Université de Poitiers,
Sciences Fondamentales et Appliquées

RAPPORT DE PROJET

U.E : Programmation Orientée Objet

Thème : Colossal Cave Adventures

Année universitaire : 2017-2018

Auteurs :

- Sangbé Narcisse SIDIBE
- William FLEURQUIN

Encadrants :

- M. Samuel PELTIER
- M. Stéphane JEAN

Table des matières

I. MANUEL UTILISATEUR 2
A. Scénario 2
B. Commandes 2
C. Carte du jeu 3
D. Comment gagner ? 4
II. CONCEPTION 4
A. Utilisation de l'Orienté Objet 4
B. Diagrammes UML 4
1. Diagramme de classe 5
2. Diagrammes de séquence 6
3. Diagramme d'activité : Jouer une partie 8
III. REALISATION 8
IV. TESTS 9
V. ORGANISATION 10
CONCLUSION 11

I. MANUEL UTILISATEUR

A. Scénario

Vous êtes Octave Charpentier, un explorateur qui a dédié sa vie à la recherche d'information sur le fameux Capitaine Némo et son sous-marin le Nautilus. Lors d'une escale en Jamaïque, vous faites la rencontre d'un équipage et de son capitaine, également à la recherche du Nautilus, et qui prétend avoir des informations sur la localisation de la célèbre machine. Vous décidez donc de vous joindre à eux pour la suite de votre aventure. Après plusieurs jours en mer, il s'avère que les informations détenues par le capitaine étaient juste, le Nautilus se trouve au fond de l'océan juste en dessous de votre navire et vous vous portez volontaire pour explorez la machine de guerre.

Le but de votre plongée est de récupérer le livre de bord du Nautilus dans lequel le Capitaine Némo aurait décrit son voyage et décrit son voyage. Restez sur vos gardes car, d'après les légendes que vous ont raconté les matelots, la carcasse du Nautilus serait habitée par différents monstres marins. Il se pourrait même qu'il soit hanté…

B. Commandes

Les différentes commandes disponibles dans le jeu sont les suivantes :

- ***look* :**

Cette commande permet d'afficher la description d'un objet ou d'une pièce.

look → affiche la description de la pièce dans laquelle le héros se trouve, la liste des objets présents et toutes les sorties disponibles vers d'autres pièces.

look nomObjet → affiche la description de l'objet si et seulement si ce dernier se trouve dans l'inventaire du héros.

- ***go :***

Cette commande permet de se déplacer dans le jeu, allé d'un lieu à un autre, à condition qu'il existe une sortie entre ces deux lieux.

go nomDeLaPièce → permet de se rendre dans la pièce indiquée.

- ***use :***

Permet d'utiliser un objet de son inventaire. Toutefois, certains objets ne s'utilisent que sur des cibles tandis que d'autres non.

use oxygène → utilise l'oxygène pour régénérer son énergie

use cléSalleDepart couloir → utilise la clé pour déverrouiller la porte entre la salle de départ et le couloir.

Lors d'un combat, il tout à fait possible d'utiliser une arme pour se défendre.

use harpon Mégalodon → utilise le harpon sur le Mégalodon.

- ***take :***

Permet de ramasser un objet dans la pièce dans laquelle on se trouve pour la mettre dans son inventaire.

take nomDeLObjet → ramasse l'objet spécifié.

- ***attack :***

Permet de se défendre à mains nus lors d'un combat. Inutile de spécifier l'adversaire, car cette commande n'est valide que lors des combats.

- ***inventory :***

Permet d'afficher la liste des objets dont le héros dispose ainsi que leurs quantités.

- ***map :***

Permet d'afficher la carte du jeu.

