Rapport



**Java & LibGDX**

Présenté à :

**RIZZOTTI Aïcha**

**PASIN Marcelo**

Étudiants :

**VAUCHER Joël**

**OMBANG NDO Charles**

**RODRIGUES LOURENÇO Daniel**

Abstract

Ce rapport présente notre projet académique P2 réalisé durant l’HES d’été.

Table des matières

[1. Introduction 3](#_Toc460918748)

[2. Conception 3](#_Toc460918749)

[2.1. Les Uses Cases 3](#_Toc460918750)

[2.2. Diagrammes de séquence 3](#_Toc460918751)

[2.3. Schéma de classes 5](#_Toc460918752)

[2.4. Architecture de logicielle 6](#_Toc460918753)

[3. Outils de conceptions et développement 6](#_Toc460918754)

[4. Fonctionnement 6](#_Toc460918755)

[5. Tiled 6](#_Toc460918756)

[6. LibGdx 7](#_Toc460918757)

[6.1. PlayScreen, World & JdcGame 7](#_Toc460918758)

[6.2. Sprite 7](#_Toc460918759)

[6.3. InteractiveTileObject 7](#_Toc460918760)

[6.4. Collision & Mask 7](#_Toc460918761)

[6.5. Son 7](#_Toc460918762)

[6.6. Blabla 8](#_Toc460918763)

[7. Problèmes rencontrés et solutions 8](#_Toc460918764)

[8. Améliorations possibles 8](#_Toc460918765)

[9. Conclusion 9](#_Toc460918766)

[10. Bibliographie 10](#_Toc460918767)

# Introduction

# Conception

Ce chapitre présente les éléments importants de la phase de conception.

## Les Uses Cases

Le diagramme des cas d’utilisations métier du projet est représenté ci-dessous.

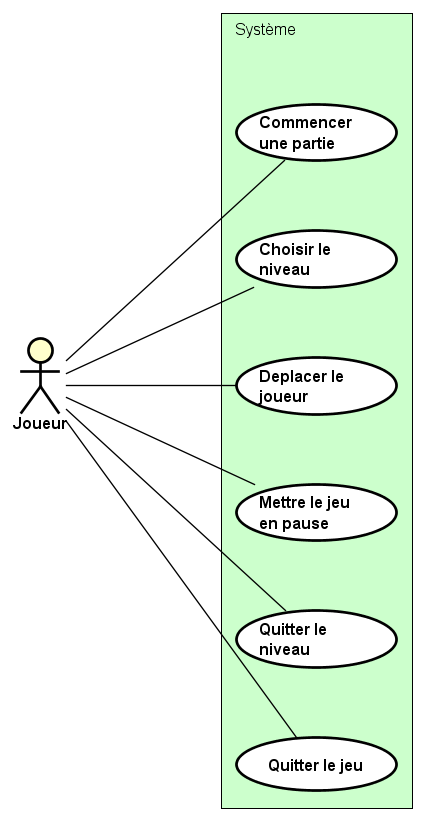


Figure 1 - Diagramme des cas d'utilisation

## Diagrammes de séquence

Le diagramme d’Uses cases ci-dessous montre les différentes fonctionnalités que le logiciel doit offrir à ses utilisateurs.

### Séquence dans le menu

Ce schéma présente le déroulement des interactions lorsque l’utilisateur est dans le menu principal. Le déroulement des interactions dans les sous-menus est le même donc un seul diagramme est présenté.

