Rapport du Projet
de Programmation et de Conception Orientée Objet

-The Last Defender-

1. Description du programme fourni

1)Présentation général

The Last Defender est une sorte de A-RPG. Votre ordinateur est infecté par plusieurs virus, ces derniers n'ont été stoppés par aucun antivirus présent. Vous êtes le dernier antivirus : Avast. Votre mission est simple : neutraliser tous les virus qui ont contaminé votre ordinateur avant qu'il ne soit trop tard.

Cela sera possible en passant de salles en salles, c'est-à-dire de composants en composants de votre ordinateur, mais attention chaque composant doivent être débloqué afin d'arriver au coeur du virus.

Faites preuve d'ingéniosité et de courage afin de battre chaque virus présent dans vos composants. Utilisez des objets, communiquez avec les différents personnages, mais surtout, ne confondez pas vitesse, et précipitation !

The Last Defender vise un public s'initiant à l'informatique et qui souhaite apprendre sur l'univers des virus et anti-virus tout en s'amusant.
Difficulté : PEGI 7.
Temps nécessaire : environ 10 minutes.

2) Les différentes actions possibles

Pour pouvoir réussir à vaincre les différents virus, voici quelques actions qui pourraient vous être utile :

- Speak :

Permet de parler avec l'un des différents personnages présents dans la salle dans laquelle vous vous trouvez. Si vous parlez à un personnage amical, ce dernier est susceptible de vous aider d'une quelconque manière que ce soit. En revanche, si vous parlez à un virus, il ne fait nul doute que le dialogue devra se régler en combat ! (Le cas du combat est décrit ci-dessous)

- Take :

Permet de récupérer l'un des différents items présents dans la salle dans laquelle vous vous trouvez. Une fois récupéré, l'item concerné est stocké dans votre inventaire.

- Inventory :

Permet de gérer l'inventaire. L'inventaire n'a qu'une seule utilité : stocker des items. Il peut être consulté hors et pendant un combat. (Le cas de la gestion de l'inventaire est décrit ci-dessous)

- Look Room :

Permet de dévoiler clairement les différents personnages ainsi que les différents items présents dans la salle dans laquelle vous vous trouvez.

- Previous Room :

Permet de se déplacer afin de regagner la salle précédente.

- Next Room :

Permet de se déplacer afin de gagner la salle suivante.

- Help :

Permet d'avoir accès à plusieurs indications concernant les actions réalisables.

- Quitter :

Quitte la partie en cours.

- Cas du combat:

Comme indiqué préalablement, dans le cas d'une demande d'interaction avec un virus, un combat se déclenche. Le système de combat utilisé dans ce jeu est fortement inspiré du système inventé par le jeu de société Risk®. En effet, le combat se déroule suivant des lancés de dés de la part des 2 opposants. Le personnage obtenant le meilleur score remporte la manche, en cas d'égalité, c'est l'adversaire qui remporte la manche.
Chaque personnage dispose d'un kit de dé, chaque dés dispose d'une borne minimum et d'une borne maximum. Ces paramètres peuvent être modifiés en interagissant avec différents items.

Lors d'un combat, plusieurs actions sont possibles :

- Attack :

Chaque personnage obtient un score de dés aléatoire généré en fonction de leurs paramètres respectifs. Si un des 2 opposants obtient, après cette manche, autant de points que de manches requises pour gagner le combat, il remporte le combat. Les conséquences du combat sont alors indiquées adéquatement.

- Use :

Donne accès à l'inventaire et permet d'utiliser un item.

- Run Away :

Abandonne le combat. Les items utilisés pendant le combat ne sont pas récupérés.

Gestion de l'inventaire (hors combat) :

Plusieurs actions relatives à l'inventaire sont possibles:

- Release :

Permet de relâcher un item dans la salle dans laquelle vous vous trouvez actuellement.

- Activate :

Permet d'activer un item, après une activation, l'item disparaît.

- Know More :

Permet d'obtenir des informations sur l'item concerné.

