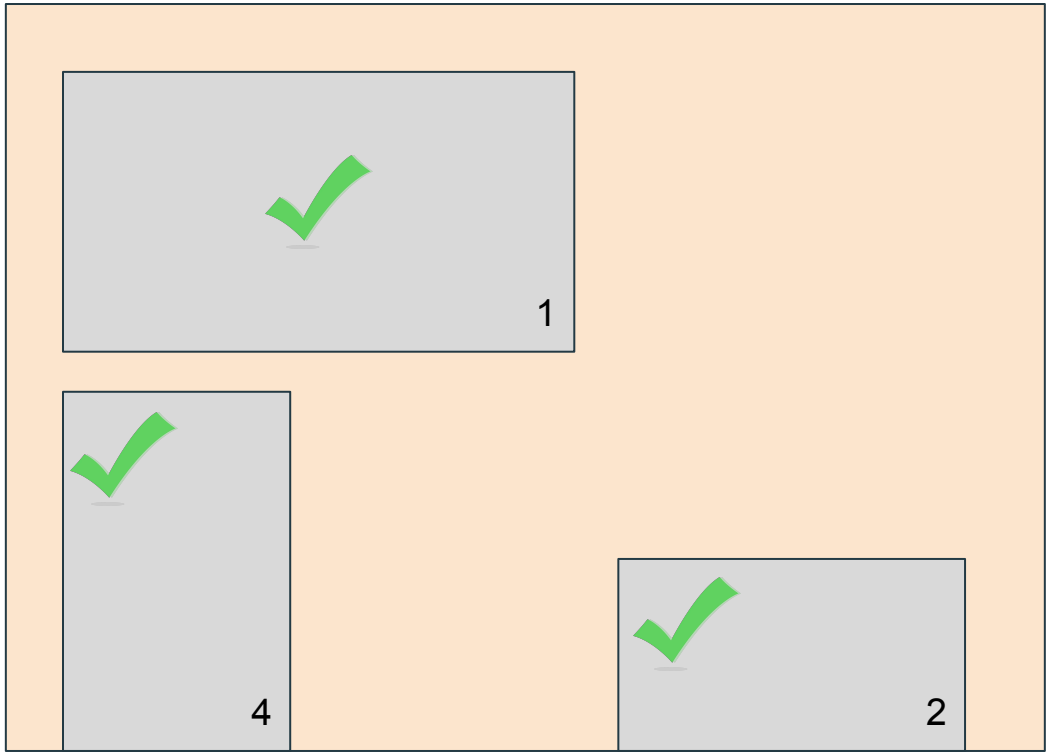


Exemple avec 6 itérations :

3. new Room()

WorldMap.rooms =

Génération des salles



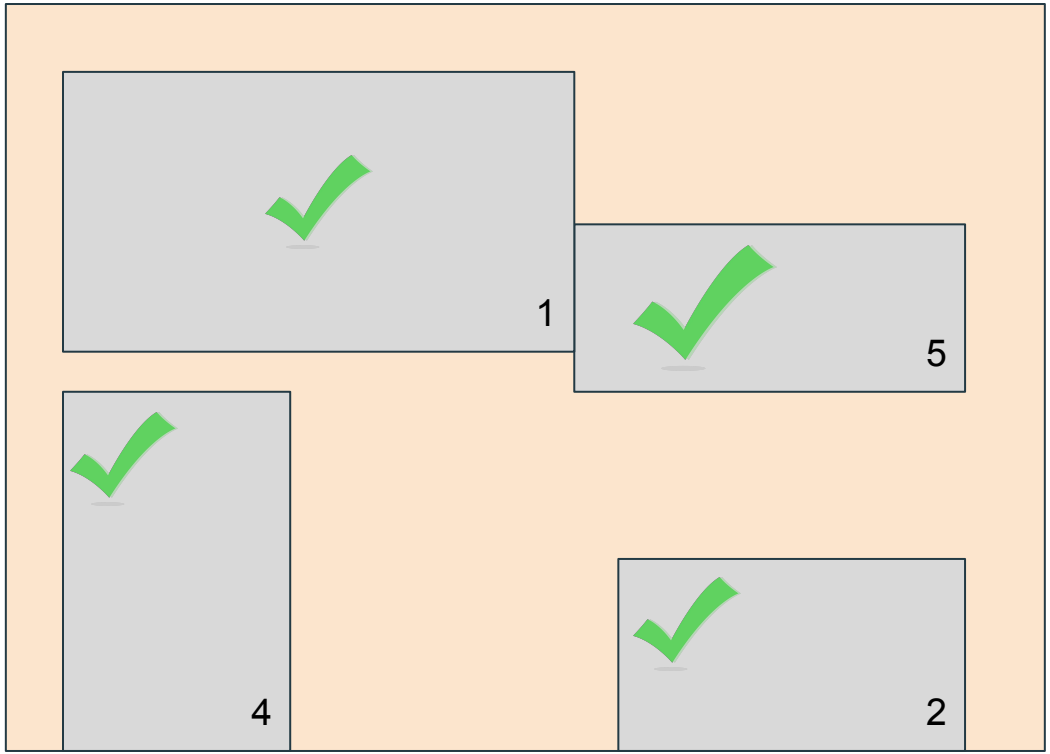
Exemple avec 6 itérations :

4. new Room()

WorldMap.rooms =

1	2	4

Génération des salles



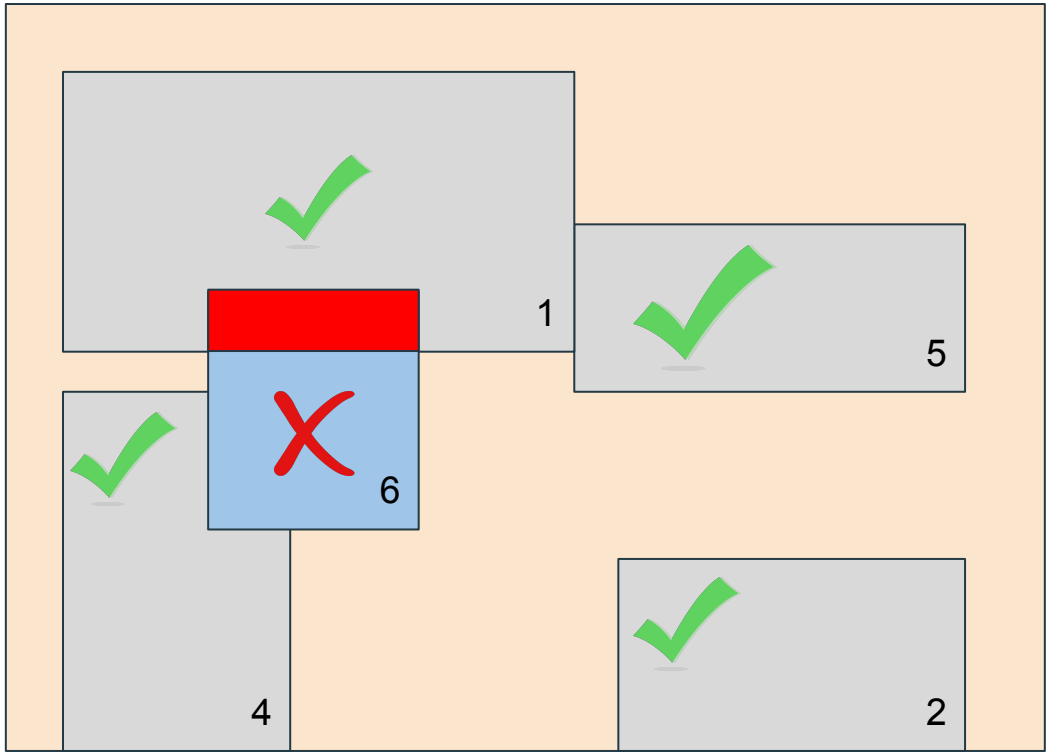
Exemple avec 6 itérations :

5. new Room()

WorldMap.rooms =

1	2	4	5

Génération des salles



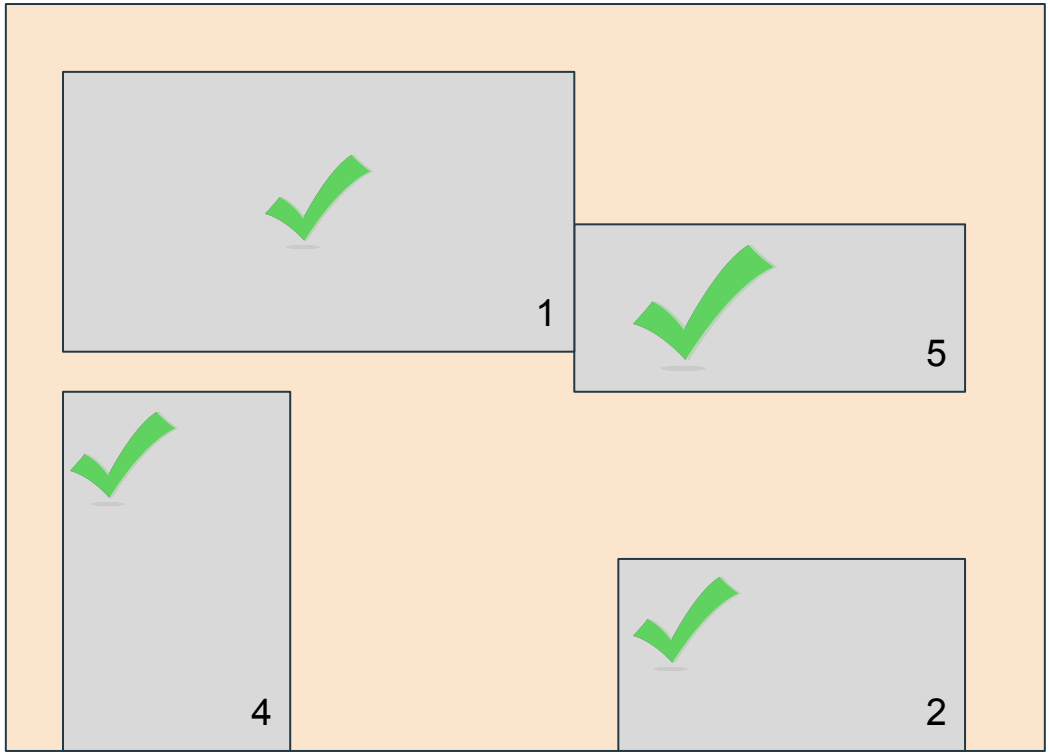
Exemple avec 6 itérations :

6. new Room()