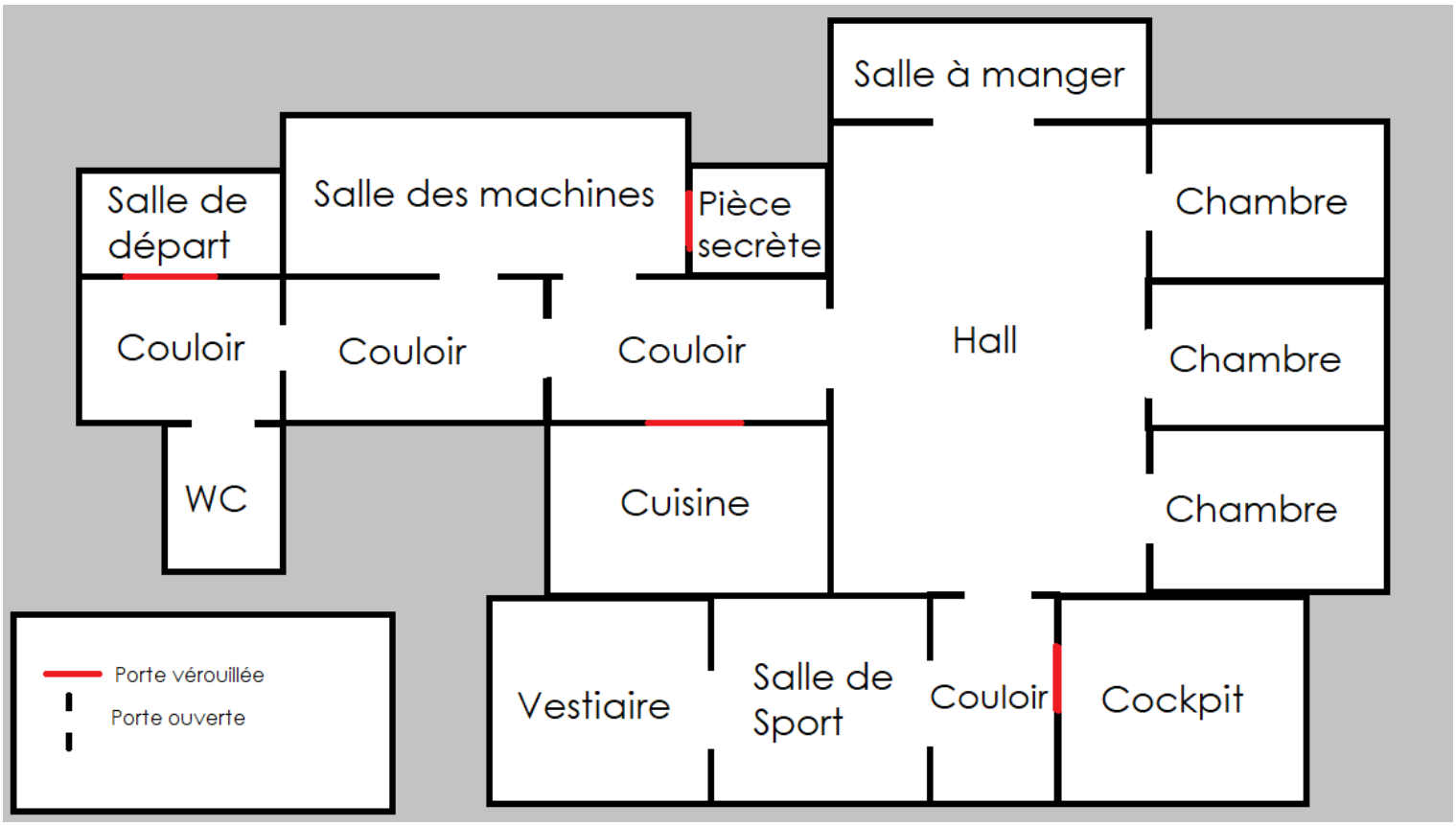
- ***help :***

Permet d'afficher l'aide.

- ***quit :***

Permet de quitter la partie en cours.

C. Carte du jeu



D. Comment gagner ?

Votre objectif est de trouver le *carnet de bord* du *Nautilus*. Ce dernier se trouve dans le *cockpit*, cependant vous n'avez pas accès à cette pièce au début de la partie, il vous faudra d'abord trouver la *clé du cockpit* qui se trouve dans la *salle de sport.* La *clé du cockpit* est protégée par le *Mégalodon*, monstre le plus puissant du jeu.