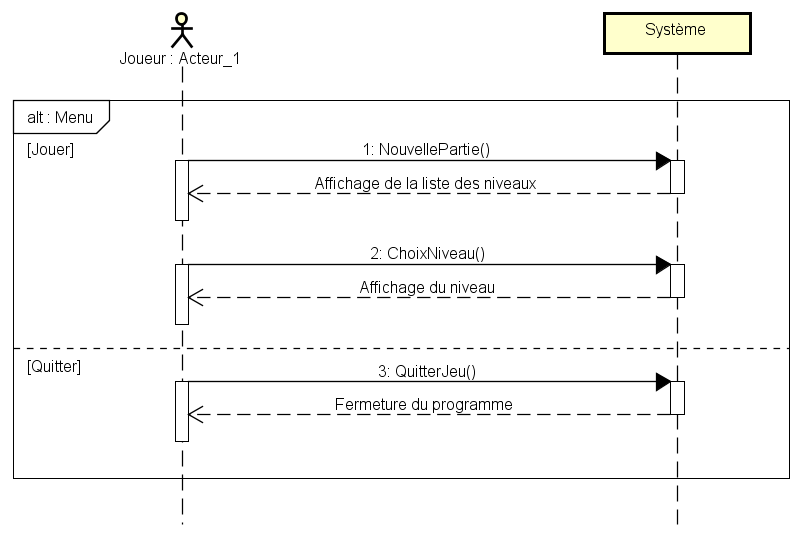


Figure 2 - Diagramme de séquence dans le menu

### Séquence durant le jeu

Voici le diagramme de séquence lors du déroulement d’une partie.

