

# 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



# 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

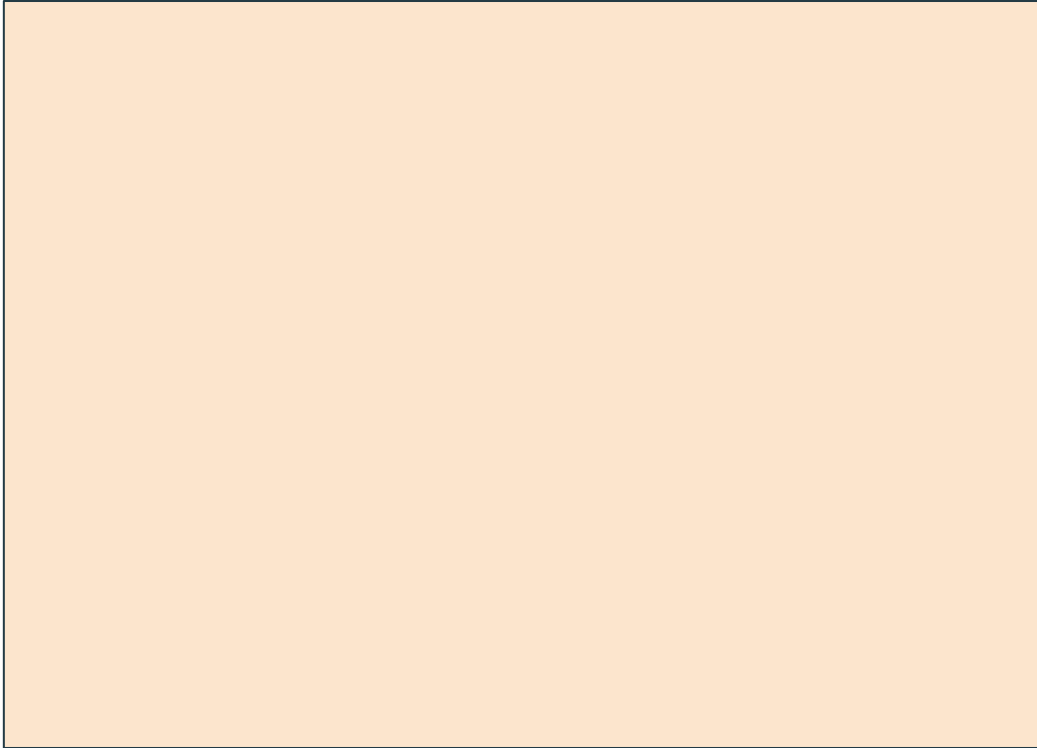
- | List<AbstractItem> items

Corridor =

- | int id

- | List<Position> positionList

# Génération des salles

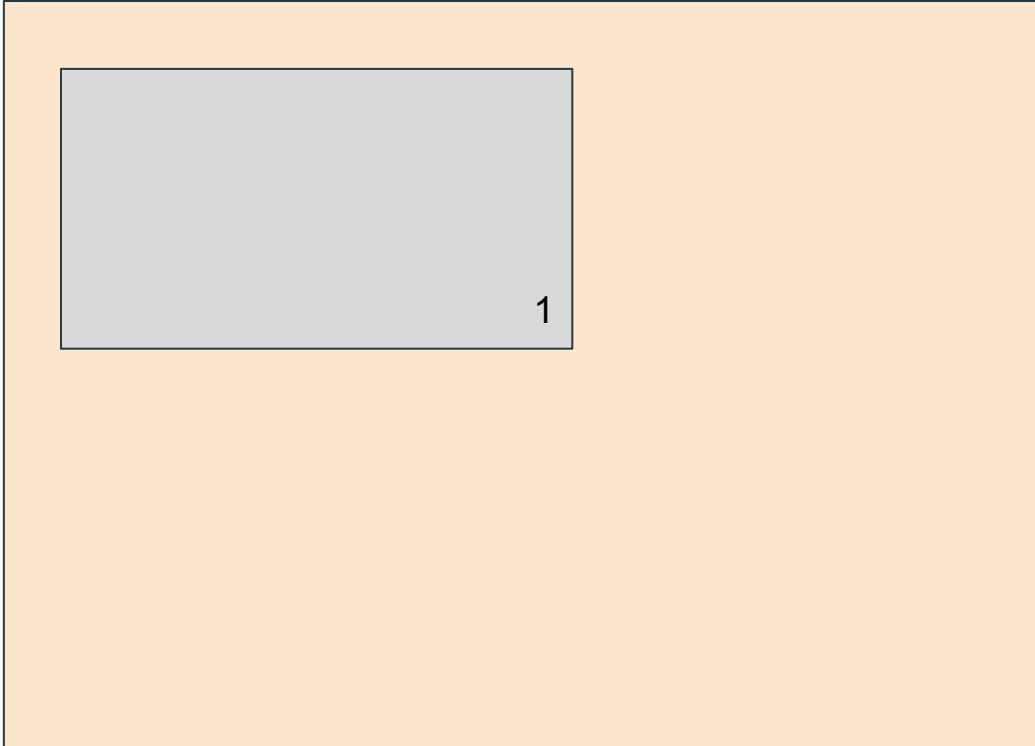


Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =

# Génération des salles

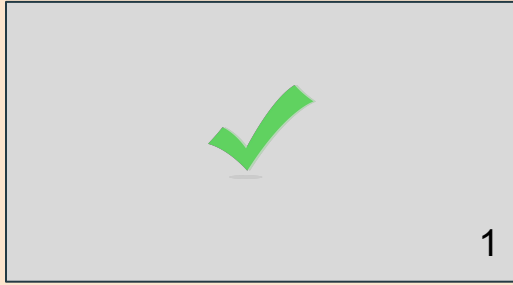


Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =

# Génération des salles



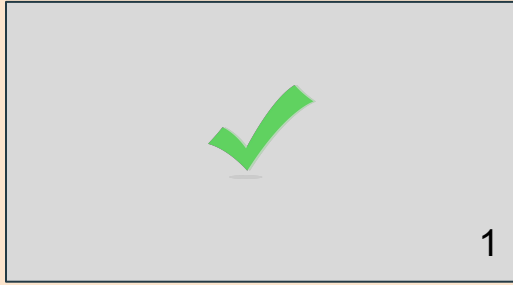
Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =



# Génération des salles



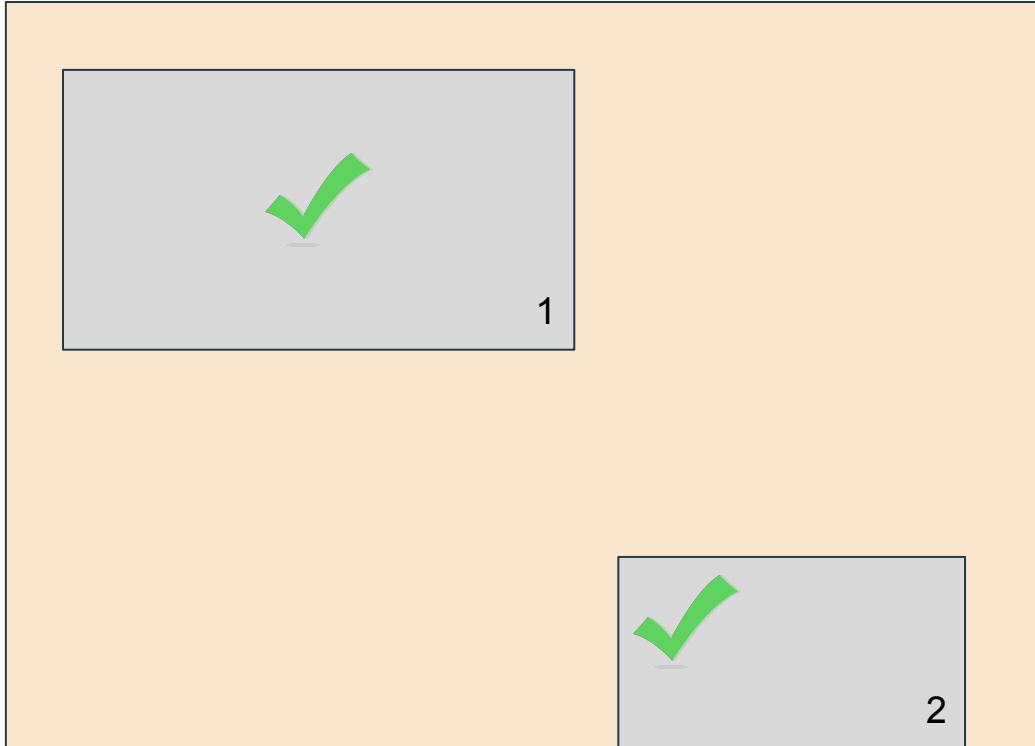
Exemple avec 6 itérations :

1. new Room()