WorldMap.rooms =

1	2	4	5

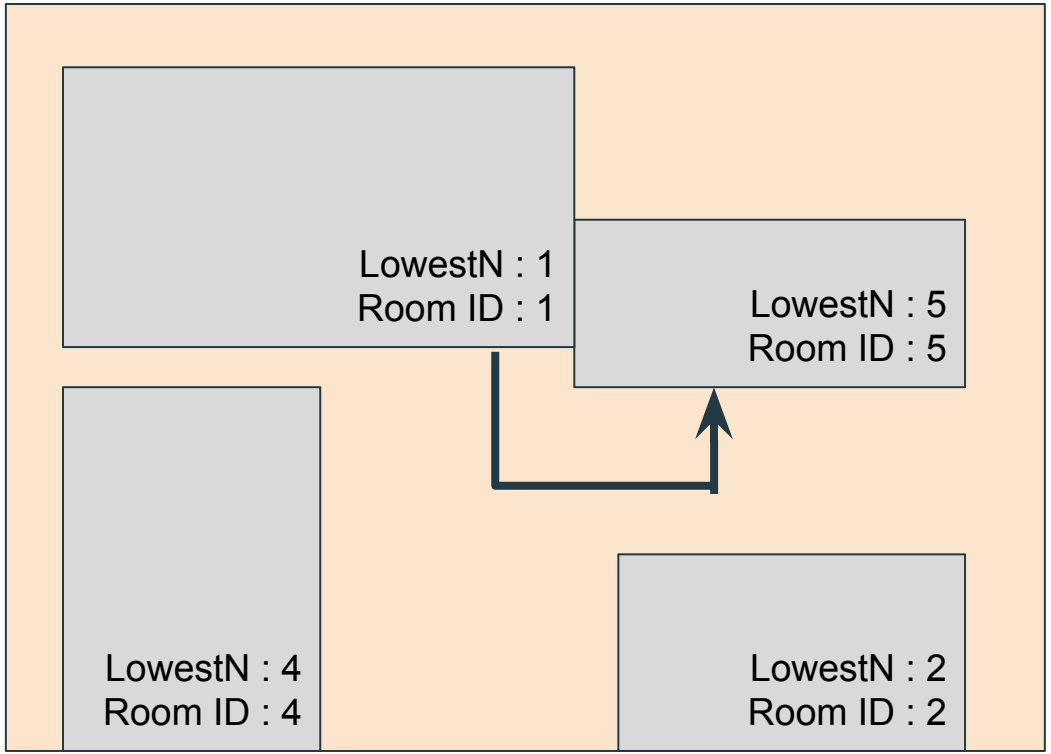
Génération des salles



WorldMap.rooms =

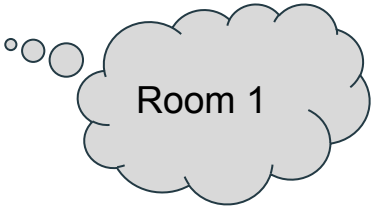
1	2	4	5

Les couloirs [Corridor]

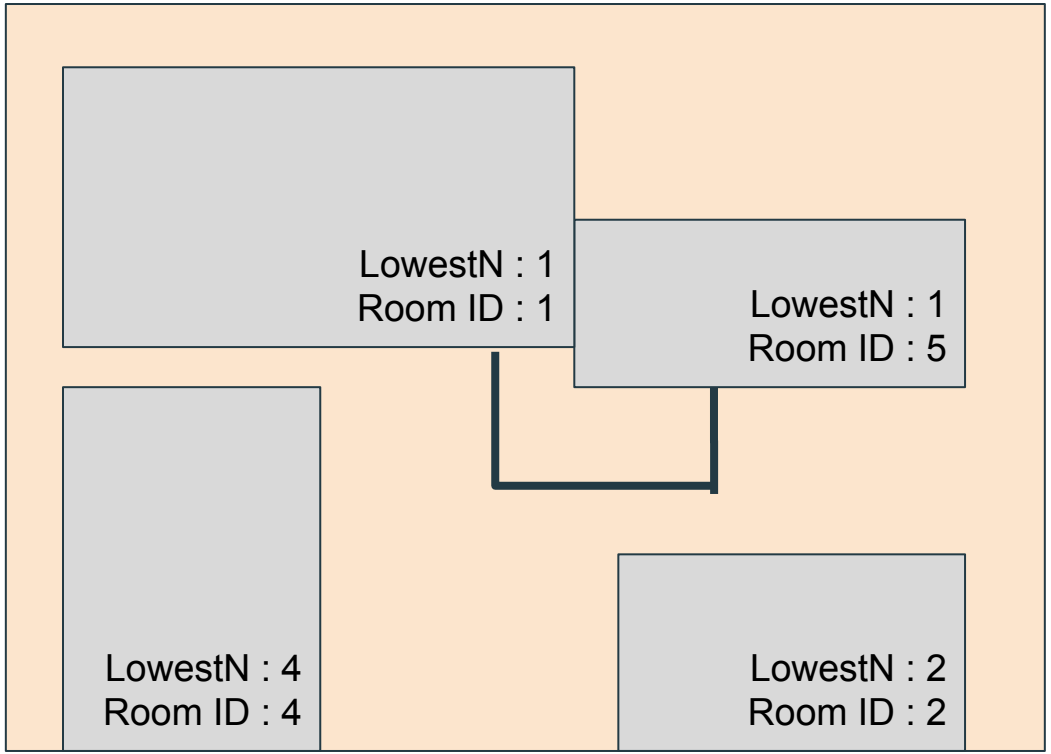


Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)



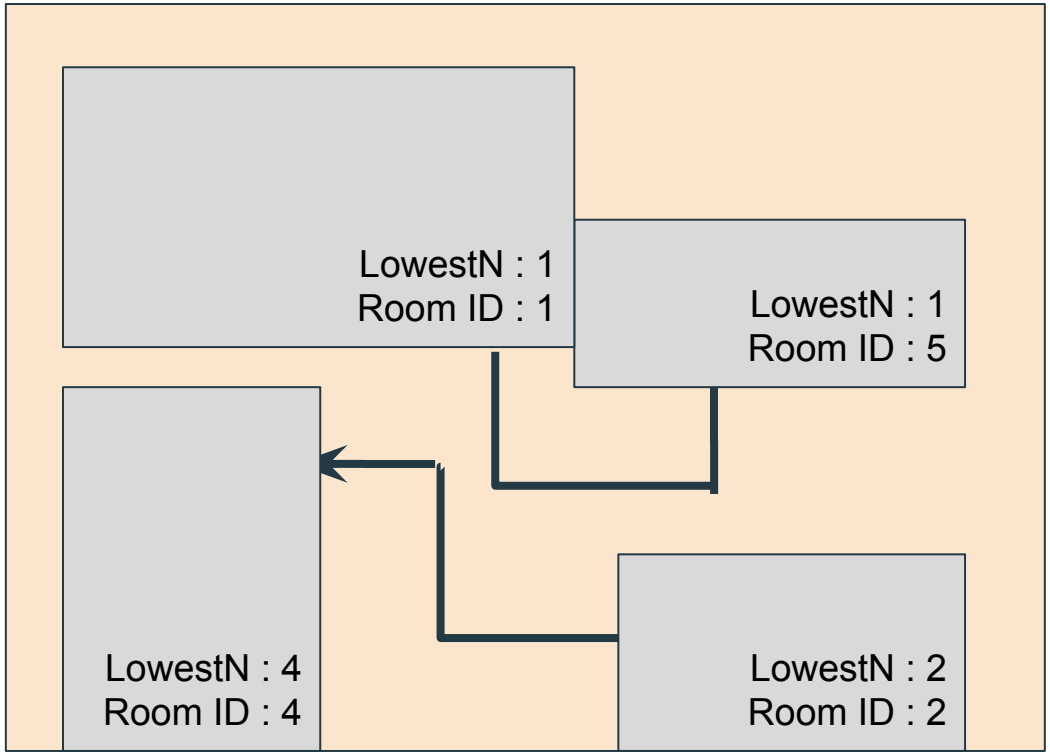
Les couloirs [Corridor]



Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

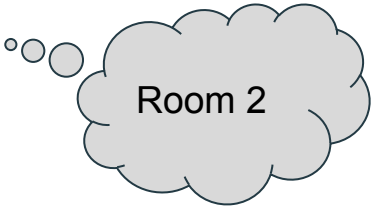
BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)

Les couloirs [Corridor]

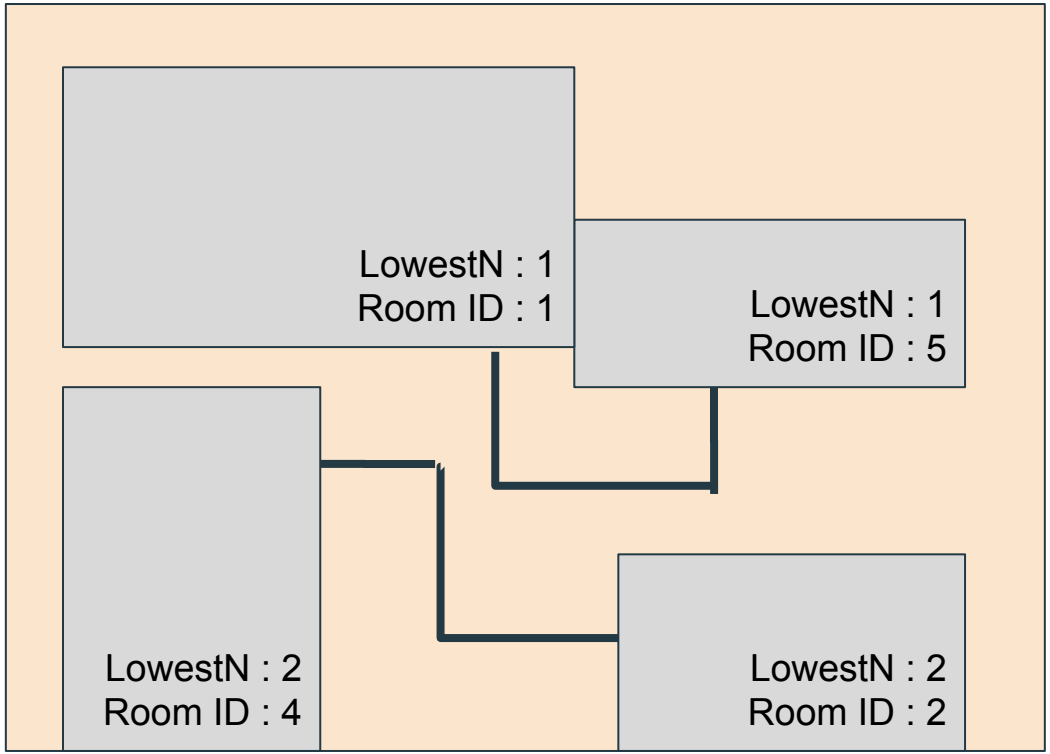


Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)