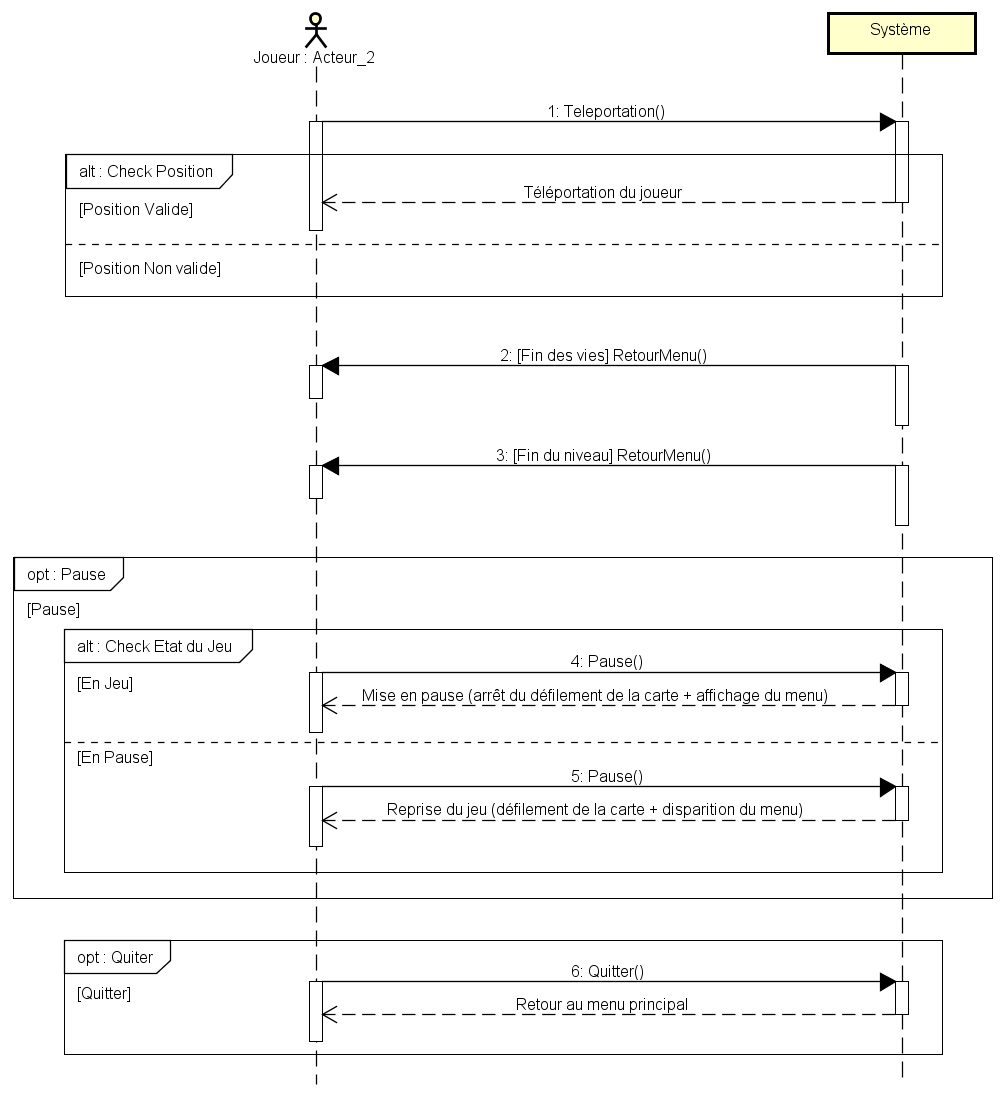


Figure 3 - Diagramme de séquence durant le jeu

## Schéma de classes

Voici le schéma de classes du projet. La version de ce schéma correspond au projet dans son état actuel. Le schéma prévu initialement se trouve en annexe.

