

# Enigma

## Rapport de projet

26 février 2020



Louka DOZ, Quentin RAMSAMY- -AGEORGES, Loïc SENECAAT, Jorys-Micke ALAIS

## Remerciements

Merci à Luc HERNANDEZ pour avoir accepté d'être notre tuteur, ainsi qu'à toute l'équipe enseignante, pour leur suivi durant ces deux années.

Remerciement spécial à Florent MADELAINE, pour nous avoir conseillé sur la structure de notre jeu, et avoir conçu un TD en relation avec notre sujet.

Les ressources ont été, majoritairement, récupérées sur itch.io. Merci à tous les auteurs de les partager !

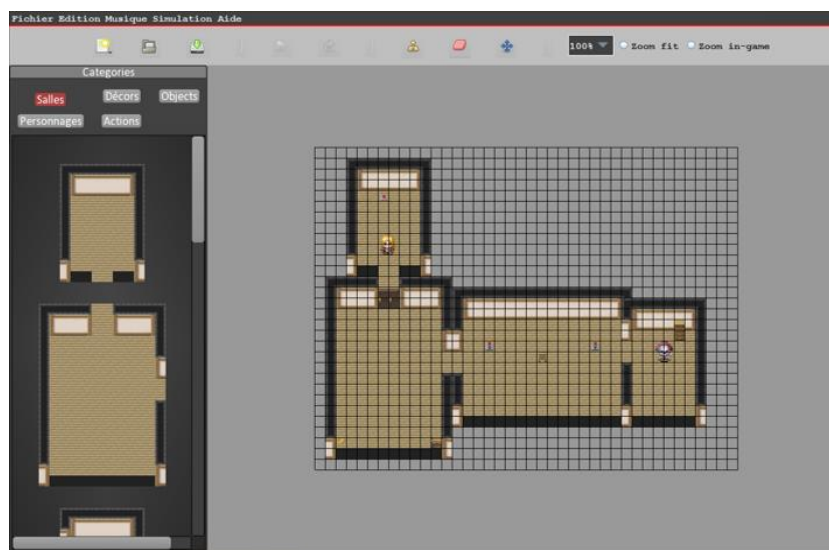
Merci à vitalzigns pour le Game Design Document (GDD) préparé.

## Synthèse

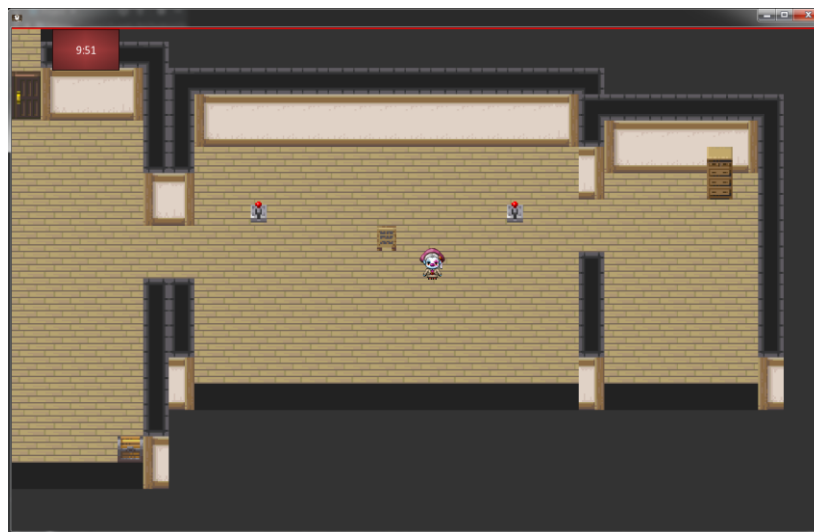
Ce rapport présente Enigma, un logiciel permettant de créer des escape games et un lanceur, permettant d'y jouer.

Le créateur met à disposition des objets préconçus tels que des bibliothèques, coffres, livres ainsi que des salles que l'utilisateur va placer pour créer son escape game. Les énigmes sont construites à partir de "morceaux" d'énigmes (des conditions pour sa résolution, et des opérations à effectuer) que nous avons préparés et qu'il ne reste qu'à assembler. (Le nombre d'énigmes différentes possibles est infini)

Le lanceur a été créé pour permettre de jouer seul ou à plusieurs, cependant, le multijoueur n'est pas encore disponible.



(Image 1 - Créateur de map)



(Image 2 - Partie en cours)

## Code source

Le code source de l'application est disponible à l'adresse suivante :  
<https://github.com/PT-2019/Enigma>

## Exécution

Note : le JRE par défaut doit être au minimum JRE10.

### .jar

Pour lancer le programme, déplacez-vous dans le dossier source à la racine du programme et exécutez la commande : `java -jar ../Enigma.jar --Dfile.encoding=UTF-8`.

### .exe

Pour lancer le programme, déplacez-vous à la racine du programme et double cliquez sur l'exécutable.

## Sommaire

1. Présentation du projet .....	7
1-1. Point de départ .....	7
1-2. Membres.....	7
1-3. Réalisation .....	8
1-4. Fonctionnalités supplémentaires .....	10
1-5. Fonctionnalités à venir .....	12
2. Présentation de l'éditeur .....	13
2-1. Map .....	13
2-2. Entités.....	17
2-3. Menu de gestion.....	21
2-4. Drag and drop .....	22
2-5. Enigmes .....	24
2-6. Musiques .....	28
2-7. Barre d'outils .....	29
2-8. Inventaire .....	40
2-9. Notifications .....	40
3. Présentation du jeu .....	41
3-1. Lancement.....	41
3-2. Collision .....	47
3-3. Correction Y .....	47
3-4. Enigmes .....	50
3-5. Joueurs .....	51
3-6. Boîtes de dialogues.....	52
3-7. Inventaire .....	53
3-8. Timer .....	54
3-9. Musique dans le jeu .....	54
4. Présentation de l'API.....	54
4-1. Pourquoi une API ?.....	54
4-2. Refonte de l'interface Swing .....	54
4-3. Utils.....	60
4-4. LibGdx.....	60
4-5. Annotations .....	60

5. Charte graphique.....	61
6. Tests .....	62
6-1. Exemples de tests.....	62
6-2. Système de Log .....	62
7. Problèmes .....	63
7-1. Migration de la gestion de maps de l'éditeur .....	63
7-2. Git .....	63
7-3. Problèmes matériels .....	63
7-4. LibGdx.....	63
8. Organisation .....	64
8-1. Classe de configuration.....	65
9. Outils et technologies.....	65
9-1. Java 11 .....	65
9-2. LibGdx 1.9.10.....	65
9-3. Tiled.....	65
9-4. Systèmes d'exploitation.....	66
9-5. Discord.....	67
9-6. GitHub.....	68
9-7. IntelliJ IDEA .....	68
9-8. JUnit 4 .....	69
9-9. Trello .....	69
9-10. TexturePacker (version LibGdx) .....	71
10. Concepts appris et utilisés.....	72
10-1. Concepts issus des enseignements scolaires.....	72
10-2. Concepts issus d'apprentissages extra-scolaires.....	83
11. Bilan .....	85
12. Crédits et ressources .....	87
13. Glossaire.....	88
14. Annexes.....	89
14-1. Classes .....	89
14-2. Guide utilisateur .....	90
14-3. Jeu.....	102
14-4. Cahier des charges .....	103

## 1. Présentation du projet

### 1-1. Point de départ

L'idée de départ était de faire un jeu. Le jeu se devait d'être suffisamment complet et sérieux afin que ce ne soit pas un projet quelconque mais une réelle démonstration de nos compétences, y compris pour notre avenir.

Avec cet objectif et l'envie de certains membres du groupe de développer une application plutôt qu'un jeu, s'est ajouté au projet un éditeur de maps ([voir glossaire](#)).

Après une période de réflexion, un jeu d'échappée game a facilement convaincu le groupe puisqu'il possède l'avantage d'avoir une grande diversité d'énigmes et donc une grande diversité de mécanismes.

Pour la création de maps, il existe déjà un logiciel nommé TILED (<https://www.mapeditor.org/>) qui a inspiré notre éditeur de maps. (Plus d'informations dans la partie Outils)

### 1-2. Membres

Le groupe est composé de quatre personnes :

- Louka DOZ, le chef de projet
- Quentin RAMSAMY-AGEORGES, le Scrum Master
- Loïc SENECAAT, le Gantt Master
- Jorys-Micke ALAIS

Outre ces rôles, toute l'équipe a participé au développement du projet.

L'équipe et le tuteur sont les clients de l'application.

Le tuteur du projet est l'enseignant Mr. Luc Hernandez de l'IUT Sénart-Fontainebleau (77 - Seine et Marne).

Ce projet a été réalisé dans le cadre des Projets tutorés de 2ème année de DUT Informatique, commencé le 03 octobre 2019. Le responsable des projets tutorés est Mr. Pierre Valarcher.

### 1-3. Réalisation

La majeure partie des demandes du cahier des charges initial, a été faite. Beaucoup de tâches ont été ajoutées pendant le développement (voir partie suivante).

L'éditeur est en grande partie fini, le jeu ne permet que des parties solos ([voir glossaire](#)) et seulement un minimum de fonctionnalités sont disponibles.

Les spécifications ont une priorité selon le fonctionnement MoSCoW (ordre de priorités (Must Should Could Would) d'une tâche, de M les tâches à haute priorité jusqu'à W les tâches optionnelles).

Editeur		
M	Création de la librairie.	Fait
M	Déplacement des entités de la librairie vers la map	Fait
M	Ajouter des énigmes aux cases.	Fait
M	Définir le début et la fin de l'escape game	Fait
M	La map est sauvegardée dans un fichier ou une base de données	Fait
M	On peut définir le contenu d'objets (livre, panneau)	Fait
C	On peut cacher des zones et définir un déclencheur pour les afficher	Editeur uniquement
C	On peut faire pivoter les salles	Non fait
C	Personnages non-joueurs peuvent vendre des objets (donc on peut définir quoi)	Non fait
C	Définir les conditions de fins personnalisées.	Fait
W	Importer et Exporter des maps en ligne	Non fait
W	Ajouter ses propres éléments (sprites, ...)	Non fait
W	Possibilité d'ajout de scénarios	Non fait
W => M	Pouvoir lancer une simulation de l'escape game pendant la création	Fait



Lanceur et Jeu		
M	On a une liste des escapes games disponibles	Fait
M	Le joueur peut créer une partie solo ou multijoueur	Fait
M	Sélection du personnage et d'un nom	Non fait
M	Consulter son inventaire, pouvoir prendre un objet en main	Non terminé
M	Si arrivée à la sortie avant la fin du temps, victoire. Si le temps est écoulé, défaite.	Fait
M	Consulter son inventaire, pouvoir prendre un objet en main	Non terminé
M	Les joueurs peuvent interagir avec tout (ou presque) les éléments du décor. Un message est affiché à chaque fois.	Fait
S	Pouvoir engager un combat avec des entités hostiles	Non fait
S	Menu des paramètres et de gestion du serveur	Non fait
W	Faire pivoter la caméra de 90°. (Graphiquement impossible)	Non
W	Jouer sur des maps en ligne	Non fait

## 1-4. Fonctionnalités supplémentaires

Des spécifications ont été ajoutées au fur et à mesure du développement, selon les retours de notre tuteur, nos envies, les besoins des futurs utilisateurs et les retours des testeurs.

Editeur		
M	Ajout de musiques et de sons	Fait
M	Ajout de la gomme	Fait
M	Exporter et Importer (Hors-ligne)	Fait
M	Pouvoir consulter la liste des énigmes d'une entité	Fait
M	Raccourcis	Fait
M	Afficher les cases bloquantes	Fait
S	Ajout de Undo et Redo	Fait
S	Ajout de la possibilité de zoomer	Fait
S	Sauvegarder-sous	Fait
S	Faire une version multi-langues	Non terminé
C	Interface propre à notre application	Fait
C	Pouvoir donner un nom aux Personnages, un nom de base est déjà attribué.	Fait
W	Guide Utilisateur interne à l'application	Fait
W	Pouvoir définir des objets comme n'étant pas bloquants	Fait

Lanceur et Jeu		
M	Pouvoir importer/Exporter des parties. (Hors-ligne)	Fait
M	Gérer la collision (avec les autres, et les décors)	Fait
M	Pouvoir changer les états des objets (bouton appuyé, bouton non appuyé...)	Fait
M	Pouvoir interagir avec les NPC (dialogue)	Fait
M	Pouvoir consulter l'inventaire des entités	Non terminé

## 1-5. Fonctionnalités à venir

Editeur	
M	Repenser la librairie pour la rendre plus flexible
S	Afficher les entités supprimées au survol de la gomme
S	Ajouter d'autres raccourcis. (CTRL+N, CTRL+O, ...)
S	Menu des paramètres dans l'éditeur (changer raccourcis, langue...)
S	Liste de toutes les énigmes de l'escape game
S	Copier des énigmes
C	Ajouter des énigmes préconstruites au menu correspondant (vide)
C	Une énigme peut avoir comme condition une autre
C	Ajouter une console dans l'éditeur
W	Zoom sur la position de la souris
W	Des sons sont ajoutés de base lorsque des actions sont faites (par exemple ouvrir une porte).

Lanceur et Jeu	
S	Implémenter l'aide des énigmes, un personnage peut par exemple donner des aides
W	Miniature des map

## 2. Présentation de l'éditeur

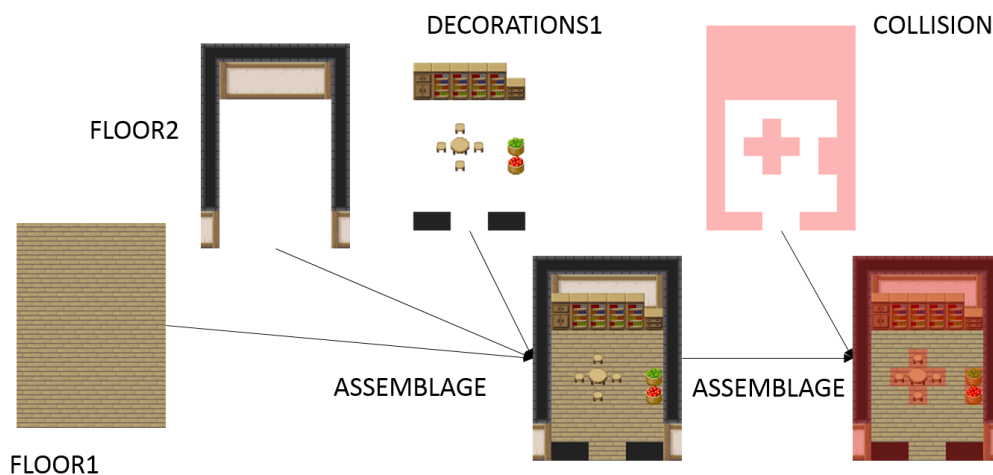
Cette partie présente tous les éléments et les fonctionnalités de l'éditeur.

### 2-1. Map

#### 2-1-1. Niveaux

Comme présenté dans la section (voir 9. Outils - qu'il est recommandé de lire avant de revenir ici), le format des map est celui du logiciel Tiled.

Dans tiled, un utilisateur va placer des tiles (on a choisi des carrés de 16 pixels car cela était la taille des ensembles de tiles que l'équipe s'est procurée) sur des niveaux, ce qui après superposition va former des salles.



(Image Map1 - Les niveaux)

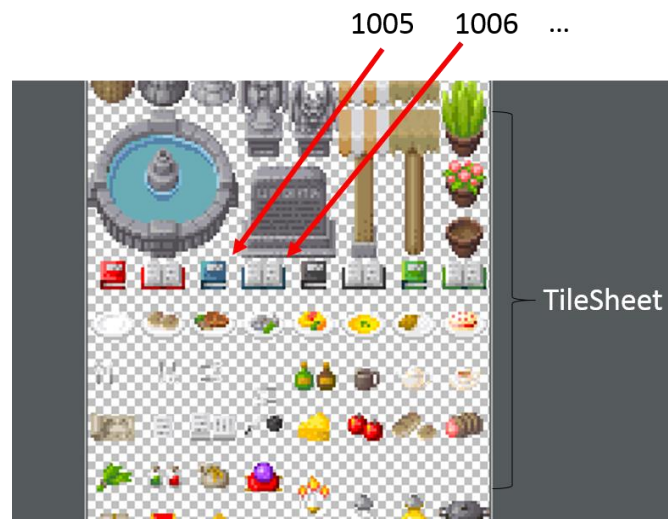
En outre, il a été ajouté un niveau nommé "COLLISION", qui contient des tiles rouges pour indiquer une case non accessible (donc on ne peut pas marcher dessus) et aucune tile pour indiquer une case accessible.

Il a été décidé de créer nous même les salles (voir partie Librairie). Elles sont conçues dans tiled, puis utilisées pour créer des map dans notre jeu. La collision est prédéfinie, pour éviter aux utilisateurs de perdre du temps ou de se tromper.

La map ne contient pas directement les textures, mais plutôt un identifiant unique qui permet de retrouver l'image (texture/tile), qui est-elle même contenue dans un ensemble d'image appelé TileSheet (image qui contient un ensemble de tiles). Par exemple le tile 1005 représente dans l'application, un livre bleu.

On rassemble les images dans une seule, pour diminuer le nombre de lectures de fichiers (dites entrées et sorties ou IO) qui coûtent cher en ressources.

## 2-1-2. Textures



(Image Map2 - TileSheet et Tiles)

Pour comprendre, imaginons que l'application contient 4 TileSheet :

- wall, contient 40 tiles
- pipo contient 2000 tiles
- collision contient 20 tiles
- extra contient 8000 tiles

Pour l'attribution des id, on commence à zéro, le tileSheet wall va avoir les id de 0 à 39, pipo de 40 à 2039, etc... Ce sont ces valeurs là que la map va conserver.

On sait que la texture fait LARGEUR pixels (par exemple pipo fait 128 pixels) et qu'il y a NB tiles (pipo contient 2000) tiles, et que chaque tile fait une dimension de 16 pixels, alors on découpe les textures comme suit :

- On déduit le nombre de tile dans une ligne en faisant  $LARGEUR/16$ 
  - Ce qui donne, pour pipo,  $128/16 = 8$  textures par ligne
- Sachant qu'il y a 8 textures par ligne et NB textures au total, alors il y a  $NB/8$  lignes
  - Ce qui donne, pour pipo,  $2000/8 = 250$  lignes

On peut ensuite récupérer l'image correspondante depuis son id.

Pour plus de confort, car cela suppose que le développeur sache que 1005 correspond au tile qui représente un livre bleu, alors, les textures ont été indexés dans un fichier .atlas à l'aide de l'utilitaire de la LibGdx (voir outils, Texture Packer (LibGdx) ) pour récupérer une texture depuis un nom.

### 2-1-3. Sauvegarde et lecture

La map est sauvegardée dans un fichier xml avec l'extension tmx car la LibGdx utilise ce format de fichier. L'API DOM de Java, a été utilisée : le principe est de charger en mémoire un arbre représentant les balises de la map puis l'API se charge de créer le fichier xml. L'utilisation de cette API au lieu de celle SAX, car elle consomme moins de mémoire, est plus facilement accessible et la taille des maps est moins importante. A titre d'exemple une map de dimension 75x75 avec 48 entités ne fait seulement que 98 ko.

Toutes les maps sont sauvegardées dans le même dossier, l'utilisateur ne peut pas choisir le dossier.

La sauvegarde s'organise de cette manière :

Toutes les informations qui décrivent la map sont d'abord écrites, les dimensions, la taille en pixel d'une case par exemple.

Ensuite des balises indiquent le chemin des textures et gère les id. Toutes les maps ont accès aux mêmes textures.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE map SYSTEM "../../assets/map/mapDtd.dtd">
<map height="30" infinite="0" nextlayerid="1" nextobjectid="3" orientation="orthogonal" renderorder="right-to-left">
  <tilesheet columns="8" firstgid="1" name="0" tilecount="48" tileheight="16" tilewidth="16">
    <image height="96" source="../../map/tiles/ground/dungeon1.png" width="128"/>
  </tilesheet>
  <tilesheet columns="8" firstgid="49" name="1" tilecount="1992" tileheight="16" tilewidth="16">
    <image height="3984" source="../../map/tiles/pippo.png" width="128"/>
  </tilesheet>
  <tilesheet columns="8" firstgid="2041" name="2" tilecount="64" tileheight="16" tilewidth="16">
    <image height="128" source="../../map/collision.png" width="128"/>
  </tilesheet>
  <tilesheet columns="80" firstgid="2105" name="3" tilecount="6400" tileheight="16" tilewidth="16">
    <image height="1280" source="../../map/Player.png" width="1280"/>
  </tilesheet>
  <!-- ... -->
</map>
```

(Image SauvegardeMap.1 - Textures)

La prochaine étape est de sauvegarder tous les niveaux de la map, pour chaque case d'un niveau. Chaque chiffre représente un id d'une texture. Le niveau collision fonctionne différemment, lorsqu'un id est égal à 2041, cela veut dire que c'est une case bloquante, donc le joueur ne pourra pas marcher dessus.

[illegible]

(Image SauvegardeMap.2 - Sauvegarde d'un Niveau)

Un fichier Dtd permet de spécifier l'organisation de notre fichier xml à l'aide de contraintes, par exemple : une balise map doit obligatoirement avoir un attribut "width". Ce fichier est utile lors du développement et des mises à jour de la sauvegarde car il affiche une erreur si le fichier est au mauvais format.



## 2-2. Entités

### 2-2-1. Présentation

On considère comme entité, tout élément de notre jeu, soit les salles, les meubles, les objets et les personnages. Les entités sont rangées en différentes catégories qui ont chacune leur propre fonctionnement :

- Activable, (bouton, levier, commutateur.)
- Contenu, représente les objets qui ont du texte (livre...)
- Conteneur, entités qui possède d'autres entités (coffre, bibliothèque...)
- Verrouillable, entités qui peuvent être verrouillé (porte, coffre...)

Il y a 3 types d'entités particuliers :

- Player (représentant un joueur humain)
- Monster (représentant une simple entité vivante)
- NPC (une simple entité vivante qui peut avoir un inventaire)

Ces entités peuvent bouger sur la map, elles nécessitent donc un traitement avancé en termes de sauvegarde (sauvegarde des sprites).

### 2-2-2. Librairie

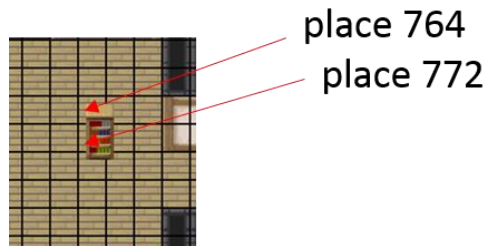
La librairie est un ensemble de fichiers json, chacun contenant un tableau d'entités.

```
{
  className: common.entities.items.Library
  path: assets/map/datas/library.png
  hover: une bibliothèque
  category: DECORS
  height: 2
  tiles: {
    FLOOR2:
    [
      764,
      772
    ]
    COLLISION:
    [
      2041,
      2041
    ]
  }
}
```

(Image Librairie.1 - Une entité de la librairie)

Les entités contiennent toutes la classe Java (attribut ([voir glossaire](#)) « className ») à laquelle elles correspondent, pour permettre leur création.

Elles contiennent un chemin vers le fichier .png qui est celui qui sera utilisé pour les afficher dans le menu de la librairie. Alternativement, elles possèdent leurs tiles, ce qui permet de placer ces valeurs sur la map, pour afficher l'image correspondante. (L'image de l'entité est ce qui est déplacé lors du maintien du clic, puis au relâchement, ces tiles sont enregistrées sur la map. L'image est supprimée. La map se met à jour environ 60 fois par secondes donc affiche automatiquement les nouvelles tiles).



(Image Librairie.2 - Une entité de la librairie)

Chaque entité contient également un champ "hover" qui sera affiché au survol de l'élément dans la librairie.

Elles peuvent contenir des attributs "height" sa hauteur, ou "width" sa largeur, qui permettent d'accéder rapidement aux informations sur ses dimensions. La valeur de base de ces champs est 1.

Enfin, elles contiennent la catégorie dans laquelle elles seront affichées.



(Image Librairie.3 - Catégories)

## 2-2-3. Sauvegarde

On sauvegarde les entités présentes sur la map dans le fichier de la map, les entités sont mises dans une balise xml "object". On demande à chaque entité, quelles informations elle veut sauvegarder et elle renvoi une liste de valeurs pour la propriété "name" et leur contenu correspondant.

```
<object height="1.0" id="6" name="entity" width="1.0" x="16.0" y="12.0">
  <properties>
    <property name="className" value="common.entities.items.Switch"/>
    <property name="ACTIVATED" value="false"/>
  </properties>
</object>
```

(Image SauvegardeEntité.3 - Sauvegarde d'une entité)

Le chargement des niveaux et des textures est fait par la Libgdx qui est capable de fournir une map à partir d'un fichier tmx. La libgdx va fournir les entités sous formes d'objet (voir glossaire) "MapProperties" avec comme attributs des chaînes de caractères (ex : "Activated", "className"...). En fonction du className (voir ci-dessus) on va pouvoir créer l'entité correspondante. Enfin chaque entité à un parseur qui lui permet de s'initialiser à partir d'un objet "MapProperties" donc chaque entité reçoit toutes les propriétés dans la balise properties et peut instancier ses variables en fonction des valeurs trouvées.

Il y a 3 types d'entités qui ont un mécanisme supplémentaire :

- Player
- Monster
- NPC

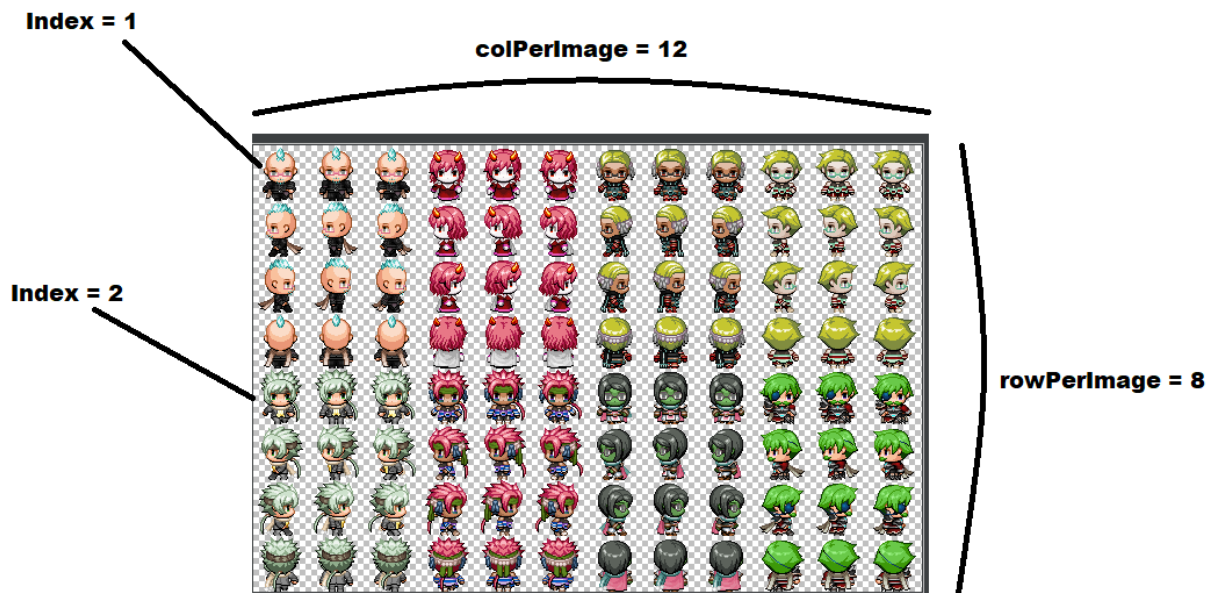
Ces entités ont besoin de "sprite" (voir l'image SauvegardeEntité.5, il s'agit d'une image représentant un personnage dans une direction. L'ensemble est appelé SpriteSheet) pour être affichées car elles peuvent se déplacer sur la map. Il y a dans leur propriété un chemin d'accès vers un fichier Json qui permet de récupérer les sprites.