2. Conception et spécification

1) Les changements dans la spécification

En ce qui concerne l'ensemble du projet, c'est-à-dire notre vision globale et nos principales attentes, le résultat obtenu est très conforme à ce que nous attendions. Cependant, et bien évidemment, nous avons été contraint de prendre des décisions qui diffèrent avec notre planification de début de projet.

Premièrement, pour la réalisation de l'interface, nous n'avions presque aucune idée de la manière dont nous allions la développer. En effet, aucun de nous deux n'avions déjà mis en place une interface graphique aussi complète et dynamique en Java et nous ne savions pas quels outils ou techniques nous allions utiliser. Ce point là a donc été une grande découverte et un aspect du projet qui nous a apporté énormément.

La mise en place de l'interface graphique nous a amené à changer plusieurs choses quant à la vision que nous avions sur la finalité de ce projet.

Pour commencer, nous nous sommes vite rendu compte que la création d'un personnage pouvant se déplacer où il le souhaite à l'intérieur même d'une pièce est extrêmement compliqué. De même le déroulement dynamique d'un combat au sens graphique du terme relève de défis extrêmement difficile à relever. Nous avons donc été forcé de repenser totalement notre vision de ces deux sujets.

Tout d'abord, en ce qui concerne la présentation de l'univers du jeu au joueur, nous avons trouvé très intéressant le fait de "raconter" les différents objets et éléments du jeu présents dans l'environnement du joueur. Cependant utiliser ce style de déroulement de jeu tend à risquer de tomber dans un jeu ennuyeux et peu dynamique. En effet aujourd'hui énormément d'animation dans les jeux vidéos résident dans l'aspect graphique.
Nous avons donc essayé de forcer le joueur à imaginer sa propre vision du monde du jeu en lui indiquant tout de même par des courts textes les différents éléments présents dans la salle dans lequel le joueur se trouve, avec à chaque fois un rapport avec le thème que nous voulions mettre en avant : l'univers des composants informatiques.

Or, l'idée d'un jeu dynamique sans graphisme ne pouvait tenir debout que si le joueur avait un minimum de choix, d'actions à réaliser. Pour ce faire, nous avons créé plus d'une dizaine de boutons permettant au joueur d'interagir avec les divers éléments du jeu mais aussi à lui donner l'impression que c'est bien lui qui maîtrise le déroulement du jeu, et que ce n'est pas une simple histoire (par exemple, "look room" force le joueur à faire lui même la démarche de mener son enquête quant à ce qui se passe dans la salle dans laquelle il se trouve, cela peut sembler anecdotique mais rend le jeu bien plus dynamique).

Nous sommes finalement plutôt satisfait de ce que nous avons pu réaliser en terme de dynamisation du gameplay de notre jeu.

Deuxièmement, au début du projet, nous hésitions beaucoup sur la langue que nous utiliserions pour le jeu. Nous avions commencé par coder le jeu en français mais, en cours de route, nous avons décidé de coder en anglais puisque nous pensons qu'à l'avenir, nous coderions la grande majorité du temps dans cette langue et que prendre l'habitude et les bons réflexes en ce qui concerne les noms de variables et de fonctions n'est pas une mauvaise idée pour la suite.
En revanche, pour le côté extérieur du jeu, c'est à dire le côté utilisé par le joueur, nous avons décidé d'utiliser la langue française. En effet nous voulions faire de notre jeu un jeu ludique et simple de compréhension pour un public jeune ou totalement étranger du monde informatique. Ayant beaucoup de descriptions et d'explications, par exemple pour les points historiques des virus, la langue anglaise nous aurait fait perdre cette notion de facilité d'accès au public auquel nous voudrions, par la suite, partager ce projet.

Néanmoins, nous avons utilisé l'anglais pour les actions faites par le joueur car cela rend, selon nous, l'expérience du jeu plus immersive. En effet le monde informatique est, même en France, inondé de termes anglo-saxons. Le fait de mettre les actions en anglais rend donc, d'une certaine manière, plus crédible l'univers du jeu, d'autant plus depuis le regard d'un jeune débutant en informatique.