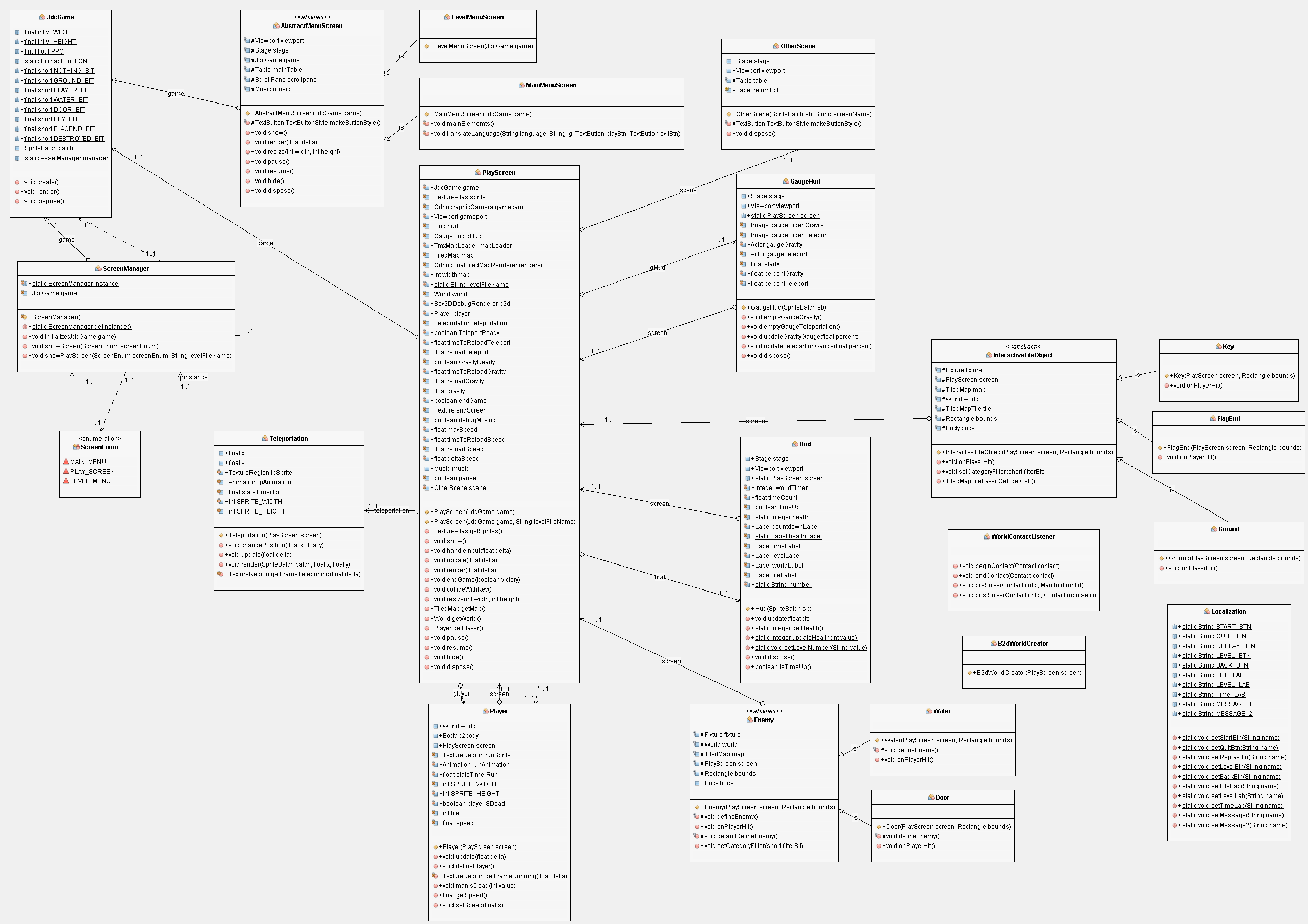


Figure 4 - Schéma de classes

## Architecture de logicielle

# Outils de conceptions et développement

La mise en place de ce projet a nécessité des outils assez simples d’utilisation et surtout, des outils utilisés quotidiennement pour notre formation.

Tableau 1 - Liste des outils utilisés

|  |  |
| --- | --- |
| Nom | Description |
| NetBeans | C’est l’IDE retenue pour la programmation de notre logiciel. |
| LibGdx | FrameWork pour les jeux 2D/3D en Java |
| GitHub | Comme outils de collaboration. |
| Astah | Pour les diagrammes de modélisation utilisés. |

Nous avons également pu compter sur un tutoriel expliquant la base de LibGdx :

<https://www.youtube.com/playlist?list=PLZm85UZQLd2SXQzsF-a0-pPF6IWDDdrXt> (Aureli, 2015)

# Fonctionnement

# Tiled

Tiled est un logiciel de création de map possédant différent niveau d’image (background, foreground) et des objets de familles différentes (mur, clé, sol). Le logiciel exporte les fichiers en .tmx qui grâce à une librairie de LibGdx peuvent être très facilement récupérer dans le code Java.

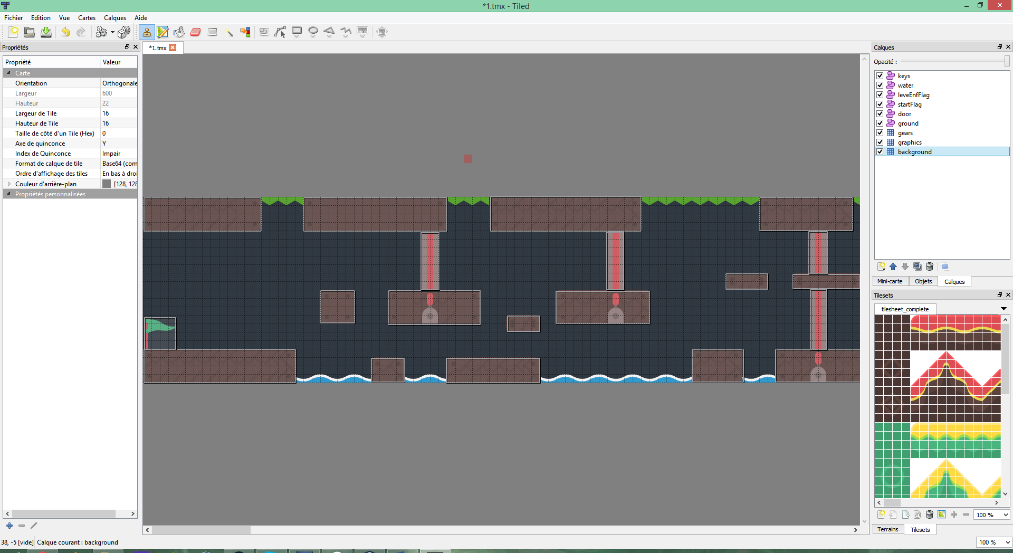


Figure 5 - Interface de Tiled Map

Le fichier contient différents calques (background, graphics & gear), ainsi que des sélections (ground, door, startflag, endflag, water, keys).

# LibGdx

## PlayScreen, World & JdcGame

Nous avons suivi la structure proposée par le tutoriel qui conseillait d’utiliser les classes de com.badlogic.gdx Screen, World et Game.