```
{
  name: Jean,
  path: "assets/entities/players/plb.png"
  cols: 3
  rows: 4
  colPerImage: 12
  rowPerImage: 8
  index: 4
  duration: 0.3f
},
```

(Image SauvegardeEntité.4 - Json)

Dans un fichier Json, il y a plusieurs personnages représentés, le nom sert à les différencier. Il y a le chemin vers les sprites. Les attributs "cols" et "rows" représentent le nombre de sprites que nous devons charger. Duration représente la durée en milliseconde pour afficher un sprite à l'écran.

Les autres attributs servent à trouver le sprite dans l'image. Les images que nous utilisons sont organisé comme ceci :



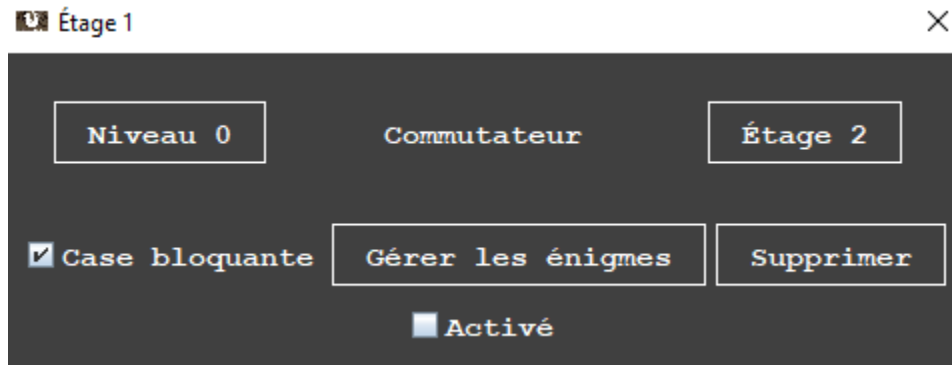
(Image SauvegardeEntité.5 - Fichier des sprites ou SpriteSheet)

## 2-3. Menu de gestion

Pour gérer les objets du jeu l'éditeur a un menu est spécialisé lorsqu'on fait un clic gauche sur une case. Ce menu permet de se balader entre les différents niveaux de la map et de faire des actions sur les entités présentes.

Lorsqu'il y a une entité sur la case et le niveau l'éditeur propose différentes options en fonction du type de l'entité. C'est sur cette interface que l'utilisateur va pouvoir définir l'état des entités, les dialogues etc...

Par exemple il est possible d'activer ou désactiver un commutateur :



(Image MenuGestion.1 - Commutateur)

Si c'est une bibliothèque, l'éditeur proposera d'autres options :



(Image MenuGestion.2 - Bibliothèque)

Au moment du clic, on regarde l'entité la plus intéressante sur la case (un livre est plus intéressant qu'une salle sinon il n'y aurait pas de raison de cliquer explicitement sur cette case) et il lui ait demandé quels sont ses types. Des menus seront ajoutés en fonction de sa réponse :

- Si c'est un objet verrouillable, on affiche un moyen de choisir si elle est fermée ou non
- Si c'est un objet pouvant avoir un nom, alors on affiche un moyen le choisir
- Une entité peut avoir plusieurs types. Chaque bouton (supprimer, ajouter des objets...) contient le ou les types que l'entité doit avoir pour qu'il s'affiche. Tous les boutons sont parcourus et ceux pour lesquels cette condition est vraie sont affichés.

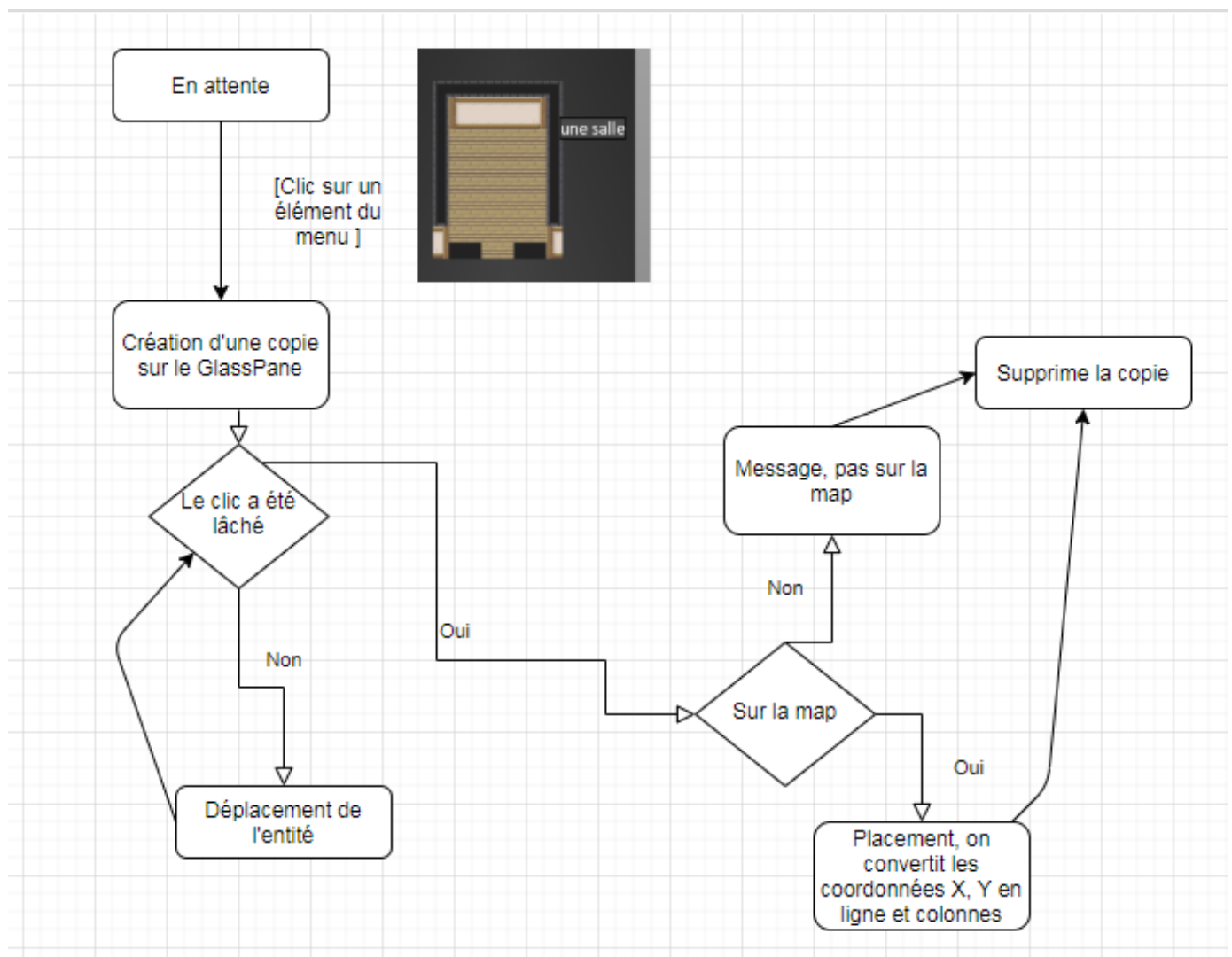
## 2-4. Drag and drop

Le drag and drop a été fait presque en intégralité, car la première version, avec les classes de SWING ne marchait pas, sous Windows, avec la LibGdx.



(Image DragAndDrop.1 : Niveaux de la map)

Le concept est le suivant, les entités sont disposées dans la librairie. Lorsque l'on clique sur l'une d'entre elles, alors elle se duplique, la copie de l'entité est créée dans un niveau supérieur (comme le GlassPane en SWING). Ce niveau est supérieur à tous les autres donc l'entité apparaît toujours au premier plan. Lorsque le clic est lâché, si l'entité est sur la map, alors on la place, sinon elle est supprimée et un message est affiché.



(Image DragAndDrop.2 : DragAndDrop)

## 2-5. Enigmes

L'équipe a choisi de définir une énigme comme des conditions qui, lorsqu'elle sont remplies, provoquent une série d'opérations.