WorldMap.rooms =



# Génération des salles



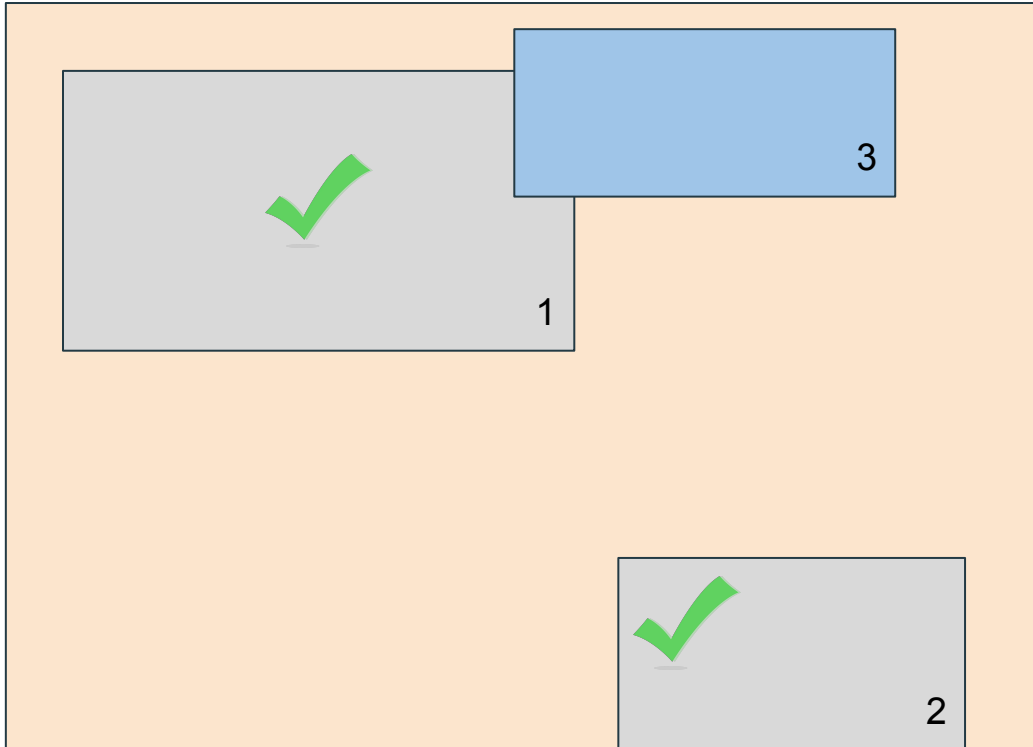
Exemple avec 6 itérations :

2. new Room()

WorldMap.rooms =



# Génération des salles



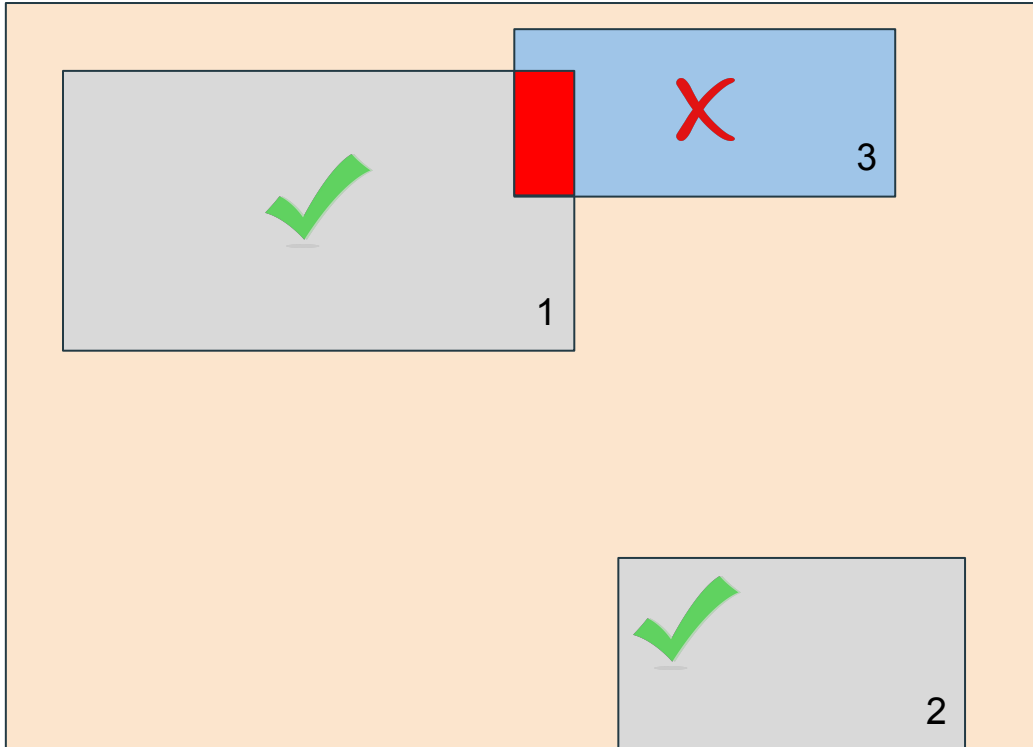
Exemple avec 6 itérations :

3. new Room()

WorldMap.rooms =



# Génération des salles



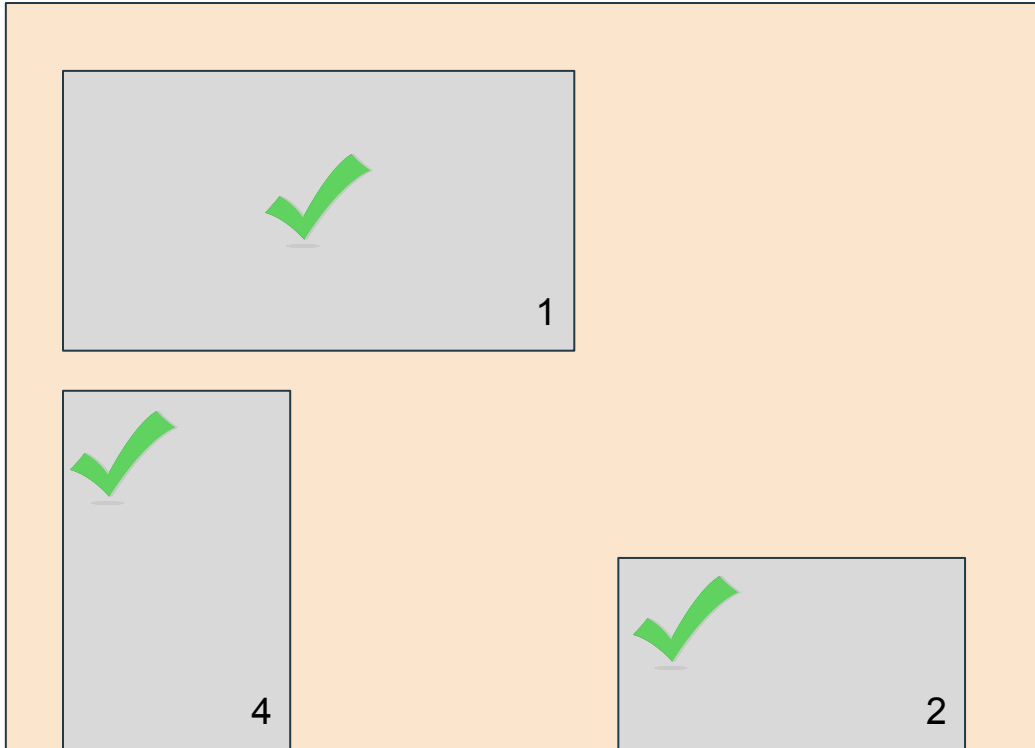
Exemple avec 6 itérations :

3. new Room()

WorldMap.rooms =



# Génération des salles



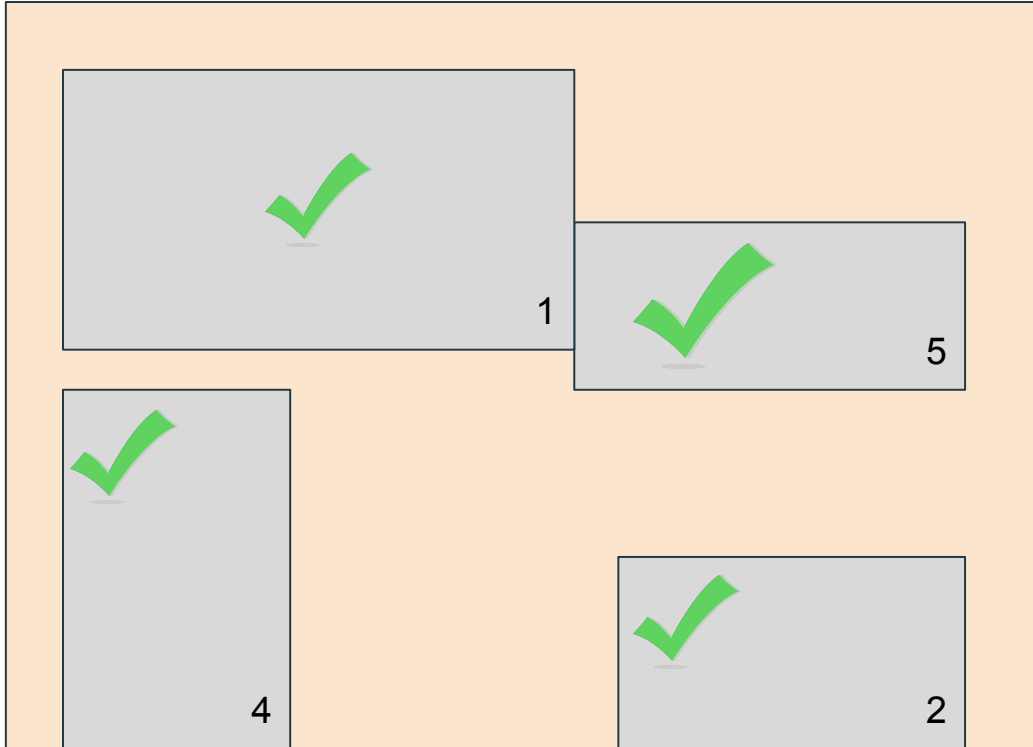
Exemple avec 6 itérations :