Après être sorti de la première pièce, vous entrerez directement en combat face à une *pieuvre*. Dans cette pièce se trouve un *couteau*, pour finir rapidement le combat sans trop perdre de point de vie, ramassez le *couteau* dès le début du combat. En effet ce dernier permet de tuer la *pieuvre* en un seul coup. Vous devrez ensuite vous rendre dans la *salle à manger* dans laquelle vous trouverez la clé pour entrer dans la *pièce secrète.* Dans cette pièce vous trouverez notamment le *harpon*, outil indispensable pour vaincre le *Mégalodon* mais également de l'*oxygène* et des *munitions*. Essayez de collecter le plus d'*oxygène* possible (vous en trouverez dans les *toilettes* et dans la *cuisine*) et ne les utiliser qu'en cas d'extrême urgence. Vous trouverez également des *munitions* et de l'*oxygène* dans les différentes chambres. Economisez vos *munitions* de *harpon* car elles vous seront très utile (voire indispensable) pour affronter le *Mégalodon*.

Une fois prêt, rendez-vous dans la *salle de sport*. Le *Mégalodon* possède 150 points de vie et inflige 20 points de dégât par tour. Faites-en sorte d'arriver avec au moins 2 capsules d'*oxygène* ou le combat sera compliqué. Une fois le combat terminé, saisissez-vous de la clé et rendez-vous dans le *cockpit*.

II. CONCEPTION

A. Utilisation de l'Orienté Objet

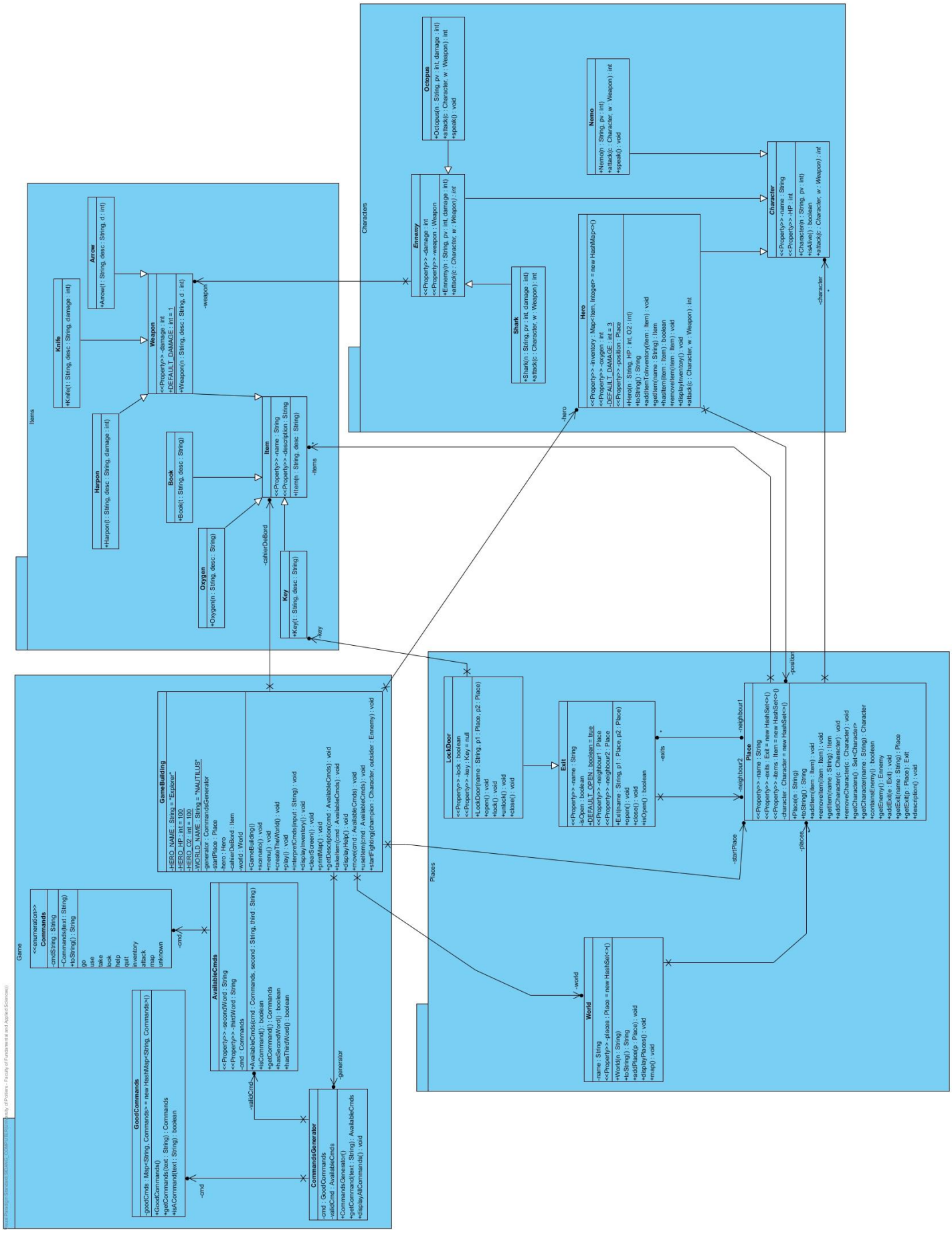
L'intérêt de l'orienté objet apparait très clairement dans un projet comme celui-ci car cela simplifie énormément le code et permet de retranscrire beaucoup plus simplement la pensée et la logique qui se cache derrière le jeu.

