Projet JustCOPS

Jeu d’action, plate-forme en Java avec « libGDX »

Projet tuteuré S2

Par Lucas Salvato,

Duc Nguyen et

Oscar Lefèvre

Tuteur :

Roza Lemdani

Table des matières

[Introduction 2](#_Toc422949351)

[1) Analyse organisationnelle 3](#_Toc422949352)

[a) Les acteurs du projet informatique 3](#_Toc422949353)

[b) L’analyse des besoins 3](#_Toc422949354)

[Tableau « QQOQCPC » d’analyse des besoins de l’utilisateur 3](#_Toc422949355)

[Diagramme « bête à corne » 3](#_Toc422949356)

[c) La définition des tâches 4](#_Toc422949357)

[d) Le cahier des charges 4](#_Toc422949358)

[e) Gestion du temps et allocation des ressources 4](#_Toc422949359)

[Durée des activités 5](#_Toc422949360)

[Diagramme prévisionnel de Gantt 6](#_Toc422949361)

[f) Mode d’emploi 6](#_Toc422949362)

[2) Analyse technique 7](#_Toc422949363)

[a) Langages et technologies utilisés 7](#_Toc422949364)

[b) Cas d’utilisation et scénarii 7](#_Toc422949365)

[Diagramme de cas d’utilisation 7](#_Toc422949366)

[Scénarii 7](#_Toc422949367)

[c) Maquette et Prototypage 9](#_Toc422949368)

[Prototype 9](#_Toc422949369)

[Premier essai pour le *Sprite* du personnage 9](#_Toc422949370)

[Maquette d’une carte en format *.txt* 10](#_Toc422949371)

[Les premières tuiles 10](#_Toc422949372)

[Mise en commun de la carte, des tuiles et du personnage 10](#_Toc422949373)

[d) Diagramme de classes 11](#_Toc422949374)

[3) Résolution des problèmes rencontrés 12](#_Toc422949375)

[Conclusion 13](#_Toc422949376)

[Bibliographie 14](#_Toc422949377)

# Introduction

JustCops est un jeu imaginé par *Lucas Salvato* et *Duc Nguyen*. Le jeu reprend le concept du jeu d’arcade « Elevator Action ».

Le principe est simple, deux policiers perquisitionnent un immeuble de trafiquants de drogue. Ils doivent gravir les étages un à un en arrêtant le maximum de trafiquants tout en récupérant toutes les matières illicites se trouvant dans l’immeuble; Jusqu’au toit où ils s’en vont en hélicoptère.

L’esthétique du jeu ainsi que la musique est de type 8-bits.

Le jeu est codé en Java avec l’aide de la librairie « libGDX ».

L’équipe de ce projet est constitué de *Lucas Salvato*, *Duc Nguyen*, *Oscar Lefèvre* et tuteuré par *Roza Lemdani*.

# Analyse organisationnelle

## Les acteurs du projet informatique

Le maître d’œuvre est *Oscar Lefèvre* qui va ainsi analyser les besoins des futurs utilisateurs du logiciel et le maître d’ouvrage est *Duc Nguyen* qui va définir les différentes activités qui vont être mises en œuvre afin de pouvoir satisfaire les besoins des futurs utilisateurs. *Lucas Salvato* étant plus compétent, il a pris naturellement le rôle de chef de projet et est ainsi aussi bien maître d’œuvre que maître d’ouvrage.

## L’analyse des besoins

### Tableau « QQOQCPC » d’analyse des besoins de l’utilisateur

|  |  |  |
| --- | --- | --- |
|  | **JustCops** | |
| **QQOQCPC** | **Questions** | **Réponses** |
| Quoi ? | De quel besoin s’agit-il ? | Le divertissement |
| Qui ? | Qui est concerné ? | Les utilisateurs d’un ordinateur n’ayant pas de problème d’acuité visuelle, ni de problème neurologique comme l’épilepsie et pouvant se servir de leurs deux mains. |
| Où ? | À quel endroit ? | Devant un ordinateur |
| Quand ? | À quel moment ? | Durant les temps libres |
| Comment ? | Quel est le moyen d’interaction ? | En utilisant ses deux mains |
| Pourquoi ? | Quelle est la raison ? | Pour se divertir seul ou à deux devant un ordinateur |
| Combien ? | Quelle est la quantité dont on a besoin ? | Un seul |

### Diagramme « bête à corne »

La « bête à corne » permet d’identifier la fonction principale de notre jeu JustCops.