```
----- Conditions -----  
[Doit être activé: Commutateur (id=6) ]  
----- Opérations -----  
  [Déverrouille: Porte (id=12) ]  
  [Déverrouille: Porte (id=9) ]  
[Jouer un son: avec Musique ouverture porte (id=19) ]  
(Image Enigme.1 - Exemple d'une énigme)
```

On peut lire l'énigme (image Enigme.1) comme ceci : "Si le commutateur (dit 6) est activé, alors les portes dites 12 et 9 sont déverrouillées et je joue un son d'une porte qui s'ouvre". (Les nombres 6, 12, 9 sont des identifiants permettant retrouver les objets choisis).

### 2-5-1. Conditions

Les conditions ont pour but de vérifier l'état d'une caractéristique d'un objet de type GameObject ([voir annexe](#)) au cours du jeu. Cela va de : vérifier si le joueur possède tel objet dans son inventaire, à vérifier si tel levier est activé par exemple.

A chaque appel de l'énigme et tant que celle-ci n'est pas complétée, les conditions sont de nouveau vérifiées. De cette manière, il est assuré qu'un état n'a pas changé entre deux appels de l'énigme.

Les conditions disponibles sont :

- Un objet doit être activé
- Le joueur doit donner une réponse
- Une salle doit être découverte/non découverte
- Avoir l'objet dans ses mains/inventaire



## 2-5-2. Opérations

Les opérations ont pour objectif de réaliser un événement au cours du jeu. Par exemple, donner un objet au joueur ou déverrouiller une porte.

Contrairement aux conditions, les opérations ne sont relancées, lors d'un nouvel appel de l'énigme, uniquement si l'événement n'a pas pu se lancer pour une quelconque raison, lors du premier appel de l'énigme. Donc les opérations savent si elles ont pu effectuer leur travail ou non.

Les opérations disponibles sont :

- Donne un objet à l'utilisateur
- Invoque une entité
- Affiche/Cache une salle
- Déverrouille un objet
- Joue un son
- Change la musique ambiante
- Fin du jeu

## 2-5-3. Indices

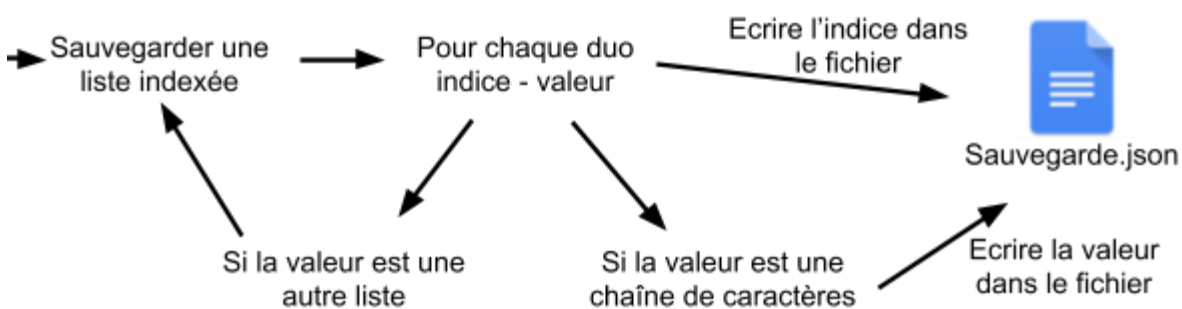
Les énigmes possèdent aussi une liste d'indices. Au début, l'énigme ne dévoile aucun indice et ne renvoie qu'un message pré configuré : "Aucune aide pour l'instant".

Cependant, il est possible de passer à l'indice suivant et ainsi changer le message renvoyé par l'énigme par celui de l'indice.

## 2-5-4. Sauvegarde

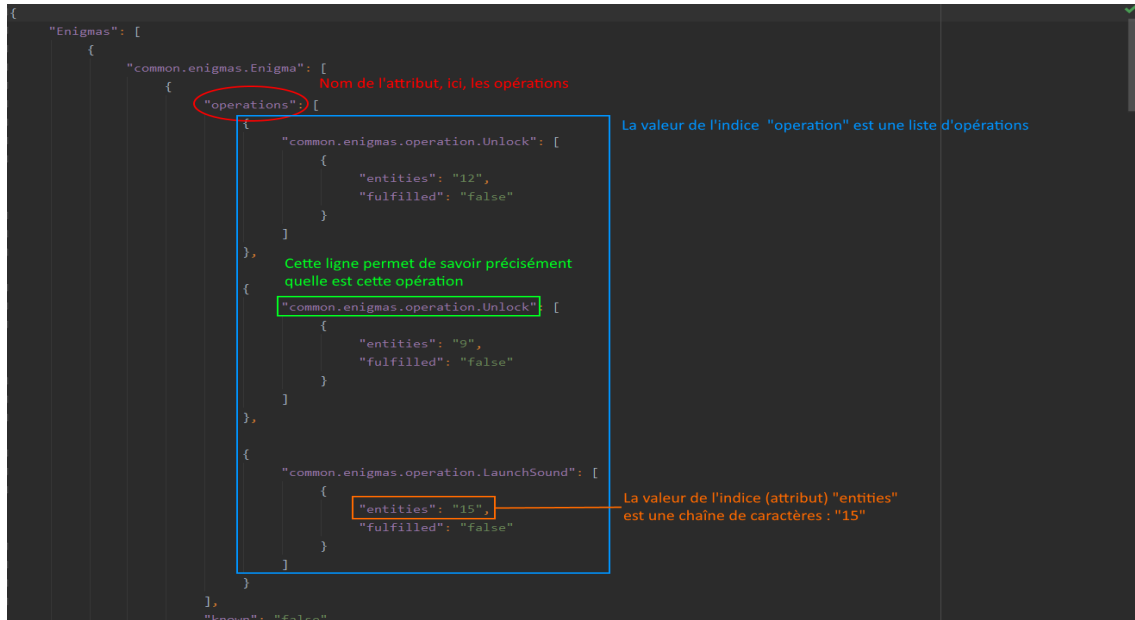
La sauvegarde des énigmes se fait au format JSON, dans un fichier portant le nom de la map. Pour sauvegarder l'énigme, il faut récupérer tous ses attributs ([voir glossaire](#)). Pour y arriver, on récupère une liste indexée, c'est-à-dire qu'elle est composée de duos indice - valeur, où les indices sont uniques. Dans cette liste, les indices sont les attributs et les valeurs sont des éléments pouvant être soit une liste de caractères, soit une autre liste indexée, c'est le cas des conditions, opérations et indices, desquelles il faut récupérer de la même manière leurs attributs, ce qui renvoie une liste indexée.

Grâce à cette structure, il est possible de sauvegarder l'énigme de façon récursive, qui signifie : propager une action faite sur un élément, à ses enfants.



(Image SauvegardeDesEnigmes.1 : mécanisme)

Les conditions et les opérations effectuent toutes des actions différentes. Afin de retrouver les bons objets ([voir glossaire](#)) lors de la lecture de la sauvegarde, le chemin des classes ([voir glossaire](#)) est écrit dans la sauvegarde afin de connaître le type d'objet auquel on a à faire.



(Image SauvegardeDesEnigmes.2 : une partie de la sauvegarde)  
(Voir annexe pour plus de détails sur les classes de l'image)

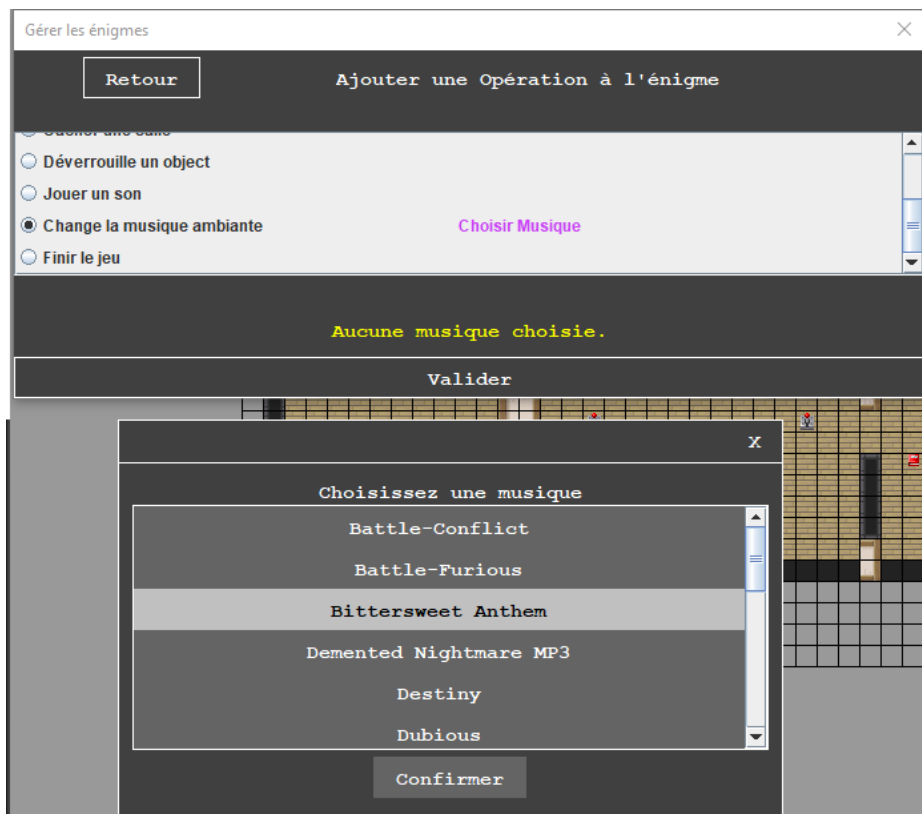
Lors de la lecture de la sauvegarde, les objets sont créés via le chemin de la classe et leurs attributs, qui ont été rassemblés sous forme de liste indexée, leur sont donnés par le constructeur, qui est le moyen de créer une classe.



(Image SauvegardeDesEnigmes.3 : une partie de la lecture de la sauvegarde)  
(Voir annexe pour plus de détails sur les classes de l'image)

## 2-6. Musiques

La musique est un élément important dans un jeu vidéo, c'est pour cela que l'on a décidé de permettre d'en rajouter sur les maps. La LibGdx est capable de gérer la musique et les sons si on lui donne un fichier au bon format (mp3, wav, ogg). Les chemins qui permettent d'accéder aux musiques sont stockés dans le fichier de la map. La musique est considérée comme une entité de taille 0 et de position (0,0) dans le fichier de la map. L'utilisateur peut définir une musique de fond qui passera en boucle. Des sons peuvent se lancer lorsqu'une énigme se finit et la musique de fond peut être changée. Les sons et musiques à choisir, sont fournies pour l'instant.



(Image Musique.1 - Choix d'une musique depuis le menu d'énigmes)

## 2-7. Barre d'outils

### 2-7-1. Zoom

Nous n'avons pas directement géré le zoom, il a été géré par la libGdx. La valeur du zoom de base était de 1, faire varier cette valeur changeait le niveau de zoom.



(Image Zoom.1 : Le zoom)

Cependant, pour choisir quelle valeur passer, c'est beaucoup plus complexe.

- Une valeur inférieure à 1, augmente la taille de la map
- Une valeur supérieure à 1, réduit la taille de la map

#### 2-7-1-1. Voir toute la map

C'était très compliqué de trouver une formule qui permettent de voir toute la map en utilisant ces valeurs du zoom très particulières, on a procédé par tests.

Le premier test a constitué à manuellement changer le zoom pour trouver la valeur pour laquelle la map est entièrement dans l'écran.

- Pour une map de 100 cases, la valeur est de 1.84
- Pour une map de 300 cases, la valeur est de 5.50

On déduit, par produit en croix, que le ratio pour une map de 1 case est  $1.84 / 100$  soit 0.0184. On vérifie en multipliant par 300 et on obtient 5.52 ce qui est proche de 5.50.

La hauteur (on considère que les écrans sont plus large que haut donc si on fait rentrer la map verticalement, alors elle sera aussi rentrée horizontalement dans l'écran) de l'écran est fixe ici : 881 pixels.

- Pour une map de 100 cases, un écran de 703 pixels, la valeur est de 2.3
- Pour une map de 100 cases, un écran de 881 pixels, la valeur est de 1.84
- Pour une map de 100 cases, un écran de 493 pixels, la valeur est de 3.30

On en déduit la formule suivante :  $\text{ratio} = 1.84 * (\text{taille écran} / 881)$ .

Vérification pour un écran 703 pixels :  $1.84 * (703 / 881) = 2.32$  ce qui est proche des 2.3 trouvé manuellement.

### 2-7-1-2. Zoom de base

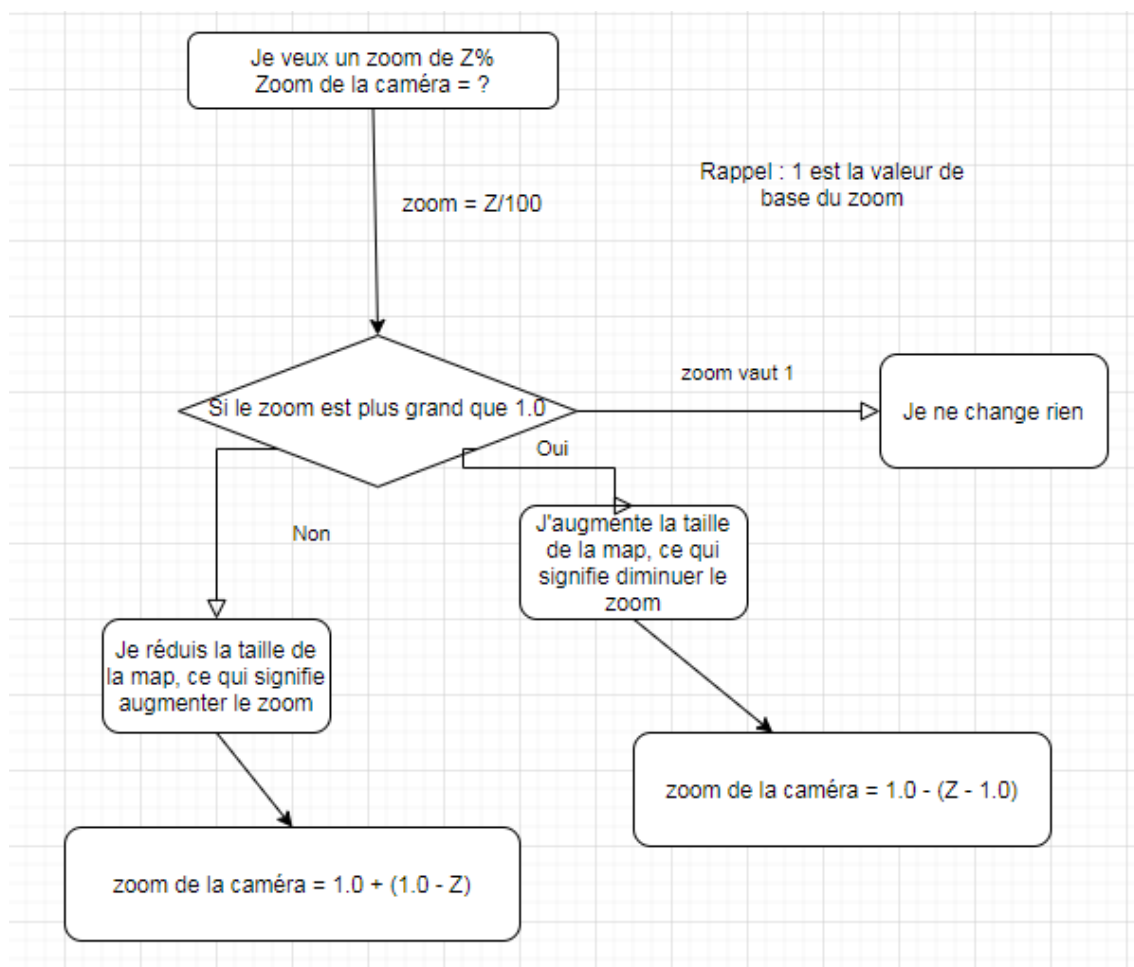
Il s'agit tout simplement d'une réinitialisation du zoom, soit remettre sa valeur à 1.

### 2-7-1-3. Zoom du jeu

Il s'agit de la valeur du zoom en jeu, elle correspond à 150% de zoom.

### 2-7-1-4. Zoom selon un pourcentage

Un zoom au pourcentage demandé.



(Schéma Zoom.1 - Zoom selon pourcentage)

### 2-7-2. Undo et redo

Le Undo (annuler) et Redo (annuler son annulation) permettent un peu de flexibilité dans les erreurs, mais il existe des restrictions dessus comme le nombre maximum de retour est limité (35 de base) et que toutes les actions ne peuvent pas être annulées.

### 2-7-2-1. Actions sujettes au undo et redo

Le undo et redo est seulement possible, pour l'instant, pour les actions suivantes :

- Ajout et suppression d'une entité
- Ajout et suppression d'une énigme
- Changement du nom d'une entité
- Changement du contenu d'une entité (contenu d'un livre)
- Ajout et suppression d'entités contenues dans une autre entité (par exemple des livres dans un coffre)

### 2-7-2-2. Fonctionnement

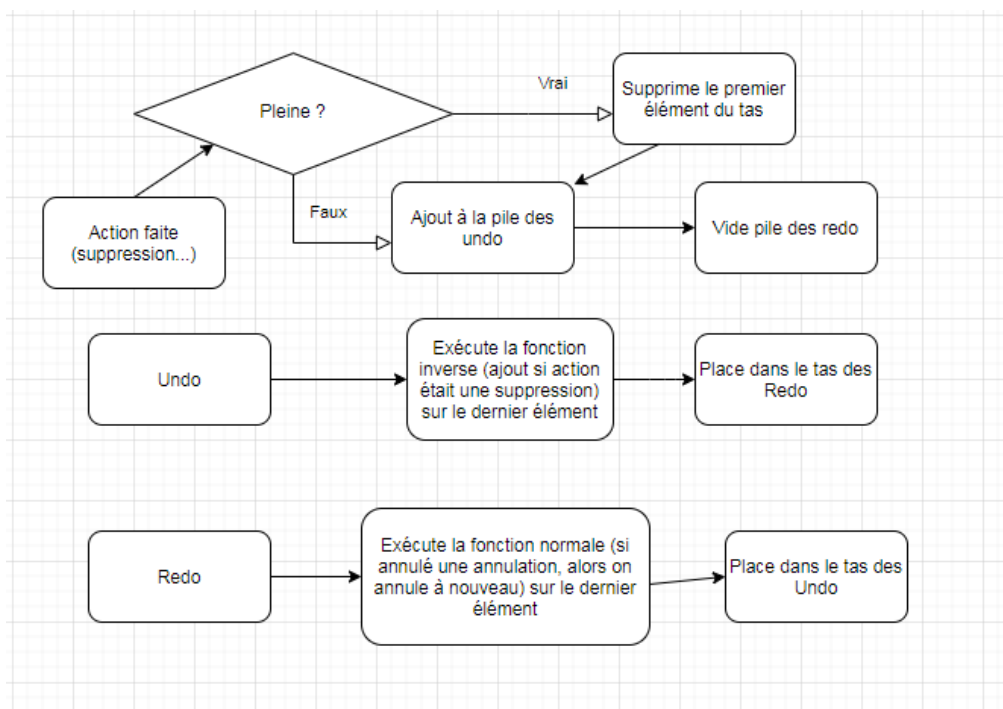
Lorsqu'une action est faite, alors sa réalisation est signalée.

Une action possède deux états, sont état normal et son état inverse. (Par exemple l'état normal d'une suppression d'une entité est la suppression d'une entité, son état inverse est l'ajout de cette même entité).

On a choisi de représenter ça comme deux "tas", l'un pour les undo, l'autre pour les redo. Chaque action faite est ajoutée au tas de "undo". Ajouter une nouvelle action vide le tas des redo.

Si je fais un undo, alors je veux faire l'inverse de ma dernière action, j'appelle donc la fonction inverse de mon action et je la stocke dans la pile de redo.

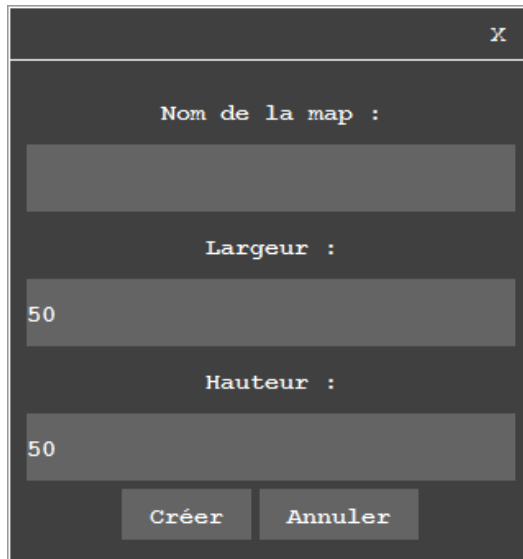
Si je veux faire un redo, alors je refais l'action normale et j'ajoute mon action à la pile des undo.



(Schéma Undo, Redo.1 : Fonctionnement)

### 2-7-3. Création d'une map

La création d'une map, est majoritairement la sauvegarde d'une map vide.

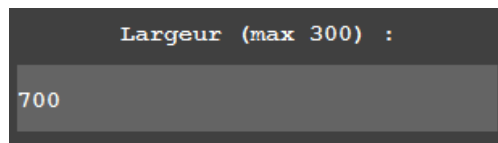


(Image Création.1 - Création d'une map)

Les vérifications sont les suivantes :

- Le nom ne doit pas être vide, ou seulement constitués de caractères blancs (comme espaces)
- La largeur doit être comprise entre 0 (non inclus) et LARGEUR MAX (une constante, vaut 300 de base).
- La hauteur doit être comprise entre 0 (non inclus) et HAUTEUR MAX (une constante, vaut 300 de base).

Si une valeur n'est pas correcte, alors un message est affiché pour prévenir l'utilisateur.



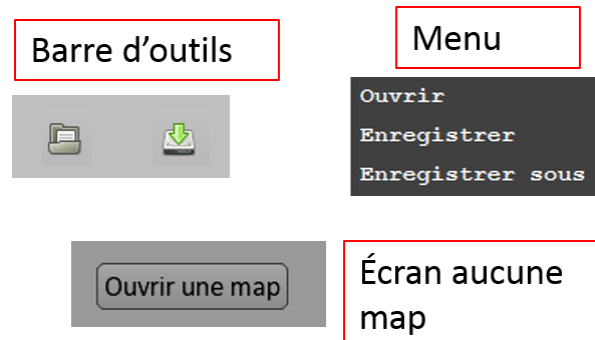
(Image Création.2 - Erreur dans un champ)



## 2-7-4. Sauvegarde et lecture

La barre d'outils permet la sauvegarde dite rapide (on écrase l'ancien fichier avec le fichier modifier), la sauvegarde dans un nouveau fichier. Et l'ouverture de la map.

Sont sauvegardés, la map (.tmx), son fichier d'énigmes (.json) et un fichier de données (.tmx) également. Pour plus d'infos, veuillez-vous référer aux parties respectives.



(Image Sauvegarde.1 - Plusieurs boutons pour les mêmes fonctionnalités)

Un message est affiché à chaque sauvegarde.

## 2-7-5. Importation et exportation de maps

Le principe de l'importation et de l'exportation est de pouvoir partager ses créations avec les autres utilisateurs de l'application.

Une map est sauvegardée sous trois fichiers : **la map**, **les énigmes** et **les données de la map** (qui pour l'instant, ne contiennent que le nom du créateur).

### 2-7-5-1. Exportation

L'exportation consiste à rassembler le contenu de ces trois fichiers dans un seul selon la disposition suivante :

**Nom de la map \0 données de la map \0 map \0 énigmes \0**

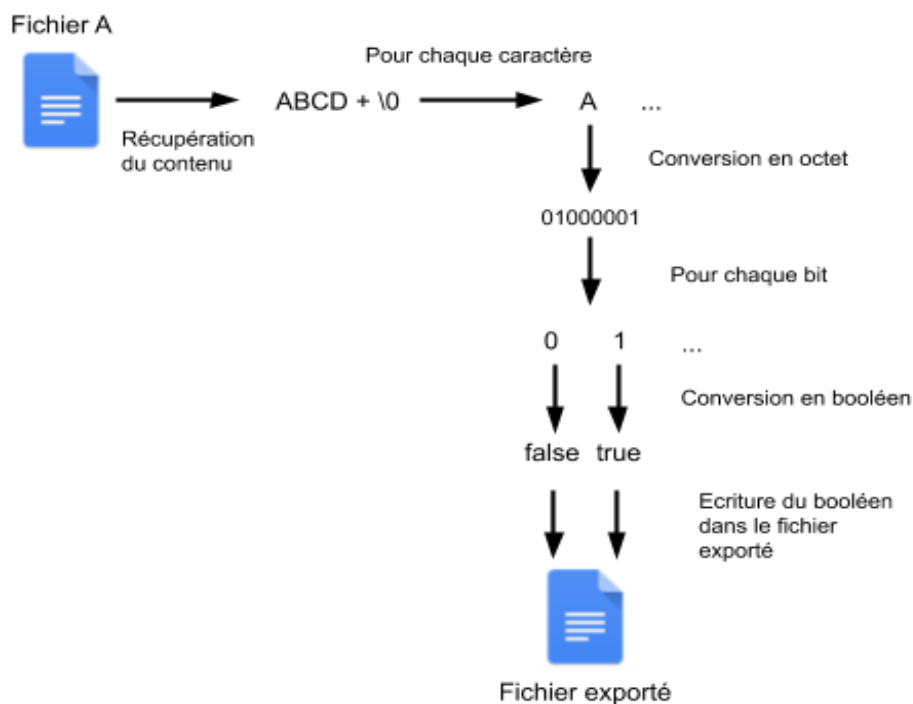
Les caractères '**\0**' indiquent la fin d'un contenu ce qui permet de les délimiter afin de les retrouver lors de l'importation.

Pour limiter un maximum la modification de ces informations en dehors de l'application, les fichiers exportés sont écrits en bits pour réduire la lisibilité.

Ainsi, le contenu des fichiers de la map est récupéré sous forme de chaîne de caractères à laquelle on ajoute le caractère de fin de contenu '**\0**'.

Ensuite, pour chaque caractère de cette chaîne, on récupère sa valeur en octet. Chaque bit de cet octet sont écrits un par un dans le fichier exporté sous forme de booléen (qui équivaut un bit). Il n'est pas possible d'écrire les octets directement car les éditeurs de textes interprètent automatiquement l'octet sous forme de caractère ce qui va à l'encontre de l'objectif de lisibilité réduite.

L'avantage d'écrire des booléen plutôt que des caractères, autre que la lisibilité, est de réduire considérablement la taille du fichier.



(Image Exportation.1 : fonctionnement)

## 2-7-5-2. Importation

L'importation consiste à effectuer l'inverse de l'exportation, c'est-à-dire, qu'une série de 8 bits (afin de faire un octet) est lue dans le fichier importé.

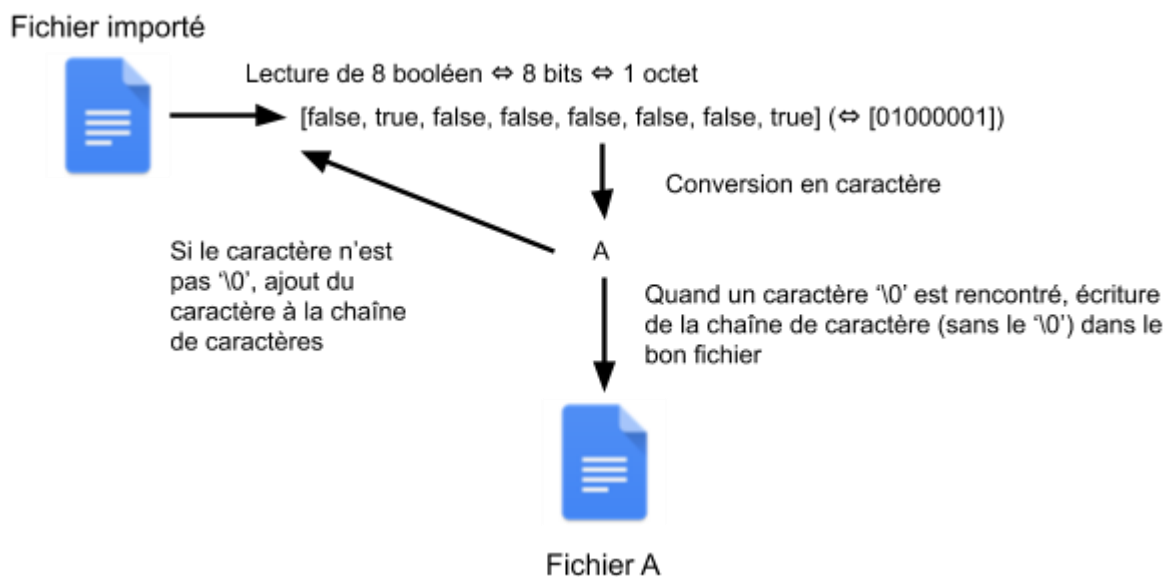
Cet octet est converti en caractère. Les caractères sont réunis dans une chaîne de caractères au fur et à mesure.

Lorsque le caractère '\0' est rencontré, la chaîne de caractères est écrite dans le bon fichier.

Ainsi, les trois fichiers de la map sont retrouvés depuis le fichier importé.

Il est à noter que l'exportation et l'importation d'une partie est faisable et fonctionne exactement de la même manière à la différence que le fichier exporté est composé du contenu des **trois fichiers de la map** et d'un fichier en plus contenant **les données de la partie** (durée, nombre de joueurs, etc...) :

Nom de la map \0 nom de la map \0 données de la map \0 données de la partie \0 map \0 énigmes \0



(Image Importation.1 : fonctionnement)

## 2-7-6. Raccourcis

Afin de permettre aux utilisateurs plus expérimentés de profiter de moyen plus efficaces et rapides pour créer des escapes game, quelques raccourcis ont été ajouté. Ce sont des combinaisons de touche que l'on retrouve sur tous les logiciels.

### 2-7-6-1. Clavier

Les raccourcis basiques sont disponibles tels que :

- Ctrl + s, sauvegarde la map
- Ctrl + z, undo
- Ctrl + y, redo
- Ctrl + alt + s, sauvegarde de la map en choisissant le nom du fichier
- Ctrl + molette, zoom/dézoom
- Ctrl + "+", zoom
- Ctrl + "-", dézoom

### 2-7-6-2. Menu popup

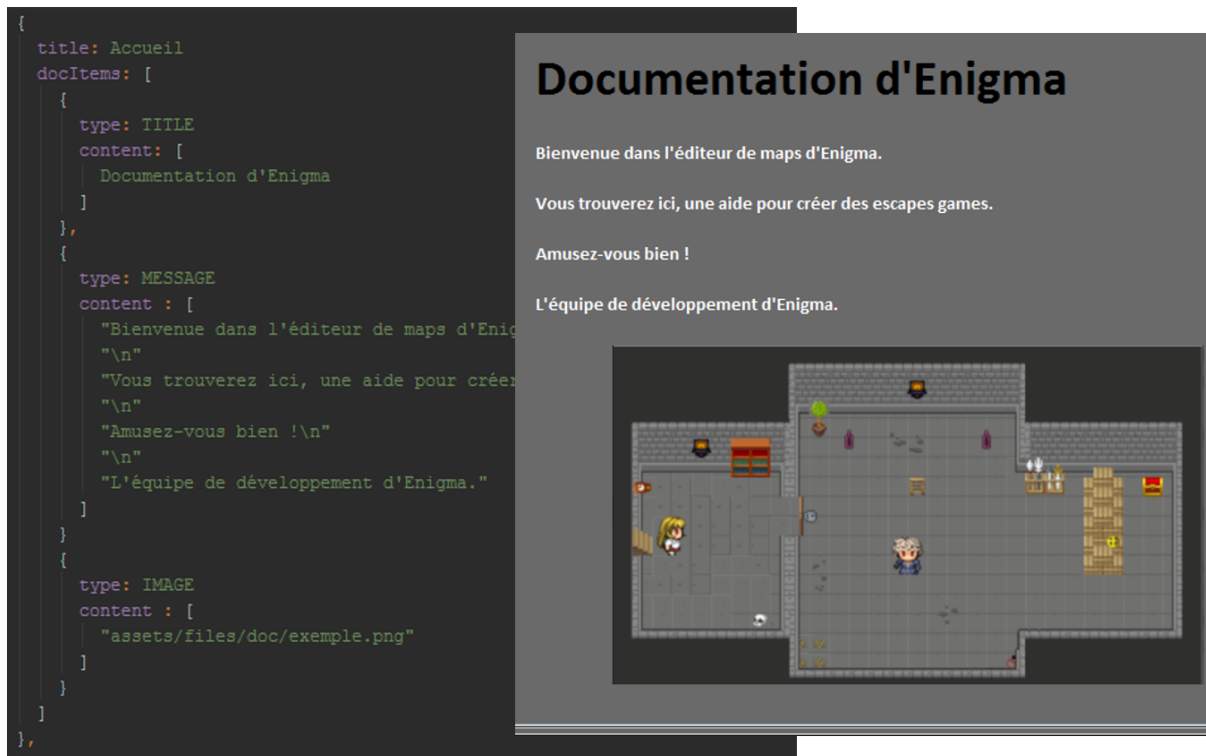
Dans l'éditeur le clic droit de la souris permet d'ouvrir un petit menu avec quelques fonctionnalités :

- Le zoom du jeu, avoir un aperçu de toute la map et revenir au zoom qui permet de faire le drag and drop
- On peut changer le type d'action que la souris va réaliser (pinceau, gomme, déplacement)
- On peut visualiser en rouge toutes les cases de la map qui sont bloquante

## 2-7-7. Documentation dans le logiciel

Comme on peut le retrouver dans de nombreux logiciels, comme RPG Maker ou Game Character Hub (qui sont dans le même style 2D de notre application), la possibilité de lire le guide utilisateur dans l'application, a été ajoutée.

Il est stocké dans un json (pour permettre plus de flexibilité, et une version pour chaque langue).



(Image Documentation.1 : Une page de documentation et son rendu)

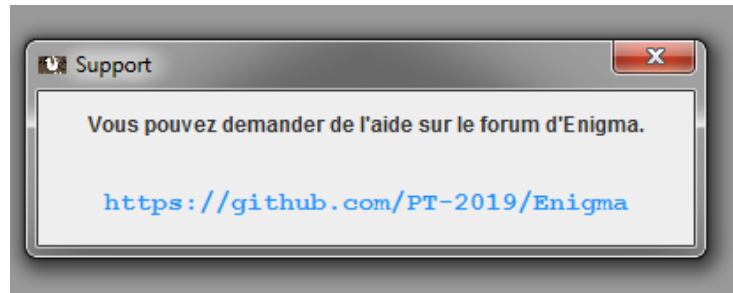
Le premier argument "title" désigne le nom du lien qui permet d'arriver sur cette page. "docItems" désigne le contenu de la page.

Actuellement les types supportés sont :

- TITLE : un titre
- MESSAGE : un texte à afficher
- IMAGE : une image
- TITLE\_SMALL : un sous-titre

## 2-7-8. Onglet support

L'onglet de support est assez simple, il ne contient qu'un lien qui ouvre le navigateur sur la page du github d'Enigma.



(Image Support.1 : Onglet de support)

## 2-7-9. Simulation

L'onglet simulation permet de tester son escape game, sans quitter l'éditeur. Tout est similaire au mode de jeu normal, mais les cases non accessibles sont affichées.

Il est prévu d'offrir plus de support lors de la simulation, par exemple commencer à un certain point...



(Image Simulation.1 : menu de simulation)

## 2-7-10. Gomme

La gomme permet de supprimer des entités en cliquant dessus, cependant, dû au manque d'une texture correcte, la précision n'est pas très haute. Vous pouvez supprimer une entité avec plus de précision en mode "pinceau" en cliquant sur l'entité puis "supprimer".

Supprimer une salle, supprime toutes ce qu'elle contient. Il n'est pas possible de supprimer un objet utilisé dans une énigme. (Un message est affiché).

Le fonctionnement de la suppression est assez-simple. Tous les objets qui "touchent" l'objet que l'on veut supprimer, sont vérifiés. Ainsi, les entités concernées sont supprimées (les tiles de la map sont retirés) et remplacées par les entités qu'elles auraient pu écraser.

Par exemple, supprimer une bibliothèque dans une salle, va retirer les tiles de la bibliothèque (entre autres) et remettre celles de la salle.

## 2-7-11. Langue

L'équipe a commencé à préparer le logiciel pour qu'il soit disponibles dans plusieurs langues (pour l'instant nos objectifs sont Français et Anglais). Toutes les interactions entre le programme et les mots à traduire se font par une Façade, qui selon la langue choisi renvoi le mot dans cette langue.

Par exemple, on demande à la façade le mot qui correspond à PLAY :

- Si la langue est anglaise, alors "play" est renvoyé
- Si la langue est française, alors "jouer" est renvoyé

Actuellement, seulement le programmeur peut changer la langue du jeu.

## 2-8. Inventaire

Des entités peuvent être ajoutées à une entité Container (**voir annexe**) via le menu de gestion des entités.

Une nouvelle boîte de dialogue apparaît lorsqu'on clique sur un objet présent sur le menu des catégories. Elle permet de sélectionner l'objet que l'on souhaite inclure dans le conteneur. Lorsqu'on choisit une entité celle-ci est envoyée à la boîte de dialogue qui se charge de vérifier que c'est une entité acceptable, à titre d'exemple on ne peut pas mettre une salle dans un coffre. Si l'entité est correcte alors une nouvelle entité est créée. La sauvegarde de l'inventaire n'est pas encore implémentée.

## 2-9. Notifications

Beaucoup d'actions se font, sans que l'utilisateur ne puisse voir ce qu'il se passe, ou ne puisse comprendre ce qu'il s'est passé. Dans la même idée que les Toast sous Android, donc un message qui apparaît et disparaît tout seul, l'équipe a créé son propre toast et l'utilise pour afficher de nombreux messages.



(Image Notification.1 : Toast  
Android - source)



(Image Notification.2 : Notre Toast)

Quelques exemples de messages possibles :

- La création de la map rate.
- Le drag and drop rate
  - Préviens si on est sur la map, mais qu'il y a déjà une entité
  - Préviens si on n'est pas sur la map
  - Préviens si le drag and drop a été désactivé car le zoom a été modifié
- Fin d'une action (import/export/sauvegarde/undo/redo...)

Un popup spécial est affiché lors de l'import et de l'export car cela peut prendre du temps.



(Image Notification.3 : Exportation)



### 3. Présentation du Jeu

Dans cette partie, il sera présenté tous les éléments et les fonctionnalités du jeu.

#### 3-1. Lancement

Avant de lancer le jeu, il faut sélectionner la partie à lancer. Dans cette interface, on retrouve plusieurs affichages permettant de créer, lancer ou rejoindre une partie. Voici ces affichages :