Nous avons utilisé autant que possible l'héritage et les **classes abstraites** (*Item, Character, Enemy*) car cela nous permettait d'ajouter assez rapidement des nouveaux éléments au jeu en ayant simplement à redéfinir des méthodes si besoin. Au début nous avions prévu d'intégrer une interface pour définir des objets transportables mais, après réflexion, cela nous a paru superflu car l'intérêt d'objets non transportables aurait été limité. L'utilisation d'un type énuméré nous a paru la solution la plus simple et la plus naturelle pour déclarer la liste des commandes.

B. Diagrammes UML

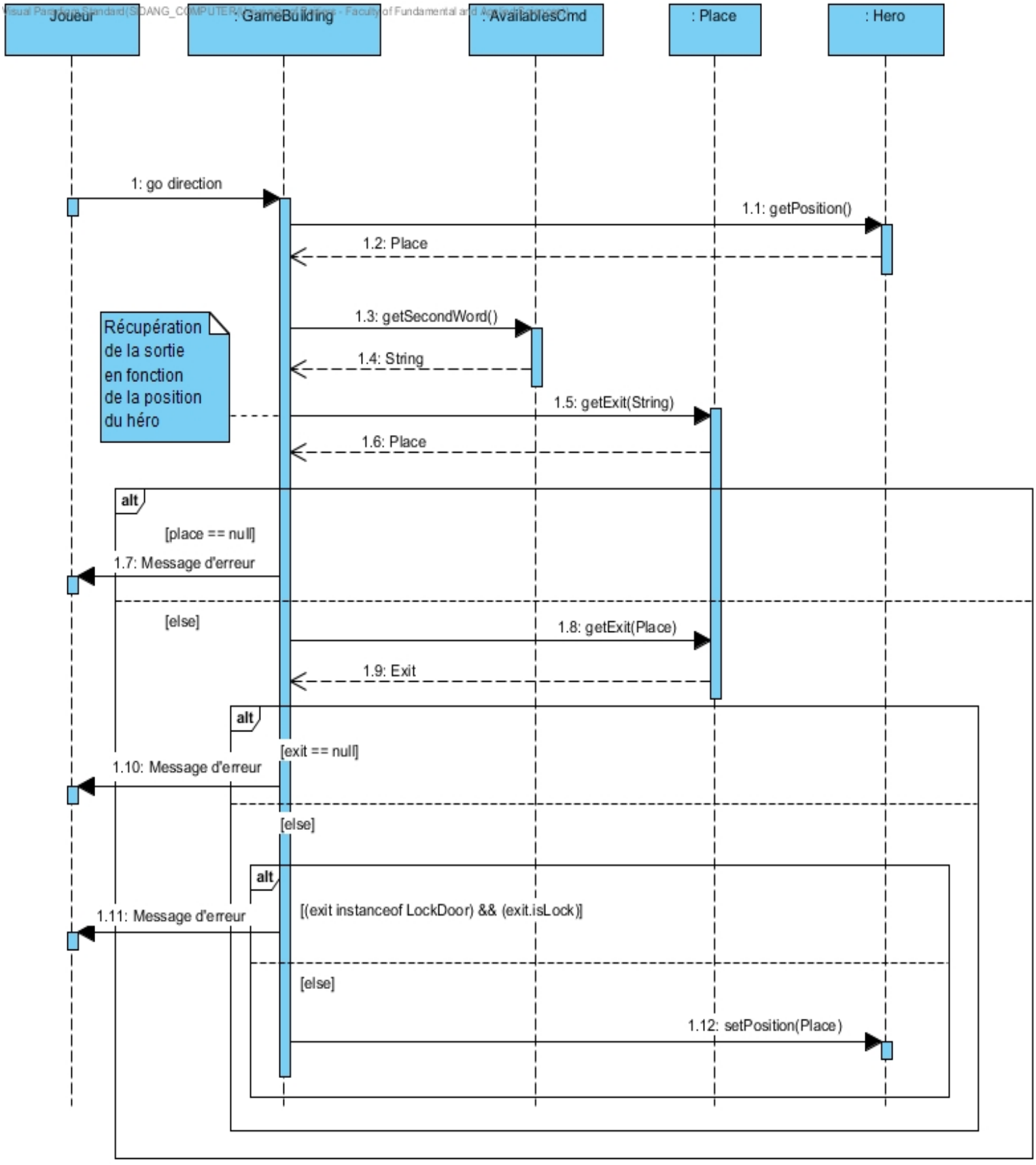
.