4. new Room()

WorldMap.rooms =



# Génération des salles



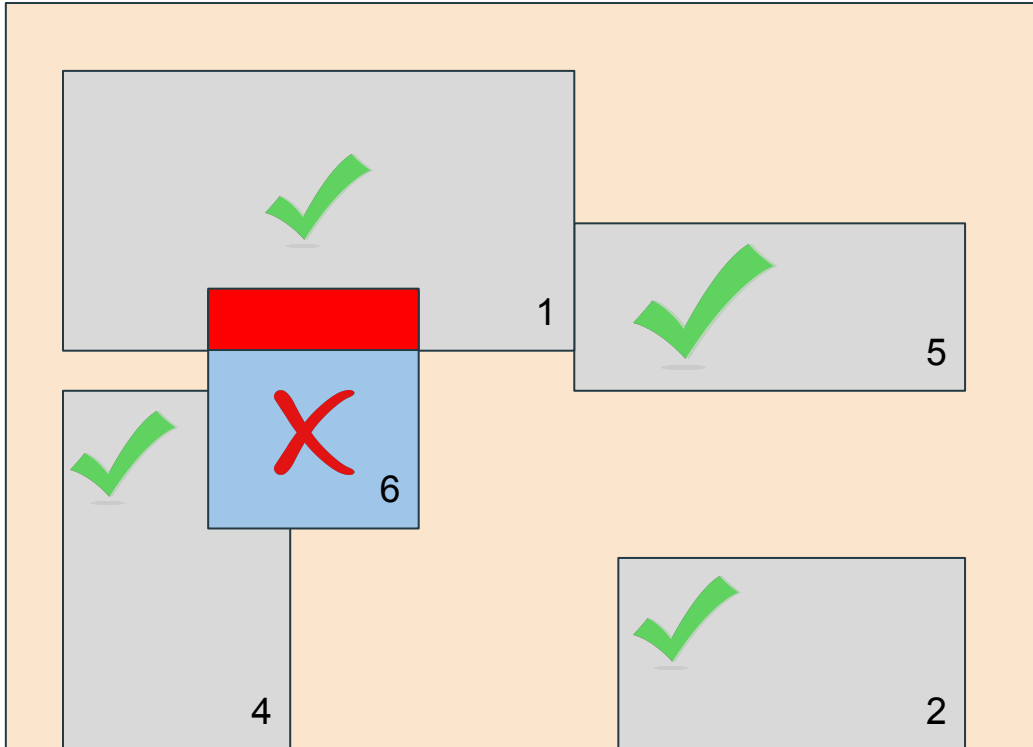
Exemple avec 6 itérations :

5. new Room()

WorldMap.rooms =



# Génération des salles



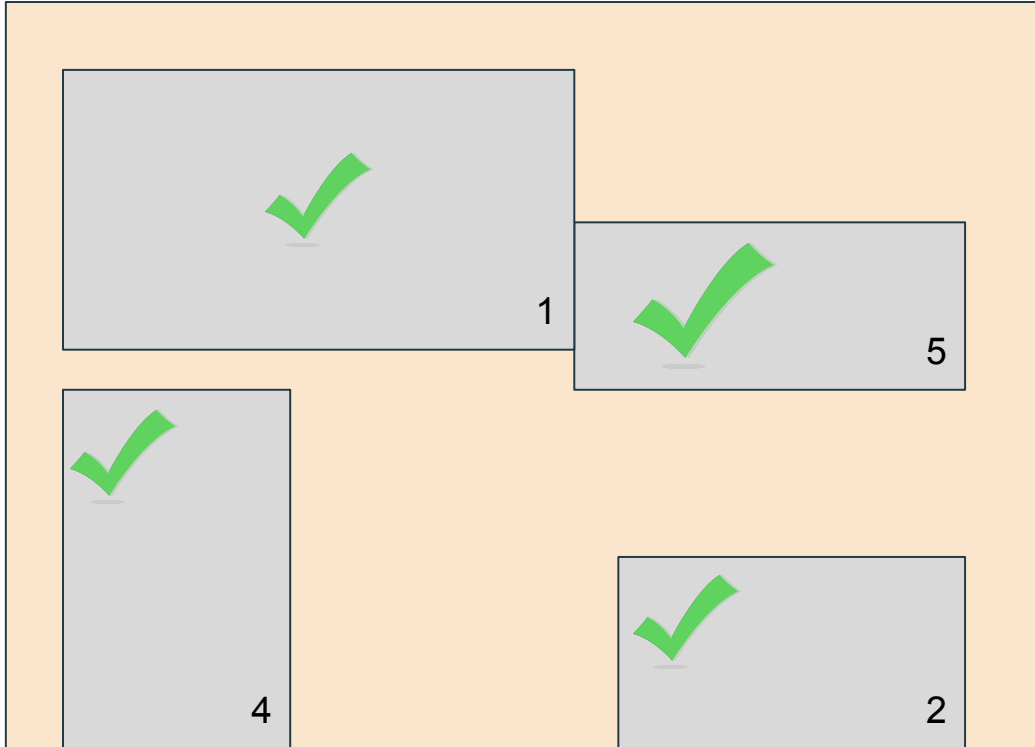
Exemple avec 6 itérations :

6. new Room()

WorldMap.rooms =



# Génération des salles

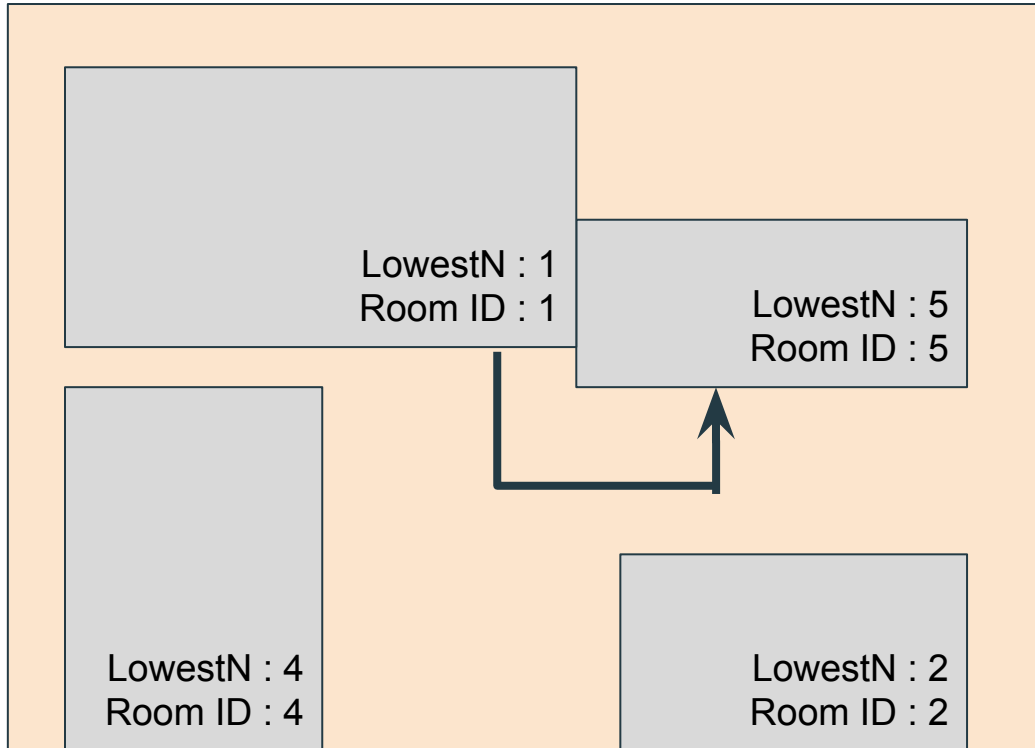


WorldMap.rooms =





# Les couloirs [Corridor]



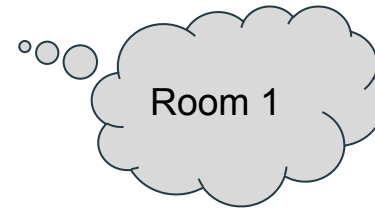
Pseudo-code :

createCorridor:

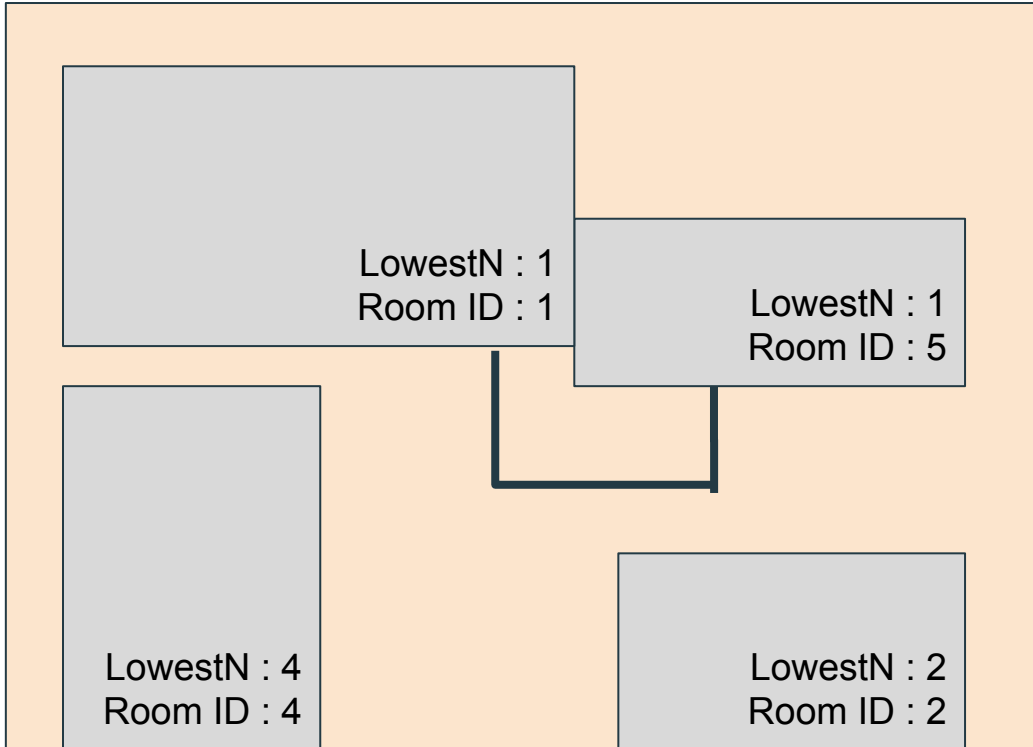
- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)



# Les couloirs [Corridor]



Pseudo-code :

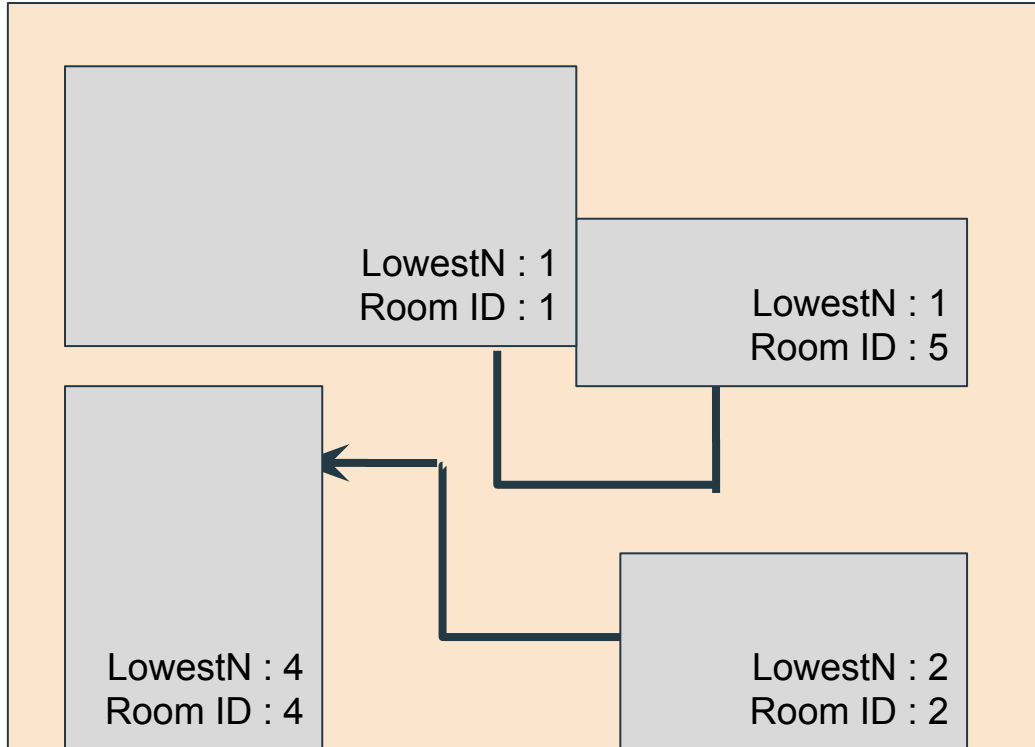
createCorridor:

- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)

# Les couloirs [Corridor]



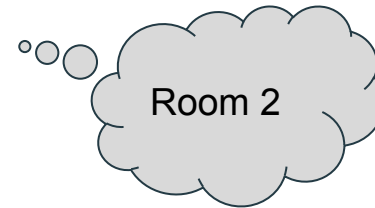
Pseudo-code :

createCorridor:

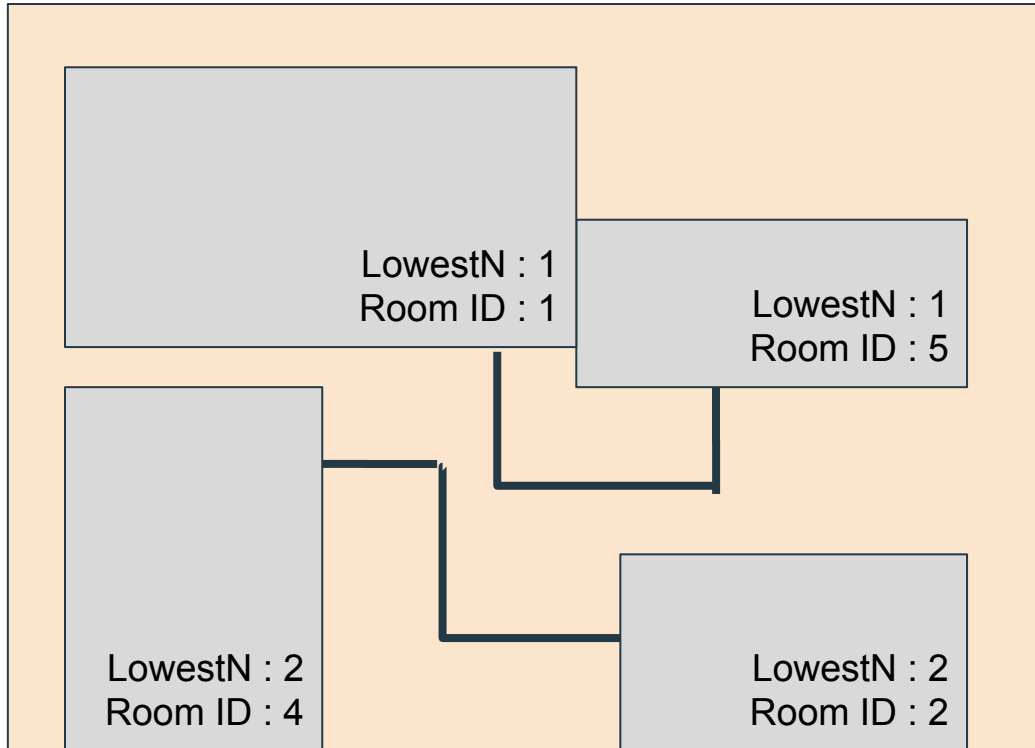
- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)



# Les couloirs [Corridor]



Pseudo-code :