##### 3-1-1. Barre de navigation

La barre de navigation permet à l'utilisateur de naviguer entre les écrans. L'écran actuel apparaît en rouge.

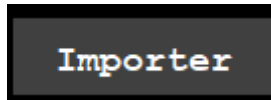


(Image BarreDeNavigation.1 : multijoueur)

Si l'utilisateur tente de changer de page alors qu'il est dans une page où l'action de la quitter peut entraîner des problèmes, il lui est demandé s'il souhaite vraiment changer de page. Par exemple, quitter le lobby en étant chef de la partie annule totalement la partie et affecte donc aussi les joueurs qui avaient rejoint.

### 3-1-2. Barre d'action

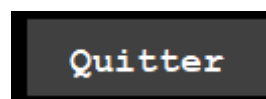
La barre d'action permet à l'utilisateur, comme son nom l'indique, d'effectuer une action. Elle varie selon les affichages :



(Image BarreD'Action.1 : affichage : solo ou multijoueur)



(Image BarreD'Action.2 : affichage : lobby et chef de la partie)



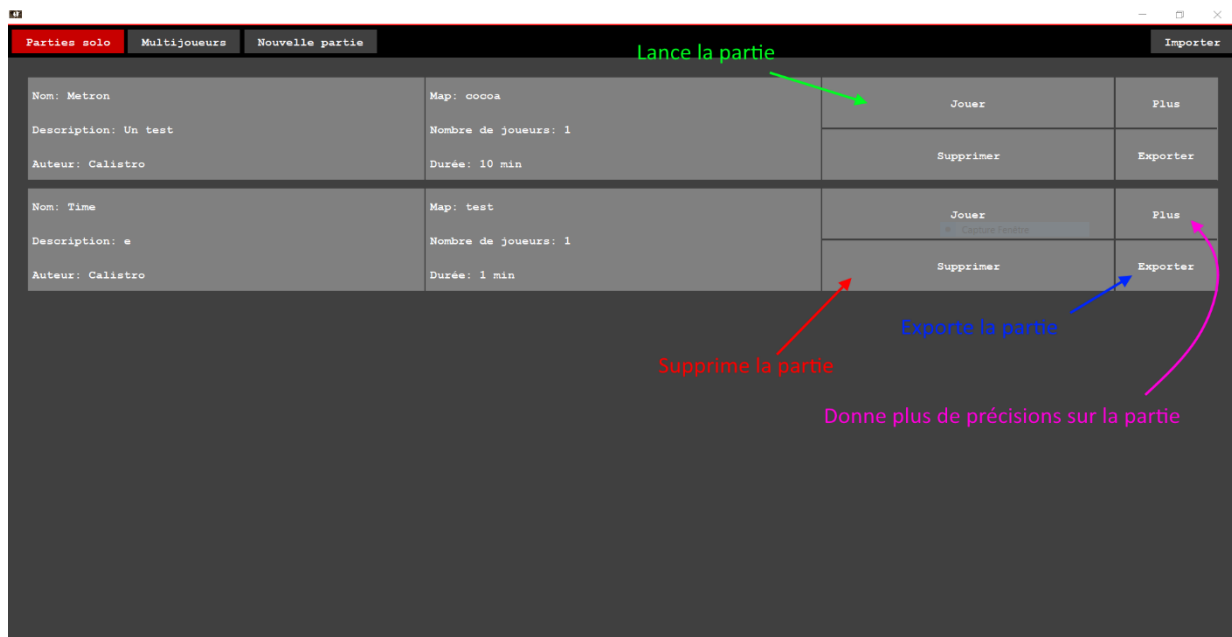
(Image BarreD'Action.3 : affichage : lobby)



(Image BarreD'Action.4 : affichage : création de partie)

### 3-1-3. Solo

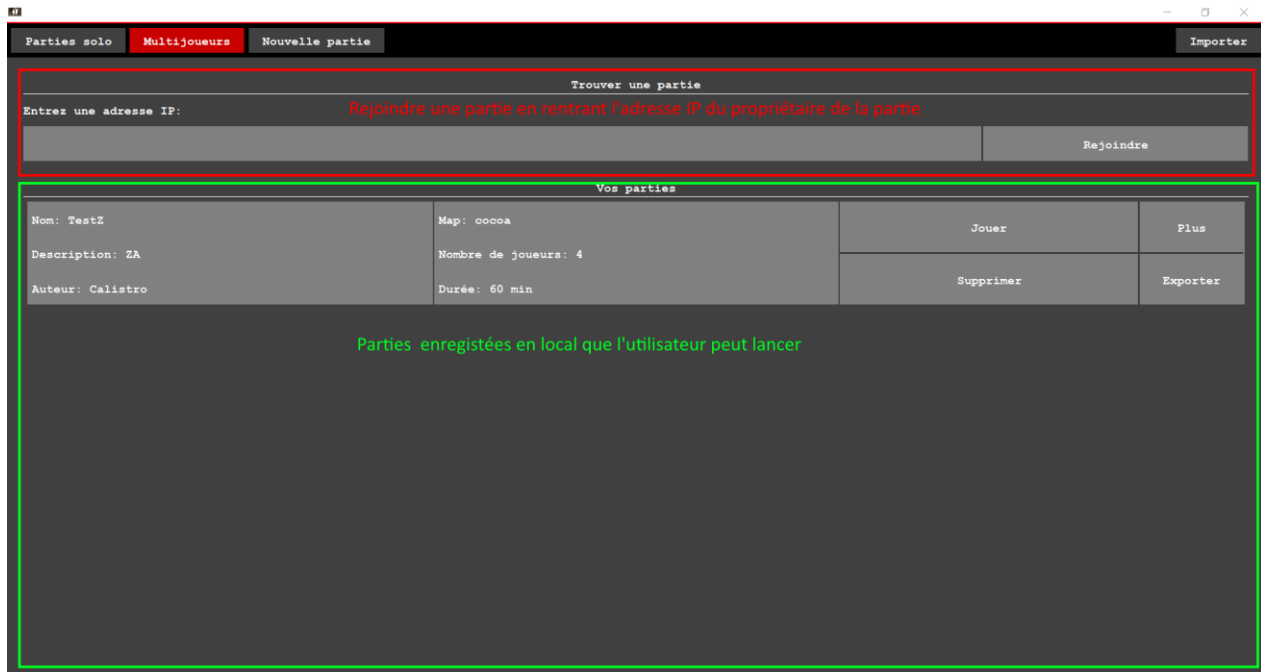
Le menu des parties solo est le premier menu qui apparaît. Il affiche la liste des parties solos stockées en local. Le joueur n'a plus qu'à sélectionner « Jouer » sur la partie souhaitée pour que le jeu démarre.



(Image PartiesSolos.1 : Liste)

### 3-1-4. Multijoueur

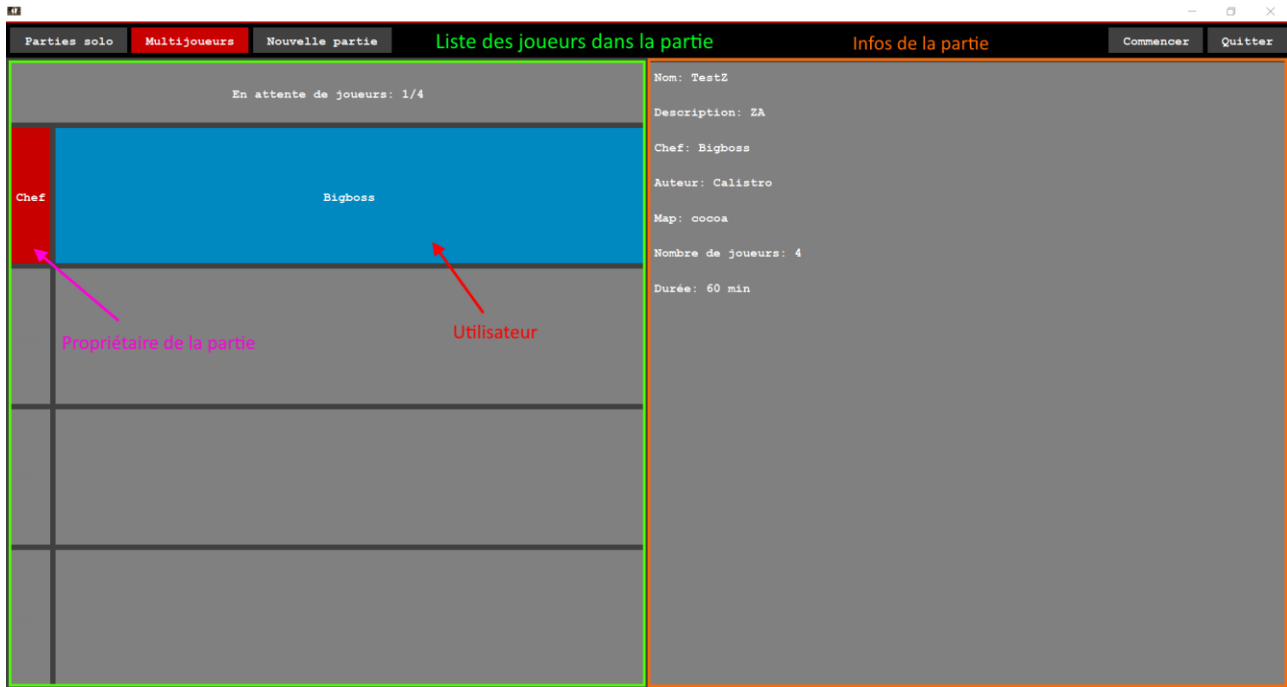
Le menu des parties multijoueur est quasiment le même que celui des parties solos. Il affiche la liste des parties multijoueur stockées en local. L'élément en plus, c'est la présence d'une zone de texte dans laquelle l'utilisateur peut rentrer l'adresse IP du propriétaire de la partie, afin de rejoindre une partie en ligne.



(Image PartiesMultijoueur.1 : Liste)

### 3-1-4-1. Lancer

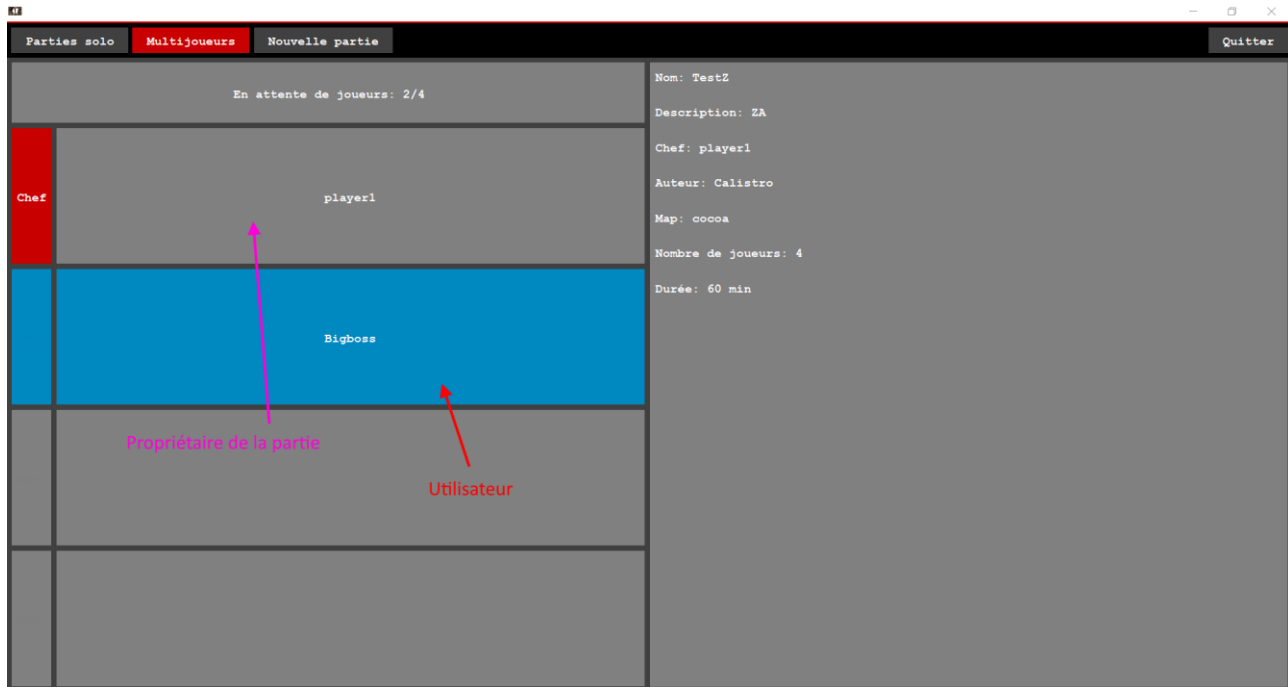
Lorsque l'utilisateur clique sur « Jouer », il ouvre (théoriquement car ce n'est pas encore réalisé) alors un serveur, qui sera le lien entre les différents joueurs durant la partie, dont il est le propriétaire. Le temps que les autres joueurs rejoignent, il est placé dans un lobby, avec les autres joueurs. Il ne peut lancer la partie que lorsque que le nombre de joueurs requis est atteint. S'il quitte, il annule la partie.



(Image Lobby.1 : chef de la partie)

### 3-1-4-2. Rejoindre

Lorsque l'utilisateur entre une adresse IP valide et rejoint la partie, il est mis en attente dans un lobby, le temps que le nombre de joueurs nécessaires soit atteint et que le propriétaire de la partie lance la partie.




(Image Lobby.2 : partie rejoint)

### 3-1-5. Créer une partie

L'utilisateur a la possibilité de créer ses parties. Pour cela, il doit renseigner :

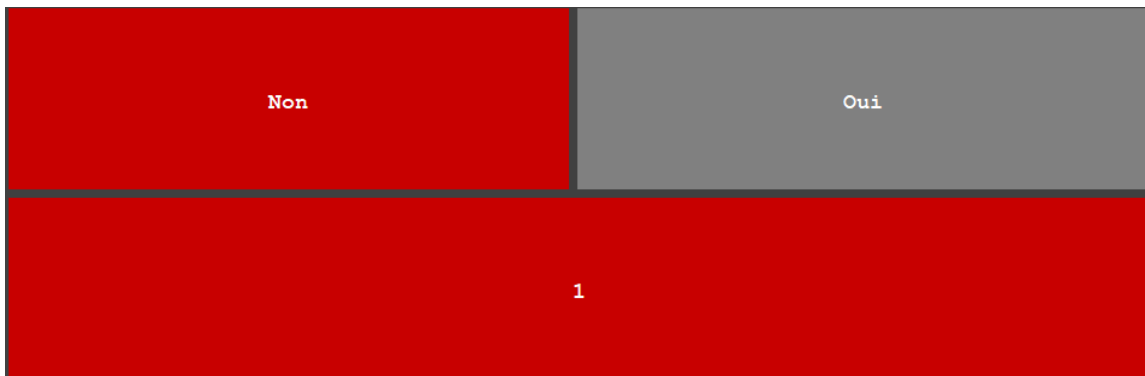
- Le nom de la partie
- Une description (facultatif)
- Une map sur laquelle la partie va se jouer
- Si la partie est multijoueur ou non
- Le nombre de joueurs
- La durée en minutes



Parties solo		Multijoueurs		Nouvelle partie		Créer
Nom:						
Description:						
Map:						
Multijoueurs:				Non	Oui	
Nombre de joueurs:				1		
Durée (min) :						

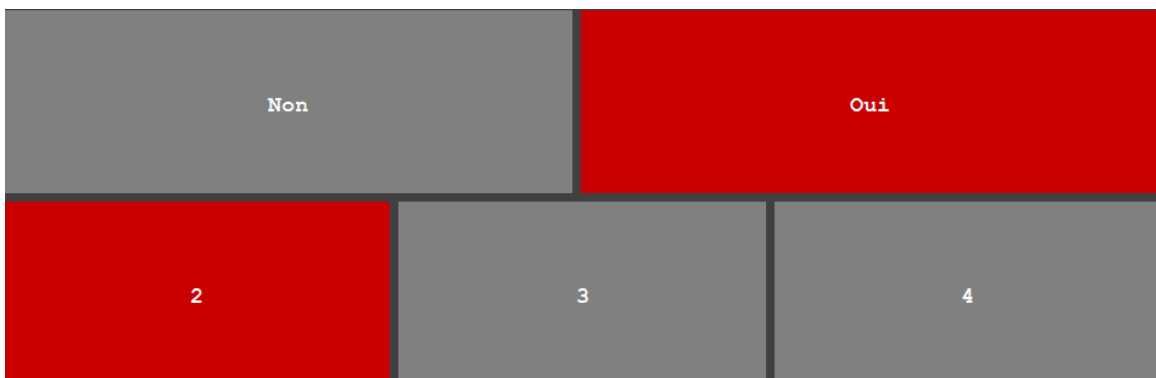
(Image CréerPartie.1 : formulaire)

Le nombre de joueurs que l'utilisateur peut choisir change en fonction de si la partie est multijoueur ou non :



Non	Oui
1	

(Image CréerPartie.2 : partie solo)



Non	Oui	
2	3	4

(Image CréerPartie.3 : partie multijoueur)

### 3-1-6. Importation et exportation de parties

Tout comme les maps, il est possible d'importer et d'exporter une partie. Le fonctionnement est exactement le même (voir importation et exportation de map) que pour les maps, la seule différence, c'est qu'il y a un fichier supplémentaire qui contient les données de la partie :

- Nom du créateur
- Nom de la map
- Nom de la partie
- Description de la partie
- Nombre de joueurs
- Durée

### 3-2. Collision

Il est naturel que si on essaye de marcher sur un mur, le mur étant solide, alors on soit bloqué. Cela s'appelle la gestion de la collision.

#### 3-2-1. Collision avec la map

La collision avec la map est gérée très simplement, comme vu dans la partie éditeur, la map possède un niveau collision qui contient une tile rouge (id=2041) si la case n'est pas accessible.

La position dans l'espace (x,y) d'un personnage est récupérée, convertie en indices de tableaux (ligne, colonne) et il est vérifié si la case dans le layer collision contient le tile rouge (dans les faits, on regarde simplement s'il y a un tile).

#### 3-2-2. Collision avec les autres joueurs

La position du joueur est récupérée, comme expliqué ci-dessus, et il est vérifié s'il n'y a pas une collision avec toutes les entités spéciales (Monstre, Joueur, PNJ). On utilise un mécanisme de la LibGdx qui permet de comparer 2 polygones entre eux pour savoir s'ils se touchent ou non.

### 3-3. Correction Y

Le placement des entités implique une correction selon l'axe Y (ordonnées) car il faut recréer la profondeur.

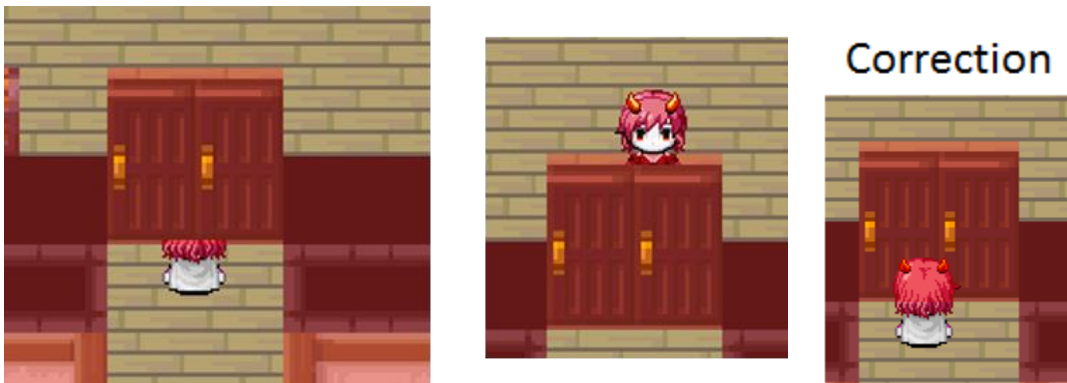
### 3-3-1. Correction Y des entités immobiles

Comme abordé dans la partie Map (2.1) et la partie Librairie (2.2.2) de l'éditeur, les entités (bibliothèques...) sont déjà placées sur la map à un niveau. Si le niveau de joueur est égal à celui de l'entité, alors elle sera dessinée dessus. Cela pose un problème si le joueur est derrière l'objet, il ne devrait pas être dessous, mais dessus. (Et inversement).

La correction appliquée, est que les entités dont le Y est inférieur au joueur sont déplacées au même niveau. Les entités dont le Y est supérieur au joueur sont déplacées dans un niveau inférieur, donc seront dessinées avant.

Exemple : Les portes sont toujours construites au Layer "Decoration1" et le joueur se trouve au Layer "Floor2", juste en dessous.

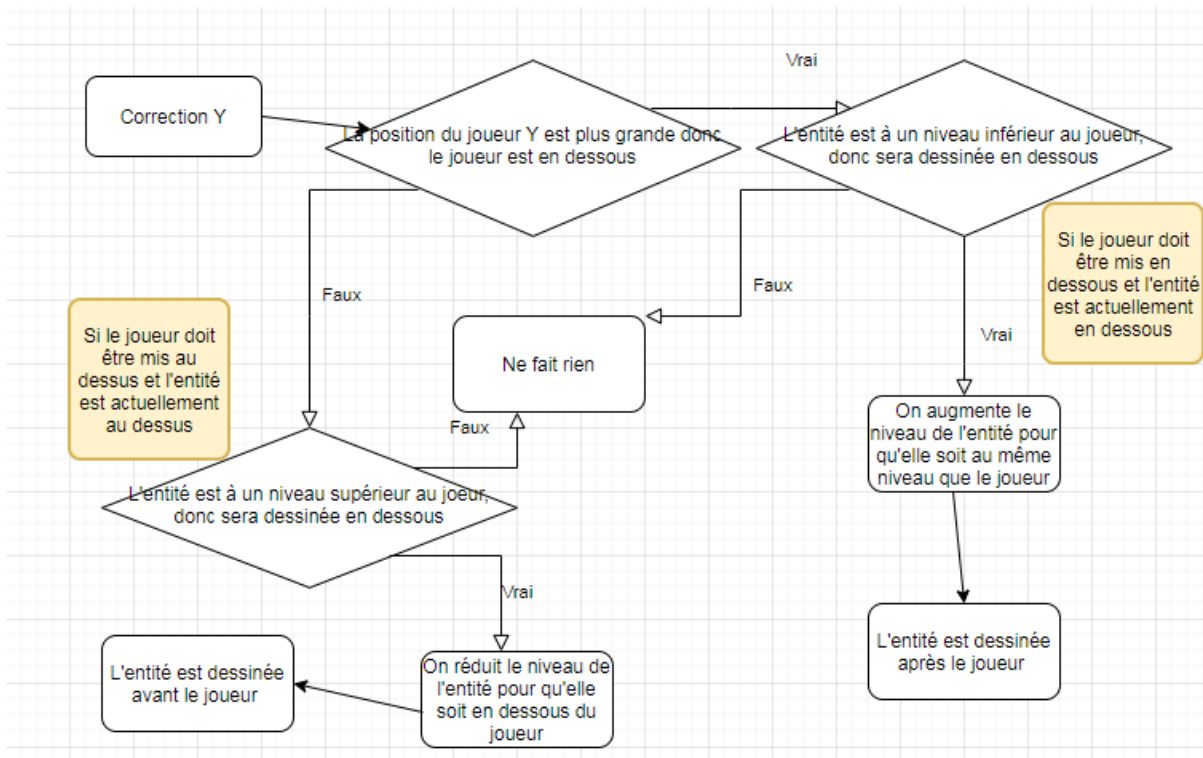
Si le joueur est en haut (Image centrale de Correction.1), alors tout ce passe bien, il est dessiné avant. Cependant sur l'autre image (Image gauche de Correction.1), si le joueur est en dessous, alors car il est sur un layer inférieur, il est dessiné avant.



(Image Correction.1 - Désactivation de la correction)

La correction place donc les entités en face du joueur au même niveau que lui (car il est en dessous), ce qui ici, revient à mettre la porte au niveau "Layer2". Ainsi le joueur sera dessiné dessus.





(Schéma Correction.2 - Correction Y des entités immobiles)

### 3-3-2. Correction Y des entités mobiles

Les entités mobiles sont traitées différemment par la map que les entités immobiles, il s'agit simplement d'une liste. Les premiers dans la liste sont les premiers dessinés. L'axe Y commence en bas à 0 et va vers le haut. On peut donc que les entités dessinées en premières soient celles dont le Y est le plus grand. Donc les entités avec un plus petit Y vont les écraser en se mettant par-dessus. Il s'agit simplement d'un tri de la liste.



(Image Correction.3 - Correction y des Personnages)

### 3-4. Enigmes

#### 3-4-1. Type

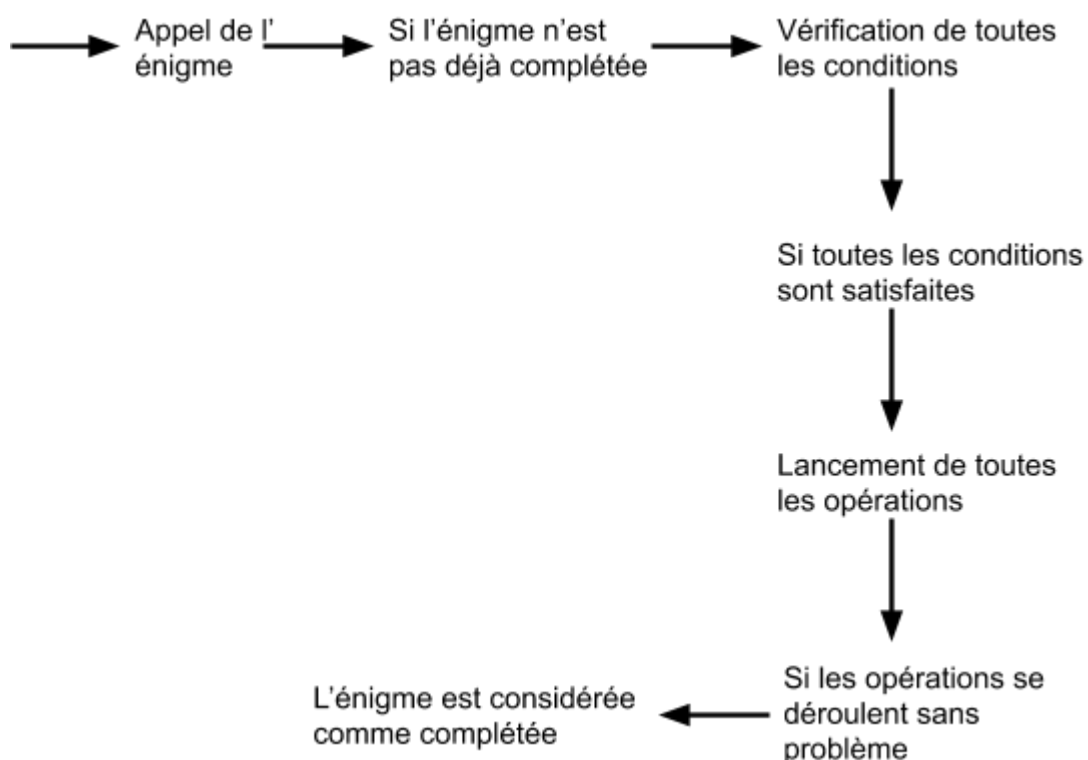
A chaque fois que le joueur fait une action : entre sur la case, quitte la case ou l'interagit sur la case (donc sur l'objet dessus), toutes les énigmes du type correspondant, sur cette case, sont prévenues.

Ainsi, les énigmes sont classées en trois types :

- ON\_ENTER : énigme se lance quand le joueur entre sur une tile
- ON\_EXIT : énigme se lance quand le joueur sort d'une tile
- ON\_USE : énigme se lance quand le joueur interagit avec une tile

#### 3-4-2. Mécanisme

Lorsqu'une énigme est lancée, elle va commencer par vérifier qu'elle n'est pas déjà complétée. Si c'est le cas, alors elle teste toutes les conditions qu'elle possède. Si les conditions ne retournent pas d'erreur, alors l'énigme lance toutes les opérations qu'elle. Encore une fois, si les opérations ne retournent pas d'erreur, alors l'énigme est considérée comme complétée.



(Schéma Enigmes.1 : Mécanisme)

### 3-4-3. Affichage d'un message à l'exécution

Après l'exécution de toutes les énigmes sur la case, on va trier par priorité les messages pour sélectionner le plus important.

Par exemple, si une énigme ne peut pas donner un objet, alors c'est plus important que de savoir qu'un son a bien été joué.

Certains messages ne seront jamais affichés si leur priorité est trop faible (on ne veut pas savoir que la condition "Avoir dans l'inventaire" a été satisfaite car cela donnerait trop d'information à l'utilisateur.)

On affichera par exemple des messages comme "Bonne réponse"/"Mauvaise réponse" après une saisie...

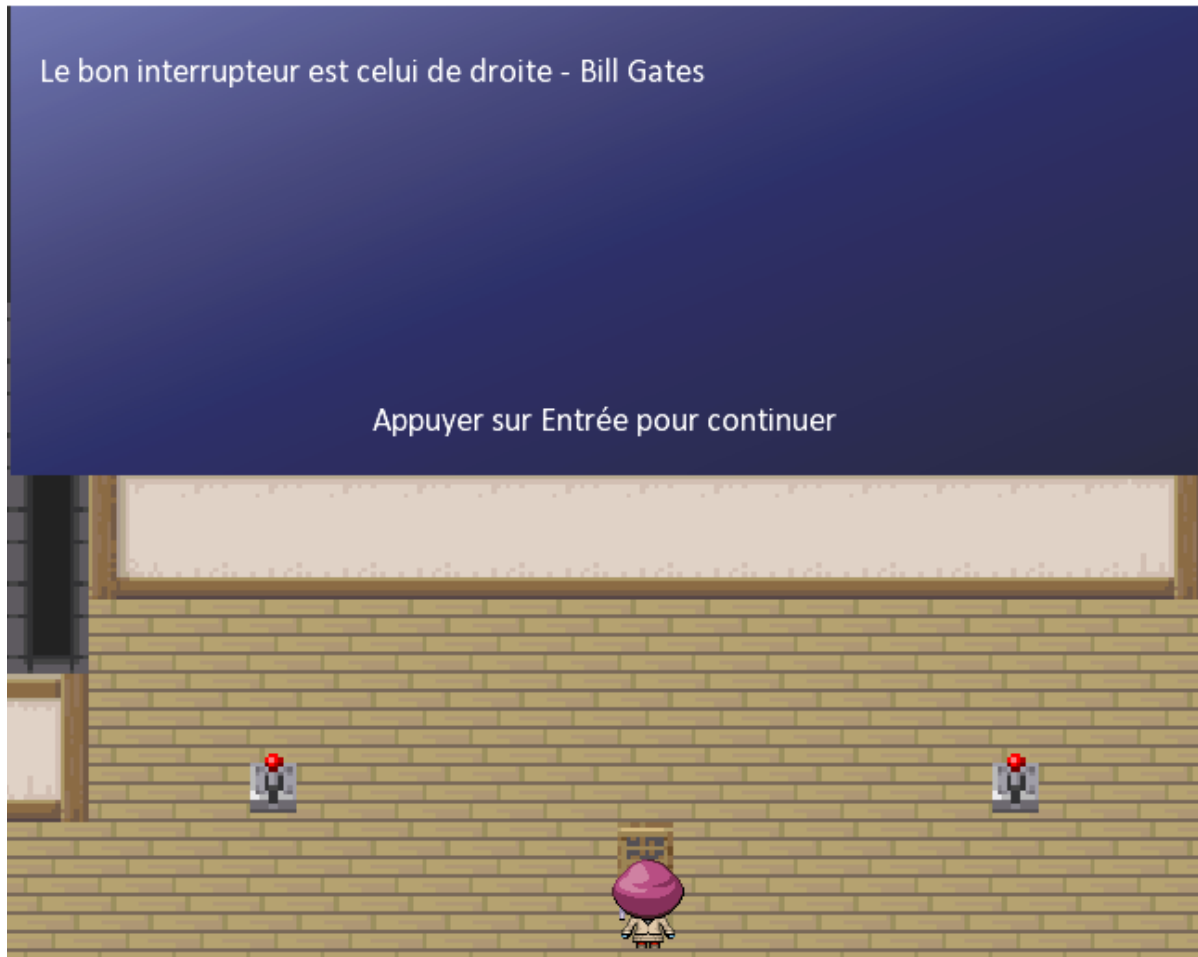
### 3-5. Joueurs

Le personnage principale se déplace à une vitesse de 10 pixel par seconde, il répond aux ordres du joueur à travers les flèches directionnelles les touches z,q,s,d. Lorsqu'une touche est enfoncée on va afficher en boucle des sprites. Le personnage interagit avec l'entité présente devant lui, s'il est au milieu de 2 entités alors l'entité choisit est celle la plus éloigné du repère.

## 3-6. Boîtes de dialogues

### 3-6-1. Texte

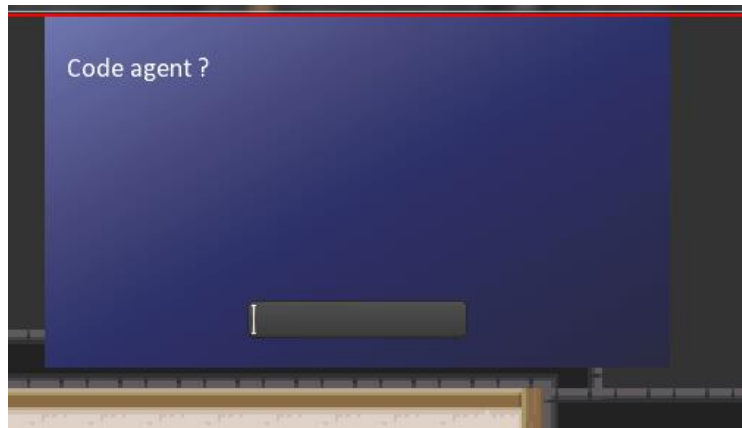
Le texte à une grande place dans le gameplay de notre jeu. Lorsque le jeu doit afficher du texte à l'écran il utilise une boîte de dialogue qui apparaît en haut de l'écran.



(Image Boîtes de dialogues.1 - Un dialogue)

### 3-6-2. Saisie d'une réponse

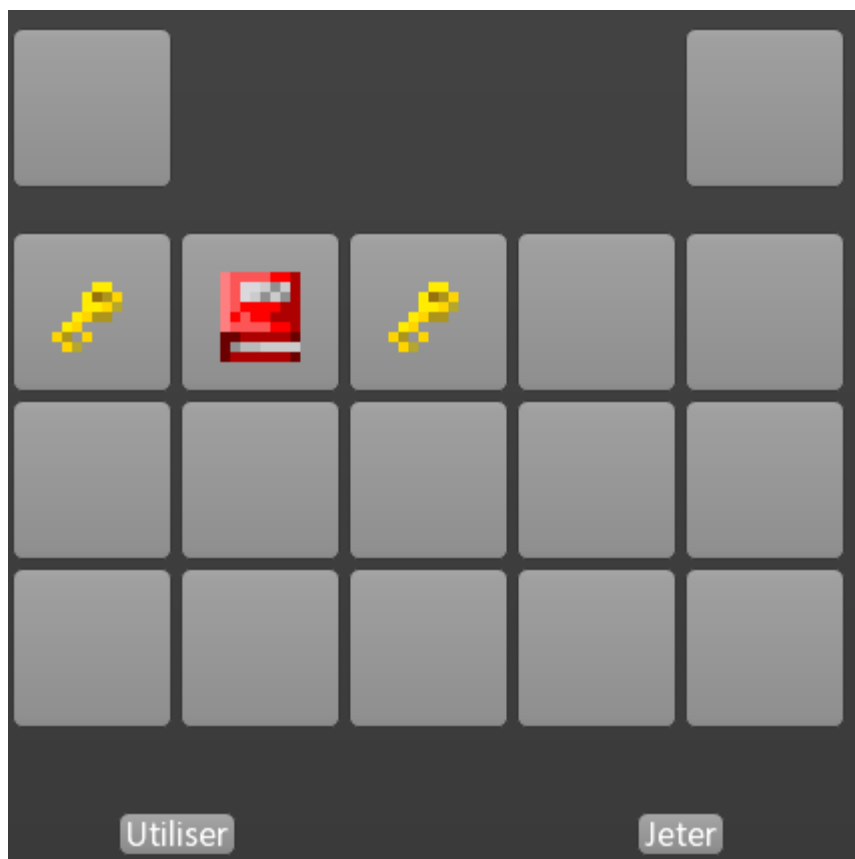
Une boîte de dialogue permettant la saisie est affichée, si une énigme nécessitant une saisie le demande.



(Image Boîtes de dialogues.2 - Saisie d'une réponse)

### 3-7. Inventaire

L'inventaire sert à visualiser les objets tenus par le joueur lorsque la touche "i" ou "espace" est appuyé.



(Image Inventaire.1 - Inventaire du joueur)

### 3-8. Timer

Le timer est en haut à gauche, il permet de savoir combien de temps il reste pour finir la map. Si aucune valeur n'est précisée, le timer prends la valeur de base (c'est une constante qui vaut 1h), sinon elle prendra la valeur que le créateur de la partie aura choisi.

### 3-9. Musique dans le jeu

On récupère les musiques grâce au fichier de la map. Si c'est la musique principale alors elle est lancée.

## 4. Présentation de l'API

### 4-1. Pourquoi une API ?

Une API est une interface de programmation, qui permet l'échange, ou l'utilisation des services d'une application dans une autre application.

Beaucoup de code que nous avons écrit, pourrait servir dans de nombreux autres projets du même genre (Inventaire, Collision ...) ou pas (Composant Swing Customizable), ce qui a donné envie de réaliser une API ([voir glossaire](#)) qui contient tout ce code que nous avons déjà écrit et que nous pourrions vouloir utiliser dans d'autres projets.

Les parties suivantes présentent, quelques domaines recouverts par notre API.

### 4-2. Refonte de l'interface Swing

Java Swing a été utilisé pour construire une grande partie de notre interface graphique. Cependant, le "look and feel" (style général de différents composant issus d'une même API) des éléments graphiques n'était pas à la hauteur de nos attentes. De plus, l'équipe souhaitait que l'application ait une signature, un style qui lui est propre et qui soit uniforme dans toute l'application. C'est pour cette raison que tous les éléments qui étaient nécessaires, ont vu leur style modifié.

Ces éléments customisés, que l'on nomme Custom[Element], voient leur style issu de Swing totalement enlevé afin d'être remplacé par nos éléments de style : Custom[Element]UI.

Nouveau CustomElementUI → Personnalisation → Ajout de CustomElementUI dans CustomElement

(Image ElémentsCustomisés.1 : Schéma)

Les éléments de style stockent donc les différents styles mais aussi l'état actuel de l'élément. Ils prennent ensuite en charge de dessiner correctement l'élément à chaque fois que cela est nécessaire.

Afin de ne pas être limité au niveau du design de l'application, les éléments de style sont personnalisables sur de nombreux points allant de la couleur de fond au style du curseur en passant par la bordure et la couleur du texte, le tout au repos (aucune action faite sur l'élément), au survol, au clic et à la sélection.

Voici les différents éléments repris et personnalisés de l'API :

## Légende

R = s'applique au repos

S = s'applique au survol

C = s'applique au clic

F = s'applique au focus

T = s'applique de la même façon à tous les états "normaux"

SR = s'applique quand sélectionné + R

SS = s'applique quand sélectionné + S

SC = s'applique quand sélectionné + C

SF = s'applique quand sélectionné + F

ST = s'applique de la même façon à tous les états "sélectionné"

A = s'applique de la même façon à tous les états

### 4-2-1. CustomAlert

(Voir annexe)

Correspond à une fenêtre popup.

Ne pas être directement modifié mais les éléments qu'elle contient le peuvent.

## 4-2-2. CustomButton

(Voir annexe)

Correspond à un simple bouton.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
R S C SR SS SC	R S C SR SS SC	R S C SR SS SC	R S C SR SS SC	R S C SR SS SC	R S C SR SS SC	T ST	A

## 4-2-3. CustomComboBox

(Voir annexe)

Correspond à une liste déroulante. Elle ne correspond pas à un seul élément elle est composée d'un CustomLabel, d'un CustomButton et d'un CustomPopupMenu. Donc certaines modifications du style doivent être faites sur ces sous composants.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
			T	T	T	T	T

## 4-2-4. CustomLabel

(Voir annexe)

Correspond à une zone de texte.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
R S C	R S C	R S C	R S C	R S C	R S C	T	T



## 4-2-5. CustomMenu

(Voir annexe)

Correspond à un menu contenu dans un CustomMenuBar. Lorsque l'on clique dessus, cela fait apparaître une liste d'options et de sous menus.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
RS		RS	T	T	T	T	T

## 4-2-6. CustomMenuBar

(Voir annexe)

Correspond à une barre de menus composé de différents CustomMenu.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
T			T	T	T	T	T

## 4-2-7. CustomMenuItem

(Voir annexe)

Correspond à une option d'un CustomMenu.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
T		T				T	T

## 4-2-8. CustomOptionPane

(Voir annexe)

Permet d'afficher différents types de fenêtres popups en fonction du besoin :

- Afficher un message pour informer l'utilisateur
- Demander une confirmation à l'utilisateur
- Demander à l'utilisateur de sélectionner une proposition parmi plusieurs propositions
- Demander à l'utilisateur d'écrire quelque chose
- Ne pas être directement modifié mais les éléments qu'elle contient le peuvent.

## 4-2-9. CustomPane

(Voir annexe)

Correspond à un conteneur. Son rôle est de contenir d'autres éléments.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
R S C	R S C	R S C	R S C	R S C	R S C	T	

## 4-2-10. CustomPopupMenu

(Voir annexe)

Correspond à une liste de CustomMenu et CustomMenuItem qui apparaît à l'endroit souhaité. Cet élément est utilisé dans les CustomMenu ou les CustomComboBox, par exemple, afin d'afficher leurs options et sous menus.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
T			T	T	T		

## 4-2-11. CustomProgressPopup

(Voir annexe)

Correspond à une fenêtre popup affichant l'état d'un avancement en valeurs brutes ou en pourcentage.

Ne pas pas être directement modifié mais les éléments qu'elle contient le peuvent.

## 4-2-12. CustomTextArea

(Voir annexe)

Permet à l'utilisateur d'écrire un texte.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
R S F		R S F	R S F	R S F	R S F	T	T

## 4-2-13. CustomTextField

(Voir annexe)

Permet à l'utilisateur d'écrire un mot quelconque.

Couleur de fond	Image de fond	Couleur du texte	Couleur de la bordure	Taille de la bordure	Bordures montrées	Type de curseur	Police
R S F		R S F	R S F	R S F	R S F	T	T

## 4-2-14. CustomWindow

(Voir annexe)

Correspond à une fenêtre simple.

Ne pas pas être directement modifié mais les éléments qu'elle contient le peuvent.

### 4-3. Utils

L'équipe a créée des classes et des méthodes pour factoriser du code que nous avons souvent utilisé :

- Convertir une chaîne avec des caractères échappés (&#10;) en caractères lisibles
- Retourner tous les fichiers d'un dossier, selon des extensions (optionnel), en le retirant (au choix)
- Retourne une chaîne sous sa notation camelCase depuis sa notation Snake (A\_SNAKE => aSnake)
- Créer une instance d'une classe depuis son nom (chaîne de caractère)
- Appel d'une méthode depuis son nom (chaîne de caractère)

### 4-4. LibGdx

Nous avons créé des classes pour nous aider à utiliser la libGdx. Par exemple, tous les personnages, les objets... sont considérés comme des "acteurs" aux yeux de la LibGdx, mais ils ont des propriétés différentes (par exemple un livre à un texture (=une image), un personnage peut se déplacer...).

Quelques classes :

- GameActor, une amélioration des plus basiques de actor avec des méthodes utiles comme la collision en plus
- GameActorAnimation, pour les GameActor pouvant se déplacer et ayant une animation de déplacement
- GameActorTextured, pour les GameActor ayant une texture

On a également créé des classes pour faciliter le chargement de la libgdx dans une application swing, sa fermeture...

### 4-5. Annotations

Nous avons créé nos propres annotations, c'est à dire des indicateurs dans le code, qui transmettent un message aux programmeurs.

Par exemple, l'annotation @Override indique la surcharge d'une méthode donc utiliser deux fonctions qui ont la même signature (nom et paramètres).

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.SOURCE)
public @interface Override {
}
```