Mise à part ces deux points, nous n'avons pas changé notre idée de la structure visible de notre jeu.

2) Les aménagements effectués

Pour commencer, nous avons dû créer un package Editor afin de mettre en place l'interface graphique. Dans ce package nous avons ajouté deux classes : EditorController.java ainsi que MainController.java. On y trouve également les fichiers .fxml Interfaces et Main.
Afin d'installer une interface graphique convenablement, nous avons utilisé l'éditeur SceneBuilder de Gluon®. C'est au travers de ce logiciel que nous avons généré nos fichiers .fxml.
Main.fxml correspond à la page d'accueil du jeu, Interface.fxml correspond à l'interface du déroulement du jeu.
A chaque fichier .fxml est relié une classe .java (Main.fxml/MainController.java et Interface.fxml/EditorController.java) dans lesquelles nous associons à chaque bouton une action.

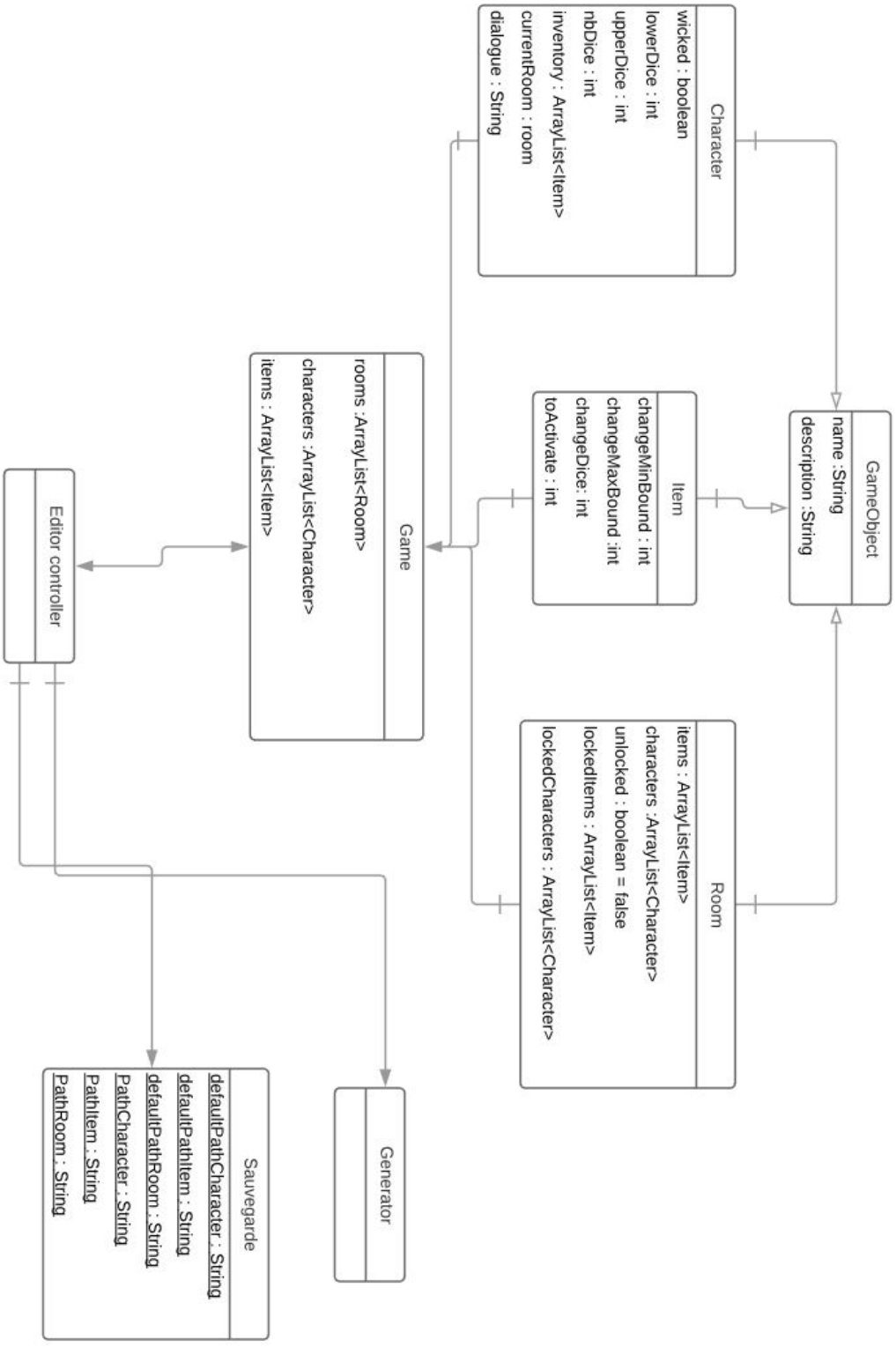
Cela nous a forcé à réorganiser notre boucle main présente dans Classes/Game.java puisque il était nécessaire de générer une scène et, d'autre part, la gérance des actions du joueur devaient être organisées différemment. En effet nous avons commencé à coder en utilisant uniquement le terminal, donc en utilisant des system.out.println ainsi que des scanners, pour communiquer avec le joueur.