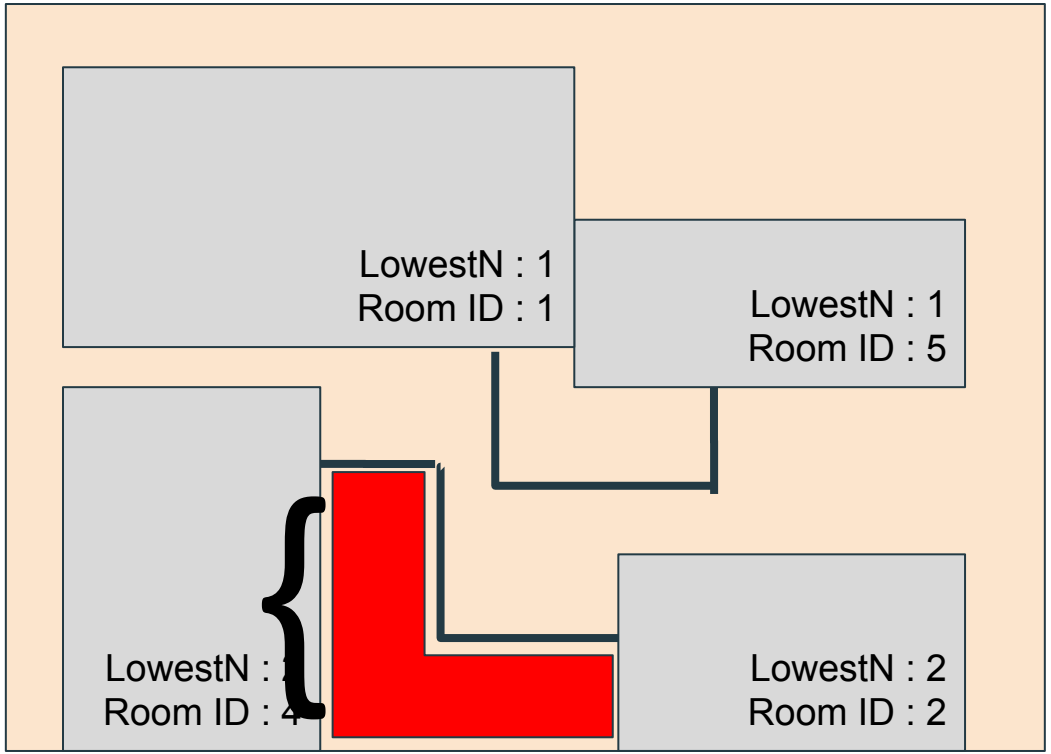
Les couloirs [Corridor]



Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)

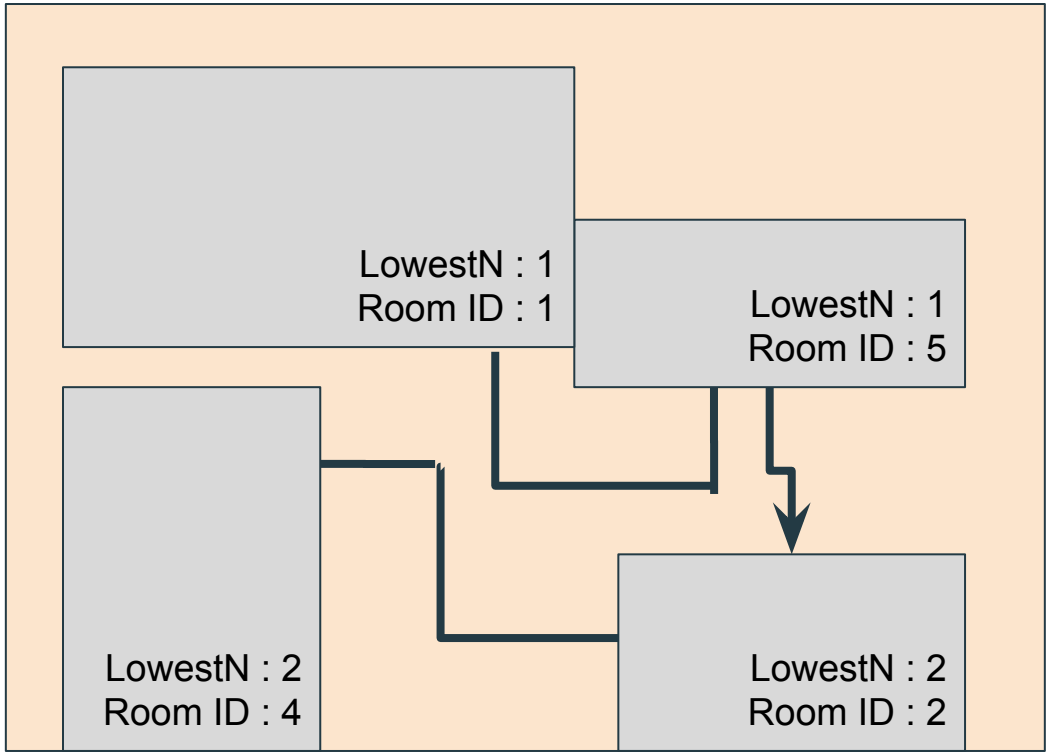
Les couloirs [Corridor]



Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)

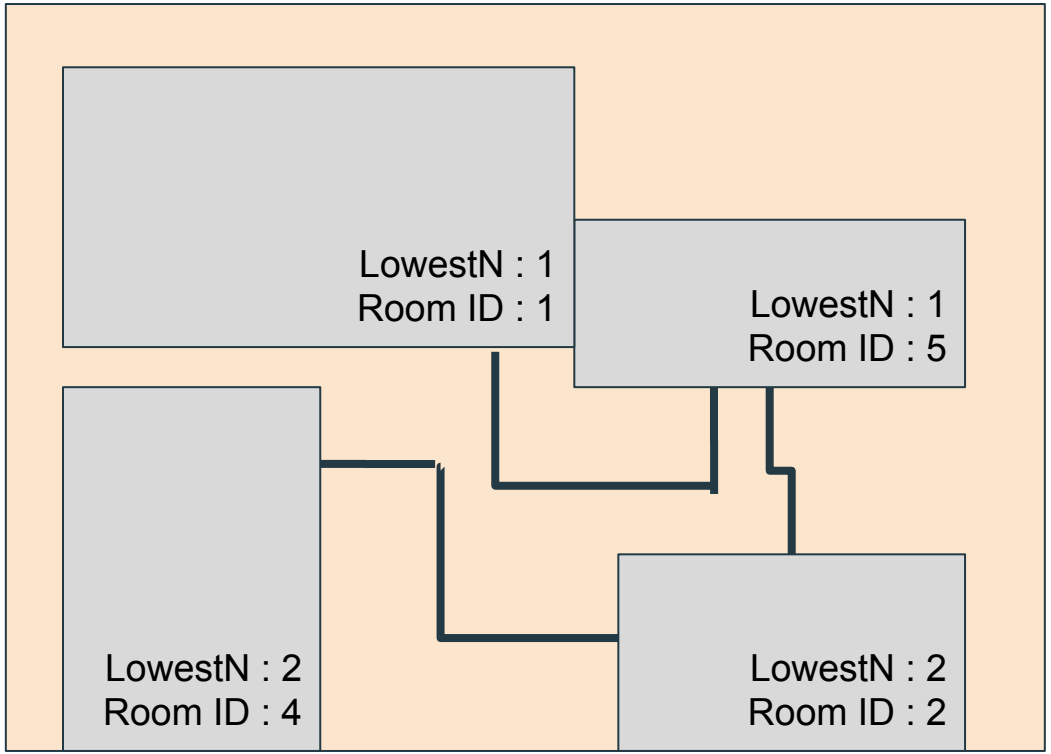
Les couloirs [Corridor]



Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)

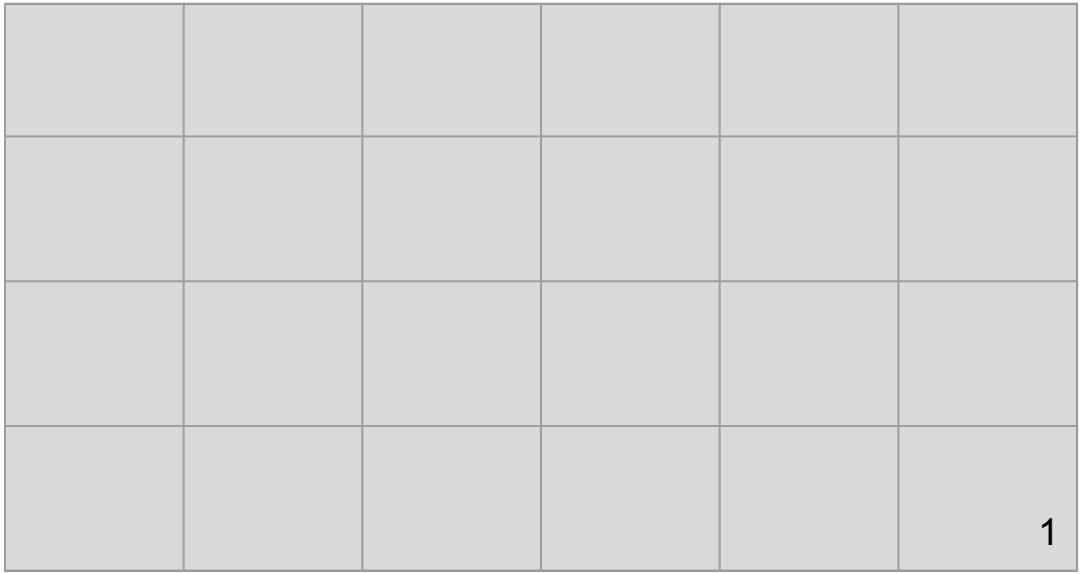
Les couloirs [Corridor]



Pseudo-code :
createCorridor:
- for R in rooms:
-- choose door D in R
-- start BFS_modified from D

BFS_modified(start):
- choose a neiborgh N of start
- if N ∈ R with different lowestN:
-- create_corridor
-- set_lowestN
- else : BFS_modified(N)