1. Diagramme de classe

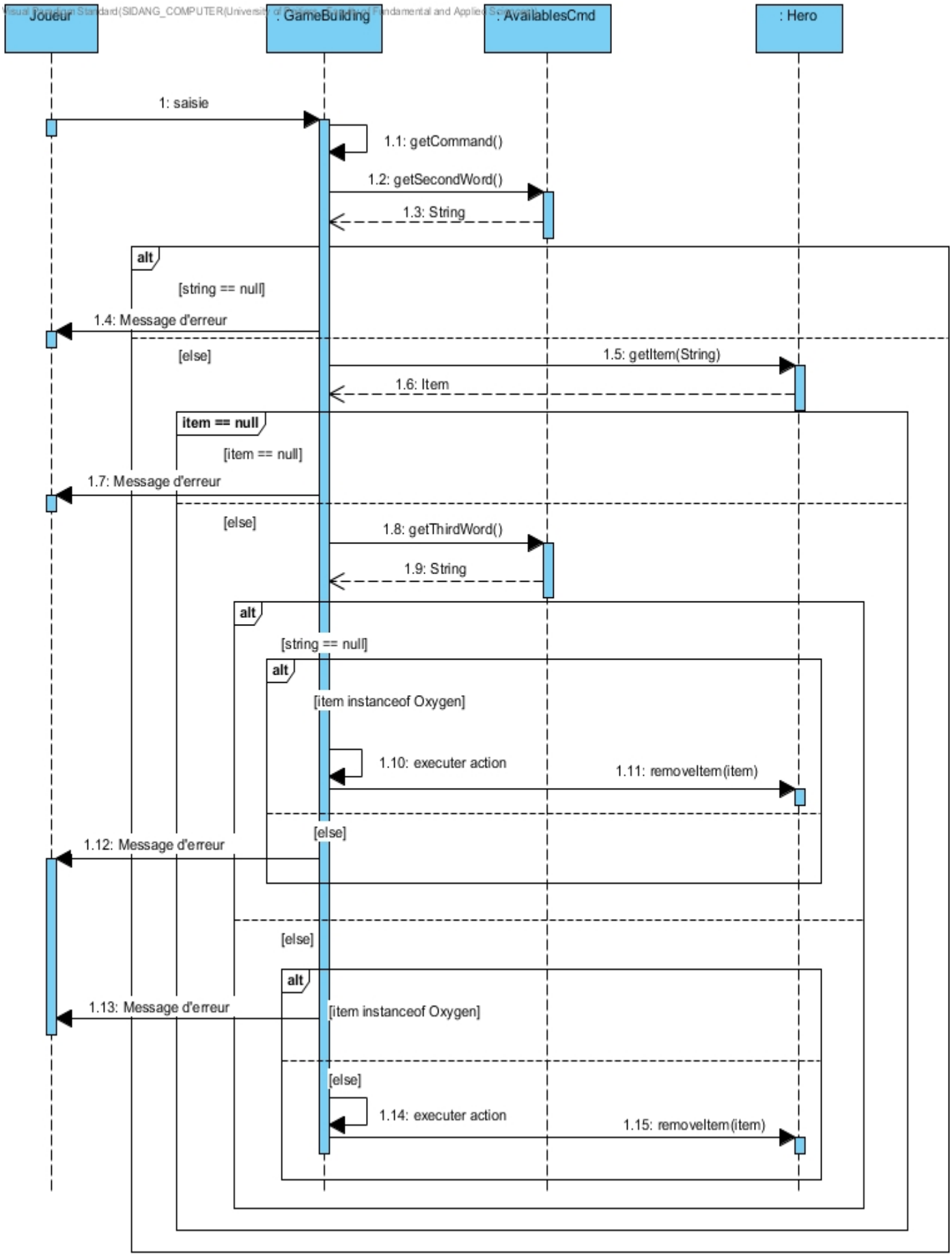


2. Diagrammes de séquence

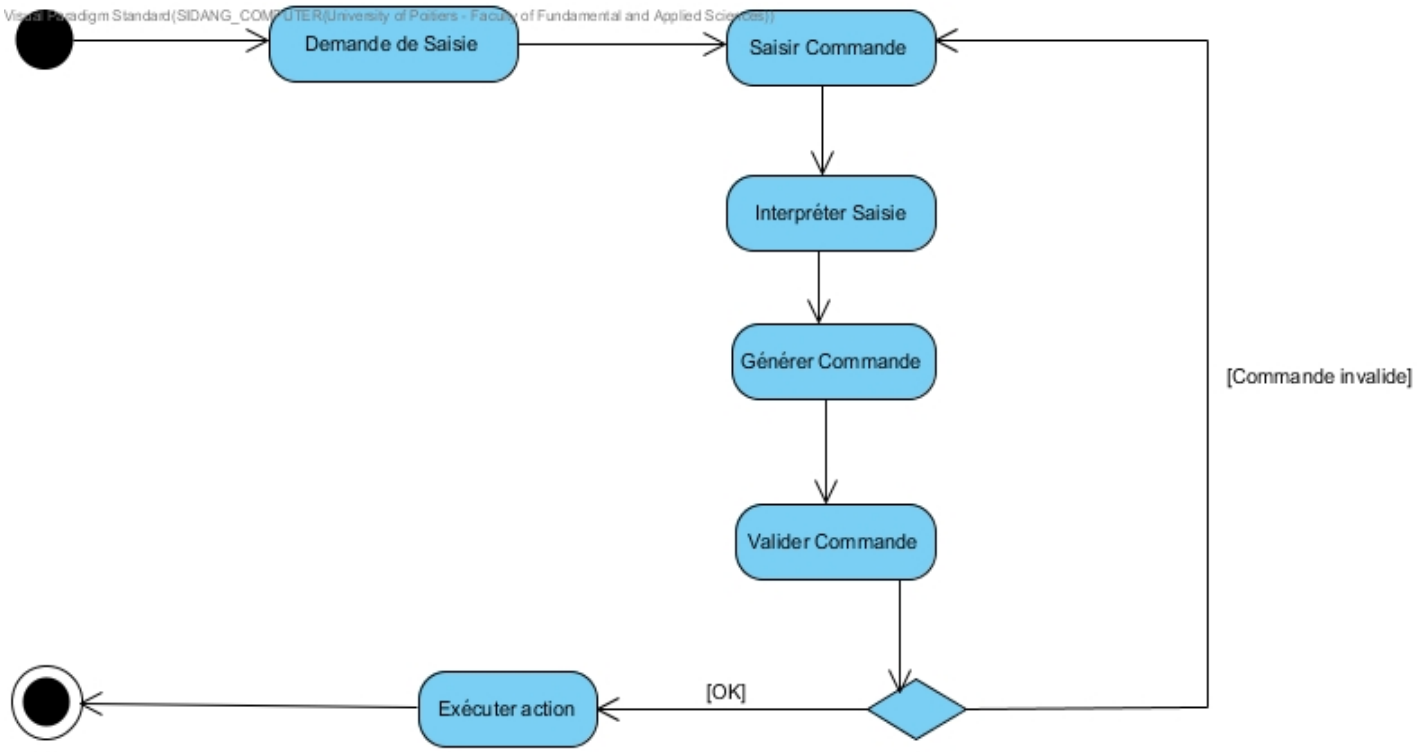
a) Déplacement dans le jeu :



b) Utilisation d'un objet :



3. Diagramme d'activité : Jouer une partie



III. REALISATION

Colossal Cave Adventure étant un jeu d'aventure en console, cela impliquait donc l'interprétation des saisies du joueur.