D'autre part, l'ajout de la sauvegarde nous a également pousser à penser notre code différemment. En effet, la mise en place d'une telle fonctionnalité nous a forcé à créer des fichiers .csv dans lesquels nous devions stocker toutes les données modifiables du jeu. Nous avons profité de ces fichiers pour rendre plus simple l'initialisation de notre jeu.
En effet, nous avons chargé des fichiers .csv des données par défaut du nouvelle partie. Ainsi, dans le cas d'une demande de nouvelle partie, le jeu s'initialise à partir de ces fichiers .csv.
Il existe un fichier .csv pour chacune des 3 ArrayLists décrites ci-après.

Nous avons organisé les éléments du jeu (rooms, items, characters) dans des ArrayLists publiques attribuées à chaque élément initialisées dans Game.java. Cela nous permet d'avoir accès aux différents éléments facilement depuis EditorController.java.
Pour appeler chacun des éléments, nous avons créé des fonctions statiques qui, à partir d'un nom (String) renvoie l'élément correspondant recherché dans une ArrayList.

Par ailleurs, nous avons trouvé inutile de créer une classe spécialement conçu pour le joueur, puisque ce dernier présente exactement les mêmes propriétés que les autres personnages du jeu. Le joueur est toujours le premier élément de la liste des personnages.

3) Diagrammes de classe


GameObject
name :String
description :String
Character
wicked : boolean
lowerDice : int
upperDice : int
nbDice : int
inventory : ArrayList<Item>
currentRoom : room
dialogue : String
Item
changeMinBound : int
changeMaxBound :int
changeDice: int
toActivate : int
Room
items : ArrayList<Item>
characters :ArrayList<Character>
unlocked : boolean = false
lockedItems : ArrayList<Item>
lockedCharacters : ArrayList<Character>
Game
rooms :ArrayList<Room>
characters :ArrayList<Character>
items : ArrayList<Item>
Editor controller
Generator
Sauvegarde
defaultPathCharacter : String
defaultPathItem : String
defaultPathRoom : String
PathCharacter : String
PathItem : String
PathRoom : String

3. Évaluation de la réalisation

1) Ce qui a été implémenté ou non

Dans le projet final, vous avez la possibilité de lancer une nouvelle partie. Dans cette derrière, l'utilisateur pourra effectuer les différentes actions rappelées dans la première partie. À savoir, la possibilité de parler aux personnages présents dans votre salle ou de les combattre afin d'obtenir des objets ou tout simplement de pouvoir aller dans la salle suivante, mais aussi de ramasser des objets afin de les avoir dans votre inventaire pour pouvoir gagner des bonus ou des malus.