Divertir l’utilisateur

JustCops

Le projet JustCops étant un jeu vidéo, on en conclut qu’il doit divertir l’utilisateur pour que celui-ci soit satisfait, ce sera donc la fonctionnalité principale du projet.  
Mais il faudra aussi que le jeu soit ergonomique, que l’utilisateur puisse modifier certains paramètres, comme modifier la taille de la fenêtre. Le jeu doit aussi être en mesure d’afficher des informations sur la partie en cours comme le score.

## La définition des tâches

Avec les besoins que nous avons établis précédemment, nous allons donc imaginer les différentes tâches à accomplir pour que le projet soit correctement réalisé. Tout d’abord, il faudra apprendre à utiliser la librairie « libGDX » qui va nous faciliter la programmation de notre jeu. Ensuite, il faudra réaliser le diagramme de classe en UML et divers diagrammes avec « Visual Paradigm » et programmer notre jeu en Java puis réaliser les graphismes de notre jeu avec des logiciels comme « Paint », « Gimp » ou « InkScape ». Nous devrons aussi créer la musique de notre jeu et mettre en forme le rapport de stage.

## Le cahier des charges

Le cahier des charges est un tableau qui devra être respecté au cours de la réalisation de notre projet.

|  |  |
| --- | --- |
| L’objet étudié | JustCops |
| L’utilisateur | Personne possédant un ordinateur et un écran relié à celui-ci, n’ayant pas de problème d’acuité visuelle, ni de problème neurologique comme l’épilepsie et pouvant se servir de ses deux mains. |
| Le besoin fondamental de l’utilisateur | Se divertir |
| Les principales fonctions | F1-Réagir suite à l’action (appuie sur une touche)  F2-Afficher le déroulement des différentes actions  F3-Etre ergonomique  F4-Modifier la taille de la fenêtre en fonction des envies de l’utilisateur  F5-Afficher des détails sur la partie en cours (objectifs, vie)  F6-Mettre fin à la partie  F7-Faire se déplacer certains éléments tout seul (ennemis et ascenseurs)  F8-Fermer l’application correctement |
| Les contraintes de réalisation | Respecter le temps imposé  Apprendre une nouvelle librairie |

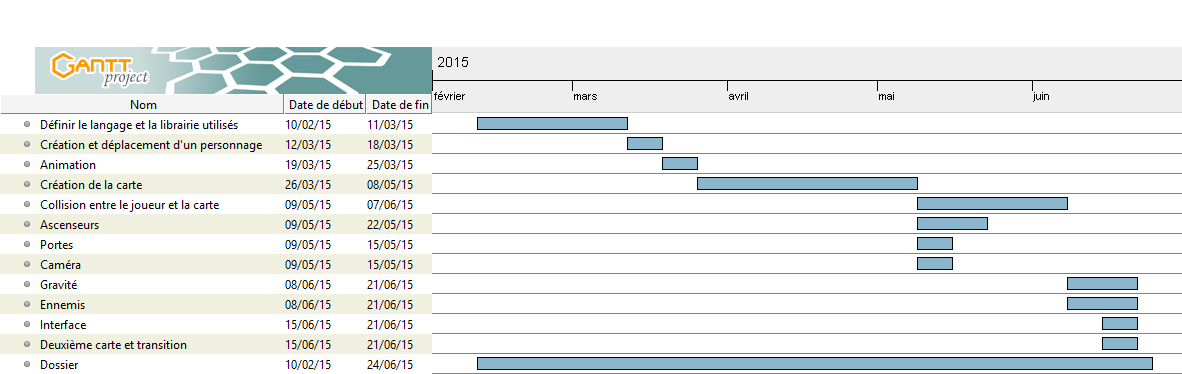
## Gestion du temps et allocation des ressources

### Durée des activités

La durée des activités a été rassemblée sous forme de tableau disposant de la description des activités au côté de leur durée.