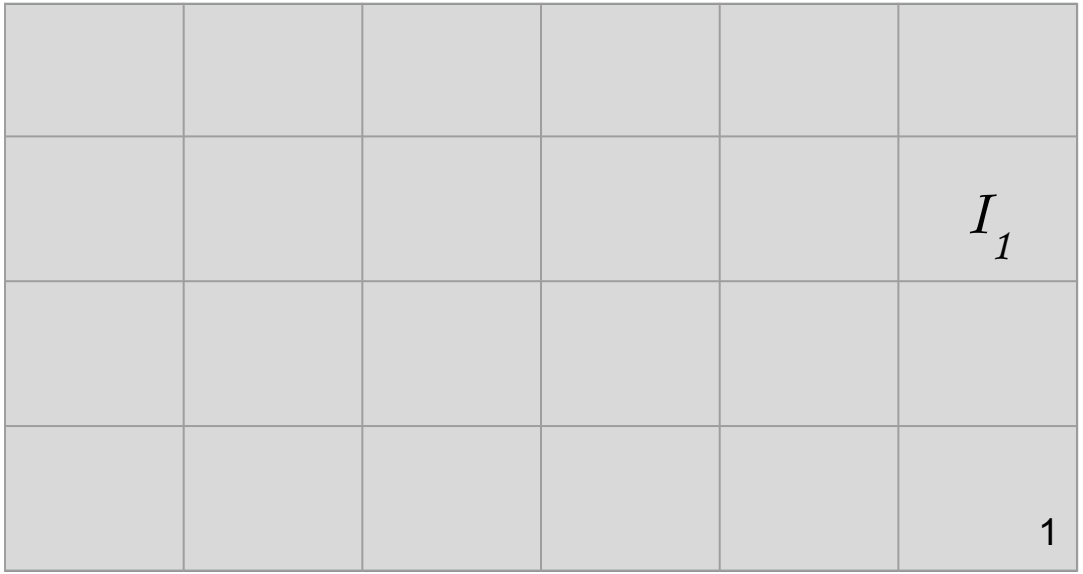
Emplacement des entités | items



this.currentItems = { }
this.monsters = { }

```
void putRandomEltInRoom():
        putItems();
        putMonsters();

void putItems():
        for i in RAND1:
                put obstacle or item

void putMonster():
        for i in RAND2:
                put monster
```

Emplacement des entités | items



this.currentItems = { I_1}
this.monsters = { }

```
void putRandomEltInRoom():
        putItems();
        putMonsters();

void putItems():
        for i in RAND1:
                put obstacle or item

void putMonster():
        for i in RAND2:
                put monster
```

Emplacement des entités | items

this.currentItems = { I_1}
this.monsters = { }

```
void putRandomEltInRoom():
        putItems();
        putMonsters();

void putItems():
        for i in RAND1:
                put obstacle or item

void putMonster():
        for i in RAND2:
                put monster
```

Emplacement des entités | items

	$O2$		$O2$		
I_2					I_1
		$O2$	I_3		
				$O2$	1

this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { }

```
void putRandomEltInRoom():
        putItems();
        putMonsters();

void putItems():
        for i in RAND1:
                put obstacle or item

void putMonster():
        for i in RAND2:
                put monster
```

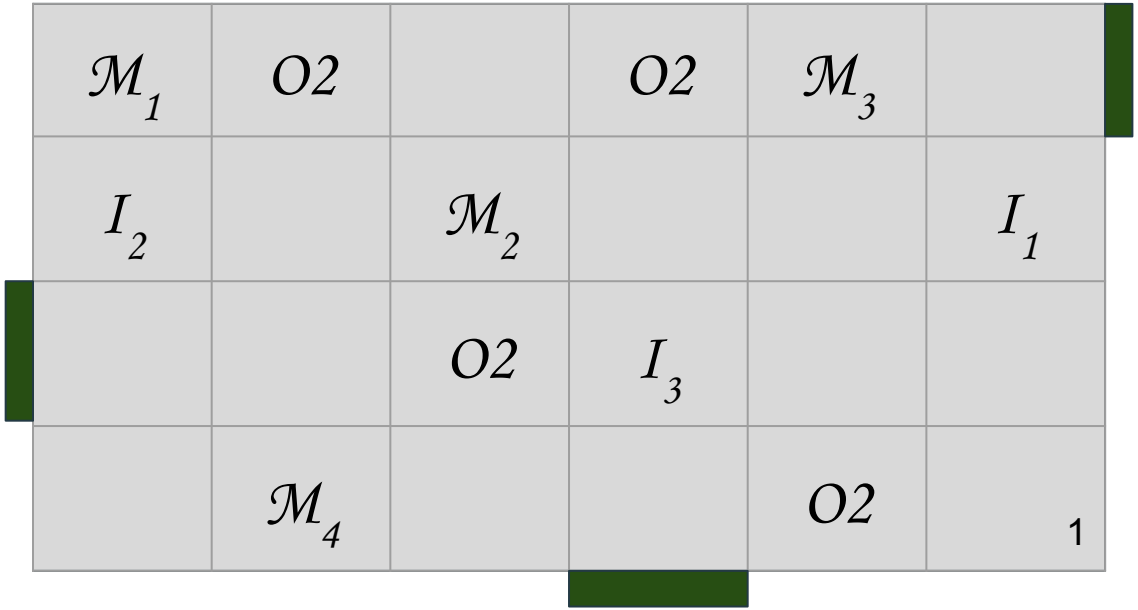
Emplacement des entités | items

$\mathcal{M}_1$	$O2$		$O2$	$\mathcal{M}_3$	
I_2		$\mathcal{M}_2$			I_1
		$O2$	I_3		
	$\mathcal{M}_4$			$O2$	1

this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}

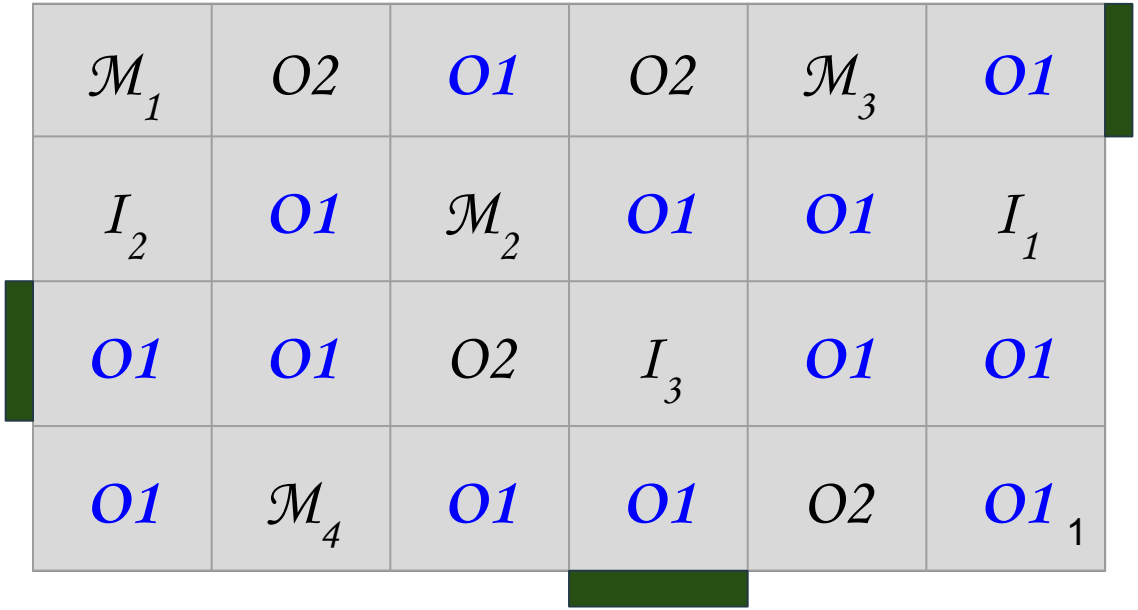
```
void putRandomEltInRoom():
        putItems();
        putMonsters();

void putItems():
        for i in RAND1:
                put obstacle or item

void putMonster():
        for i in RAND2:
                put monster
```

Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

Les obstacles

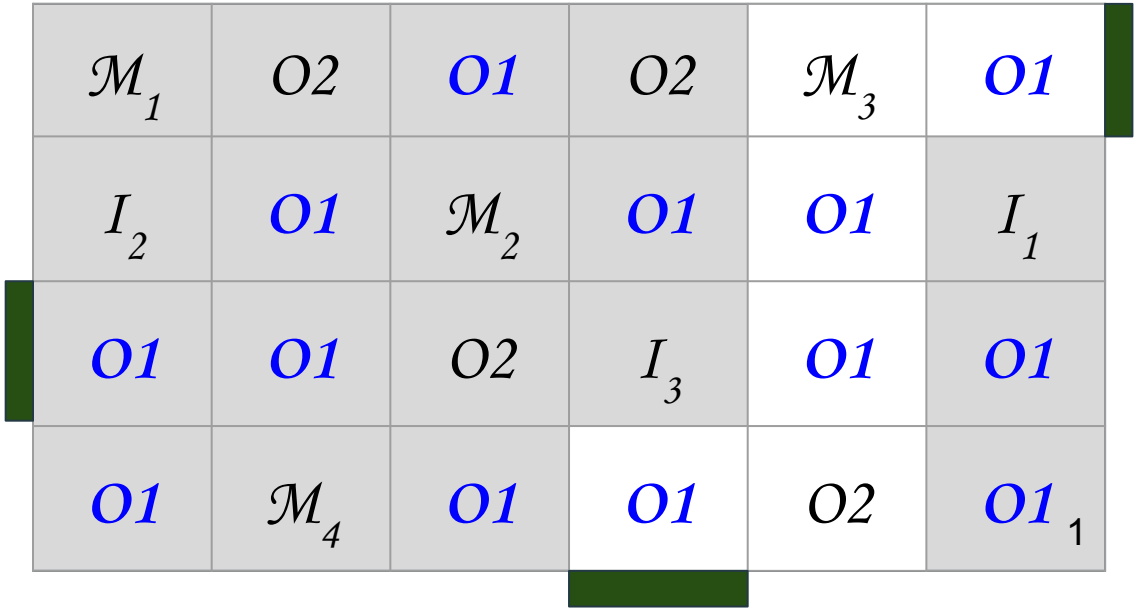


this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

O1 = obstacles de type forêt
O2 = obstacles peu fréquents

But :
remplir la salle avec max de O1

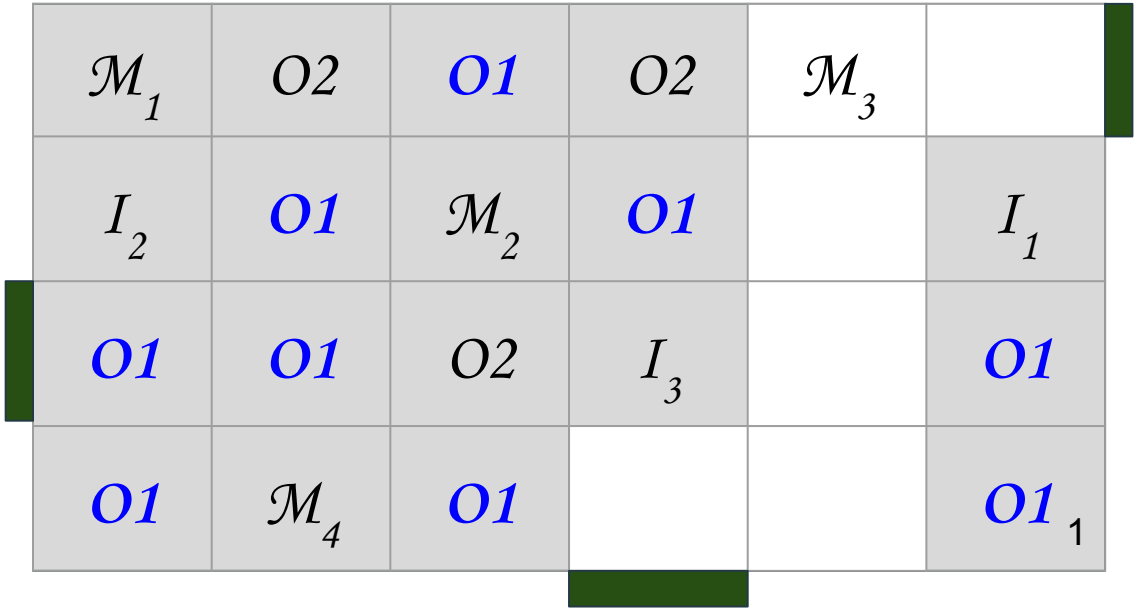
Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