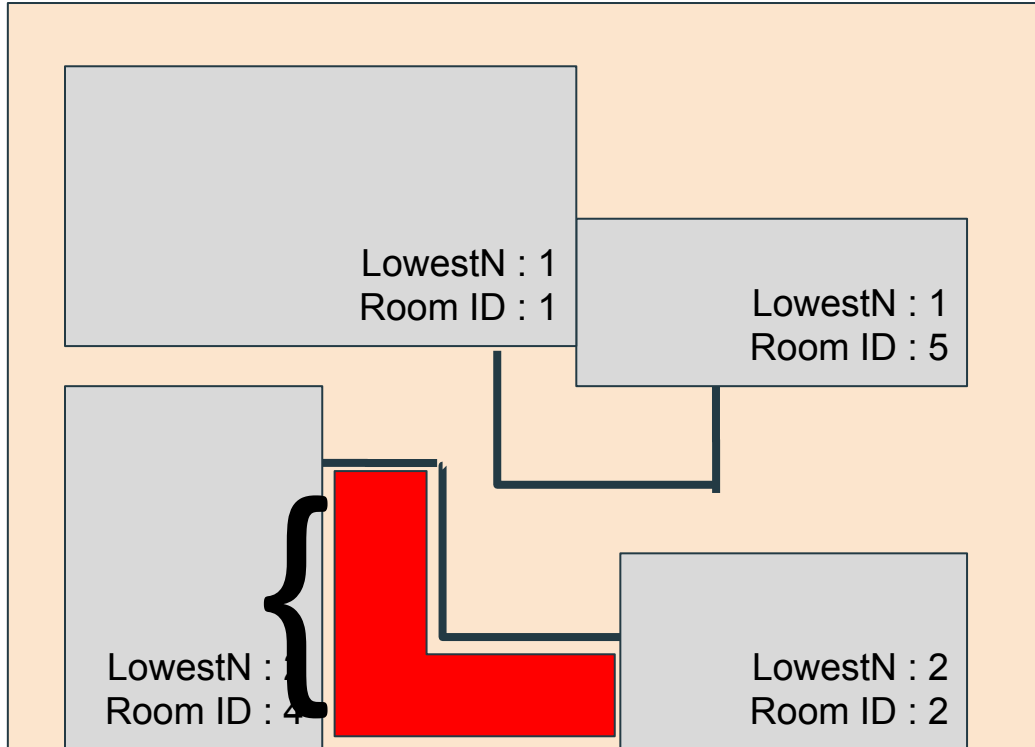
createCorridor:

- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)

# Les couloirs [Corridor]



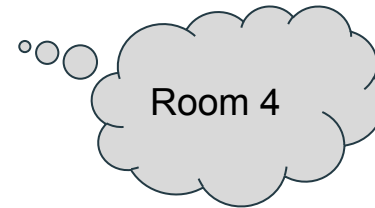
Pseudo-code :

createCorridor:

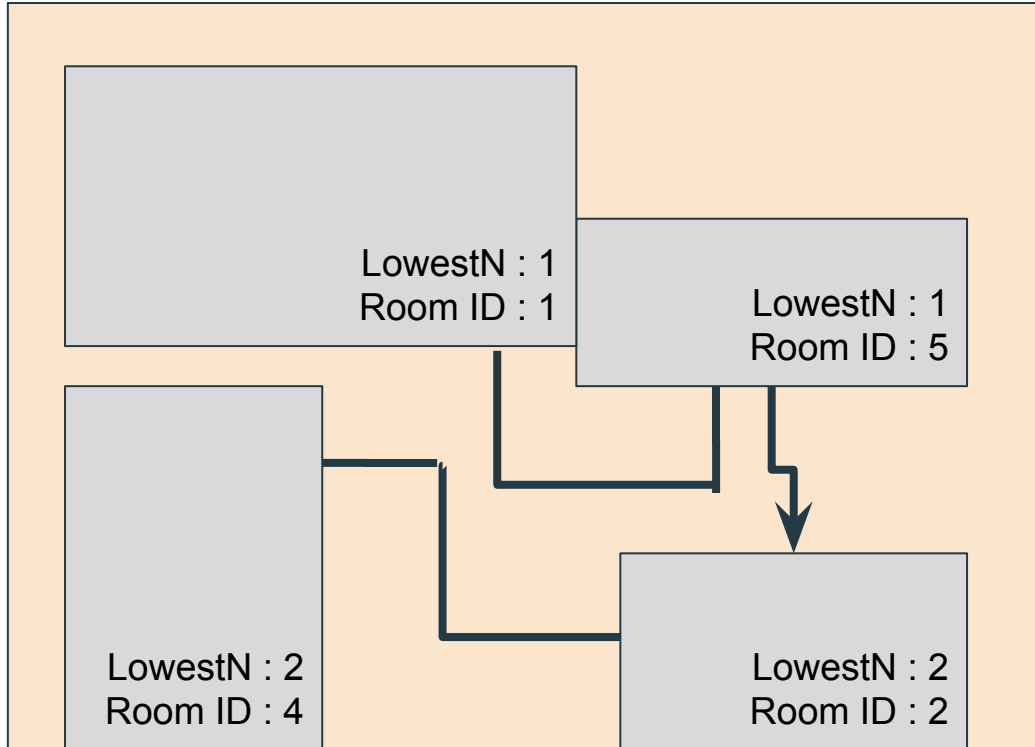
- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)



# Les couloirs [Corridor]



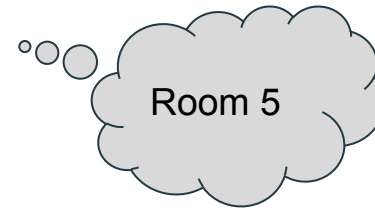
Pseudo-code :

createCorridor:

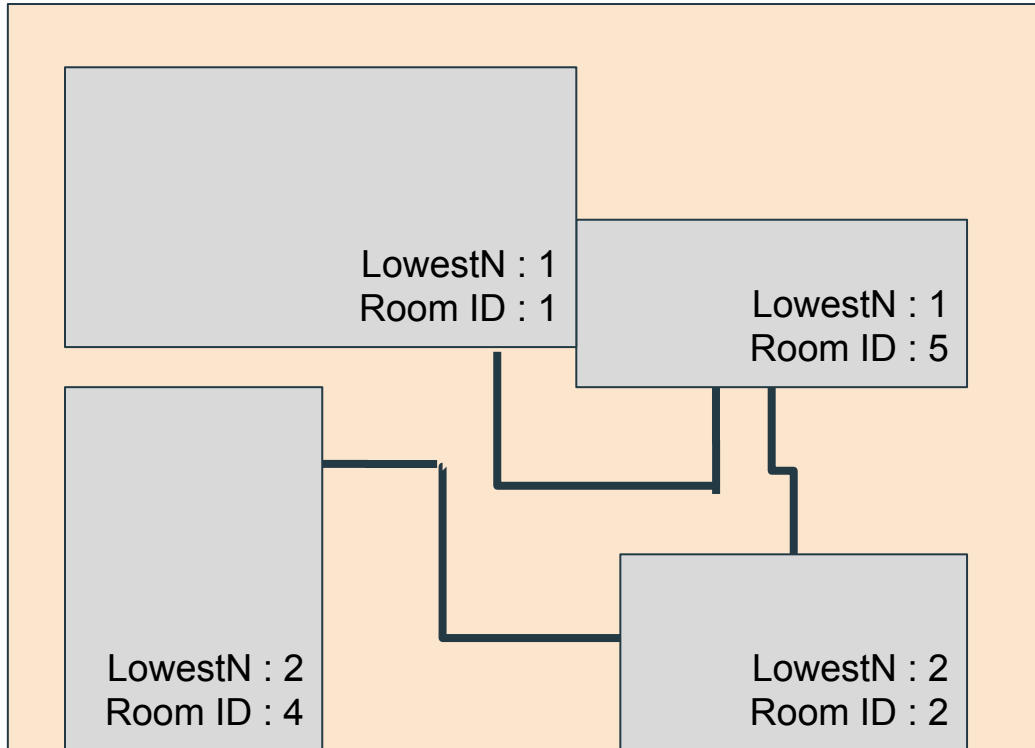
- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)



# Les couloirs [Corridor]



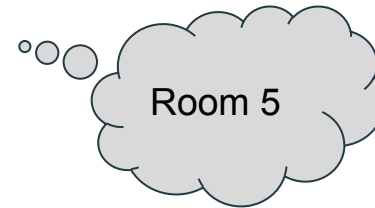
Pseudo-code :

createCorridor:

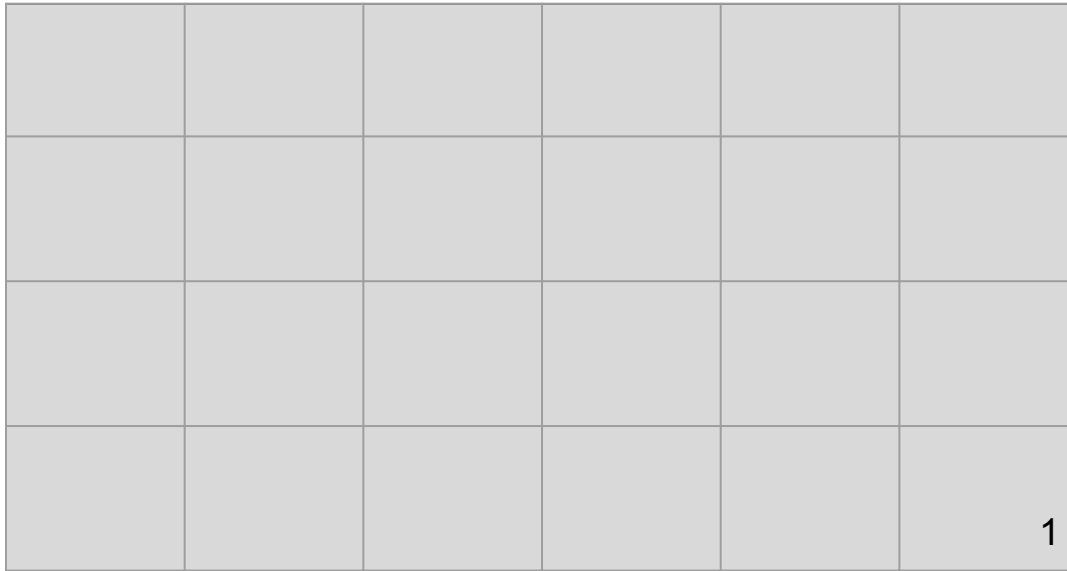
- for R in rooms:
- choose door D in R
- start BFS\_modified from D