Dans un premier temps, nous avons associé des chaines de caractères à chaque commande, à l'aide d'un ***HashMap***, d'où la classe « ***GoodCommands*** ». Grâce à cette classe il nous était alors possible de savoir si une saisie du joueur correspondait bien à une commande. Ensuite, partant du principe qu'une saisie correcte ne comporte que trois mots maximums, c'est-à-dire, ***cmd objet cible***, nous avons créé la classe « ***AvailableCmds*** » représentant ce principe.
Enfin, nous avons créé la classe « ***CommandsGenerator*** » qui à partir des deux classes ci-dessus, génère la commande correspondante à la saisie de l'utilisateur.

Nous avons séparé le code en différents packages pour améliorer sa lisibilité globale. Nous avons choisi de répartir les classes de manière intuitive en 4 packages :

- **Game** : contient les classes ayant un rapport avec la construction du jeu en lui-même.
- **Places** : contient les classes ayant un rapport avec les déplacements et les positions.
- **Characters** : contient les classes représentant les personnages du jeu.
- **Items** : contient les classes représentant les objets présents dans le jeu.

Quant au programme principal, nous l'avons laissé dans le package Game.

IV. TESTS

Nous avons effectué des tests unitaires sur les classes « ***Hero*** », « ***Exit*** », « ***Place*** » et « ***World*** », car nous avons jugé que ces classes étaient les plus importantes du projet.

Classes	Description	Valeurs en entrées	Résultats attendus
Exit	Création d'une sortie	(null, null, null) et (null, couloir, hall) et (couloir_hall, null, hall)	nom : null voisin 1 : null voisin 2 : null
		P1_to_P2, couloir, cockpit	nom : P1_to_P2 voisin 1 : couloir voisin 2 : cockpit
World	Ajouter une place	null	Ens_Place : inchangé
		cockpit, cockpit	Ens_Place : contient cockpit qu'une seule fois
Hero	modifier de la position	null	position: inchangée
		couloir	position: couloir
	Ajouter un objet	null	inventaire: inchangé
		couteau	inventaire: contient couteau
	Retirer un objet	null	Inventaire: inchangé
		couteau	Inventaire: (taille-1) et ne contient plus couteau
Place	Ajouter une sortie	null	Ens_Sortie : inchangée
		P1_to_P2	Ens_Sortie : contient P1_to_P2
	Ajouter un objet	null	Ens_Objets : inchangé
		couteau	Ens_Objets : contient couteau
	Retirer un objet	null	Ens_Objets : inchangé
		couteau	Ens_Objets: ne contient plus couteau
	Ajouter un personnage	null	Ens_Perso : inchangé
		octopus	Ens_Perso : contient octopus
	Retirer un personnage	null	Ens_Perso : inchangé
		octopus	Ens_Perso : ne contient plus octopus

V. ORGANISATION

Etant donné qu'en parallèle de ce projet, nous avions un projet de programmation système C plutôt complexe, nous avons décidé de désigner un chef de projet pour chacun des projets.
Sangbé SIDIBE avait la charge du projet de JAVA et William FLEURQUIN celle du projet de C.

La charge de travail était répartie en **60/40** sur chacun des projets ce qui nous a permis d'avancer efficacement sur les deux tout en ne négligeant aucun d'eux.

Le travail préliminaire à la programmation (réflexion, commandes à implémenter, diagramme de classe, organisation…) a été réaliser à deux durant les premières séances de TP. Concernant la programmation, Sangbé SIDIBE s'est occupé des packages *Game* et *Places* ainsi que du *Main*. William FLEURQUIN à, quant à lui, réalisé les packages *Characters* et *Items*.

CONCLUSION

Le rendu final du jeu nous satisfait même si nous n'avons pas pu consacrer autant de temps que voulut à ce dernier.

La difficulté de ce projet résidait principalement dans l'organisation puisqu'il est arrivé dans une période durant laquelle nous avions beaucoup d'autres projets et Contrôles Continus ainsi que différents projets personnels à gérer. Malgré quelques difficultés à démarrer, ce travail nous a permis de mettre en œuvre la plupart des notions vue en cours et de mieux nous familiariser avec le fonctionnement et l'utilité des classes abstraites et des Maps notamment.