But :
créer des chemins valides entre la première porte et toutes les autres.
On cherche le plus court chemin entre D1 et D2

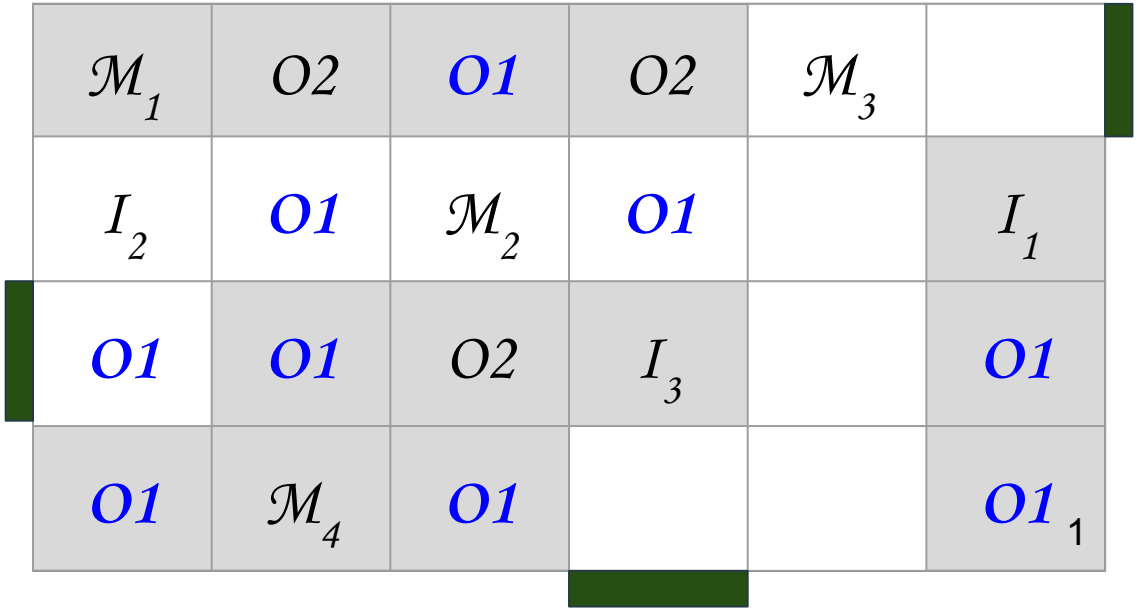
Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

But :
pour que le chemin soit accessible, il faut supprimer tout obstacle (O1 ou O2) qui se trouve sur ce chemin

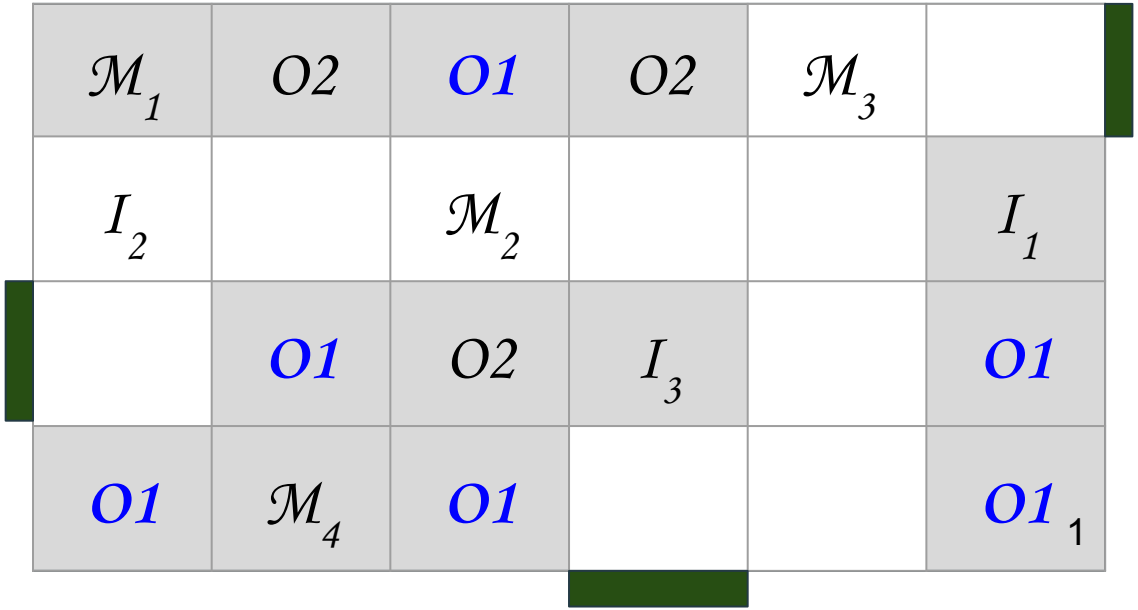
Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

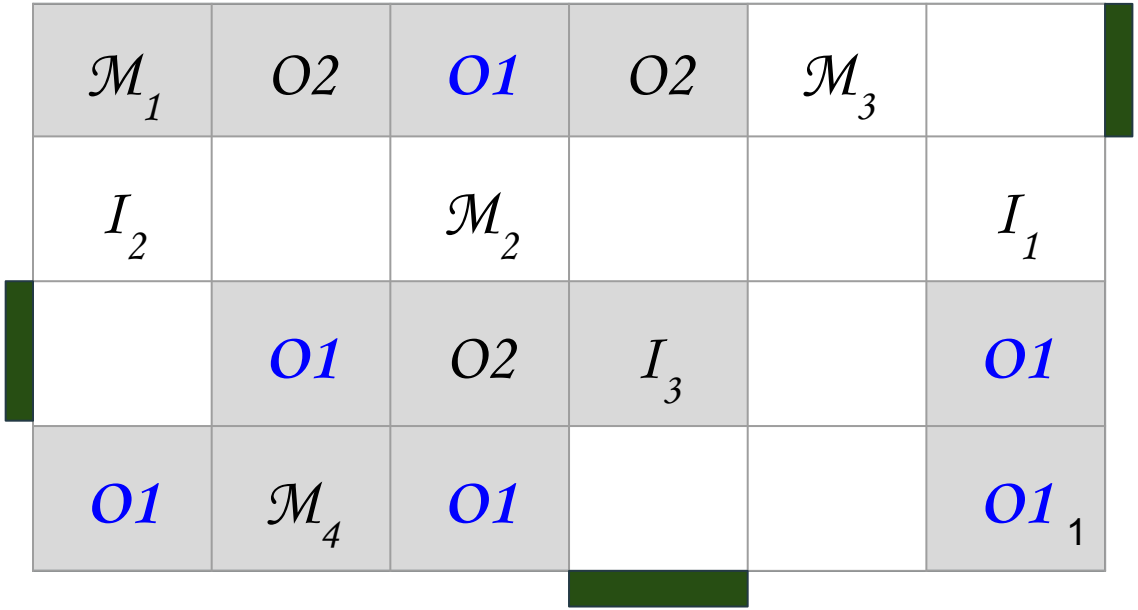
But :
lier D1 avec D3

Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

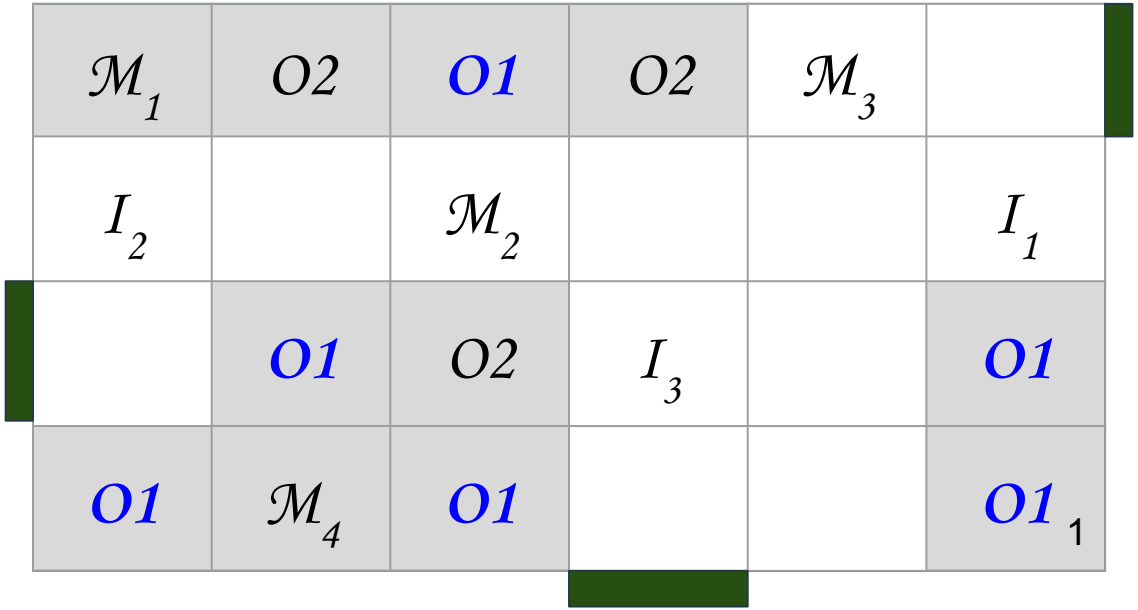
Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