|  |  |  |  |
| --- | --- | --- | --- |
| N° de l’activité | Nom de l’activité | Description de l’activité | Durée de l’activité |
| 1 | Définir le langage et la librairie utilisés | Avant toute chose, nous devions choisir comment serait réalisé le jeu | 1 mois |
| 2 | Création et déplacement d’un personnage | L’une des étapes les plus basiques consiste à réussir à déplacer un personnage à droite, à gauche et de sauter | 1 semaine |
| 3 | Animation | Le personnage bouge lorsqu’il se déplace à l’aide d’un *Sprite* qui est ensuite animé | 1 semaine |
| 4 | Création de la carte | Création d’une carte en utilisant un système de tuiles | 1 mois et 2 semaines |
| 5 | Collision entre le joueur et la carte | Le personnage doit se heurter contre les murs | 1 mois |
| 6 | Ascenseurs | La carte doit contenir des ascenseurs pouvant bouger et le personnage peut y entrer pour changer d’étage | 2 semaines |
| 7 | Portes | La carte contient des portes avec lesquels le personnage peut interagir | 1 semaine |
| 8 | Caméra | Une caméra permet d’avoir une distance appropriée et suit les personnages | 1 semaine |
| 9 | Gravité | Lorsque le personnage ne touche pas le sol, il tombe | 2 semaines |
| 10 | Ennemis | Création, mouvement et placement des ennemis aléatoirement | 2 semaines |
| 11 | Interface | Ajout du nombre d’objectifs et de la vie | 1 semaine |
| 12 | Deuxième carte et transition | Création d’une deuxième carte plus élaborée et transition à celle-ci une fois la première terminée | 1 semaine |
| 13 | Dossier | Faire le dossier | 5 mois |

### Diagramme prévisionnel de Gantt

Le diagramme de Gantt permet de faire une prévision des différentes tâches que l’on devra réaliser au cours de ce projet.

## Mode d’emploi

Appuyer sur la touche « flèche droite » pour faire bouger le personnage à droite.

Appuyer sur la touche « flèche gauche » pour faire bouger le personnage à gauche

Appuyer sur la touche « flèche haut » pour faire sauter le personnage dans les airs.

Appuyer sur la touche « flèche haut » quand le personnage est devant une porte objectif pour changer la porte de couleur et ainsi accomplir l’objectif.

Cliquer sur le bord de la fenêtre du jeu et déplacer la souris pour agrandir ou réduire la fenêtre du jeu.

Cliquer sur le bouton « réduire » pour minimiser la fenêtre du jeu dans la barre des tâches ou pour la remettre à sa taille normale.

Cliquer sur le bouton « plein écran » pour passer la fenêtre du jeu de petite taille à plein écran et inversement.

Cliquer sur le bouton « croix » pour fermer la fenêtre du jeu.

# Analyse technique

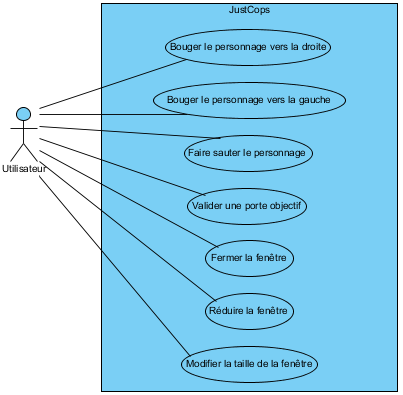
## Langages et technologies utilisés

Nous avons décidé de programmer la totalité de notre projet avec le langage orienté objet « Java » et la librairie « libGDX ». Nous allons aussi faire certains diagrammes avec le logiciel « Visual Paradigm ». Les graphismes seront réalisés sur les logiciels « Paint » et « Gimp ». L’interpréteur en lignes de commande « Bash » permettra aussi de créer des scripts pour la création des cartes, et le logiciel « Tiled » se chargera de faciliter l’assemblage des cartes.

## Cas d’utilisation et scénarii

### Diagramme de cas d’utilisation

Le diagramme de cas d’utilisation de JustCops permet de connaitre les interactions possibles de l’utilisateur sur le jeu.