Pour effectuer ces actions, nous avons mis en place un interface graphique avec plusieurs boutons, un comboBox et un TextArea permettant de voir la répercussion de vos actions. Cela facilite l'interaction avec le joueur. Cette interface se compose de plusieurs scènes. La première, la page d'accueil, permet à l'utilisateur de choisir de lancer une nouvelle partie ou une ancienne partie. La deuxième, la page principale, est composées de trois grandes parties : une partie contrôle où se trouve les boutons de commande, une partie description où le joueur peut voir le résultat de ces actions (dialogues, combats, etc.) et pour finir, une partie où le joueur peut voir où il se trouve avec qui et quoi. Une transition a été implémentée afin de rendre plus joli le passage entre les deux scènes. Il n'est pas possible de revenir à la page d'accueil quand nous sommes en train de jouer.

Nous avons implémenté la sauvegarde d'une partie grâce à des fichiers au format CSV. En effet, il est possible de sauvegarder votre partie : c'est-à-dire, le contenu de votre inventaire, les éventuels bonus ou malus perçus au cours de la partie, les ennemis vaincus, et le contenu des salles. Il est important de noter que notre application propose de sauvegarder une seule partie. Si vous sauvegardez une partie, vous écrasez alors l'éventuelle ancienne partie sauvegardée.

Dans la continuité de la sauvegarde, nous avons implémenté le chargement d'une partie sauvegardé.Vous recommencez dans la première salle avec votre ancien inventaire et les salles déverrouillés jusqu'au point de sauvegarde. Si vous n'avez sauvegardé aucune partie, l'application chargera une nouvelle partie.

Notre application ne propose pas de carte pour guider l'utilisateur. Nous avons préféré afficher des informations qui nous semblent plus utiles pour le joueur comme par exemple le nom de la salle avec ce qu'elle contient.

Le déplacement du joueur n'a pas été implémenté avec les points cardinaux. Les salles représentent des paliers de niveau. La première est la plus simple et la dernière est la plus compliquée. Nous n'avons donc eu besoin d'implémenter que deux boutons permettant de revenir dans la salle précédente ou d'aller dans la prochaine salle.

Nous n'avons pas implémenté l'attaque automatique d'un personnage qu'on le joueur rentre dans une salle. De même de l'action de parler. En effet, seul le joueur de décider avec qui combattre ou discuter.

2) Tableau synthétique

Objectif	**Accompli à**	**Commentaire**
1. Déplacement	100%	Les boutons previous room et next room permettent de se déplacer dans les différentes salles.
2. Observer	100%	Le bouton look room permet d'observer le contenu d'une salle (Items présents, personnages présents).
3. Personnages	100%	Des personnages alliés ou ennemis sont présents dans les salles.
4. Objet	100%	Des objets sont présents dans les salles.
5. Ramasser	100%	Le bouton took permet de ramasser un objet présent dans la salle du joueur.
6. Inventaire	100%	Le bouton inventory permet de voir ce que contient notre inventaire.
7. Interaction	100%	Le joueur peut relâcher, utiliser ou demander des informations sur un objet de son inventaire.
8. Parler	75%	Le bouton speak permet de parler à tous les personnages présents dans la salle du joueurs.
9. Combattre	60%	Le bouton attack permet d'attaquer un adversaire présent dans la salle du joueurs.
10. Prototype	100%	Le prototype a été rendu avec les fonctionnalité attendus.
11. Interface	100%	Une page d'accueil permet de lancer une nouvelle partie ou une ancienne partie. Une seconde scène permet de gérer les actions du joueur.
12. Histoire	25%	L'histoire est composé de 6 salles.
13. Carte	0%	Pas de carte disponible.
14. Sauvegarde	50%	Sauvegarde qu'une partie.
15. Bonus / Malus	50%	Les bonus et malus sont disponible en activant les objets.

3) Les éléments techniques à changer

Avant la réalisation de ce projet, nous n'utilisons pas le même IDE, il a donc fallu s'adapter afin de choisir un IDE commun. Cela nous a évité de nombreux problèmes techniques.