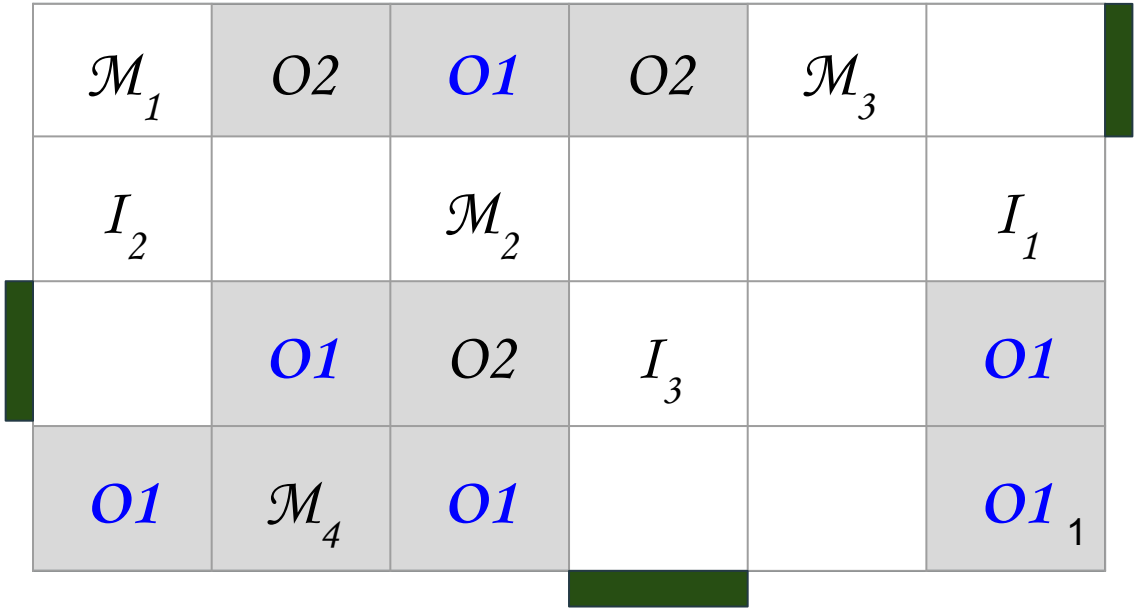
But :
une fois les portes liées, il faut s'assurer que tout item et tout monstre soit atteignable par le jouer : on vérifie qu'ils soient accessibles

Les obstacles



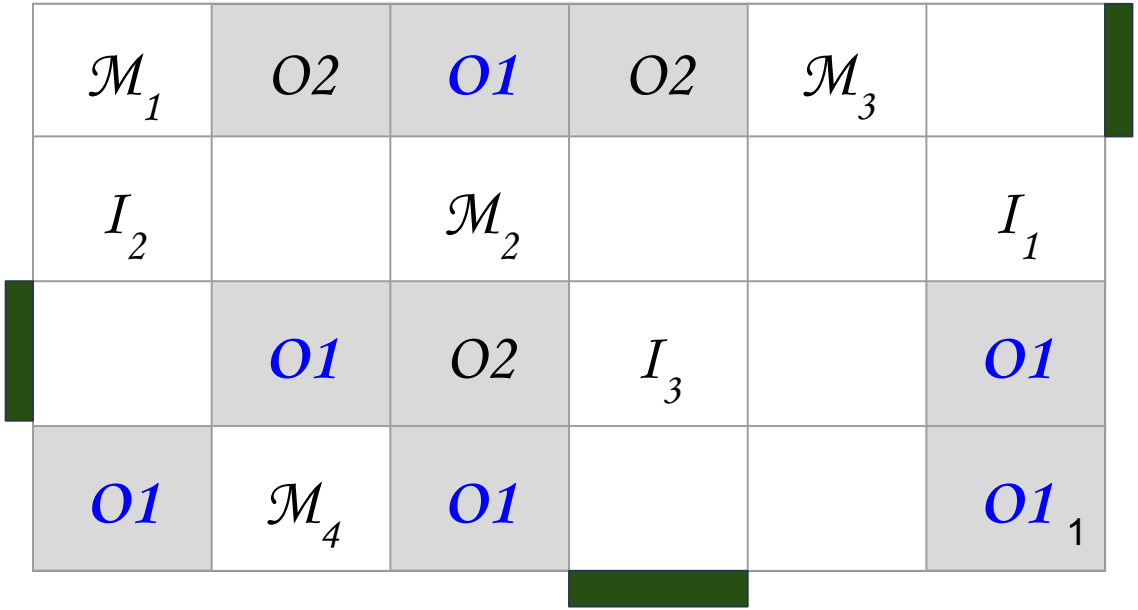
this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

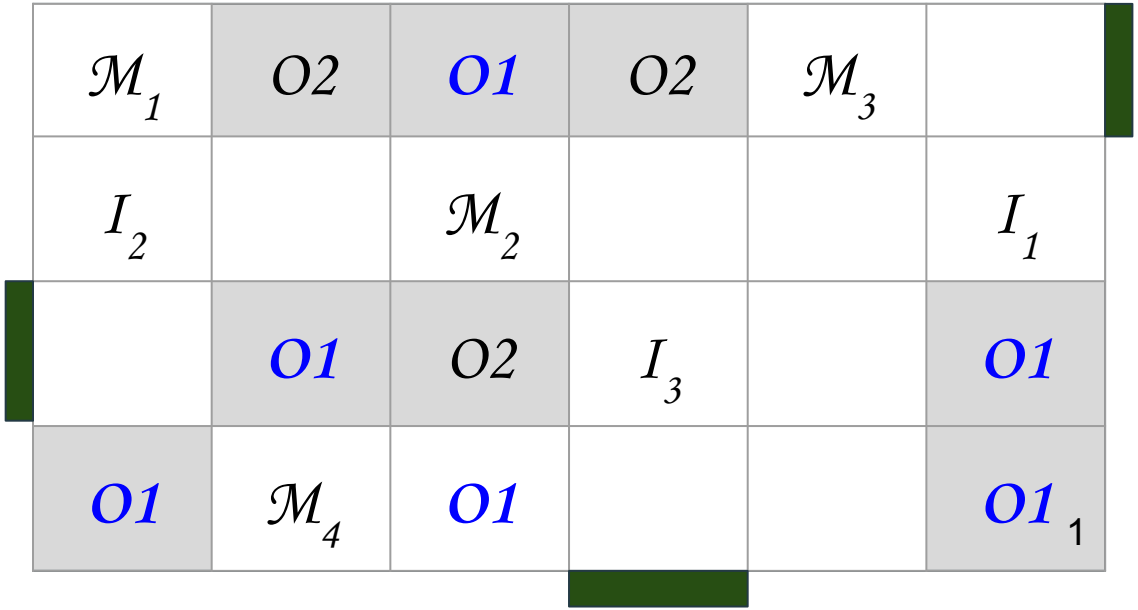
Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

But :
dans l'exemple M4 est le seul monstre à ne pas être accessible

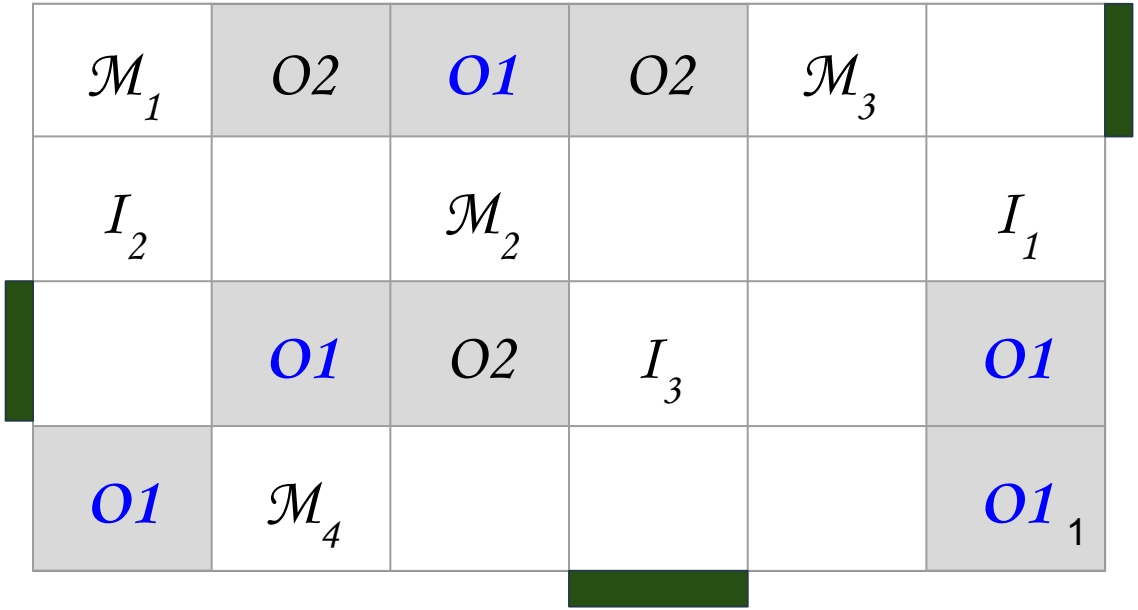
Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

But :
on applique l'algo BFS pour trouver le plus court chemin vers une cellule libre

Les obstacles



this.currentItems = { I_1 , I_2 , I_3}
this.monsters = { M_1 , M_2 , M_3 , M_4}
this.doors = {D_1 , D_2 , D_3}

But :
on supprime les obstacles sur ce chemin et finalement cette salle est une salle valide : on peut y entrer et sortir, en pouvant être attaqué par tout monstre et en pouvant ramasser tout item

Partie 2
Les entités et les items

Infos principales - Les entités et les Items

Les classes principales et leurs champs des **Entité**

AbstractEntity
- | Position
- | int attack, HP
- | EntityType
- | Strategy
- | EntityState