### Scénarii

Système : JustCops

Acteur : Utilisateur

Objectif : Divertir l’utilisateur en répondant à ses interactions

1-L’utilisateur lance le logiciel

2-Le système affiche la fenêtre de jeu et attend que l’utilisateur effectue une action

3-L’utilisateur effectue une action

4-Le système effectue l’action

5-L’utilisateur fini le jeu

6-Le système ferme la fenêtre de jeu

Exceptions :

3a-Fermeture de la fenêtre

3-L’utilisateur clique sur la croix

4-Le système ferme la fenêtre de jeu

5a-L’utilisateur effectue une autre action, retour à l’étape 3

Alternatifs :

3a-Appuie sur la touche « flèche droite »

3-L’utilisateur appuie sur la touche « flèche droite »

4-Le système déplace le personnage sur la droite

3b-Appuie sur la touche « flèche gauche »

3-L’utilisateur appuie sur la touche « flèche gauche »

4-Le système déplace le personnage sur la gauche

3c-Appuie sur la touche « flèche haut »

3-L’utilisateur appuie sur la touche « flèche haut »

4-Le système fait sauter le personnage dans les airs

3d-Appuie sur la touche « flèche haut » devant une porte objectif

3-L’utilisateur appuie sur la touche « flèche haut » devant une porte objectif

4-Le système change la couleur de la porte et décrémente le score

3e-Clique sur le bouton « réduire »

3-L’utilisateur clique sur le bouton « réduire »

4-Le système minimise la fenêtre de jeu dans la barre des tâches ou la remet à la normale

3f-Clique sur le bouton « plein écran »

3-L’utilisateur clique sur le bouton « plein écran »

4-Le système passe la fenêtre du jeu de plein écran à petite taille ou inversement