Le principal élément technique que nous regrettons est notre utilisation de trois ArrayList dans notre classe Game.java. Ces ArrayList stockent toutes les salles du jeu, tous les personnages du jeu et tous les items du jeu. Pour avoir accès, à un objet en particulier, il a fallu faire une fonction qui parcourait la List et qui retourne par exemple le personnage avec le nom JigSaw. Nous regrettons aussi le fait que ces trois List sont public ce qui nous permet d'y avoir accès dans toutes les autres classes du projet. Nous savons que cela n'est pas la bonne méthode, mais nous n'avons pas eu le temps de faire autrement. Cependant, nous avons pensé à une solution.

En effet, nous aurions pu utiliser des Maps de type qui nous aurait permis d'avoir accès aux objets (Salle, Item, Personnage) grâce à leurs noms et ainsi ne pas avoir besoin de créer des fonctions auxiliaires pour récupérer un objet particulier. Nous pensons qu'il aurait été plus facile de comprendre notre code avec ces Map car cela nous aurait éviter de récupérer les personnage avec leurs positions dans l'ArrayList.

Nous regrettons aussi l'utilisation du format CSV pour la sauvegarde car il peut être difficile de le lire quand nous ne connaissons pas le code. Mais nous l'avons choisi à cause de sa facilité d'implémentation et nous avons déjà eu l'occasion de travailler sur ce format ce qui nous a permis d'éviter certaines erreurs commises dans d'ancien projet.

4. Organisation du Travail

1)L'organisation temporelle de notre travail

- Imagination du thème, des principes du jeu (1 jour)
- Conception (1 semaine)
- Création d'un GIT et mis en place des outils qui en découlent (½ jours)
- Développement de la structure du code (1 semaine)
- Ecriture d'un scénario prototype (½ jour)
- Mise en place d'un premier système de sauvegarde (3 jours)
- Création des actions et interactions possibles (3 semaines)
- Mise en place d'une première interface graphique (1 semaine)
    - Apprentissage de scene builder

    - “Traduction” du dialogue par terminal en interaction par interface avec boutons

- Amélioration sauvegarde : ajout du chargement de partie déjà existante (1 semaine)

- Invention des salles, personnages, items (3 jours)
    - Documentation et recherches sur les virus et anti-virus afin de créer un environnement cohérent et réaliste
    - Ecriture d’un scénario
    - Mise en place de propriétés cohérentes pour chaque élément

- Perfectionnement de l’interface graphique (1 semaine)
    - Design
    - Ajout de fonctionnalités

- Corrections de bugs, tests, ajout de fonctionnalitées optionnelles (1 semaine)

- Rédaction du Rapport (1 semaine)

2) La répartition des tâches

Dawen CALIPPE :

- Conception du prototype

Au début du projet, la conception de ce dernier a été très flou pour nous deux. Nous n’arrivons pas à voir quelles classes nous allons utilisé avec quelles attributs. Mais au fur et à mesure du temps, nous avons eu une idée plus précise de nos besoins. Le prototype, nous a permis de voir les erreurs de notre première conception afin de les corriger pour la suite du projet.

- Gestion de la sauvegarde et des .csv

Je me suis occuper de la sauvegarde et de l’import d’une partie. J’ai choisi d’utiliser le format CSV. Format qui n’était pas inconnu pour moi car j’avais déjà travaillé sur la sauvegarde dans un projet précédent. J’ai décidé d’utiliser des fichiers csv pour la création d’une nouvelle partie. En effet, la nouvelle partie se crée grâce à des fichiers CSV de défaut. Une fois cette étape réussite, il a été très rapide de m’occuper de la partie export.

- Perfectionnement de l’interface graphique

J’ai perfectionné l’interface graphique en ajoutant notamment la page d’accueil qui permet au joueur de lancer une nouvelle partie ou une partie sauvegardé. J’ai ensuite rajouter du style au fichier FXML pour rendre le jeu plus agréable à regarder. Pour finir, j’ai rajouté une transition entre les deux scènes toujours dans l’idée d'ajouter du style à notre application.