Player
- | int level, hunger, money
- | List<AbstractCollectableItem>

Monster

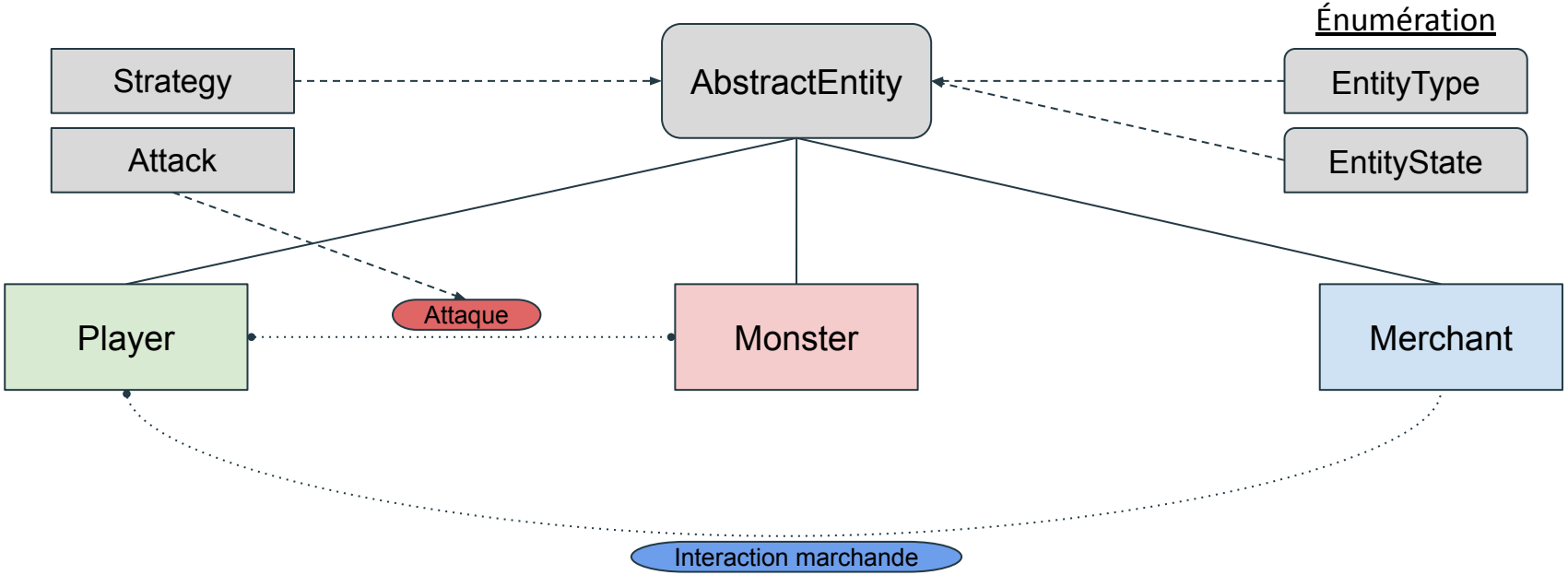
Marchant
- | int safeRoomId
- | List<AbstractCollectableItem>
- | JDialog

Les classes principales et leurs champs des **Items**

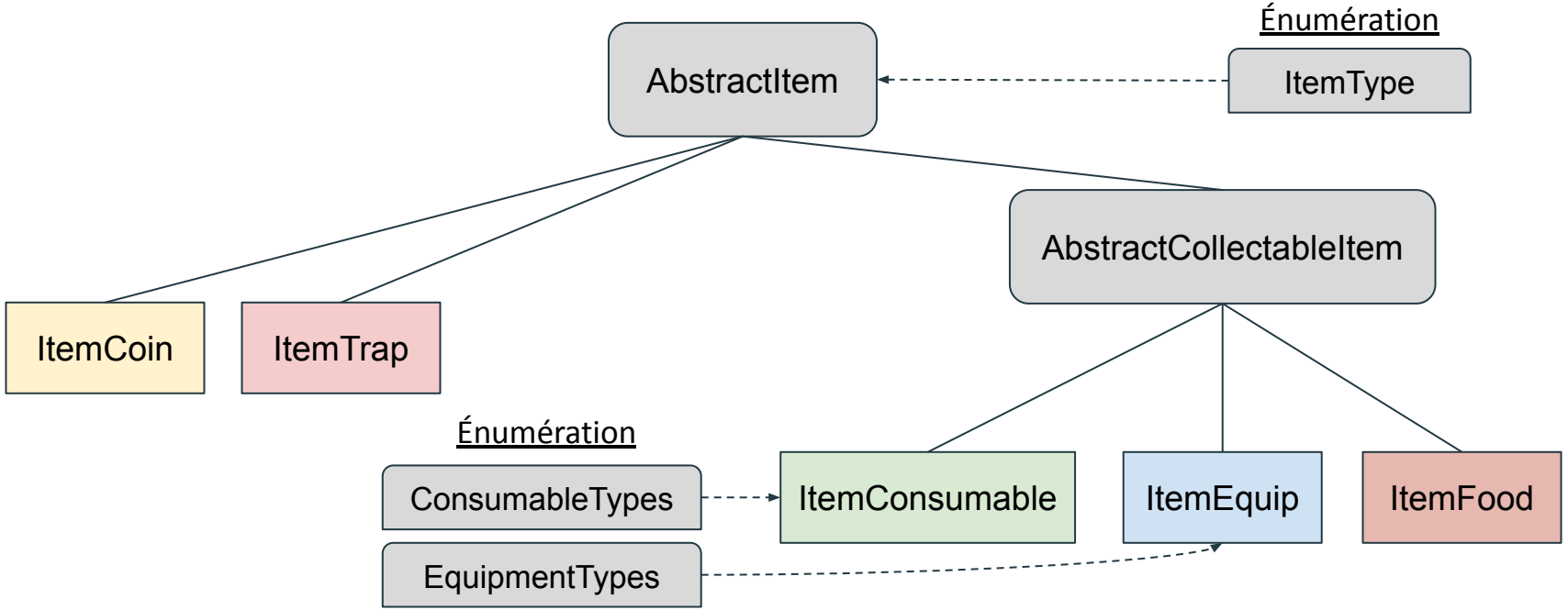
AbstractItems
- | ItemType
- | int id
- | Position
- | boolean immediateUse

AbstractCollectableItem

Entité
Schéma de dépendance



Item
Schéma de dépendance



Singleton Pattern

3 Singletons concernant les entités :

- Une instance de Player (*instancePlayer*)
- Une instance de Monster (*boss*)
- Une instance de Merchant (*instanceMerchant*)

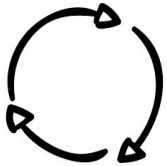
+ 1 instance statique et unique de l'item de type **END**

Factory Pattern

3 Factories concernant les entités/items :

- Une fabrique à monstres (dans classe **Monster**, *generateRandomMonster()*)
- Une fabrique d'items (dans classe **AbstractItem**, *generateRandomItem()*)
- Une fabrique d'items collectables (dans classe **AbstractCollectableItem**, *generateAbstractCollItems()*)

Game Loop



Avec l’introduction du Graphisme, ce pattern n’est plus vraiment de rigueur.

Mais pour l’affichage terminal, on avait un jeu qui tournait à l’infinie (tant que l’on ne mourrait pas). Cela était permis par une boucle.

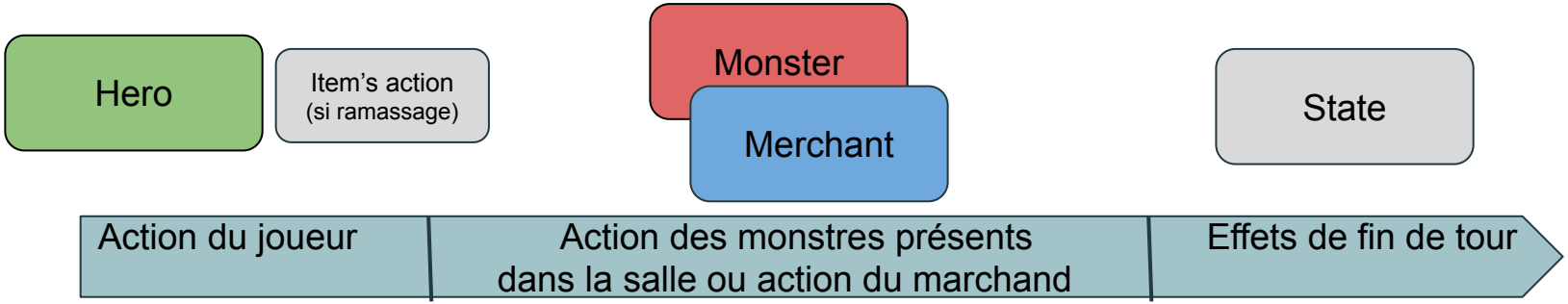
State Pattern

Le comportement d’une entité va être modifié si champs modifiés

Strategy Pattern

Une méthode qui diffère (dans classe **Strategy**, *applyStrategy()*) selon le type de l’entité

Tour par tour

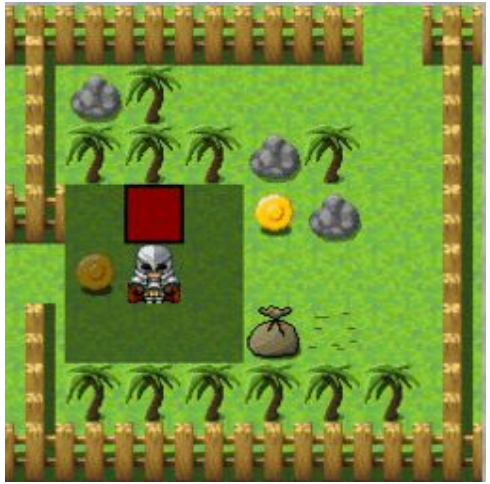


Chronologie d'un tour

Note : utiliser un item ne consomme pas le tour

Attack

Warrior



Archer

Mage

Chaque type a une attaque et range bien distinctes

Movement

Merchant

Aléatoire

Wizard

Appréhender la portée

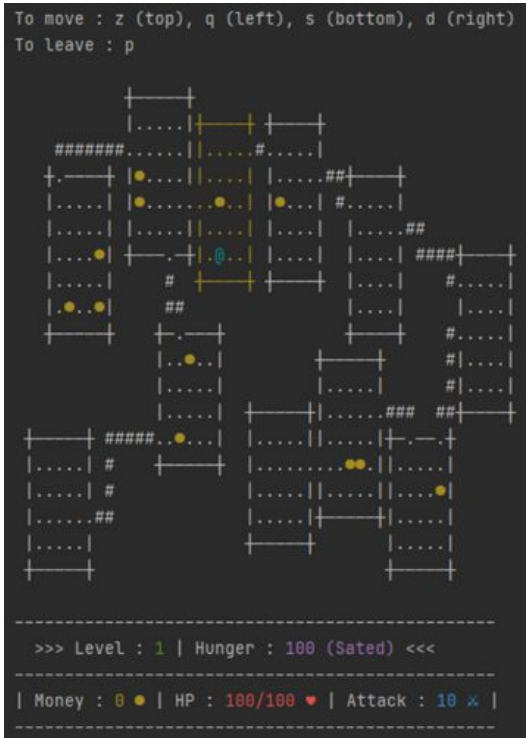
Goblin

Conditionnel

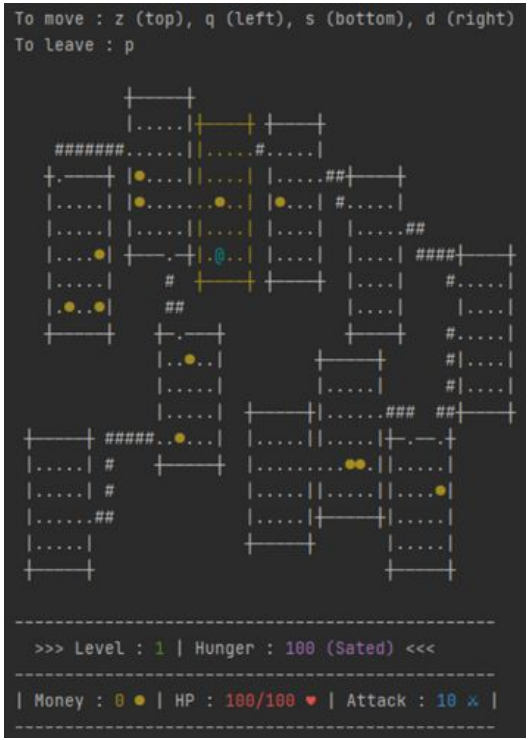
Note : State Pattern pour le cas des monstres