(Image Annotation.1 : L'annotation Override)

Les annotations ne contiennent généralement pas de code mais on peut en ajouter. Cela va déterminer les arguments de l'annotation.

Par exemple `@Temporary(reason= "une raison")`. Celà signifie que l'on a une méthode dans `Temporary` appelée "reason" et qu'on lui donne la valeur "une raison".

Quelques annotations :

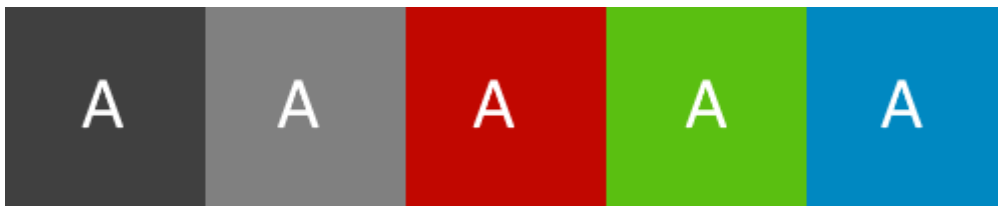
- `ConvenienceClass`, `ConvenienceMethod` sont nos annotations pour indiquer une classe/méthode qui n'existe que pour des raisons pratiques
- `Immutable`, indique une classe `Immutable`, donc on ne peut pas changer les attributs/l'objet sans en créer un nouveau

## 5. Charte graphique

La charte graphique de l'application se limite à une couleur principale : gris foncé et du blanc pour le texte.

Cependant, des couleurs secondaires peuvent être utilisées telles que du gris clair, du bleu ou du vert. La principale couleur secondaire étant du rouge.

Des nuances de ces couleurs secondaires peuvent être utilisées.



(Image CharteGraphique.1 : nuances de ces couleurs secondaires)

La couleur des éléments de survol, sauf exceptions si la situation s'y prête, est le bleu clair. Les exceptions possibles sont si la couleur de fond est changée à la place.

## 6. Tests

Comme vu dans la partie Outils (9.) avec JUnit, nous n'avons fait que quelques tests sur les parties que nous pouvions facilement tester, car pour la majeure partie, reproduire un environnement de test intéressant était trop compliqué. Nous avons également fait des tests sans JUnit, juste du code à lancer.

### 6-1. Exemples de tests

Voici quelques tests que nous avons fait :

- Test sur les énigmes (Conditions, création d'une énigme)
- Affichage d'une entité après lecture de sa texture
- Test sur la map (structure)
  - Structure d'enregistrement des entités dans la map
  - Test d'ajout de salles
  - Sauvegarde et rechargement de la map
- Test de la lecture et sauvegarde des énigmes
- Test de la distribution des identifiants aux entités
- Test de la lecture des fichiers .atlas et de leur utilisation

### 6-2. Système de Log

La majeure partie de nos "vérifications" a été faite à l'aide d'un système de debug que nous avons mis en place et donc nous consultons les messages pour vérifier que les actions se sont faites correctement. Nous avons trié les messages en types :

- debug : des messages de débogage
- debugAll : tous les messages de débogage
- error : les messages d'erreur
- info : des messages d'informations

Le programmeur peut définir quels types de messages il veut afficher.

Après l'établissement de ce système, nous affichons un maximum de message, que les actions se passent bien, ou mal.

## 7. Problèmes

Tout le développement ne s'est pas très bien passé, voici quelques un de nos problèmes.

### 7-1. Migration de la gestion de maps dans l'éditeur

À la suite de problèmes avec JAVA FX (crash sur certains ordinateurs Linux ou Windows) et car le drag and drop (Swing) ne marchait pas avec la libgdx sous Windows, nous avons décidé d'utiliser les classes Libgdx pour gérer la map, la librairie et le drag and drop de l'éditeur.

Cela a rendu énormément de code obsolète (presque toutes nos classes créées pour gérer la map, la librairie et le drag and drop) et nous a fait perdre beaucoup de temps.

### 7-2. Git

Nous avons de nombreux problèmes avec Git. Lors des merge (fusion de nos codes), il arrivait de voir du code se supprimer, ou de provoquer des problèmes non visibles, et qui ne sont détectables qu'à l'utilisation.

### 7-3. Problèmes matériels

L'ordinateur de l'un de nos membres n'était plus en état de fonctionner durant les vacances de Noël, ce qui a réduit notre avancée et nous a retardé.

### 7-4. Libgdx

La libgdx s'est révélée plus compliquée que prévue notamment en termes d'affichage mais l'association de plusieurs membres nous a permis de ne pas bloquer trop longtemps dessus.

## 8. Organisation

Nous avons décidé d'utiliser la méthode agile Scrum pour mener à bien notre projet. L'objectif initial était de réaliser toutes les 2 semaines un sprint. Nous avons une réunion en groupe le jeudi après-midi.

Nous voyions notre tuteur toutes les 2 semaines le vendredi soir où nous discutons du résultat du sprint et du prochain sprint à venir.

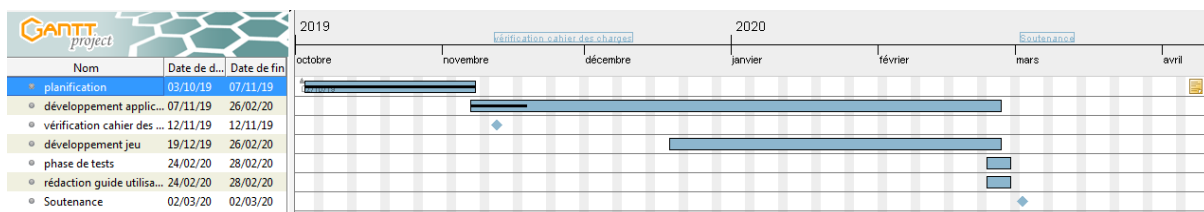
Nous avons réussi au début à tenir ce rythme avant les vacances de décembre nous avons entamé un sprint beaucoup trop ambitieux il s'est fini 1 semaine plus tard.

Les réunions avec le tuteur ont toujours été maintenues, sauf pendant la période des partiels de janvier, le projet n'ayant pas beaucoup avancé durant 2 semaines ce n'était pas nécessaire.

Nous nous sommes réparti les tâches d'abord en fonction des envies de chacun et ensuite en fonction des besoins pour avancer le plus rapidement possible et éviter de bloquer des membres du groupe qui se retrouverai sans travail.

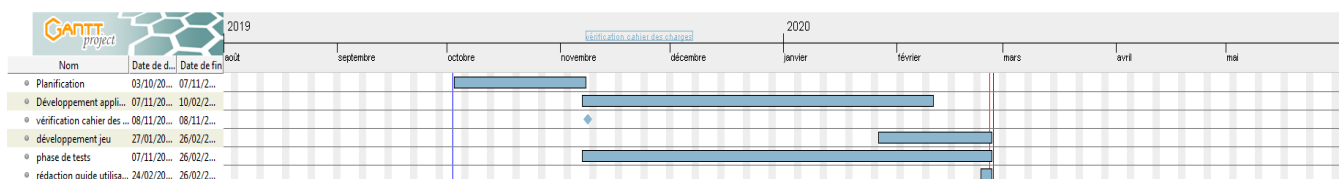
Au quotidien nous travaillons chacun sur sa machine dès qu'on était bloqué ou qu'on avait besoin d'information nous communiquons entre nous. Le jeudi après-midi nous restions tous à l'IUT pour travailler le projet. Lors de notre temps libre à l'IUT et hors IUT nous avançons également.

Voici ce que nous avons prévu initialement :



(Image Organisation.1 - Diagramme de Gantt prévisionnel)

Voici ce que nous avons fait finalement :



(Image Organisation.2 - Diagramme de Gantt réel)

Nous avons sous-estimé le travail à faire sur l'éditeur avant de commencer à développer le jeu et nous avons surestimé le travail concernant le jeu, finalement nous avons réussi à implémenter le jeu en un peu plus d'un mois. Nous avons effectué des tests tout au long du développement, ce n'est pas pertinent de faire des tests juste avant de rendre un projet car nous aurions pas eu le temps de corriger les bugs. Le guide utilisateur a bien été commencé au moment prévu.



## 8-1. Classe de configuration

Il existe des données, telles que la taille maximum d'une map, les chemins de certains fichiers, les extensions supportées... qui sont utilisées dans beaucoup d'endroits du code.

Nous les avons centralisées toutes ces données dans une classe dite "Config".

## 9. Outils et technologies

### 9-1. Java 11

Langage orienté objet, nous avons utilisés ce langage principalement parce que nous l'avons appris à l'iut. La portativité du langage est un avantage intéressant tout sa compatibilité avec un framework/bibliothèque adapté à notre projet la Libgdx.

Notre projet contient 491 classes (=491 fichiers java), avec dans certains fichiers, si c'était raisonnable, plusieurs classes.

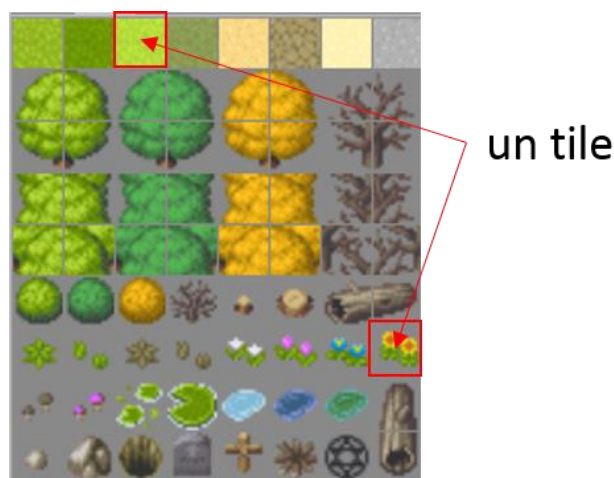
Vous pourrez voir nos techniques de programmation JAVA utilisées dans la partie 10 (Concepts appris).

### 9-2. LibGdx 1.9.10

Libgdx (<https://libgdx.badlogicgames.com/>) est un framework open source compatible avec le Java. Il permet de créer des jeux qui tourne sur Pc et également sur android. Nous avons choisi d'utiliser ce Framework car un des membres du groupe avait déjà des connaissances dessus de plus il est fait pour le Java, langage que nous connaissons tous.

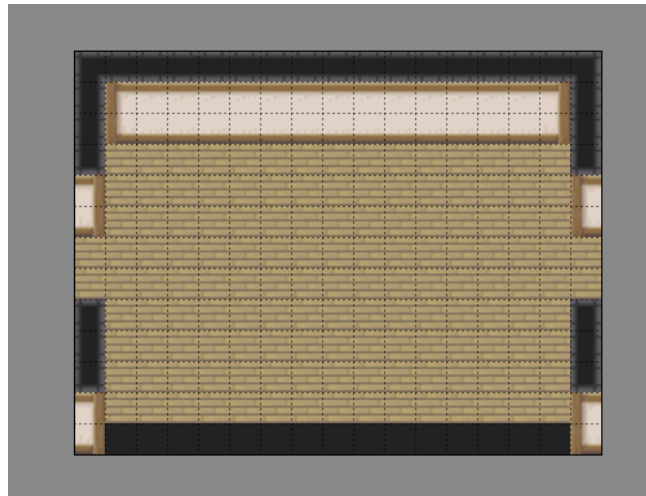
### 9-3. Tiled

Tiled (<https://www.mapeditor.org/>) est un logiciel (open-source) permettant de créer map depuis des images appelées Tiles (il s'agit de petits cubes, voir image Tiled.1).



(image Tiled.1 - Des tiles)

En assemblant plusieurs tiles, on peut créer des objets (par exemples un arbre, une salle...) et c'est avec ce logiciel que toutes nos salles ont été construites.



(Image Tiled.2 - Une de nos salles dans Tiled)

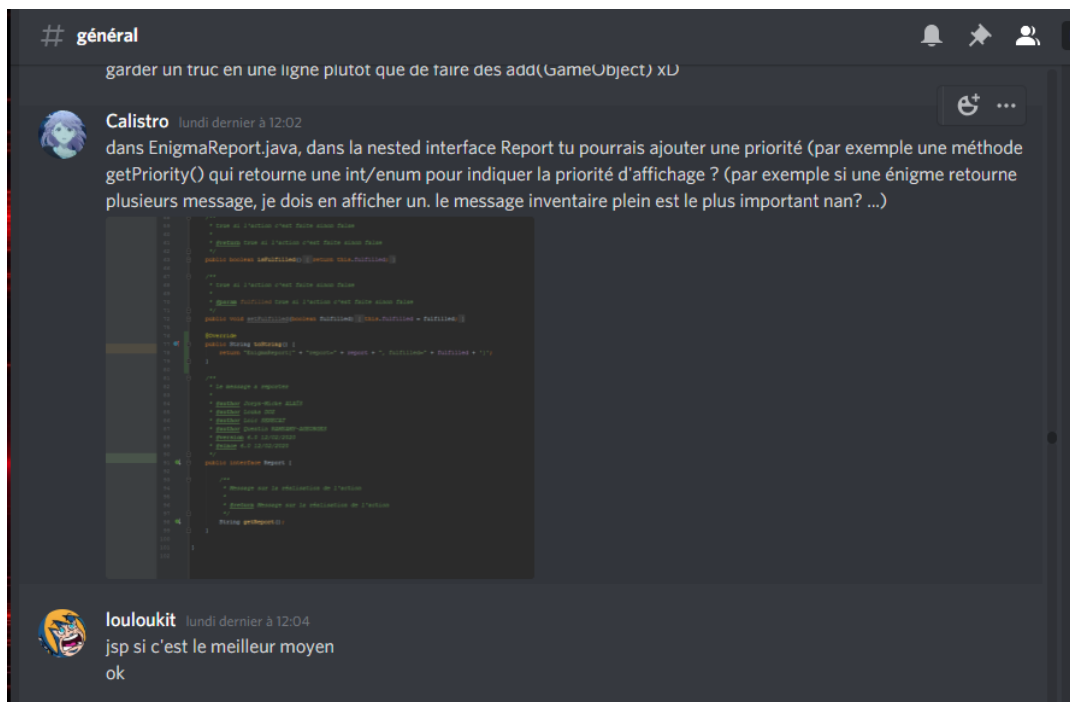
Le format de nos map est le même que celui de tiled, son extension est .tmx. Utiliser le même format, nous a permis de vérifier nos map dans tiled, quand l'éditeur n'était pas encore prêt pour afficher nos map. Et cela nous a permis d'utiliser des classes Libgdx qui comprenaient ce format. (<https://github.com/libgdx/libgdx/wiki/Tile-maps>)

## 9-4. Systèmes d'exploitation

On a codé avec sous Window/Linux/MacOs systèmes d'exploitation ce qui nous a permis de voir si il n'y a pas de problème de comptabilité ou un bug présent sur un système en particulier. En théorie Java et la Libgdx sont multi-plateformes cependant il arrive que les comportements d'un programme soient différents en fonction du système sur lequel il est lancé, d'où notre prudence et nos vérifications. Par exemple, la libgdx peut ne pas marcher sous linux si le JRE ([voir glossaire](#)) est supérieur à 1.8, ou encore JavaFx ne marche pas avec la libgdx, sur certains ordinateurs uniquement (windows/linux).

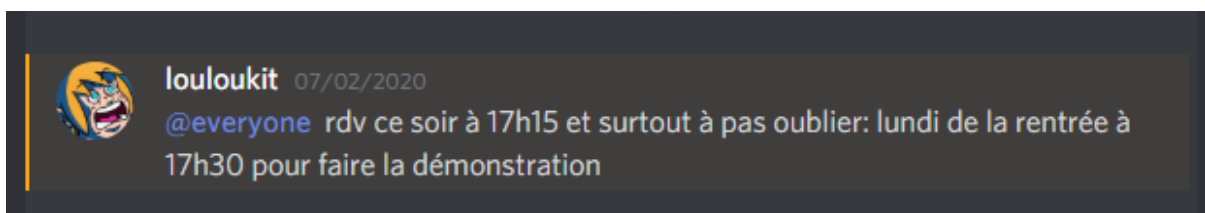
## 9-5. Discord

Discord (<https://discordapp.com/>), est un logiciel spécialisé dans le chat vocal et textuel qui a été conçu et pensé pour les joueurs de Jeux Vidéo. C'est un logiciel très pratique par sa gestion des chats textuel et vocal, l'échange de fichier est rapide et fluide. Sa disponibilité sur téléphone et sur navigateur est aussi un atout. Les développeurs ont l'habitude de s'en servir, nous avons donc décidé de l'utiliser.



(Image Discord.1 - échange entre des membres)

Ce fut un bon choix, on peut facilement notifier tous les membres en même temps.



(Image Discord.1 - notification à tous les membres)

## 9-6. Github

GitHub (<https://github.com>) est un outil de gestion de fichier qui permet d'avoir différentes versions d'un fichier et de les mettre en communs.

On a plus de 460 commits, sur plus de 36 branches (certaines ont été supprimées).

On a décidé de l'organisation suivante :

Pour chaque nouvelle fonctionnalité majeure à coder on créait une nouvelle branche sur git. On avait la branche master qui était celle de base. Pour fusionner avec master une branche devait être fonctionnelle et ne pas comporter de bug majeur. Cependant lors des merges il arrivait que des bugs s'insèrent notamment des problèmes de lancement de l'application.

On a également utilisé, le système d'issues sur Github pour signaler des problèmes.



**Problèmes liste déroulante libgdx et swing #23** New issue

Closed QuentinRa opened this issue on 29 Dec 2019 · 1 comment

QuentinRa commented on 29 Dec 2019

La liste déroulante ne s'affiche pas si on ajoute des items dedans, le problème viendrait de `setFirstElement` et la ligne `this.label.setText(item.getText());`.

Problème également si on essaye de fermer l'application et on est dans la liste déroulante. L'écran devient noir comme dans le premier cas.

utiliser ta `EnigmaComboBox` même si elle marche très bien, genre la `libgdx` se désactive si on fait `addItem` sur la combo box, je ne sais pas d'où ça vient mais je sais que dès que je commente cette ligne ça marche sauf que la combo box est vide (ce bug ce faisait déjà avant que je fasse la javadoc et tout)

ya aussi ça c'est vraiment bizarre apparemment le bug vient de `setFirstElement` (ça marche un peu dès que je l'ai commenté mais ça fait cette barre)

re	v	fit	in
25%			
50%			
100%			
125%			
150%			
175%			

Assignees: LoukaDOZ

Labels: bug

Projects: None yet

Milestone: No milestone

Linked pull requests: Successfully merging a pull request may close this issue. None yet

2 participants: LoukaDOZ, QuentinRa

(Image Github.1 - Issue #23, listes déroulantes)

## 9-7. IntelliJ IDEA

IntelliJ IDEA (<https://www.jetbrains.com/fr-fr/idea/>), un environnement de développement puissant adapté pour le Java. Il permet de gérer les imports de ressource extérieure automatiquement, de générer de la Javadoc. Il a également une interface qui permet d'utiliser git.

## 9-8. JUnit 4

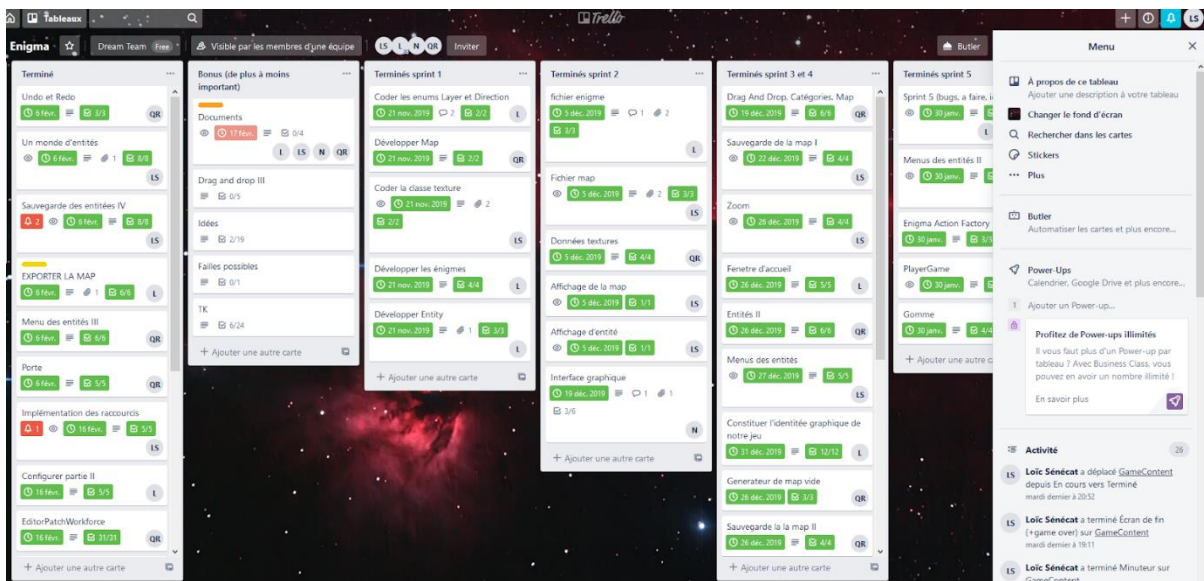
Nous avons fait quelques fichiers de test, avec JUnit4. Cependant la majeure partie de notre programme est faite lié à la vue, et il nous a paru plus efficace d'utiliser un autre système pour nos tests liés à la vue. (Voir partie 6. Test)

## 9-9. Trello

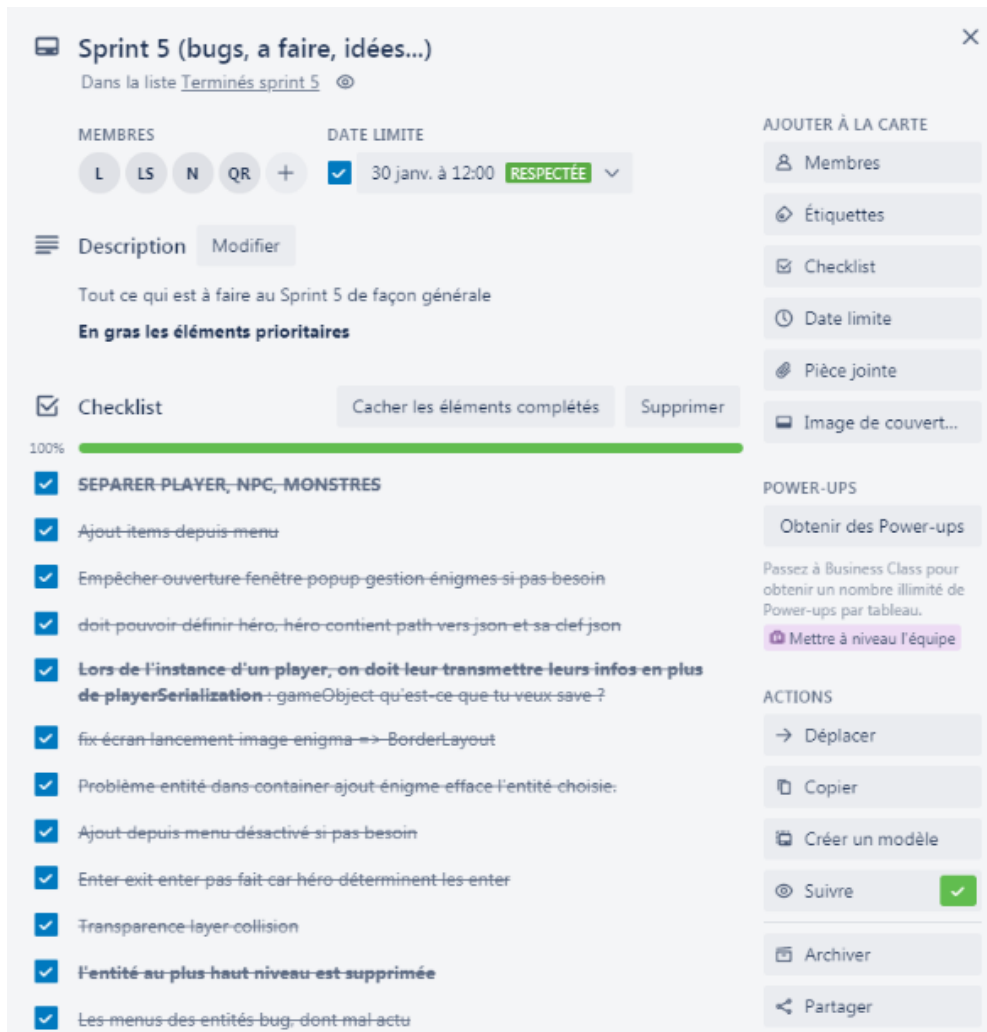
Trello (<https://trello.com/>), c'est un site web qui permet de visualiser des cartes et de pouvoir les ranger. Cela permet d'avoir un Scrum board virtuel très pratique. On peut assigner des personnes à une tâche, on peut bien décrire les tâches à réaliser. C'est sur ce site qu'on organisait le partage du travail. Chacun pouvait rajouter une carte quand il le voulait, généralement c'était à la suite de réunions de groupe qu'on établissait les cartes.