## Maquette et Prototypage

### Prototype

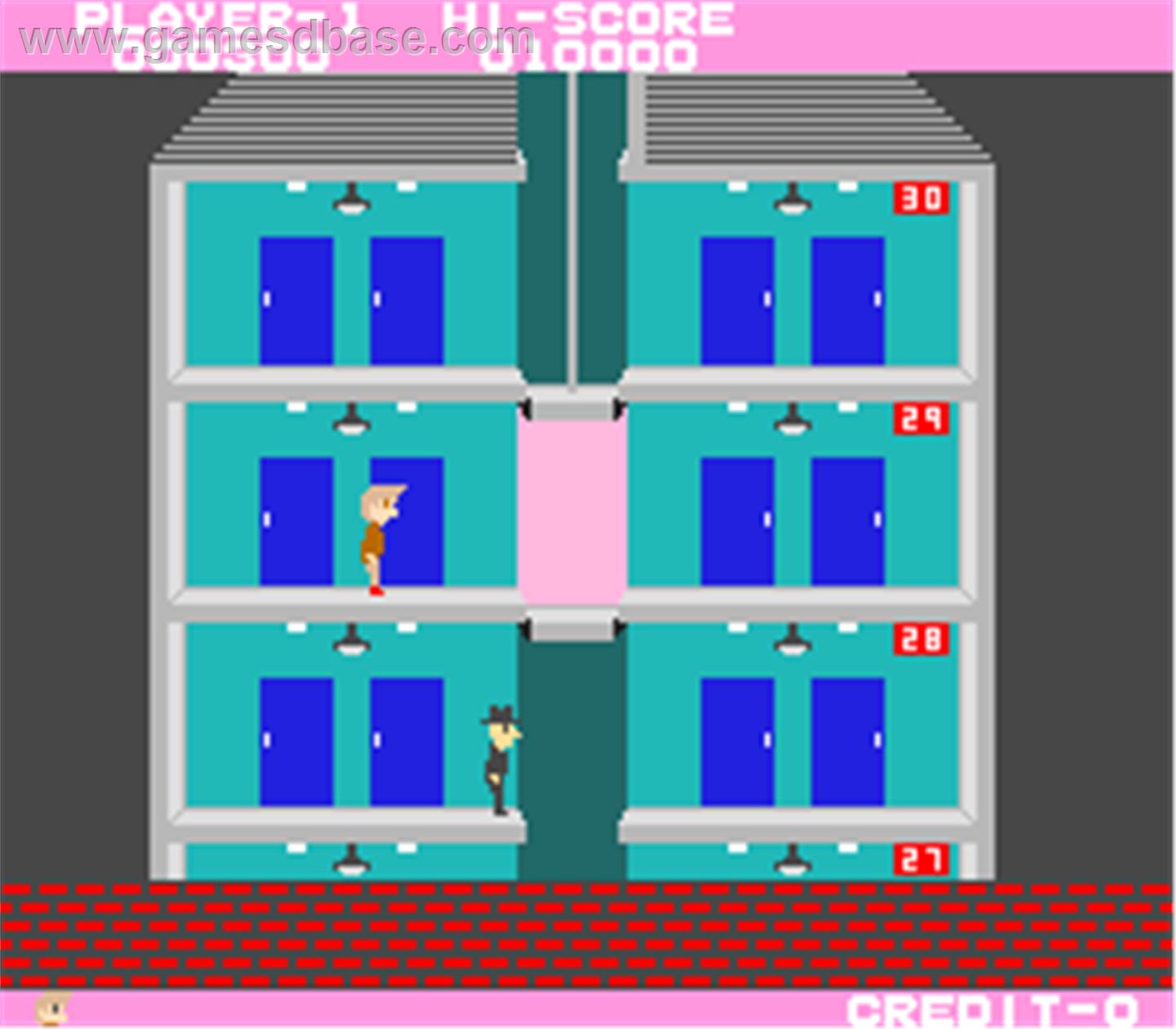


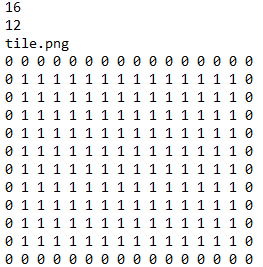
Image du jeu « Elevator Action » développé et édité par *Taito*, sorti en 1983 sur borne d’arcade dont nous nous sommes inspirés.

### Premier essai pour le *Sprite* du personnage



Le Sprite du personnage est composé de 5 images légèrement différentes permettant à la suite de créer une animation de marche.

### Maquette d’une carte en format *.txt*



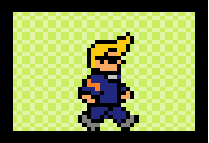
Les 0 représentent les murs et les 1 représentent le fond. Les deux premiers chiffres sont le nombre de colonnes et le nombre de ligne. La troisième ligne informe l’endroit où sont contenues les tuiles.

### Les premières tuiles



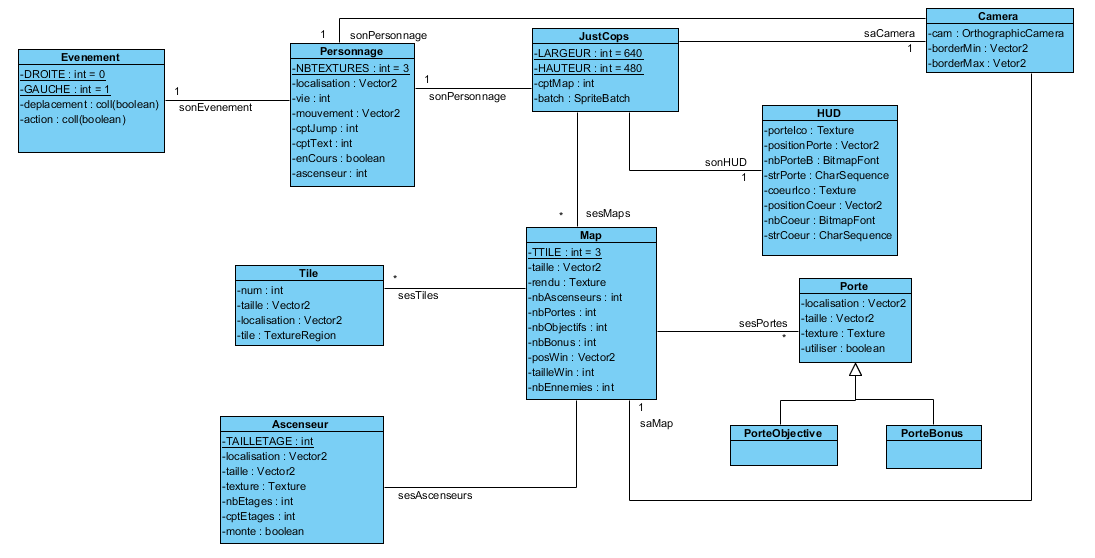
Les tuiles sont les composants de notre plateau de jeu. Le bloc noir représente les murs tandis que le vert représente le fond.

### Mise en commun de la carte, des tuiles et du personnage



Cette mise en commun permet d’obtenir une ébauche du projet final.

## Diagramme de classes



# Résolution des problèmes rencontrés

Au cours de ce projet, nous avons dû faire face à plusieurs problèmes que nous avons résolus :

\_L’installation de « libGDX » fut difficile : nous avons suivi des tutoriels sur internet.

\_Ce fut notre première utilisation d’une librairie pour programmer : nous nous sommes renseignés auprès de notre tuteur et sur divers sites internet.