Elie BOURY :

- Développement de la structure du code

J'ai commencé par créer les classes et les premières méthodes et attributs en me basant sur le diagramme de classe en cours de création. Nous avancions donc ensemble, Dawen avançant la conception, moi-même le code. Cela nous a permis de démarrer avec la même vision d'ensemble du projet et d'établir nos différents rôle par la suite.

- Mise en place d'une première interface graphique

Comme expliqué préalablement, j'ai d'abord essayé de créer une interface "à la main" mais nous nous sommes rendu compte de la complexité du travail requis et du manque d'expérience en la matière. Cela nous a donc poussé à utiliser l'éditeur SceneBuilder. J'ai d'abord retransmis la console sous la forme d'une textArea, puis j'ai installé les boutons et traduit les fonctions pour que le jeu ne soit pas obligé d'attendre un message inscrit dans le terminal mais une action exercée sur les boutons. Cela m'a contraint à réorganiser plusieurs fonctions.

- Ecriture d'un scénario prototype

Le principal problème était de ne pas faire un jeu trop répétitif, en effet le concept du jeu rend les actions quelque peu stéréotypées.

- Invention des salles, personnages, items

Je me suis efforcé à rendre le jeu ludique, de manière à respecter notre objectif initial. J'ai essayé de faire en sorte que chaque personnage, chaque item, soit en relation avec le composant dans lequel le joueur se trouve en fonction de l'histoire des virus dans le monde informatique.

Ensemble :

- Imagination du thème, des principes du jeu

Au tout début du projet, nous avons commencé par un BrainStorming et cela nous a petit à petit amené à l'idée générale que nous avons ici, dans notre jeu actuel.

- Communication

Tout au long du projet, même si nous faisions parfois des tâches très séparément, nous nous sommes toujours tenu au courant de nos avancement afin de pouvoir nous aider réciproquement et de n'avoir aucune zone d'ombre dans notre projet.

5. Perspectives

Pour conclure, ce projet nous a permis de voir à quoi pourrait ressembler le développement d'une application et nous a permis de nous rendre compte des possibles difficultés que cela entraîne. Cela nous a aussi permis d'approfondir nos connaissances, par exemple sur l'utilisation de javaFX ou de logiciel comme SceneBuilder. Le travail d'équipe et l'organisation ont été la clé de la réussite de ce projet. Cependant, si nous avions un budget et des ressources humaines et temporelles conséquentes pour finaliser ce projet, nous souhaitons :

- Une amélioration de la partie graphique. En effet, nous avons pensé à utiliser des graphismes qui permettraient au joueur de se croire à l'intérieur des composants d'un ordinateur (les salles) et de pouvoir être dans la peau de l'anti-virus. Cela impliquerait l'utilisation de 2D et une manière totalement différente de jouer.
- Un rajout d'une bande son. Les effets sonores sont tout aussi important que les aspects graphiques selon nous. C'est pourquoi nous avons pensé au rajout de son caractérisant chaque action du joueur. Par exemple, une lecture audio pour les dialogues.
- Une amélioration de l'histoire qui est pour l'instant beaucoup trop courte. Il faut rajouter des salles mais aussi des actions et des objets afin de donner plus de plaisir au joueur et d'éviter qu'il ne s'ennuie ou ne se lasse avant la fin de la partie. Le scénario doit donc être plus accrocheur, immersif, et intéressant.
- Une amélioration de notre code. Comme nous l'avons dit dans la troisième parti, il y a des aspects techniques que nous regrettons et que nous voulons changer.
- Le système de combat peut être lui aussi amélioré en prenant en compte l'expérience du joueur. En effet, un anti-virus est devient plus puissant avec le temps car il rencontre de nouveau virus et apprend à les vaincre.

Pour finir, nous tenons à remercier notre professeur encadrant qui nous a toujours pris le temps de répondre à nos questions.