On peut assigner des membres à des cartes, une date limite, donner des descriptions en Markdown (met en avant les éléments importants), déposer des fichiers utiles ou des captures d'écran.

Un autre côté très pratique est qu'il existe une application mobile.



(Image Trello.1 - Trello)



**Sprint 5 (bugs, a faire, idées...)**  
Dans la liste [Terminés sprint 5](#)

MEMBRES: L, LS, N, QR, +  
DATE LIMITE: 30 janv. à 12:00 **RESPECTÉE**

Description **Modifier**  
Tout ce qui est à faire au Sprint 5 de façon générale  
**En gras les éléments prioritaires**

Checklist **Cacher les éléments complétés** **Supprimer**  
100%  

- ✓ **SEPARER PLAYER, NPC, MONSTRES**
- ✓ Ajout items depuis menu
- ✓ Empêcher ouverture fenêtre popup gestion énigmes si pas besoin
- ✓ doit pouvoir définir héro, héro contient path vers json et sa clef json
- ✓ **Lors de l'instance d'un player, on doit leur transmettre leurs infos en plus de playerSerialization** : gameObject qu'est-ce que tu veux save ?
- ✓ fix écran lancement image enigma => BorderLayout
- ✓ Problème entité dans container ajout énigme efface l'entité choisie:
- ✓ Ajout depuis menu désactivé si pas besoin
- ✓ Enter exit enter pas fait car héro déterminent les enter
- ✓ Transparence layer collision
- ✓ **l'entité au plus haut niveau est supprimée**
- ✓ Les menus des entités bug, dont mal actu

AJOUTER À LA CARTE  
 Membres  
 Étiquettes  
 Checklist  
 Date limite  
 Pièce jointe  
 Image de couvert...

POWER-UPS  
 Obtenir des Power-ups  
 Passez à Business Class pour obtenir un nombre illimité de Power-ups par tableau.  
 Mettre à niveau l'équipe

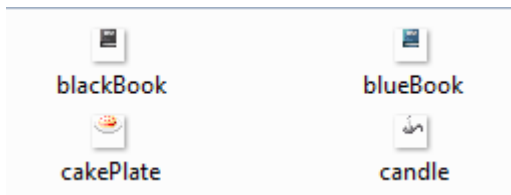
ACTIONS  
 Déplacer  
 Copier  
 Créer un modèle  
 Suivre ✓  
 Archiver  
 Partager

(Image Trello.2 - Exemple de carte Trello)

## 9-10. TexturePacker (version LibGdx)

TexturePacker (<https://github.com/libgdx/libgdx/wiki/Texture-packer>) est un logiciel très pratique permettant de fusionner plusieurs petites images en une seule grande.

L'important est de donner des noms (mémorisables) aux fichiers qu'on lui passe, car ils vont devenir des clefs, donc un moyen de retrouver cette image.



(Image TexturePacker.1 - Futures tiles)



(Image TexturePacker.2 - TileSheet)

Le logiciel génère un fichier .atlas comme vous pouvez le voir sur dans l'image TexturePacker.3. C'est ce fichier qui permet de récupérer les images depuis leur un nom.

```
items.png
size: 256,64
format: RGBA8888
filter: Nearest,Nearest
repeat: none
blackBook
  rotate: false
  xy: 56, 2
  size: 16, 16
  orig: 16, 16
  offset: 0, 0
  index: -1
```

(Image TexturePacker.3 - fichier .atlas)

On sauvegarde toutes les clefs du fichier (blackBook...) et parallèlement leurs informations et on est ainsi capables de pouvoir afficher un tile depuis son nom juste en récupérant la texture à la position x, y.

## 10. Concepts appris et utilisés

### 10-1. Concepts issus des enseignements scolaires

Concepts	Louka DOZ	Loïc SENECA	Quentin RAMSAMY- -AGEORGES	Jorys-Micke ALAIS
Programmation				
Langage JAVA				
Structures itératives bornées ou non bornées	X	X	X	X
Historique				
Classes et Objets	X	X	X	X
Héritage et abstraction	X	X	X	X
Interfaces graphique (SWING, AWT)	X	X	X	X
Mise en page	X	X	X	X
Dessin				
Substitution de Liskov, polymorphisme	X	X	X	X
Programmation événementielle	X	X	X	X
Exceptions	X	X	X	X
Manipulation des flux	X	X	X	X
Récursivité	X	X	X	
Généricité	X	X	X	X
Structures de données	X	X	X	X
Listes	X	X	X	X
Piles			X	
Files				
Arbres	X	X		
Arbres binaires				
Dictionnaires	X	X	X	X
Tableaux	X	X	X	X
Api	X	X	X	X
Javadoc	X	X	X	X



Code couleurs d'échappement ASCII			X	
Tri	X		X	
Tri fusion				
Tri par insertion	X			
Arbre binaire de tri				
Barrel Shift ou décalage				
Thread	X	X	X	
Verrous				
Synchronisation			X	
Réseau				
Client				
Serveur				
Base de données				
Assertions				
Annotations	X	X	X	X
Langage C				
Structures itératives bornées ou non bornées				
Gestion de la mémoire				
Variables globales				
Fonctions				
Manipulation des flux				
Saisie				
Fichiers				
Structures de données				
Listes				
Piles				
Files				
Structures				
Tableaux				
Récursivité				
Fonctions variadiques				
Macros				

Chaines de caractères				
Arguments à la ligne de commande				
Code couleurs d'échappement ASCII				
Tri				
Tri du nain de jardin				
Tri à bulles				
Processus & Threads				
Ordonnancement				
Signaux				
Recouvrement				
Tubes				
Groupe et sessions				
Barrel Shift ou décalage				
Verrous				
Sémaphores				
Mono-threading				
Réseau				
Stream				
Datagramme				
<b>Assembleur</b>				
Barrel Shift ou décalage				
Adressage				
Registres				
Opérations et Conditions				
Intégration d'assembleur dans du C				
<b>SQL</b>				
Gestion de base de données				
Modèle et algèbre relationnelle				
Calculs				
Permissions				

Concurrence				
Stockage				
Optimisation				
PL/SQL				
Procédures				
Trigger				
Exceptions				
Curseurs				
HTML, CSS				
Balises				
Responsive design				
Javascript				
Fonctions				
Dom				
Ajax				
Événements				
Tableaux associatif				
Programmation objet				
JSON				
XML				
PHP				
Tableaux associatif				
CodeIgniter				
Routes				
MVC				
Sessions				
Cookies				
MVC				
Sessions				

Cookies				
Injections				
Erreurs				
Bases de données				
Programmation objet				
DAO				
<b>Bash, Shell, Système (linux)</b>				
Utilisateurs et permissions				
WildCard, expression régulières et patterns				
Bases 2, 8, 10, 16 et conversions				
Shell				
Opérateurs de comparaisons, conditions				
Scripts				
Appels systèmes				
Encodage du système				
Entiers				
Floats				
Calculs				
Processus				
Fichiers	X	X	X	X
Processeur				
Partitionnement des disques				
<b>Réseau</b>				
Adressage IPV4				
Adressage IPV6				
EUI-64				
SLAAC				
Protocoles (UDP, TCP/IP, ICMP, PDU)				

Ethernet				
Tables de routage				
Adresses				
Masque de sous réseau				
Adresse de diffusion				
Notation CIDR				
Notation CHN				
Imunes				
Connaissances générales				
Maths				
Prédicats				
Graphes				
Langages				
Automates	X		X	
Probabilités				
Statistiques				
Logique	X	X	X	X
Matrices		X	X	
Modélisation	X	X	X	X
Bureautique				
Suite Office	X	X	X	X
Anglais				
Professional things				
Memo				
CV				
Hard skills				
Soft skills				
Job interview				

Job applying				
Cover letter				
Grammar and vocabulary	X	X	X	X
Tense				
Present perfect				
Passive form				
Past simple passive				
Present simple passive				
Prepositions				
Compound adjectives				
Linking words				
Collectives				
Funding a small business				
Personal finance				
Flexibility at work				
Video games behavior				
Millennials				
Brexit				
Environment				
COP 24				
Slums				
E-waste				
<b>Communication</b>				
Argumentation	X	X	X	X
Orthographe	X	X	X	X
Historique de la communication				
Rédaction	X	X	X	X
D'un rapport professionnel	X	X	X	X
De son CV				
D'une lettre de motivation				

Ergonomie	X	X	X	X
Droits d'auteurs	X	X	X	X
Chartes graphiques	X			
Entretien				
Travail de vulgarisation	X	X	X	X
Réaliser une interview				
Réaliser un portrait				
Réaliser une soutenance	X	X	X	X
La communication par l'image	X	X	X	X
Les agences de presse				
Modélisation de la communication				
Économie				
Créer une Entreprise				
Factures				
Contrôle de gestion				
Fidéliser ses clients				
Acheter, fabriquer et vendre				
L'état				
Les prix, la monnaie, les marchés				
Droits de l'informatique	X	X	X	X
Droit, la loi et les tribunaux				
Le niveau de vie				
Les échanges internationaux				
Méthodologie, Conception et Outils				
Méthode Agile-Scrum				
Points clefs (Sprint, Sprint Board, Sprint Review...)	X	X	X	X
Roles (Scrum master, Product Owner...)	X	X	X	X

Manifest Agile			X	
<b>Gestion de projet</b>				
GanTT		X		
GanTT en Agile		X		
WBS				
PERT				
Priorisation	X	X	X	X
Gestion des risques	X	X	X	X
Gérer son équipe	X			
Rédaction du cahier des charges	X	X	X	X
Cycle de vie d'un projet	X	X	X	X
<b>Organisation, techniques de programmation</b>				
Pair-programming			X	X
<b>UML</b>				
Diagrammes d'activité	X		X	
Diagrammes séquences	X			
Diagrammes de cas d'usages	X	X	X	X
Diagrammes de classes et d'objets	X	X	X	X
Synopsis	X	X	X	X
Cadres d'interaction				
StarUML	X	X	X	X
Wireflow et Wireframe	X			
Maquettes				
Diagrammes de classes vers SQL, les 7 règles de Pierre VALARCHER				
<b>IHM</b>				
Les points clef (habitudes, normes, directives, ...)	X	X	X	
10 Principes généraux de Jakob Nielsen			X	
Comprendre ses utilisateurs		X	X	



Tests				
Tests unitaires	X	X	X	X
Cycle en V				
JUnit	X		X	
Assertions			X	
Méthodes de Test				
Test-driven développement				
Behavior-driven développement				
Programmation défensive	X	X	X	X
Programmation par contrat				
RightBicep et Correct				
Patrons de conception				
Factory, AbstractFactory	X	X	X	
Façade	X	X	X	
Singleton, multi-ton	X		X	X
Composite		X	X	
Itération	X	X	X	X
Décorateur				
Memento				
NullObject			X	
MVC	X	X	X	X
Observateur		X	X	
Proxy		X		
Outils				
SublimeText3		X		
Android Studio				
Vi et Vim				
Autres				
Gcc				
Débogueur (GDB)				

Makefile				
Javac	X	X	X	X
Git (usages basiques): add, pull, commit, push	X	X	X	X
Git (Wiki, Readme)	X	X	X	X
Bibliothèque graphique de l'IUT (X11)				
Markdown	X	X	X	X

## 10-2. Concepts issus d'apprentissages extra-scolaires

Concepts	Louka DOZ	Loïc SENECA	Quentin RAMSAMY - AGEORGES	Jorys-Micke ALAIS
Programmation				
Langage JAVA				
Jar et Manifest	X		X	
Package	X	X	X	X
Javadoc avancée (since, see, deprecated, ...)	X	X	X	X
UI (Swing et Awt)	X			
Créer des annotations			X	
LibGdx	X	X	X	X
Collision		X		X
Stages		X	X	
Sons & musiques		X		
Acteurs	X	X	X	X
Dessin				
Cycle de vie		X	X	X
Textures, Animations	X	X		
Interface (menus...) et affichage de composants	X	X	X	X
Json (Libgdx)	X	X	X	
Skins		X	X	
Atlas		X		
Fichiers		X	X	
Événements	X	X	X	X
Préférences			X	
Javafx			X	X
Gestionnaires de fichiers			X	X
Connaissances générales				

Jeux vidéo				
Sprites, SpriteSheet		X		
Étapes du développement	X	X	X	X
Structure	X		X	
Game Design Document	X	X	X	
Méthodologie, Conception et Outils				
Outils				
Intellij Idea	X	X	X	X
Intellij Idea - Intégration GIT	X	X	X	X
Tiled			X	
Texture Packer			X	
Trello	X	X	X	X
Autres				
JLink			X	
Git (usages avancées) : merge, issues, branches, fork...	X	X	X	X
XML (dom), Tmx	X	X	X	
Fichier .dtd		X	X	

## 11. Bilan Commun

En conclusion, tout ce qui était prévu dans le cahier des charges n'a pas pu être réalisé tel que : le multijoueur ou les combats en temps réel, par exemple.

Les membres du groupe s'accordent à dire que ce projet a été une très bonne expérience notamment car il permet de se mettre en réelle situation professionnelle avec un délai à tenir, une organisation à mettre en place, une autonomie presque totale (puisque qu'il y avait tout de même les conseils des professeurs).

En plus de l'expérience sociale et professionnelle, ce projet a apporté un gain de maîtrise et d'expérience des connaissances tout en apportant de nouvelles.

### Louka DOZ

J'ai attendu ce projet avec impatience car c'est typiquement ce qui me motive et me pousse à m'améliorer, à donner le maximum de moi, avec en plus la promesse de pouvoir réaliser ce que l'on souhaite. Sans surprise, j'ai adoré me lancer dans le développement d'un jeu, ce qui est aussi mon objectif professionnel, et j'espère réellement avoir de nouveau l'occasion de me lancer dans ce genre de projets.

Ce qui a rendu ce projet d'autant plus intéressant, c'est l'implication de tous les membres du groupe dans l'organisation et la réalisation, avec beaucoup de discussions, débats et échanges de connaissances, ce qui a permis une amélioration flagrante de la qualité de mon travail et de mon expérience. Je n'hésiterais pas à garder la même équipe si je devais refaire un projet de cette envergure.

Je me suis porté volontaire afin d'être chef de projet, et même si je ne regrette pas d'avoir expérimenté ce poste, je considère ceci comme le point négatif de ce projet. Je ne pense pas être qualifié pour ce type de poste, du moins pas encore : je manque encore d'expérience, de connaissances et de confiance en soi.

Pour conclure, malgré toutes les difficultés, et même parfois mon manque de connaissance qui a pu poser des problèmes, je ne retiens que des points positifs de cette expérience. De plus, je compte bien terminer ce projet et aller au bout des ambitions de l'équipe.

## Loïc SENECAAT

Ce travail m'a fait découvrir le fonctionnement d'un projet informatique ambitieux, long, et à faire en groupe. J'ai eu l'opportunité d'apprendre tout au long du projet une nouvelle technologie, c'était à la fois un obstacle et une source de motivation. Le fait d'avoir utilisé le Java durant plusieurs mois et d'avoir lu du code provenant de mes camarades me laisse à penser que j'ai progressé dans la maîtrise de ce langage. Je trouve que le groupe de projet a très bien vécu, il n'y a jamais eu d'incident, le courant passait bien et je n'avais pas de problème de communication avec mes collègues. Pour ma part, plus on avançait dans le projet plus je devais passer du temps à comprendre le code de mes collègues avant de moi-même coder. J'ai découvert que pour être informaticien il faut avoir une ouverture d'esprit et adaptation à assez forte pour pouvoir travailler en groupe. Je ne regrette pas d'avoir pris ce sujet et souhaite continuer à travailler dessus.

## Quentin RAMSAMY-AGEORGES

Ayant travaillé seul pour la quasi-totalité de mes projets, j'ai vraiment apprécié cette expérience. Nous avons réussi à avoir une très bonne organisation et une bonne répartition du travail, ce qui nous a permis d'avancer très vite, malgré quelques problèmes qui nous ont fait rebrousser chemin au début du projet. J'avais originellement l'intention de me concentrer sur le jeu, mais l'idée de mes camarades d'inclure certaines fonctionnalités telles que les barres de menus, refaire l'interface de l'éditeur, choses auxquelles je n'aurais prêté aucune importance, m'ont donnée envie de réaliser un logiciel complet et j'ai fini par beaucoup m'amuser à rajouter des petites fonctionnalités pour améliorer notre éditeur. Je suis impatient de finir notre application et de voir des personnes s'amuser dessus, les retours de nos testeurs étant très positifs.

## Jorys-Micke ALAIS

Cette expérience a été vraiment enrichissante pour moi, travailler sur un projet de cette ampleur m'a beaucoup plu. J'ai découvert de nouvelle manière de travailler, de penser et j'ai beaucoup appris de mes camarades.

J'ai pu me rendre compte de certaines de mes difficultés lors de ce projet, ce qui me permettra donc de travailler dessus à l'avenir.

J'ai aimé travailler en coordination avec mes camarades tout en ayant une certaine autonomie, ainsi que notre organisation et notre ambiance de travail. L'idée de faire un jeu m'a semblé compliqué et ennuyeuse à première vue mais voir l'avancée de notre travail jour après jour, a contribué à changer mon opinion, car ça a été une motivation et un plaisir. L'entraide et la discussion au sein du groupe ont été un facteur clé car quelqu'un pouvait toujours m'aider en cas de blocage. Ayant rejoint le groupe en cours de route, je tiens à les remercier de m'avoir permis de prendre part à ce projet car même si l'optique de faire des jeux ne m'intéresse pas pour la suite, cette expérience m'a permis d'en voir les dessous et ne m'a pas déçu.

## 12. Crédits et Ressources

### Gestion du projet

- vitalzigns (<https://vitalzigns.itch.io/1-page-gdd>) : Game Design Document
- Discord (<https://discordapp.com/>) : communication entre les membres du groupe
- GitHub (<https://github.com/LoukaDOZ/Enigma>) : plateforme GIT du projet
- Trello (<https://trello.com/b/pTGVfrp6/enigma>) : plateforme pour le sprint board

### LibGdx

- *Java Game Développement with LIBGDX* (<https://www.apress.com/gp/book/9781484233238>) : explications sur l'utilisation de la LibGdx
- LibGdx wiki (<https://github.com/libgdx/libgdx/wiki>) : wiki officiel de la LibGdx
- Stack Overflow (<https://stackoverflow.com/>) : explication et exemple de fonctionnement de la libgdx

### Textures

- itch.io (<https://itch.io/game-assets>) : différentes textures
- *howtomakeanrpg* (<https://howtomakeanrpg.com/>) : explications sur la structure et de développement des jeux vidéo
- Tiled (<https://www.mapeditor.org/>) : pour créer les designs des salles
- Texture Packer (<https://www.codeandweb.com/texturepacker>) : assembler plusieurs sous-textures dans un même fichier
- Ezgif (<https://ezgif.com/sprite-cutter>) : séparer une texture

## 13. Glossaire

### A

**API** : est une interface de programmation, qui permet l'échange, ou l'utilisation des services d'une application dans une autre application

### C

**Classe** : en programmation orientée objet, la déclaration d'une classe regroupe des membres, méthodes et propriétés (attributs) communs à un ensemble d'objets (voir définition objet).

### J

**JRE** : Java Runtime Environment, permet d'exécuter du Java.

### M

**Map** : version anglaise du mot carte. Désigne l'ensemble de la zone sur laquelle l'utilisateur peut jouer (dans le jeu) ou placer (dans l'éditeur) des salles, personnages, monstres, objets, ...

### O

**Objet** : en programmation orientée objet, un objet est la concrétisation d'une classe (voir définition classe).

### S

**Solo(s)** : signifie solitaire, une seule personne.

### T

**Tile, Texture** : sont des synonymes pour une image. La particularité d'une tile est qu'il s'agit d'une image dans un ensemble d'images, l'ensemble contenu dans un seul fichier (.png, jpg...).



## 14. Annexe

### 14-1. Classes

Cette partie présente les catégories de classes mentionnées et tous les packages (dossier) auxquels elles sont associées.

#### C

**Container** (src.common.entities.types) : une entité qui peut en contenir d'autres.

#### E

**Enigma** (src.common.enigmas.Enigma) : énigmes

#### G

**GameObject** (src.common.entities.GameObject) : désigne l'ensemble des objets du jeu tels que les joueurs, les monstres, les salles, les objets, ...

#### L

**LaunchSound** (src.common.enigmas.operation.LaunchSound) : opération qui permet de lancer un son durant le jeu.

#### U

**Unlock** (src.common.enigmas.operation.Unlock) : opération qui permet de déverrouiller n'importe quel chose verrouillée.

## 14-2. Guide utilisateur

### 14-2-1. Editeur

#### 14-2-1-1. Barre D'outils



##### a. Pinceau

Le pinceau correspond au mode de l'application.

Vous pouvez déplacer des éléments d'une catégorie vers la map.

Il permet également d'accéder au menu des entités en faisant un clic gauche sur une case.

##### b. La gomme

La gomme supprime une entité en cliquant dessus.

Si c'est une salle, alors tout ce qu'elle contient est supprimé.

Pour plus de précision, vous pouvez supprimer directement une entité via son menu (cf image plus haut).

##### c. Déplacement

L'outil de déplacement vous permet de quitter le mode pinceau.

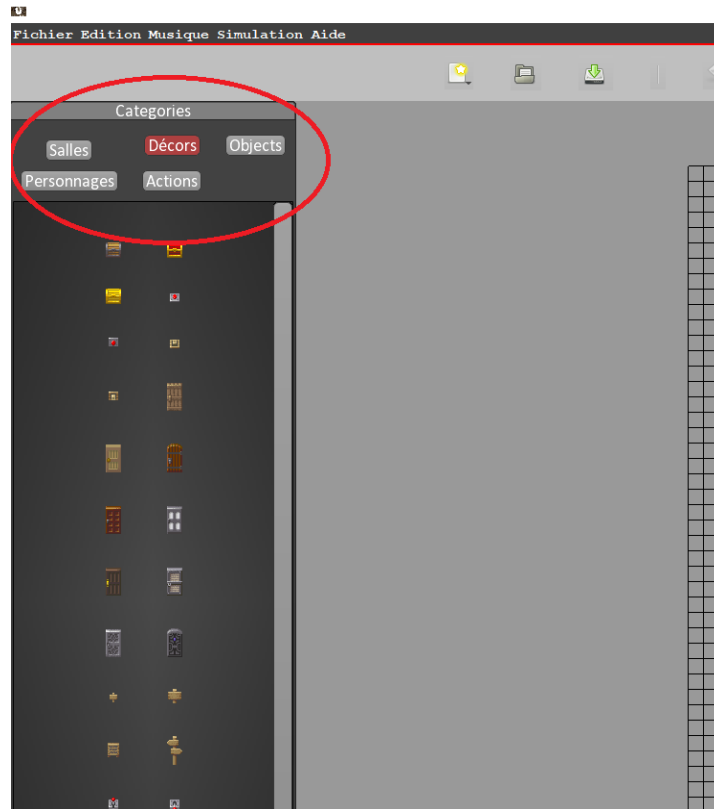
Si vous cliquez alors sur la map, vous pouvez vous déplacer avec les flèches de votre clavier.

### 14-2-1-2. Musique

Pour définir la musique de fond il faut aller dans **Musique > Musique ambiante** et ensuite vous aurez un menu avec toutes les musiques.

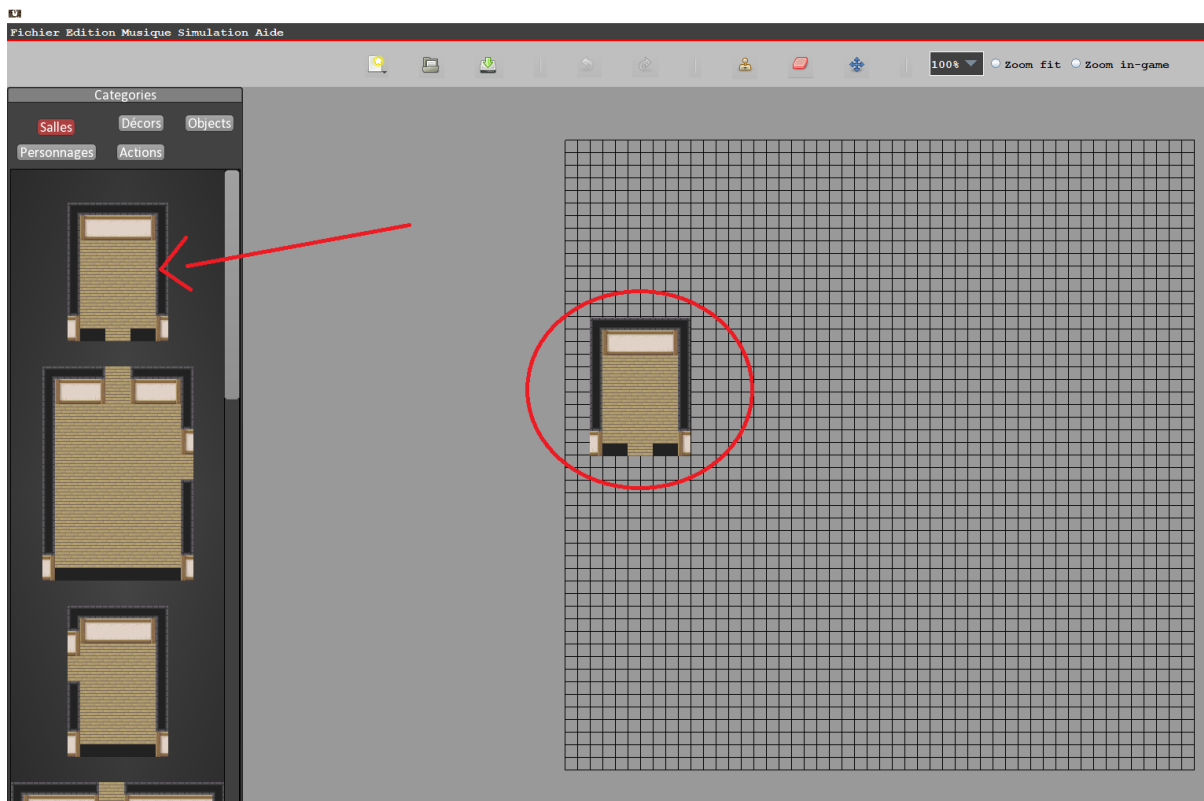
### 14-2-1-3. Catégories

Ce menu permet de sélectionner différent type d'objets que vous pouvez sélectionner puis glisser sur la carte à droite.



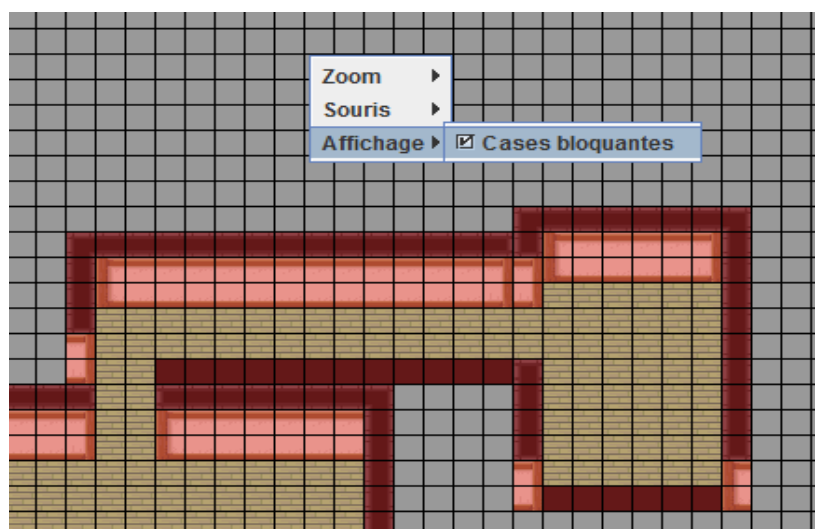
## Salles

Vous pouvez placer des salles en cliquant dans la catégorie salles. Pour créer une salle vous devez maintenir votre souris sur une des salles disponibles à gauche de la fenêtre puis faite glisser sur la carte présente à droite.



La disposition des salles dans la map est libre. Les salles peuvent se chevaucher, à condition que cela n'écrase pas d'éventuelles entités.