Screen représente notre application est sert de « main ». Elle possède nos instances de world et game ainsi qu’une instance viewport qui est la fenêtre qui apparait à l’écran et gamecam qui est la position logique de ce qui apparaitra de l’objet world dans le viewport.

World contient les objets qui ont une apparence dans le jeu ou une interaction avec (comme des senseurs). Il ne faut pas le confondre avec la gamecam qui est juste la proportion du monde qu’on affiche à l’écran. Chaque corps qui est ajouté dans le monde doit posséder un Body contient des coordonnées formant un Polygon ainsi qu’un type (StaticBody ou DynamicBody, KinematicBody n’a pas été utilisé). Les StaticBody comme les plateformes ou les pièces ne subissent pas la physique comme la gravité ou la propulsion d’un joueur, alors que les DymanicBody sont affectés.

Game qu’on a appelé JdcGame pour Joel-Daniel-Charles-Game contient les paramètres du jeu et ces ressources. On y trouve les constantes de proportions, les sons ainsi que les bits de mask.

## Sprite

Un Sprite est une image contenant toutes les différents frames d’une animation.

C:\Joel_Vaucher\INF2-b\HesETE-p2-Vaucher\hesETE-jdcGame\core\assets\sprites\Run_sprite.png

Figure 6 - Exemple de Sprites

LibGdx possède une classe Sprite afin de simplifier leurs intégrations au jeu.

Il s’agit de Body se trouvant dans le monde ayant une animation (pour nous, il y a juste le joueur et la téléportation, mais il y aurait pu avoir une rotation des pièces). Pour faire fonctionner le Sprite, il créer un objet Animation et lui donner un tableau de Texture, la taille d’une Texture et le temps entre deux frames.



## Téléportation

Comme expliquer brièvement dans un autre chapitre, il y a différent objet pour l’affichage et il est important de pouvoir les différencier.

* Viewport (fenêtre)
* Gamecam (contenu de la fenêtre)
* World (contenu total)

Pour la téléportation, il faut que un clic dans la viewport nous donne la position dans le world où placer le joueur.

De plus, la position sur l’écran est en pixel alors que celui du monde est en « mètre ». Cependant, pour aider, les objet fournisse des getters permettant de récupèrer les distance et avec des règles de trois, de récupérer la position du joueur dans le world.

## InteractiveTileObject

Parmi les objets qu’on a créé dans le monde, certain devront effectuer certaines actions au contact du joueur (mort, bonus).

Le tutoriel proposait une classe abstraite laissant une méthode virtuelle pour que les différentes spécifications fassent eux-mêmes les conséquences d’une collision au joueur (mur donne la mort, les pièces donne des points, le sol ne fait rien, …).

Nous avons fait deux classes abstraites Enemy et InteractiveTileObject qui fonctionne de la même manière et pour les mêmes raisons, mais dont la famille d’objet est différente et dont les extensions futures pourraient être problématique s’il n’y avait pas deux classes.

## Collision & Mask

Une gestion des collisions assez courante dans les IDE de 2D/3D est la gestion par calque. Cela signifie qu’on définit que certains objets font partie d’un groupe (ou plusieurs) et qu’il ne peut entrer en collision qu’avec seulement certains autres groupes.

Ainsi, on peut définir quelles collisions le joueur pouvait faire. La liste des bits de calque sont des constantes de JdcGame et par défaut, les objets ont un bit de calque de 1 (GROUND\_BIT)



Nous avons également un bit DETROYED\_BIT qui correspond aux objets qu’on ne veut plus qu’il entre en collision (porte cassé, ancienne pièce). C’est pourquoi on ne le voit pas dans la liste.

## Son

## Blabla

# Problèmes rencontrés et solutions

Ce chapitre présente quelques problèmes rencontrés et leurs solutions.

# Améliorations possibles

Ce chapitre présente quelques améliorations possibles du projet.

# Conclusion

Neuchâtel, le 9 septembre 2016

Vaucher Joël

Ombang Ndo Charles

Rodrigues Lourenço Daniel

# Bibliographie