Partie 3
Swing

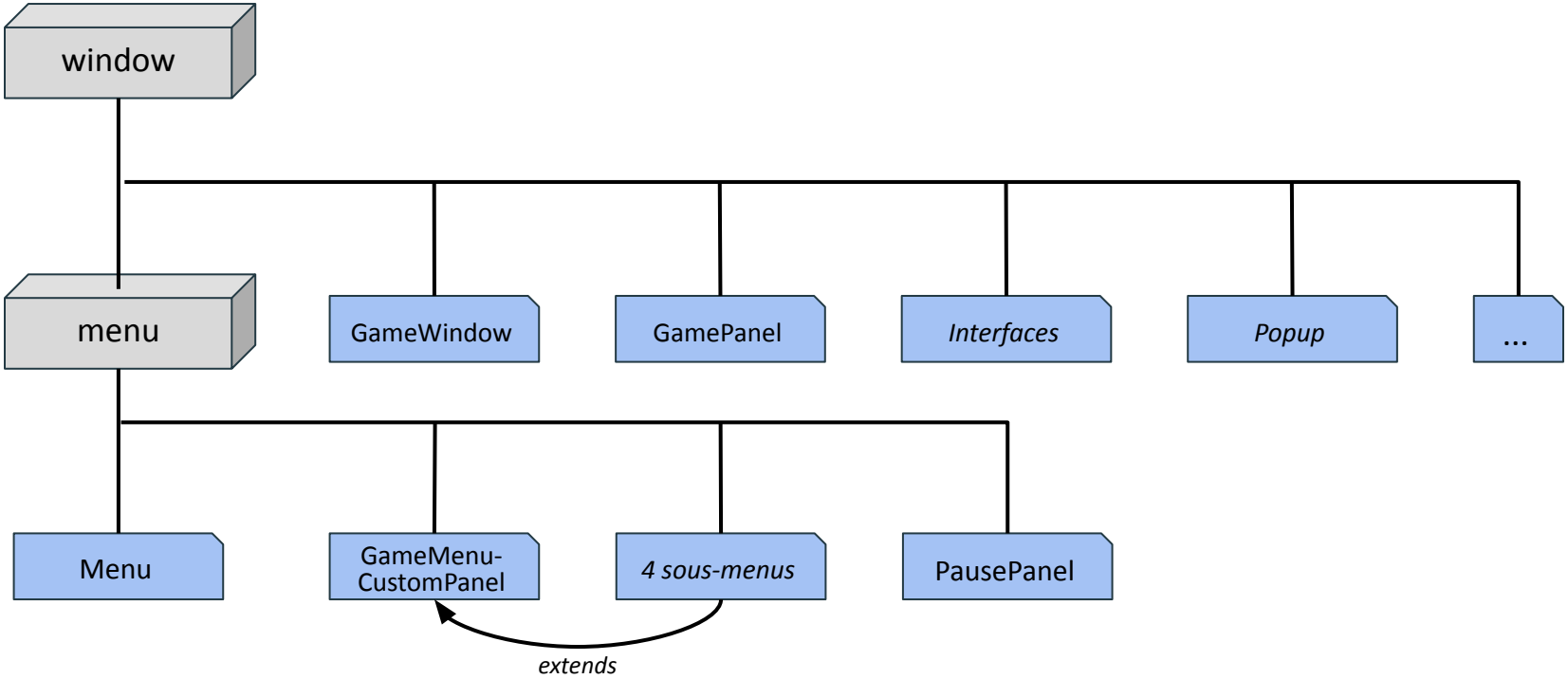
Du terminal...


To move : z (top), q (left), s (bottom), d (right)
To leave : p
>>> Level : 1 | Hunger : 100 (Sated) <<<
| Money : 0 ● | HP : 100/100 ♥ | Attack : 10 ⚔ |

Du terminal... à la fenêtre graphique



Les classes graphiques



Le menu


That Time the Hero saved the Village
Nouvelle Partie
Options
Aide
Classement
Quitter le Jeu

Classement

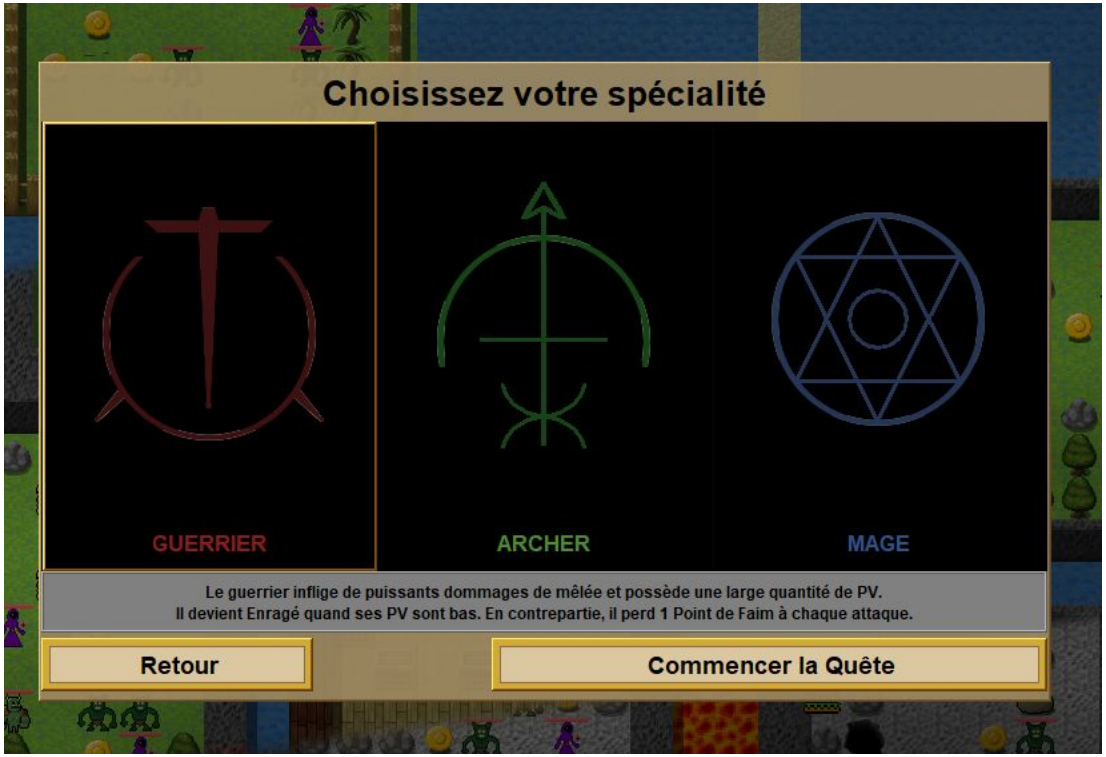

Date
Nom
Fin
Difficulté
Spécialité
Niveau
Etage
28/04/2021 Tilla Abandon Facile ARCHER 8 5
28/04/2021 Lotherad Mort par PV Facile GUERRIER 1 1
28/04/2021 Mirav Mort par PV Facile MAGE 1 1
28/04/2021 Evert Mort par famine Facile GUERRIER 2 1
Retour
Effacer

Section « aide »

Options


Options
Sélectionnez la langue
Son du jeu
Résolution d'écran
1440 x 900
— Choisissez la difficulté —
Facile
Moyen
Difficile
Cauchemar
Sans fin
Retour
Valider

Choix du personnage



La phase de jeu

Les thèmes : **Forêt**, **Îles**, **Donjon**, **Marchand** et **Boss Final**

La phase de jeu

Les thèmes : **Forêt**, **Îles**, **Donjon**, **Marchand** et **Boss Final**

Les états : **Brûlé**, **Gelé**, **Paralysé**, **Empoisonné**, **Enragé**, **Soigné** et **Invulnérable**

La phase de jeu

Les thèmes : **Forêt**, **Îles**, **Donjon**, **Marchand** et **Boss Final**

Les états : **Brûlé**, **Gelé**, **Paralysé**, **Empoisonné**, **Enragé**, **Soigné** et **Invulnérable**

… Et le **son** !

La phase de jeu

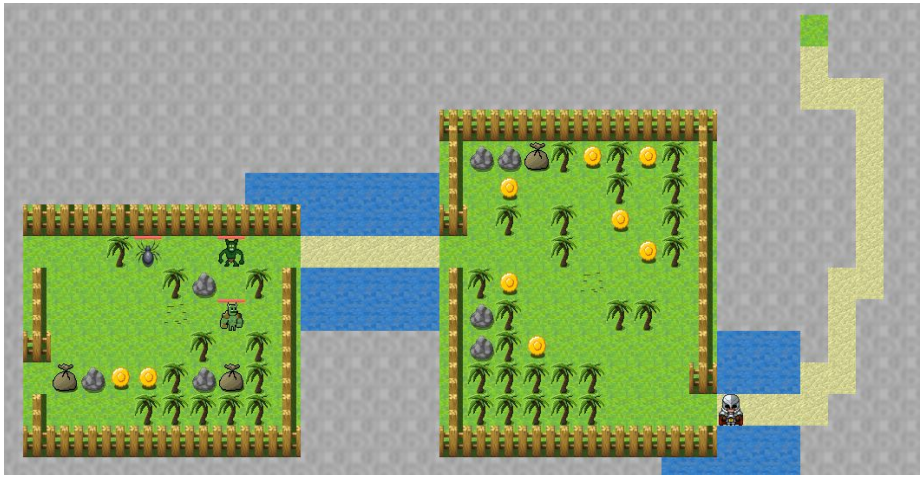
Menu Options
(en jeu)

Inventaire

Génération du brouillard

Le brouillard de guerre cache toute salle et couloir qui n’ont pas encore été visités par le héro.

Génération du brouillard



La fonction removeFog() est appelée quand le héros pénètre dans un couloir ou dans une chambre.

Génération du brouillard

Pas de brouillard dans la chambre du boss final.

Partie 4
Démo du jeu

Fin

Merci de votre attention

Lien vers la présentation sur google drive : click me