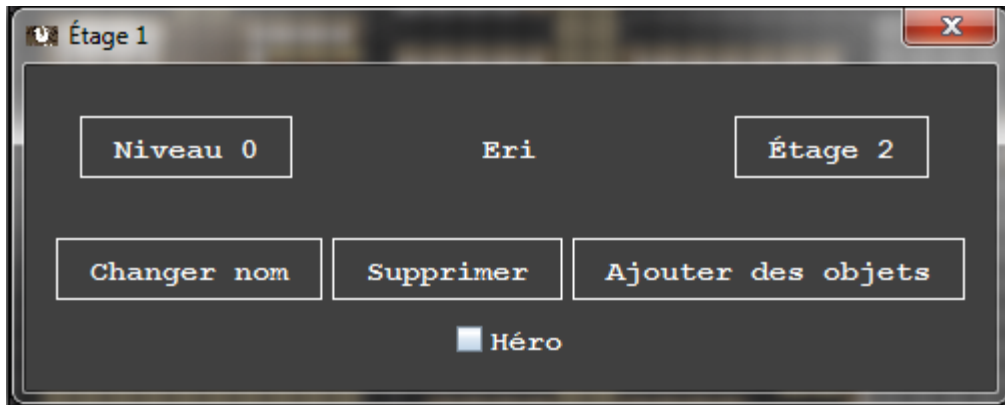
Attention ! Bien que la disposition soit libre, il est de votre devoir de tester que tous les endroits restent accessibles. Afficher les cases bloquantes peut grandement vous aider, en cochant : **Clic gauche > Affichage > Cases bloquantes**



## Personnages

Vous pouvez ajouter depuis la catégorie "Personnages" des monstres et des joueurs.

Pour qu'un joueur soit contrôlé par un humain, alors il doit être déclaré comme "héros".



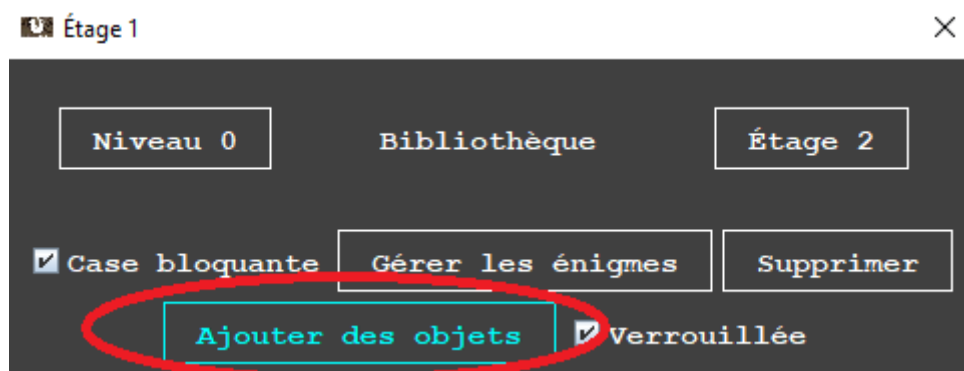
Vous pouvez changer le nom de vos personnages et leur donner des objets dans leur inventaire.

## Décors

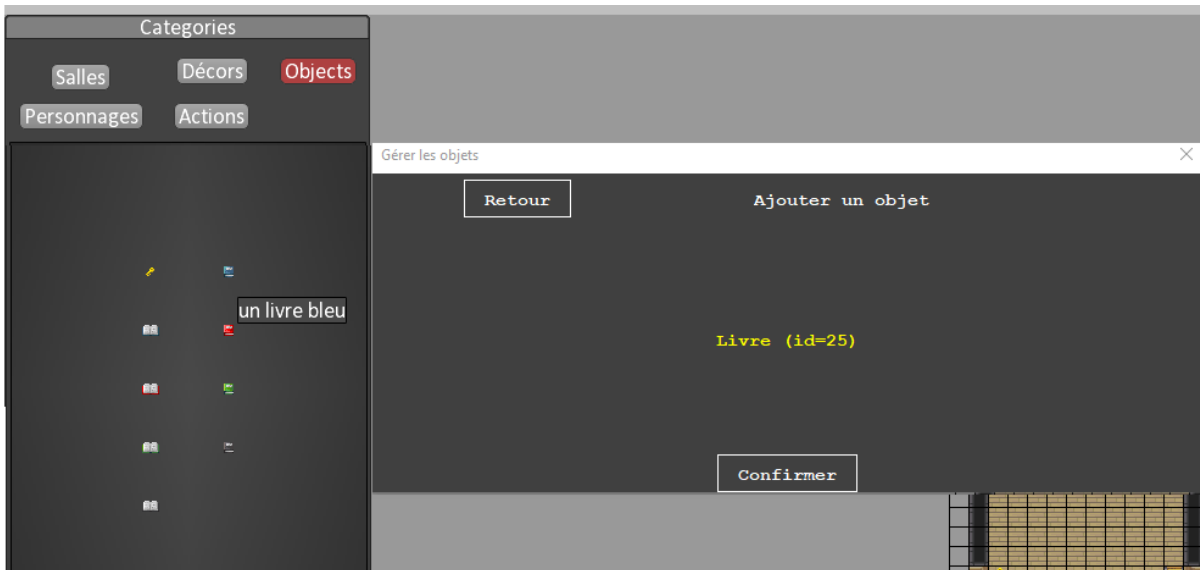
Vous trouverez dans cette catégorie tous les éléments de décors que vous aurez besoin pour créer vos énigmes, coffre, porte, bibliothèque, armoire, levier et bouton.

## Objets

Les objets sont des éléments qui peuvent être contenu dans un inventaire d'un joueur ou d'un élément du décor comme une armoire. Par exemple pour ajouter un objet à une armoire ou à tout autre élément du décor qui peut contenir un objet, on clique sur le décor en question, on appui sur le bouton ajouter des objets :



Maintenant vous devez aller dans **catégories > objets** et cliquer sur l'objet que vous souhaitez ajouter.



Vous aurez un menu différent qui permet de voir et d'effectuer des opérations sur les objets sélectionnés tout en pouvant ajouter d'autres objets.

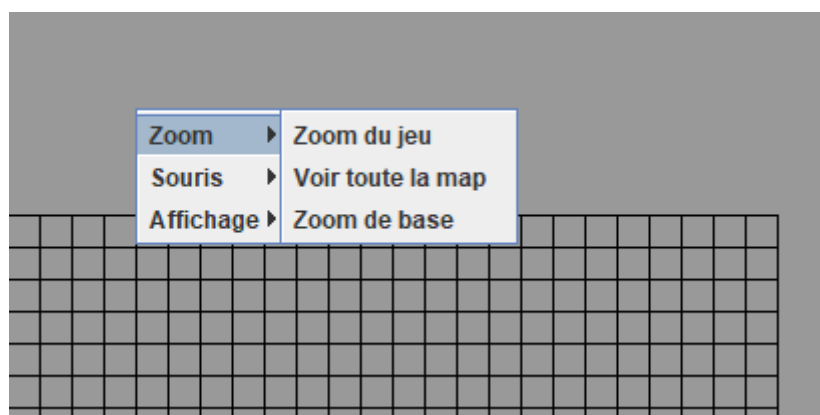
## Actions

C'est dans cette section que l'on va pouvoir placer la sortie de notre jeu.

### 14-2-1-4. Zoom

#### IMPORTANT

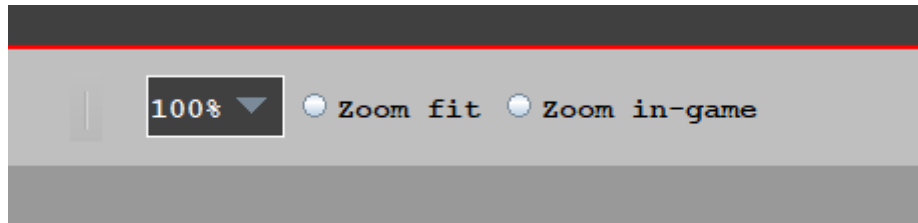
Lorsque vous changez le zoom initial de l'éditeur vous ne pouvez plus rien déposer sur la carte. Si vous désirez déposer un objet vous devez d'abord remettre le zoom de base pour cela **clik droit > Zoom > Zoom de base**



On peut zoomer et dézoomer avec : **CTRL + MOLETTE**.

On peut avoir un aperçu de toute la carte en cochant la case Zoom fit ou bien en faisant **clic droit > Zoom > Voir toute la map**.

Pour avoir un aperçu de la carte lorsqu'elle sera en jeu on peut cocher la case Zoom in-game ou en faisant **clic droit > Zoom > Zoom du jeu**



### 14-2-1-5. Enigme

Les énigmes seront le fer de lance de votre map, vous pouvez en ajouter sur tous les éléments de votre map excepté les personnages et les monstres. L'éditeur vous permet de poser des conditions pour qu'une énigme soit réussie. Quand l'énigme est réussie vous pouvez ordonner au jeu de faire des actions. Vous pouvez également ajouter des indices qui vont apparaître, à l'écran de l'utilisateur.

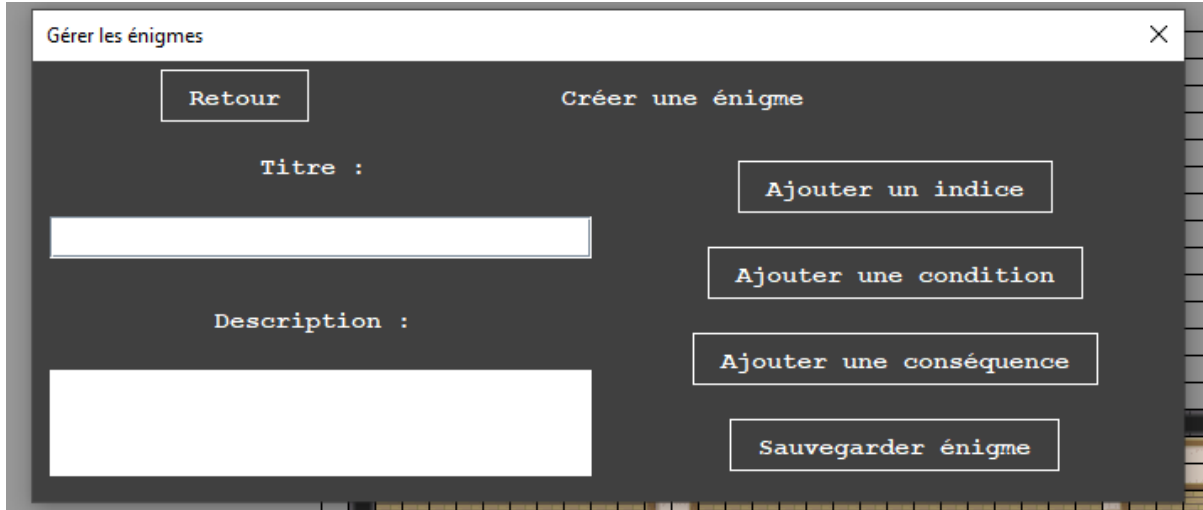
On va prendre l'exemple d'un bouton (b1) ouvrant une porte (p1) :

Je vais ajouter une condition (c1), si b1 est actionné la condition est satisfaite.

Je vais ajouter une conséquence, si toutes les conditions sont validées (c1) alors j'ouvre la porte p1.

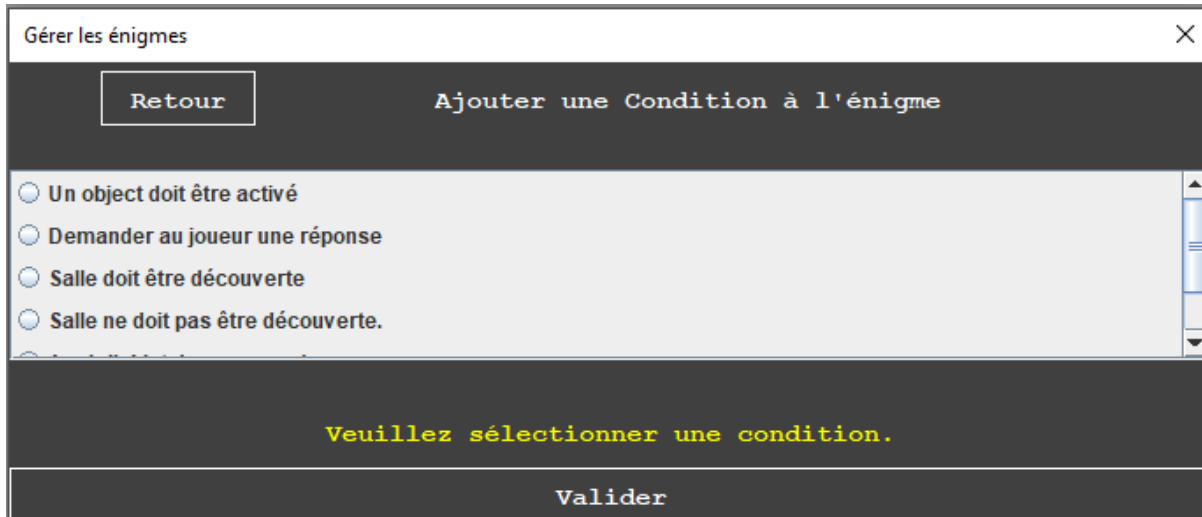
Nous allons voir maintenant comment le faire dans l'éditeur.

Pour ouvrir le menu de gestion des énigmes vous devez faire un clic gauche sur un élément du décor pouvant contenir une énigme, **clic gauche > Gérer les énigmes > Créer énigme**. Vous tombez sur cet écran :



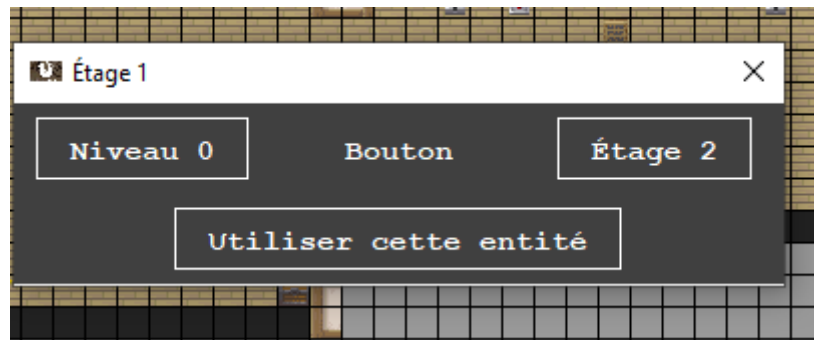
Le titre et la description sont des moyens mnémotechniques pour vous souvenir de vos énigmes.

Pour ajouter une condition choisissez en sélectionnant le type de condition que vous voulez rajouter :



Une fois coché, vous allez devoir **sélectionner un objet** sur la carte qui peut être activé, comme un bouton. Pour sélectionner vous devez juste faire un **clic gauche** dessus. Une fenêtre va apparaître à l'écran :



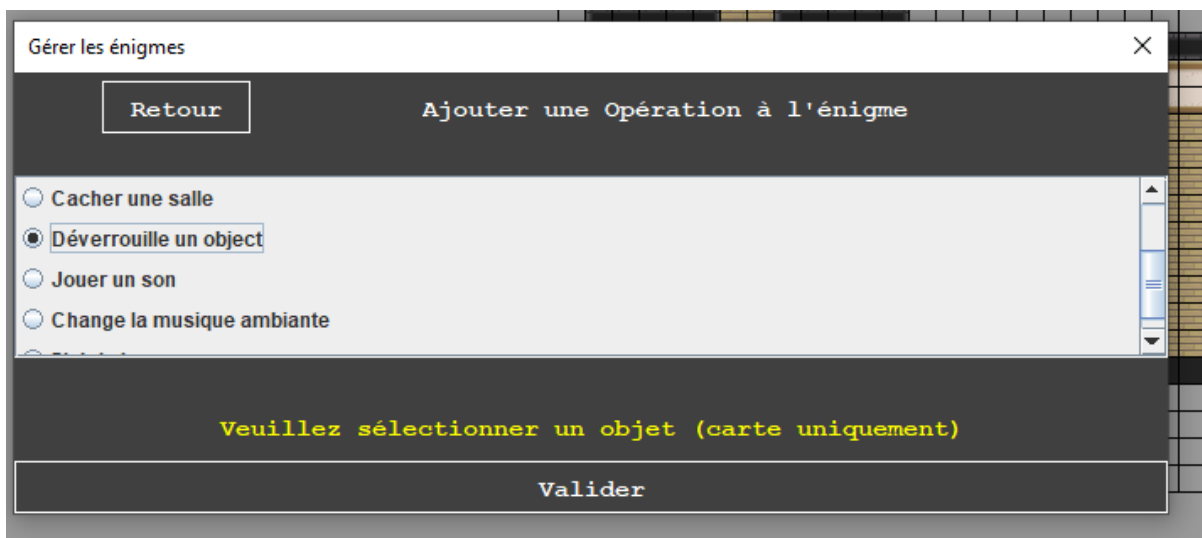


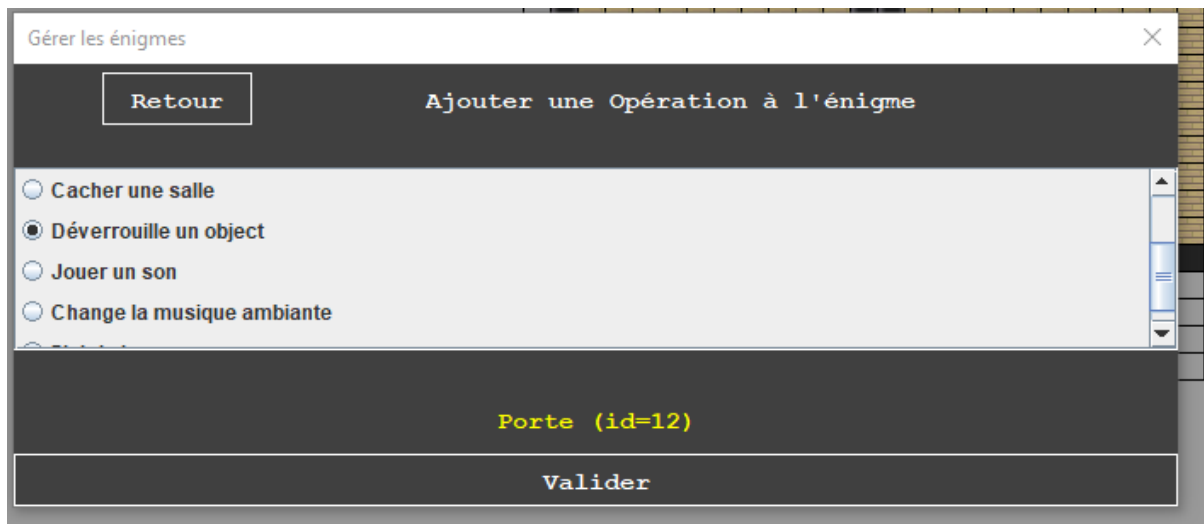
Appuyer sur **Utiliser cette entité** et vous la verrez sur le menu des conditions.



Faite **valider** et la condition est ajouté.

Le fonctionnement est le même pour ajouter une Opération :





Une fois le titre, les conditions et les conséquences définie vous pouvez sauvegarder l'énigme.

## Les conditions

Voici un tableau de toutes les conditions qui sont à votre disposition :

Type de condition	Objet compatible
Activation	Interrupteur, levier, commutateur, bouton
Demande une réponse	*
Salle découverte	Salle
Salle pas découverte	Salle
Tenir un objet dans les mains	Uniquement les objets (voir section catégories)
Tenir un objet dans l'inventaire	Uniquement les objets (voir section catégories)

\* C'est l'objet sur lequel on crée l'énigme qui demandera la réponse au joueur

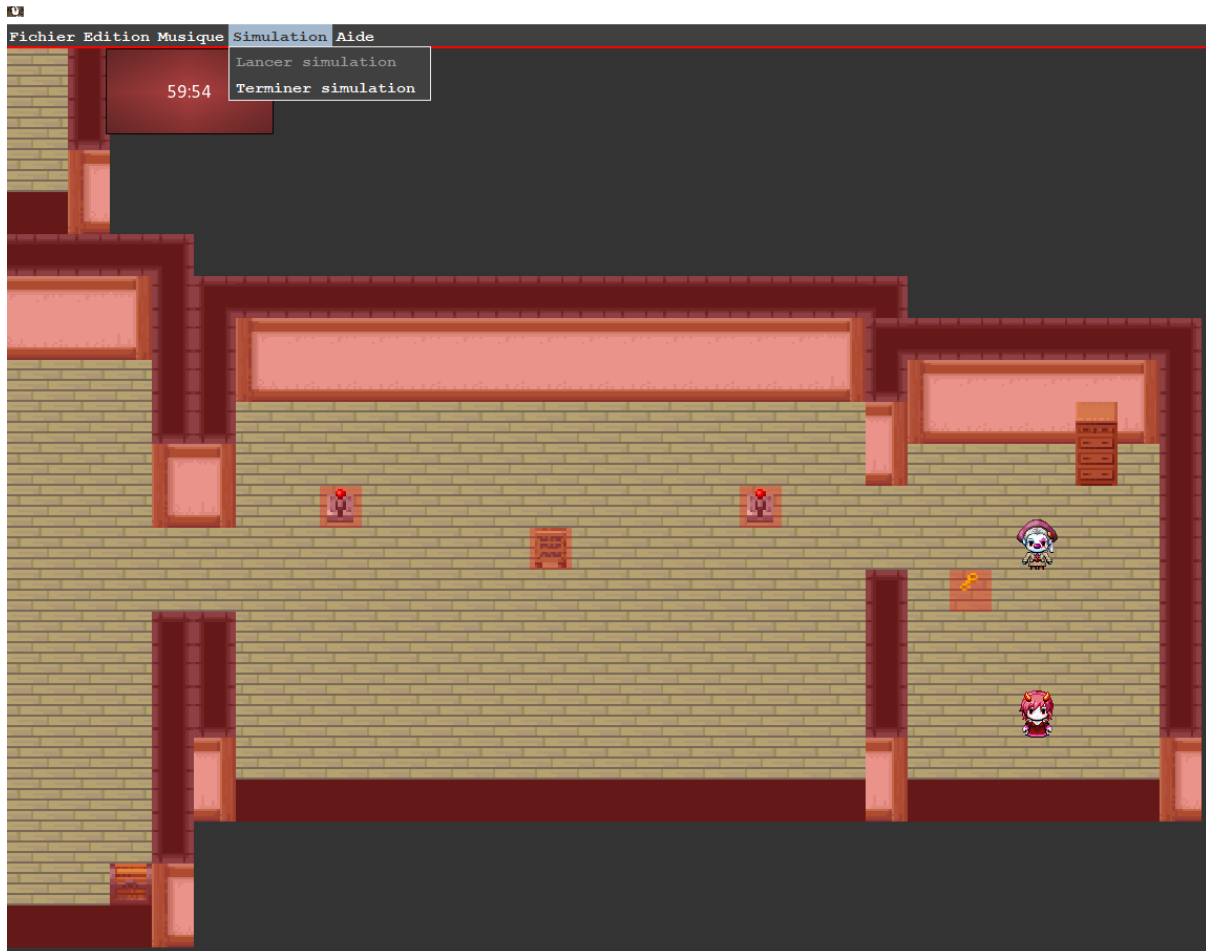
## Les opérations

Voici un tableau de toutes les opérations qui sont à votre disposition :

Type opération	Objet compatible
Donner un objet	Uniquement les objets (voir section catégories)
Invoquer une entité	Personnages
Afficher une salle	Salle
Cacher une salle	Salle
Déverrouiller	Porte, coffre, armoire, bibliothèque
Lancer un son	
Changer musique ambiante	
Finir le Jeu	

## 14-2-1-6. Simulation

Lorsque vous êtes dans l'éditeur vous pouvez tester votre carte. Avant toute chose vous devez sauvegarder votre carte **fichier > enregistrer**. Vous pourrez lancer le jeu en suivant ce chemin **Simulation > Lancer simulation**.



Une fois dedans vous verrez en rouge toute les cases qui ne sont pas accessible au personnage. Vous pourrez tester votre carte (Voir le tutoriel du Jeu). Pour revenir à l'éditeur il suffit d'aller dans l'onglet **Simulation > Terminer simulation**.

## 14-3. Jeu

### Déplacement

Pour déplacer votre personnage dans le jeu, respectivement en haut, à gauche, en bas et à droite, utilisez ces touches **z**, **q**, **s**, **d** ou les **flèches directionnelles**.

### Interaction

Pour interagir avec les éléments du jeu, utilisez la touche **e** ou **Entrée**. Pour fermer les boîtes de dialogue, appuyez de nouveau.

### Inventaire

Pour ouvrir l'inventaire, appuyez sur la touche **i** ou **la barre d'espace**. Pour fermer l'inventaire, appuyez de nouveau.

Une fois dans l'inventaire, pour y déplacer un objet, maintenez le clic gauche et déplacez la souris, puis relâchez sur la case souhaitée. Pour affecter un objet dans vos mains, faites un clic droit sur l'objet.

Pour utiliser l'objet, faites un clic gauche dessus.

Pour jeter l'objet, sélectionner-le, puis cliquez sur **jeter**.

# Enigma

## Cahier des charges

## Synthèse

L'objectif est la réalisation d'un éditeur permettant de créer des escapes games en 2D et d'un lanceur permettant d'y jouer.

La création d'escape games se fera à partir d'une librairie créée l'équipe. Les ressources seront fournies.

Le jeu permettra le lancement de parties solo et, si possibles de parties multi-joueurs.

L'équipe, composée de 4 membres, dispose de 5 mois, 3 mois seront consacrés à l'éditeur et 2 au jeu.

L'enseignant Luc HERNANDEZ est le tuteur du projet.

Le code source, sera disponible sur GITHUB : <https://github.com/PT-2019/Enigma>



## Sommaire

1. Présentation du projet et de l'équipe.....	103
1-1. Présentation du projet.....	103
1-2. Présentation de l'équipe .....	104
2. Spécifications fonctionnelles .....	105
2-1. Créateur d'échappement .....	105
2-2. Lanceur d'échappement .....	107
3. Spécifications techniques .....	109
3-1. Langage et Framework.....	109
3-2. Architecture .....	109
3-3. Modèle de données .....	109
3-4. Conventions .....	110
3-5. Outils .....	110
4. Charte Graphique .....	111
5. Planning et Organisation .....	111
6. Gestion des risques .....	112
7. Annexes .....	113
7-1. Bilan des réunions .....	114
7-2. Échanges avec le tuteur .....	118
7-3. Autres .....	121

## 1. Présentation du projet et de l'équipe

NB. Un Game Design Document est disponible en annexe, décrivant, en une page, le produit attendu.

### 1-1. Présentation du projet

**Livrable** : Est attendu, pour le 26 février 2020, une application permettant de la création d'escape games et d'y jouer en local uniquement.

L'objectif est la création d'un éditeur permettant la création d'escape games, les fonctionnalités attendues sont

- Ajout d'entités (salles, meubles, objets...) sur la map (carte) depuis une librairie
- Ajout d'énigmes basiques
- Personnalisation des entités
- Simulation d'une partie
- Scénarios
- Importer et Exporter des maps

L'application permet également de lancer une partie d'escape game, les fonctionnalités du lanceur et du jeu attendues sont

- Commerce avec des personnages non joueurs
- Liste des escapes games disponibles
- Sélection d'une partie
- Inventaire
- Interactions
- Caméra
- Multijoueur
- Paramètres

**Livrable** : L'application, dont le créateur et le lanceur, sont attendus pour le 26 février 2020 au plus tard.

Le lanceur devra permettre, le jeu d'une partie simple pour un joueur. Il devra être possible de créer des escapes games simples dans l'éditeur.