BFS\_modified(start):

- choose a neighbor N of start
- if  $N \in R$  with different lowestN:
- create\_corridor
- set\_lowestN
- else : BFS\_modified(N)



# Emplacement des entités | items



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

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

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

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



# Emplacement des entités | items

					$I_1$
					1

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

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

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

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

# Emplacement des entités | items

			$O_2$		
					$I_1$
					1

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

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

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

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

# Emplacement des entités | items

	$O_2$		$O_2$		
$I_2$					$I_1$
		$O_2$	$I_3$		
				$O_2$	1

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

```
void putRandomEltnRoom():  
    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 = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }

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

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

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

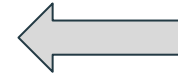
## Les obstacles

$\mathcal{M}_1$	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 = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }  
this.doors = {  $D_1, D_2, D_3$  }

## Les obstacles

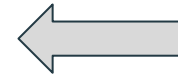
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$	O1
$I_2$	O1	$\mathcal{M}_2$	O1	O1	$I_1$
O1	O1	O2	$I_3$	O1	O1
O1	$\mathcal{M}_4$	O1	O1	O2	O1 <sub>1</sub>



this.currentItems = {  $I_1, I_2, I_3$  }  
this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }  
this.doors = {  $D_1, D_2, D_3$  }

## Les obstacles

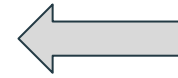
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$	O1
$I_2$	O1	$\mathcal{M}_2$	O1	O1	$I_1$
O1	O1	O2	$I_3$	O1	O1
O1	$\mathcal{M}_4$	O1	O1	O2	O1 <sub>1</sub>



this.currentItems = {  $I_1, I_2, I_3$  }  
this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }  
this.doors = {  $D_1, D_2, D_3$  }

## Les obstacles

$\mathcal{M}_1$	O2	<b>O1</b>	O2	$\mathcal{M}_3$		
$I_2$	<b>O1</b>	$\mathcal{M}_2$	<b>O1</b>		$I_1$	
<b>O1</b>	<b>O1</b>	O2	$I_3$		<b>O1</b>	
<b>O1</b>	$\mathcal{M}_4$	<b>O1</b>			<b>O1</b> <sub>1</sub>	

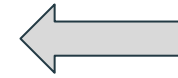


`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`



## Les obstacles

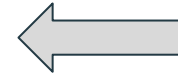
$\mathcal{M}_1$	O2	<b>O1</b>	O2	$\mathcal{M}_3$		
$I_2$	<b>O1</b>	$\mathcal{M}_2$	<b>O1</b>		$I_1$	
<b>O1</b>	<b>O1</b>	O2	$I_3$		<b>O1</b>	
<b>O1</b>	$\mathcal{M}_4$	<b>O1</b>			<b>O1</b> <sub>1</sub>	