\_Le jeu vidéo est un monde inconnu pour nous, nous ne savions pas par où commencer dans la création de notre projet : nous nous sommes renseignés auprès de notre tuteur et de nos camarades et sur divers sites internet, nous avons alors décidé de bâtir en premier un diagramme de classe complet.

\_Se renseigner sur le *Wiki* officiel de « libGDX » rédigé en anglais : nous avons utilisés diverses applications de traduction et nous avons recherché un tutoriel « libGDX » en français.

\_La complexité de « libGDX » : nous avons trié toutes les capacités de « libGDX » et nous nous sommes orientés sur certaines fonctions précises comme « SpriteBatch ».

\_La création d’une carte fut dure à imaginer : nous nous sommes renseignés auprès de nos camarades, nous avons alors décidé d’utiliser le « Scanner de fichier » de *Java* pour lire le corps de la carte dans un fichier *.txt* .

\_La collision entre le personnage et les murs de la carte fut difficile à concevoir : nous avons procédés à de multiples tests en passant par de nombreux échecs jusqu’à ce que cela marche.

\_La carte n’était pas dans le format approprié pour le logiciel : nous avons utilisé « Bash » pour supprimer les virgules et les identifiants des tuiles puis nous avons vérifié manuellement et minutieusement la bonne disposition de la carte sur l’éditeur de texte.

\_Les portes et ascenseurs ont dû être ajoutés manuellement dans le fichier texte de la carte : nous avons copié/collé minutieusement les identifiants des portes et ascenseurs depuis le fichier *XML* de la carte vers le fichier texte.

\_La caméra modifie la position de tous les objets et du personnage, on ne peut donc pas obtenir la position réelle de ces derniers ce qui est dérangeant pour faire une marge pour la caméra et pour le placement des portes et des ascenseurs sur la seconde carte : nous avons désactivé temporairement la caméra pour obtenir les positions réelles puis nous l’avons réactivée une fois les objets placés.

# Conclusion

Le jeu est opérationnel et dispose de différentes fonctionnalités comme les déplacements, la vulnérabilité face aux ennemis, la validation des objectifs, le changement de carte, etc…

Le projet fut trop ambitieux dès le départ ce qui nous a valu le soucis de devoir enlever certaines améliorations comme une intelligence artificielle aux ennemis, de la musique d’ambiance, des bruitages, la possibilité de tirer pour tuer ses ennemis, ou encore l’ajout de nouvelles cartes pour chacune des portes objectif. Toutes ces améliorations pourraient être ajoutées dans une version ultérieure du projet.

Pour conclure, nous avons appris grâce à ce projet comment créer un jeu vidéo, utiliser une librairie et adapter nos connaissances à un projet concret. Cela nous a aussi appris à quel point il était difficile de travailler en groupe, coordonner chaque personne et chaque tâche lors d'un projet demande des compétences particulières que nous avons pu développer durant ce projet. Cependant le niveau de compétence à atteindre pour ce projet était très élevé ce qui a rendu la coordination délicate, prendre un sujet conséquent nous a donc permis de nous améliorer sur cette aspect de gestion.

# Bibliographie

<http://fr.wikipedia.org/wiki/Elevator_Action> (Page *Wikipédia* d’ « Elevator Action », le jeu dont nous nous sommes inspirés)

<http://gamesdbase.com/> (Permis d’obtenir des informations sur Elevator Action)

<http://libgdx.badlogicgames.com/> (Le site de « libGDX »)

<https://github.com/libgdx/libgdx/wiki/Gradle-on-the-Commandline#packaging-the-project> (Le *gitHub* de « libGDX » où nous avons trouvé la plupart des informations importantes sur la librairie)

<http://openclassrooms.com/courses/apprenez-a-programmer-en-java> (Un cours qui nous a permis de répondre à certaines de nos interrogations sur la programmation en *Java*)

<https://projets-tuteures.iut-orsay.fr/> (Le site de gestion des projets tuteurés de S2)

<http://spritedatabase.net/> (Grande base de données sur les *Sprites* de personnages et objets de décor pour jeux vidéo)

<http://www.visual-paradigm.com/> (Le site de « Visual Paradigm »)

<http://tuto-libgdx.blogspot.fr/2013/08/installation.html> (Tutoriel en français sur « libGDX »)