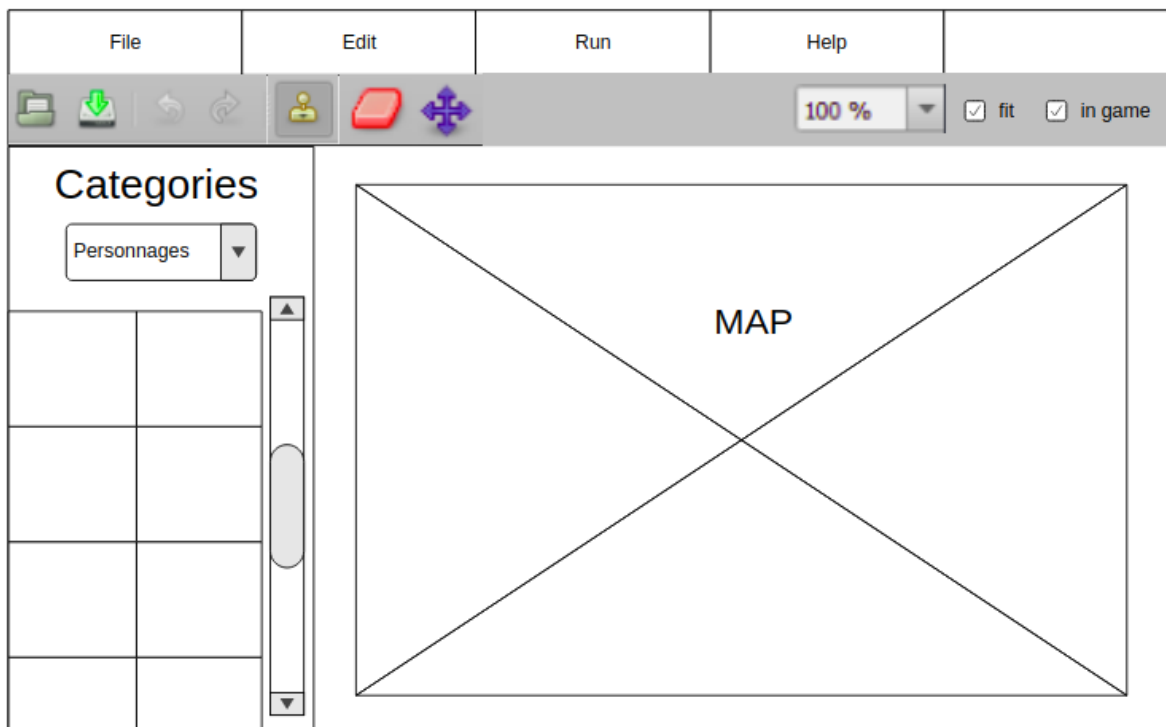
## 1-2. Présentation de l'équipe

L'équipe est composée de :

- Jorys-Micke ALAIS : programmeur
- Louka DOZ : programmeur, chef de projet
- Quentin RAMSAMY-AGEORGES : programmeur, Scrum master
- Loïc SENECAAT : programmeur, responsable de GANTT

## 2. Spécifications fonctionnelles

### 2-1. Créateur d'escape game

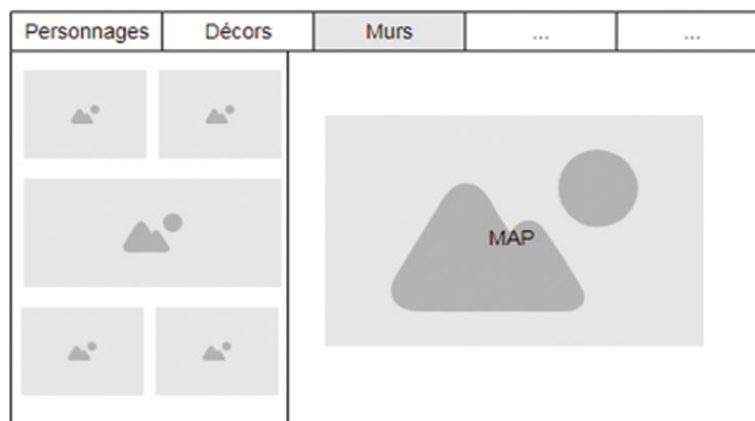


#### 2-1-1. Ajout d'entités sur la map depuis une librairie

La librairie sera composée des entités suivantes :

- Personnages : 3 personnages non-joueurs et 2 monstres
- Pièces : 4 salles
- Objets : un livre, une clef
- Meubles : Armoire/bibliothèque, un panneau, un interrupteur et une porte

Les entités sont triées par catégorie.



## 2-1-2. Ajout d'énigmes basiques

Il faut construire des énigmes de la forme suivante :

« Si <entité=joueur> <action = utilise> <item=clef> <cible =porte> alors  
<conséquence = porte s'ouvre>” »

Il s'agit de conditions pouvant être génériques (on remplace l'entité...).

## 2-1-3. Personnalisation des entités

La personnalisation comprend :

- Possibilité d'ajouter du contenu à un livre
- Ajouter des objets dans des coffres
- Déterminer si une case est accessible ou non (on peut marcher dessus)

## 2-1-4. Simulation d'une partie

Le menu de simulation devra permettre de tester son escape game en jeu.

## 2-1-5. Scénarios

On devrait pouvoir lancer un scénario au lancement d'une énigme.

## 2-1-6. Importer et Exporter des maps

On devrait pouvoir permettre d'importer ou d'exporter nos map créées en ligne.

## 2-2. Lanceur d'écape game

### 2-2-1. Liste des escapes games disponibles

On doit pouvoir consulter la liste des escapes games disponibles.



### 2-2-2. Sélection d'une partie

Sélection dans la liste, d'un escape game. On peut choisir de lancer une partie solo ou multi-joueurs. Chaque joueur peut choisir un nom et un personnage.

### 2-2-3. Inventaire

Il pourra accéder à un inventaire, dans lequel, il pourra consulter ses objets. Il est possible que la taille de l'inventaire soit limitée. Le joueur peut prendre des objets en main.

### 2-2-4. Interactions

Le joueur doit pouvoir interagir avec tout (presque) les éléments du décor (donne indice, énigme, ou rien (affiche)).

On peut lister des cibles d'interactions telles que :

- Portes/coffres, meubles, interrupteurs, panneaux, puzzles, dessins, inscriptions
- Êtres vivants (ennemis, personnages non joueurs, autres joueurs)

### 2-2-5. Caméra

Possibilité de faire pivoter la caméra.

### 2-2-6. Paramètres

Le menu des paramètres permet de changer les touches ou de les consulter. Il permet de régler le son ou encore de gérer le serveur et les personnes dessus, si vous êtes en partie.

### 2-2-7. Combats

Le joueur durant son exploration pourra rencontrer des entités hostiles, auquel cas, il pourra directement engager un combat, avec ses poings et/ou en utilisant des objets tels que des potions ou un objet efficace contre ce monstre. Il pourra être intéressant de permettre des "combats sans combat" tel un combat d'insultes.

### 3. Spécifications techniques

#### 3-1. Langage et Framework

Le langage utilisé sera le JAVA 10, avec sa bibliothèque graphique swing pour l'application de création d'escape games et le framework LIBGDX pour la partie jeu.

Il est possible d'utiliser la bibliothèque JAVAFX, en complément à Swing.

Le logiciel sera disponible sur PC, toutes versions, tous OS.

#### 3-2. Architecture

L'application sera codée dans le logiciel IntelliJ IDEA.

Elle sera sauvegardée sur github : <https://github.com/PT-2019/Enigma>

- LoukaDOZ
- QuentinRa
- SenecatLoic
- Niskey77

Au niveau de l'organisation des dossiers :

- Enigma : racine du projet
- assets : toutes les ressources, triées par catégories (hud, utils, ...)
- libs : les bibliothèques (libgdx, annotations)
- documents : les documents du projets (rapport...)
- src : contient les sources du projet
  - editor : les sources de l'éditeur
  - game : les sources du jeu

Le site trello sera utilisé pour le scrum board : <https://trello.com/b/pTGVfrp6/enigma>

#### 3-3. Modèle de données

Les données seront sauvegardées dans un/plusieurs fichiers ou dans une base de données telle que SQLite.

Si la sauvegarde dans un fichier est choisie, alors le format sera .tmx, et devra pouvoir être ouvert dans l'application Tiled.



### 3-4. Conventions

On a convenu d'appliquer les conventions telles que :

- Le code est rédigé en anglais
- La documentation est en français
- Utiliser les annotations (@NotNull, @Nullable, @MagicConstant)
- Regarder trello avant de coder

### 3-5. Outils

Il est possible d'utiliser les outils suivants :

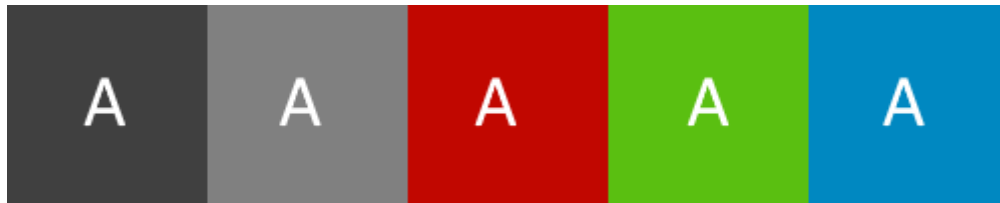
- Trello
- Github
- TexturePacker (assemblage des textures)
- Itch.io (récupération de textures)
- Tiled (construction de la librairie)

## 4. Charte Graphique

La charte graphique de l'application se limite à une couleur principale : gris foncé et du blanc pour le texte.

Cependant, des couleurs secondaires peuvent être utilisées telles que du gris clair, du bleu ou du vert. La principale couleur secondaire étant du rouge.

Des nuances de ces couleurs secondaires peuvent être utilisées.



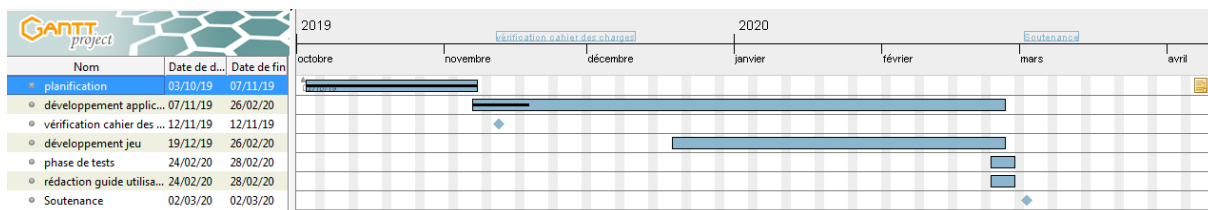
(Image CharteGraphique.1 : nuances de ces couleurs secondaires)

La couleur des éléments de survol, sauf exceptions si la situation s'y prête, est le bleu clair. Les exceptions possibles sont si la couleur de fond est changée à la place.

## 5. Planning et Organisation

L'application sera codée en suivant le modèle MVC et en appliquant la méthodologie de développement AGILE-SCRUM.

Le rythme des sprints a été déterminé : 2 semaines.



L'équipe de développement se réunira au moins deux fois par sprint. (Par exemple chaque jeudi, pour échanger sur leurs avancées et leurs problèmes).

Les rendez-vous avec le tuteur seront faits, au moins une fois par sprint.

## 6. Gestion des risques

Risque	Solution
Un membre quitte l'équipe ou ne peut pas travailler	Il n'y a pas de rôles attribués (x fait les tests, y fait la javadoc) et on se partage les rôles donc chacun d'entre nous peut tout faire
LibGdx trop compliquée	On peut utiliser swing
Une branche GIT est supprimée par erreur	Il existe au moins 4 copies du projet
Le site Trello est temporairement indisponible	Ce wiki et le cahier des charges contiennent toutes les informations nécessaires

## 7. Annexes

Des conseils ont été demandés pour la présentation du projet le 04 décembre 2019 à Luc Hernandez :

- Il souhaite voir les diagrammes expliqués (un petit texte à côté)
- La présentation est tout public : toute la complexité doit être cachée et/ou expliquée
- Pas de code (cf point précédent) ou si code, alors expliqué par exemple avec des algorithmes (diagramme d'activités)
- Montrer comme les "trucs" marchent et l'organisation

Des conseils ont été demandés pour la présentation du projet le 13 décembre 2019 à Luc Hernandez :

- Ne pas oublier d'aborder les points techniques (ne pas les cacher parce que c'est compliqué à expliquer)
- Essayer de parler de tout (ne pas se limiter à la surface)

Des conseils ont été demandés pour la présentation du projet le 20 décembre 2019 à Denis Monnerat

- Bien présenter la problématique. Pourquoi on a fait le projet ? Il y avait-il un besoin ?
- Recul par rapport au sujet/projet
- Formaliser les notions avec UML par exemples les IO
- Voir prezi si on sait bien l'utiliser
- Fluide et regards
- Pas de fautes dans les diapos et on ne lit pas les diapos (guide)
- Ne pas tout traiter (faire sélection des points important et les traiter entièrement)

## 7-1. Bilan des réunions de l'équipe

### 7-1-1. Première réunion

**Horaire** : Jeudi 17 octobre 2019 de 13h40 à 17h40

**Lieu** : Fontainebleau

L'équipe traite la partie création d'escape game, une liste des tâches a été écrite. La bibliothèque sera composée de :

- Personnages : 3 personnages non-joueurs et 2 monstres
- Pièces : 4 salles
- Objets : un livre, une clef
- Meubles : Armoire/bibliothèque, un panneau, un interrupteur et une porte

Lors de la réunion, un DCU de la partie création a été conçu. Il a été décidé de la mise en œuvre du projet : nous réaliserons à 3, les fonctionnalités « must » de la création, puis laisserons Loïc faire les « Should Could Would » pendant que nous nous occuperons de la partie jeu.

Un scénario a été réalisé et la décision à l'unanimité a été prise de ne pas faire de Diagramme séquence. La réunion du jeudi 24 octobre est annulée.

### Questions traitées

**Question** : Présenter une salle puis une autre, ... ou toute la carte d'un coup

**Décision** : Présentation de toute la carte

**Question** : Doit-on afficher des salles préconçues (avec leurs murs) et les « coller les unes aux autres » ou ajouter la salle (sans les murs), le logiciel ajoute automatiquement les murs et si on met une salle à côté, le mur qui les sépare disparaît.

**Décision** : Les salles sont préconçues et l'utilisateur peut seulement supprimer un mur, s'il est entre deux salles. (Donc toute la carte est entourée de murs).

**Question** : Comment le joueur ajoute des actions ? (Catégorie actions ou bouton +/-paramètres) ?

**Décision** : Il peut ajouter des actions depuis catégorie (actions prédéfinies tel que « dialogue ») et avec un bouton +/-paramètre sur chaque entité. Le bouton +/-paramètres permet de consulter les actions mises, en créer, les modifier et les supprimer. (On ne peut créer que les actions faisables sur l'entité).

### Fonctionnalités & propositions

On peut éditer une escape game.

Clic sur une entité, on voit les anciennes actions ajoutées, on peut les modifier/supprimer.

Lors de l'ajout d'une salle, on demande si elle doit être cachée.

Une carte est un dossier avec un sous-dossier/fichier par niveau.

Une bibliothèque contient un seul livre ou plusieurs ?

## 7-1-2. Seconde réunion

**Durée** : Lundi 04 novembre 2019 de 08h40 à 11h15

**Lieu** : Fontainebleau

L'équipe s'est chargée de la rédaction du cahier des charges et de l'élaboration du diagramme de classes du cœur du jeu. Le rythme de sprint a été décidé : 2 semaines.

Un rendez-vous avec le tuteur est pris le mardi 12 novembre 2019 à 17h45.

Les objectifs du rendez-vous sont :

- Lui présenter le cahier des charges, avoir un retour/conseils
- Discuter de l'avancement, du rythme de sprint choisi

Il est prévu pour la prochaine réunion de traiter le sprint planning.

## Questions traitées

**Question** : Comment est la map ? (Grille ou plutôt un espace)

**Décision** : C'est une grille composée de cases et des salles prennent un certain nombre de ses cases

**Question** : Comment on détermine qui affiche un mur (ex : entre deux salles)

**Décision** : On a choisi qu'un mur contient la salle qui va le dessiner, plutôt qu'un booléen que l'on devrait changer 60 fois par seconde (60 mises à jour de l'affichage par seconde).

**Question** : Comment on représente une pièce ?

**Décision** : C'est 2 parties, on dessine d'abord les cases (sol, objets, ...), puis on dessine les murs.

**Question** : Profondeur ?

**Décision** : Une case contient des niveaux, on va afficher par exemple, au niveau 0, le sol, au niveau 1, un tapis, au niveau 2, un joueur... Les niveaux sont rendus de 0 à nombre de niveaux.

**Question** : Contenu des cases ?

**Décision** : On peut placer dans une case, une entité, dont on appellera sa méthode `getTexture` pour la dessiner.

**Question** : Comment se répartir les tâches ?

**Décision** : ...

**Décision le 08/11/19** : on met les tâches dans le sprint planning et chacun en prend une et la fait. Il faut le consulter avant de coder.

**Question** : Nombre de travail à faire par semaine/horaires ?

**Décision** : ...

## Fonctionnalités & propositions

Utiliser les annotations (@Nullable, @NotNull, @MagicConstant...)

Patron de conception proxy pour éviter de charger inutilement les textures ou de les charger plusieurs fois (voir patron de conception wikilivres)

Utiliser un tableau Excel/site web pour le sprint backlog.

La personne qui écrit le code n'est pas celle qui le documente. Cela permettra une meilleure maîtrise du code car on devra apprendre le code pour bien le documenter.

La méthode getTexture prend un enum plutôt que les coordonnées de la sous-texture.

## 7-1-3. Troisième réunion

**Durée** : Jeudi 21 novembre 2019 de 13h40 à 16h10

**Lieu** : Fontainebleau

**Membres présents** : Louka DOZ, Quentin RAMSAMY-AGEORGES, Loïc SENECAAT

Réalisé durant le sprint 1 :

- On peut charger une texture, dont les informations sont dans un fichier ("path sizeTile indexMin indexMax")
- On peut charger une texture en donnant le fichier, la colonne et la ligne dans le fichier.
- On possède la structure d'une map, on peut y placer des rooms
- On possède la structure des entités (Porte, Panneau, ...)
- On possède la structure des énigmes.
- La maquette de l'interface graphique
- 

## Questions traitées

**Question** : Sauvegarde des objets ?

**Réponse** : ???

**Explication** : On veut sauvegarder l'objet (ex : porte) dans le fichier. Le problème est que l'on sauvegarde des entités. Comment retrouver que c'était une porte.

**Proposition** : newInstance et instance of

**Proposition** : sauvegarder directement les objets

**Décision le 21/11/19** : demander conseil au tuteur

**Question** : Sauvegarde des énigmes ?

**Réponse** : Une classe par exemple "Writer" va prendre une liste d'entités, récupérer leurs énigmes et les sauvegarder. Elle va écrire pour toutes les énigmes, un identifiant (int) qui permettra de retrouver à quel objet elles sont associées.

**Question** : Comment détecter qu'un "truc" est une porte (on ne possède pas les ressources pour les portes verticales) ?

**Réponse** : Si un joueur est à proximité d'une porte, le message "ouvrir <touche utilisation>" est indiqué.

## Fonctionnalités & propositions

Ajouter une colonne testings.

Ajouter une méthode getEnigmas dans la classe Entité.

Les versions sont (sprint de dernière modification/créé).(nombre de modification au sprint actuel)

## Pour le prochain sprint

**Sprint Goal** : Affichage de la map et des entités graphiquement

Map sauvegardée au format .tmx (tiled) 2 fichiers à la création 1 seul à l'exportation

Les textures d'une porte sont dans des json

```
"porte1" {  
  "chemin image"  
  "coin-supérieur-gauche" : {  
    "0"  
    "1"  
  },  
  "..."  
}
```

## Améliorations

Les membres se sont retrouvés sans travail longtemps avant la fin du sprint.



## 7-2. Échanges avec le tuteur

### 7-2-1. Échange du 12 novembre

**Durée** : Mardi 12 novembre 2019 de 17h45 à 18h50

**Lieu** : Fontainebleau

Le but principal était de lui présenter notre organisation, ce que l'on a fait et que l'on souhaite faire et qu'il valide le Cahier des charges.

## Bilan

- Améliorer le diagramme de GanTT : on y ajoute les grandes parties du développement (Editeur, Gameplay, ...)
- **(Conseil)** Ajouter une partie analyse des risque ex : une personne s'en va
- **(Idée)** Faire des synopsis qui recouvrent tous les cas possibles pour faire un diagramme de classes complet
- **(Conseil)** Ne pas passer trop de temps à faire des diagrammes
- Lui permettre d'accéder au git pour qu'il puisse facilement nous assister
- **(Idée)** classe JLayeredPane pour l'intégration de la libgdx dans swing

Au niveau du diagramme de classes de la structure :

- Pas obligé de mettre les détails d'implémentation (HashMap...)
- Renommer texture en entité car plus parlant sur le contenu
- Le nombre de niveaux est fixe, donc plus pratique d'utiliser un tableau
- Pas besoin pour une classe d'implémenter item et : Content pourrait directement implémenter item et donc tous ceux qui implémentent Content n'ont plus besoin non plus d'implémenter Item
- Pas une bonne idée de faire un enum par texture (le programme n'est pas fermé), faire plutôt une méthode comme Tiled, on donne la texture et sa gamme (images de x à y) et dès que l'on veut charger une texture (sous-image), on donne un id et on regarde dans quelle texture (fichier) elle est.

## Signatures des parties prenantes

Une signature numérique n'a pas de validité selon le tuteur

## 7-2-2. Échange du 22 Novembre

**Durée** : Vendredi 22 novembre 2019 de 17h40 à 18h45

**Lieu** : Fontainebleau

Principalement des questions sur la sauvegarde de la map et des énigmes.

### Bilan

- **(Conseil)** énigmes seraient un ensemble d'actions, traitement récursif par exemple sauvegarde
- **(Idée)** un pnj qui aide
- **(Conseil)** on demande à chaque énigme ce qu'elle veut enregistrer, qui demande à chacune de ses composantes (actions ou énigmes). Elles peuvent renvoyer true si elles sont satisfaites.
- **(Suggestion)** Avoir des branchements dans les enigmes
- Autorise l'utilisation de newInstance pour la sauvegarde et l'instanciation de classes selon string
- **(Conseil)** ne pas utiliser xml et json : choisir.
- **(Conseil)** utiliser des fichiers DTD pour vérifier le format du .tmx

## 7-2-3. Échange du 06 Décembre

**Durée** : Vendredi 22 novembre 2019 de 17h30 à 18h05

**Lieu** : Fontainebleau

Questions sur nos prévisions pour l'interface graphique.

### Bilan

- **(Conseil)** Lors d'un clic gauche sur une case, il ouvre directement le menu (ajouter énigme...) de l'entité au plus haut niveau. On peut après dans ce menu naviguer entre les entités sur la case.
- **(Conseil, idée)** Maintenir clic drag and drop embêtant donc possiblement mieux une sélection et après on en met autant que l'on veut.
- **(Idée)** Mettre d'une couleur les cases possibles de drop
- **(Idée)** Permettre scroll via molette
- **(Idée)** Raccourcis vers le zoom naturel dans le jeu
- Classes JInternalFrame ? et JSplitPane

## 7-2-4. Échange du 31 Janvier

### Bilan

- **(Idée)** Signature des fichiers pour empêcher leur modification externe
- **(Question)** Joueur est entre deux cases, gestion des énigmes
- **(Idée)** Donner des versions aux maps
- **(Idée)** Fin peut être une énigme
- **(Remarque)** data ne prends jamais de (s)
- **(Idée)** Factory pour les options des popup (inversion des dépendances)
- **(Idée)** Condition, avoir visité une salle
- **(Conseil)** popup en mode « Toast » (android)
- **(Conseil)** bien de placer les entités nous-mêmes, mais les livres (...) peuvent être placées selon ce que veut l'utilisateur

## 7-2-5. Échange du 7 février

### Bilan

- **(Conseil)** Musique à base d'événements
- **(Conseil)** Rendre exporter modal (empêcher la modification de la map pendant l'export)
- **(Conseil)** Montrer ce qui marche lors de la soutenance, le cheminement, situations, démarche, stratégies, comparaisons avec le cahier des charges

## 7-3. Autres

### 7-3-1. Annexe 1 : Game Design Document

#### ENIGMA

GAME DESIGN DOC

##### Présentation du Jeu

Un ou plusieurs joueurs sont enfermés dans un lieu. Leur objectif est de s'échapper dans le temps imparti. Pour cela, ils vont explorer l'environnement et pouvoir résoudre des énigmes leur permettant de trouver la sortie.

##### Quel est l'objectif ?

Le but principal est de mettre à défi la capacité de réflexion du joueur tout en lui permettant de profiter d'un moment ludique, si désiré, en groupe.

##### Fonctionnalités

Le jeu propose d'explorer un lieu, et d'interagir avec ses éléments, que ce soient des objets ou des personnes.

Vous pouvez explorer seul ou à plusieurs (serveur hébergé sur l'une des machines).

Vous avez également la possibilité de créer vos propres escape games.

##### Equipe et langage

3-4 personnes.

Développé en Java avec la LIBGDX. Utilisation de la méthode SCRUM. <https://discord.gg/UAgUcs9>

##### Genres

Escape game / serious game  
Réflexion / 2D / Solo / Multiplayer

##### Plateformes

Le jeu sera disponible sur PC.  
Aucune configuration minimale.  
Souris et Clavier requis.  
Support manette

##### Audience

Toutes personnes, de tout âge et de tout genre.  
La langue du jeu sera le français.

##### Ressources

Ressources graphiques & sonores majoritairement de itch.io (TileSheet, SpriteSheet, sons, musiques, ...)

#### Présentation technique

L'utilisateur lance le jeu. Il a la possibilité soit de commencer à jouer, soit de créer un escape game.

##### Créer un escape game

On devra construire une bibliothèque de pièces de l'escape game tels que les objets (portes, ...), les personnages et le terrain.

L'utilisateur utilise alors ces pièces pour créer sa carte. On doit ensuite pouvoir les customiser (ajouter action, contenu par exemple d'un livre, ou encore définir pour une entité (case de la map, joueur, caillou, ...) des "actions". Voici un exemple d'action sur une porte : si <entité=joueur> <action = utilise> <item=clef> <cible =porte> alors <conséquence = porte s'ouvre>". Il s'agit de conditions pouvant être génériques (on remplace l'entité...). On peut choisir les zones à cacher et un déclencheur pour les afficher. La map peut être sauvegardée dans un fichier ou dans une base de données.

##### Chargement d'un escape game

On devra avoir une liste des escape games disponibles (image, nom, difficulté, durée, créateur) et, si possible savoir si le niveau a été terminé, ou encore obtenir un rang en fonction de la vitesse/qualité de l'évasion.

Le joueur après avoir sélectionné une map peut choisir de la lancer seul ou avec des personnes (auquel cas, il héberge le serveur). Le joueur pourrait éventuellement, lancer une map parmi des maps mises en ligne par d'autres utilisateurs et en mettre en ligne.

Les joueurs peuvent choisir un nom et une apparence.

##### Le jeu

Le ou les joueurs commencent dans une ou plusieurs salles.

Le joueur doit pouvoir interagir avec tout (presque) les éléments du décor (donne indice, action - cf Création, ou rien (affiche))

On peut lister des cibles d'interactions telles que :

- portes/coffres, meubles, interrupteurs, panneaux, puzzles, dessins, inscriptions
- êtres vivants (ennemis, pnj, autres joueurs)

Le joueur durant son exploration pourra rencontrer des entités hostiles, auquel cas, il pourra directement engager un combat, avec ses poings et/ou en utilisant des objets tels que des potions ou un objet efficace contre ce monstre. Il pourra être intéressant de permettre des "combats sans combat" tel un combat d'insultes. Il pourra accéder à un inventaire, dans lequel, il pourra consulter ses objets. Il est possible que la taille de l'inventaire soit limitée. Le joueur peut prendre des objets en main.

Si un joueur meurt, alors il est exclus de la partie. Si la partie ne contient pas le nombre de joueurs minimum alors elle est perdue. Si tous les joueurs arrivent à la sortie avant la fin du temps, alors la partie est gagnée.

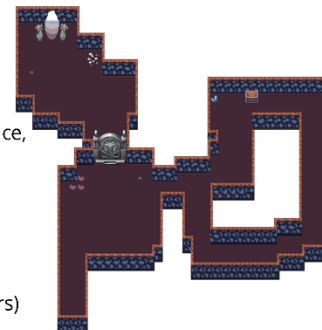
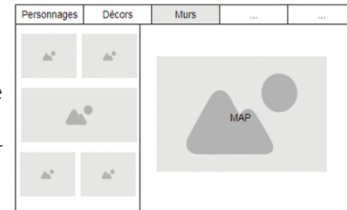
Pas de sauvegarde.

Les personnages non-joueurs (pnj) peuvent possiblement échanger des items avec les joueurs.

Il est possible de faire pivoter la caméra (de 90° par exemple), pour pouvoir voir les murs par exemple, en appuyant sur une touche.

##### Les paramètres

Le menu des paramètres permet de changer les touches ou de les consulter. Il permet de régler le son ou encore de gérer le serveur et les personnes dessus, si vous êtes en partie.



## 7-3-2. Annexe 2 : Tableaux des spécifications fonctionnelles

### Le créateur

M	Création de la librairie.
M	Déplacement des entités de la librairie vers la map
M	Ajouter des énigmes aux cases.
M	Définir le début et la fin de l'escape game
M	La map est sauvegardée dans un fichier ou une base de données
M	On peut définir le contenu d'objets (livre, panneau)
C	On peut cacher des zones et définir un déclencheur pour les afficher
C	On peut faire pivoter les salles
C	Personnages non-joueurs peuvent vendre des objets (donc on peut définir quoi)
C	Définir les conditions de fins personnalisées.
W	Importer et Exporter des maps en ligne
W	Ajouter ses propres éléments (sprites, ...)
W	Possibilité d'ajout de scénarios
W	Pouvoir lancer une simulation de l'escape game pendant la création

## Le lanceur

M	On a une liste des escapes games disponibles
M	Le joueur peut créer une partie solo ou multijoueur
M	Sélection du personnage et d'un nom
M	Consulter son inventaire, pouvoir prendre un objet en main
M	Si arrivée à la sortie avant la fin du temps, victoire. Si le temps est écoulé, défaite.
M	Consulter son inventaire, pouvoir prendre un objet en main
M	Les joueurs peuvent interagir avec tout (ou presque) les éléments du décor. Un message est affiché à chaque fois.
S	Pouvoir engager un combat avec des entités hostiles
S	Menu des paramètres et de gestion du serveur
W	Faire pivoter la caméra de 90°. (Graphiquement impossible)
W	Jouer sur des maps en ligne