`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`

## Les obstacles

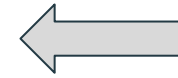
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$		
$I_2$			$\mathcal{M}_2$			$I_1$
	O1	O2	$I_3$			O1
O1	$\mathcal{M}_4$	O1				O1 <sub>1</sub>



`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`

## Les obstacles

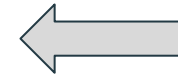
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$	
$I_2$		$\mathcal{M}_2$			$I_1$
	O1	O2	$I_3$		O1
O1	$\mathcal{M}_4$	O1			O1 <sub>1</sub>



`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`

## Les obstacles

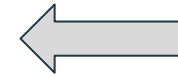
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$		
$I_2$		$\mathcal{M}_2$			$I_1$	
	O1	O2	$I_3$		O1	
O1	$\mathcal{M}_4$	O1			O1 <sub>1</sub>	



`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`

## Les obstacles

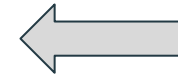
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$		
$I_2$		$\mathcal{M}_2$			$I_1$	
	O1	O2	$I_3$		O1	
O1	$\mathcal{M}_4$	O1			O1 <sub>1</sub>	



`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`

## Les obstacles

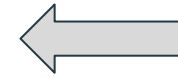
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$		
$I_2$		$\mathcal{M}_2$			$I_1$	
	O1	O2	$I_3$		O1	
O1	$\mathcal{M}_4$	O1			O1 <sub>1</sub>	



`this.currentItems = {  $I_1, I_2, I_3$  }`  
`this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }`  
`this.doors = {  $D_1, D_2, D_3$  }`

## Les obstacles

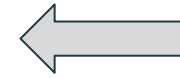
$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$		
$I_2$		$\mathcal{M}_2$			$I_1$	
	O1	O2	$I_3$		O1	
O1	$\mathcal{M}_4$	O1			O1 <sub>1</sub>	



this.currentItems = {  $I_1, I_2, I_3$  }  
this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }  
this.doors = {  $D_1, D_2, D_3$  }

## Les obstacles

$\mathcal{M}_1$	O2	O1	O2	$\mathcal{M}_3$		
$I_2$		$\mathcal{M}_2$			$I_1$	
	O1	O2	$I_3$		O1	
O1	$\mathcal{M}_4$				O1 <sub>1</sub>	



this.currentItems = {  $I_1, I_2, I_3$  }  
this.monsters = {  $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$  }  
this.doors = {  $D_1, D_2, D_3$  }





## 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

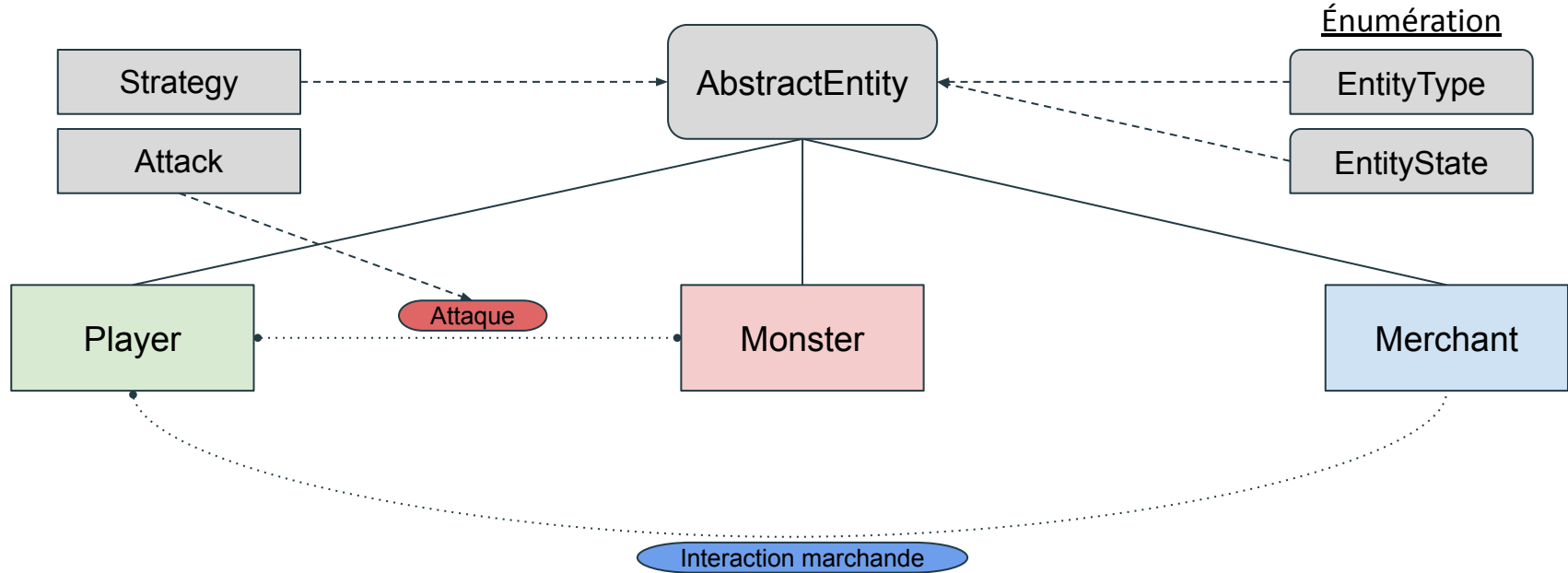
AbstractItems

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

AbstractCollectableItem

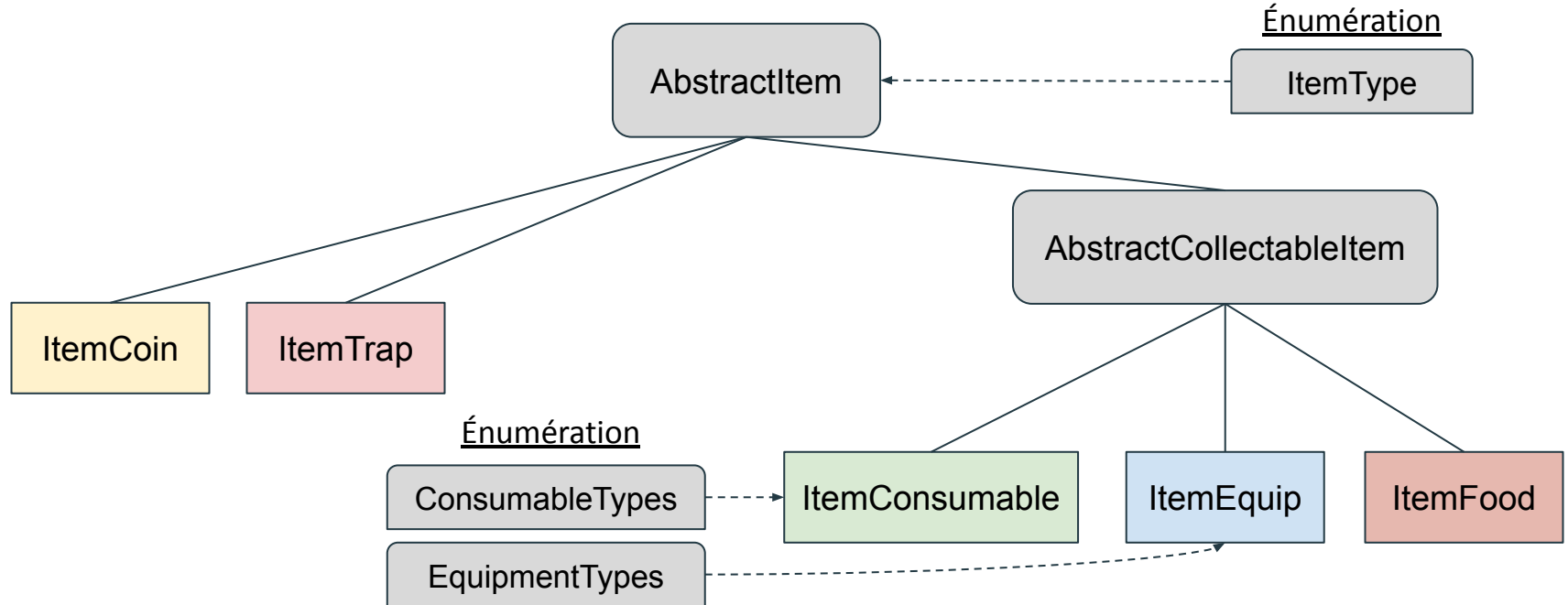
# Entité

## 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

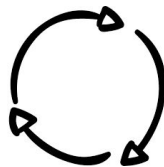
3 Factories concernant les entités/items :

- 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()*)

# Strategy Pattern

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

# 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

# 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

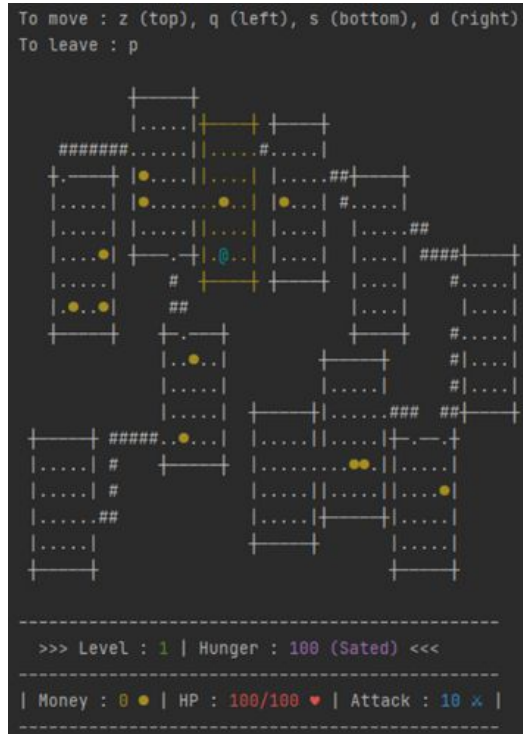
## Swing

# Du terminal...

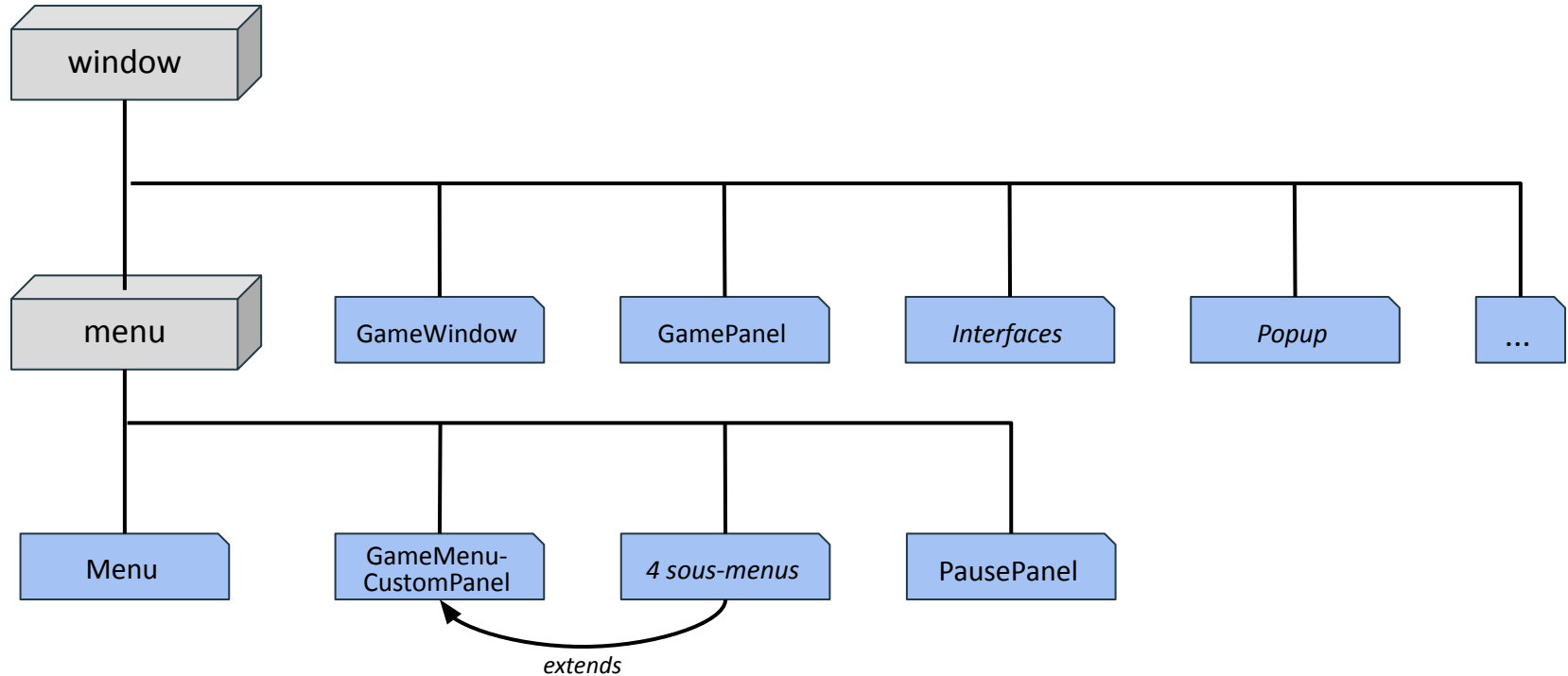
```
To move : z (top), q (left), s (bottom), d (right)
To leave : p
```

```
>>> Level : 1 | Hunger : 100 (Sated) <<<
```

# Du terminal... à la fenêtre graphique

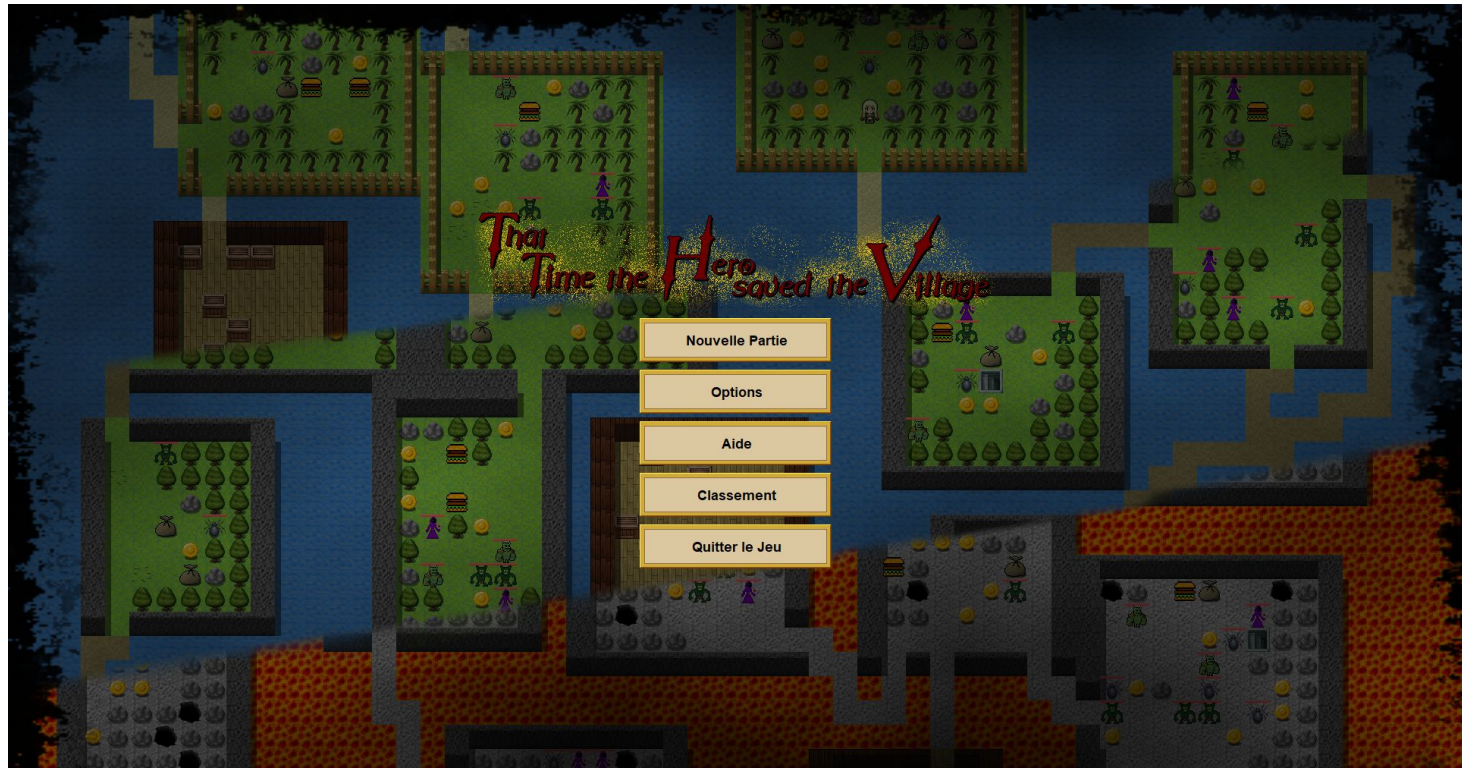


# Les classes graphiques





# Le menu



# 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

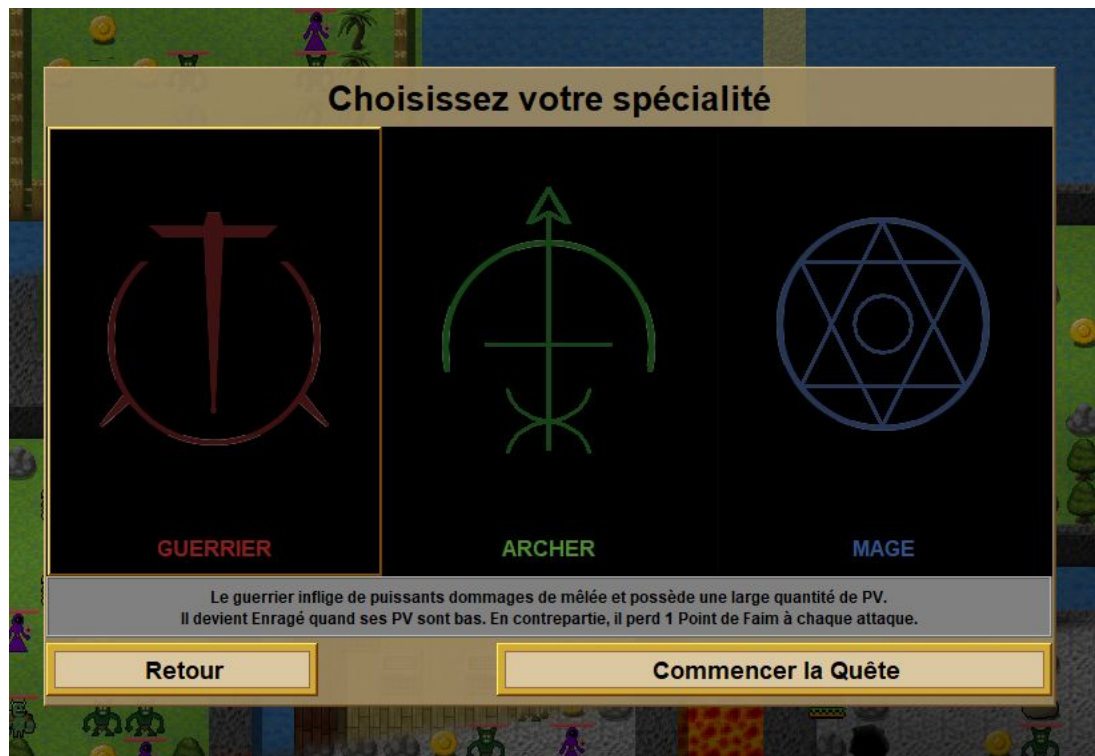
# Section « aide »



# Options



# 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



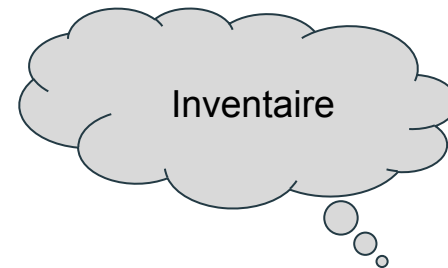
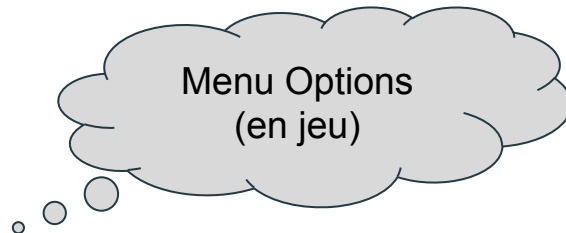
Les états : Brûlé, Gelé, Paralysé, Empoisonné, Enragé, Soigné et Invulnérable



... Et le son !



# La phase de jeu

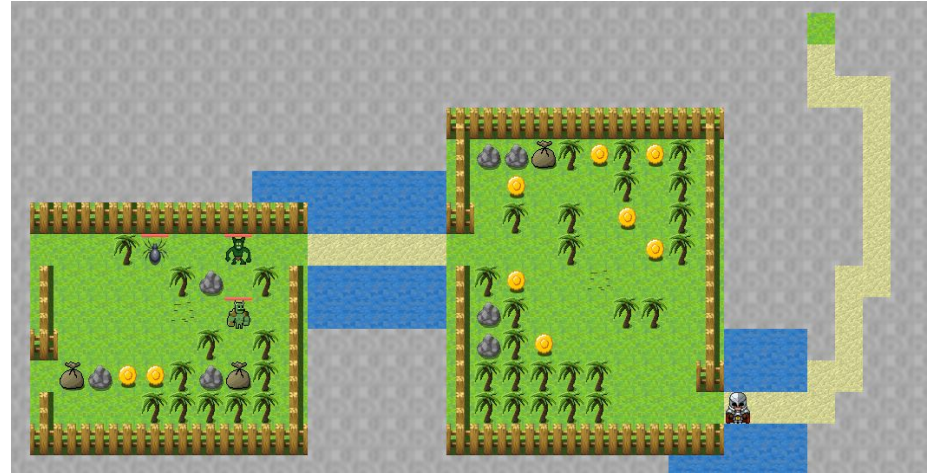


# Génération du brouillard



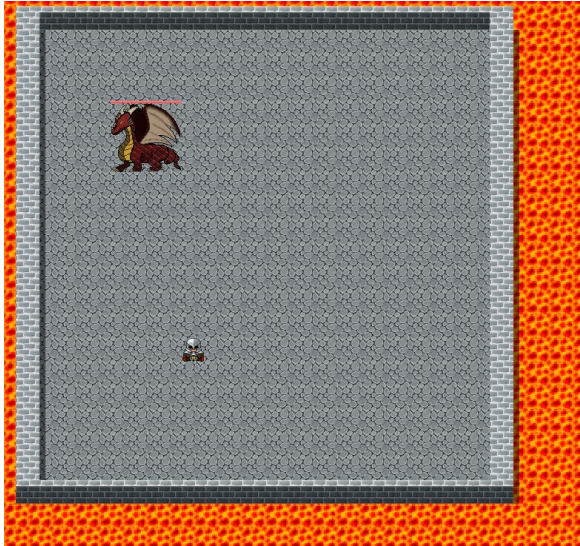
On place le héros, puis le marchand, et ensuite le brouillard de guerre.

# 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

