

Dossier du projet (20-21)

Groupe : Defoy Nathan, N'Mily Adam & Goessel Florian

1. Documents pour CPOO :

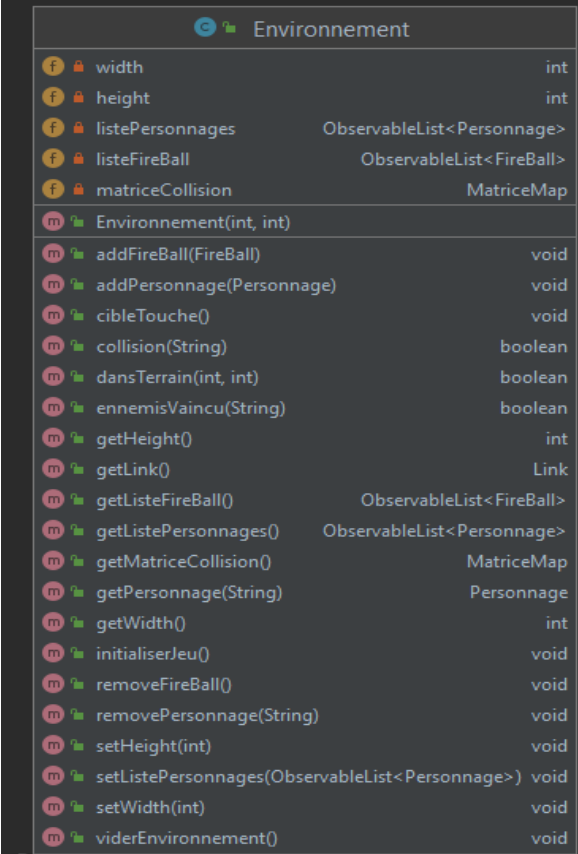
1.1) Diagramme de classe

Notre projet est constitué de trois packages principaux (modèle, vue, contrôleur) respectant le modèle MVC. Notre package modèle représente le cœur du jeu, il définit l'ensemble des fonctionnalités du jeu. Le modèle est composé de sa classe principale "Environnement", mais aussi de deux sous packages qui sont "personnages" et "objet". Ce dernier possède par ailleurs deux sous package "objetAttaque" et "objetDivers".

La classe Environnement :

La classe "Environnement" regroupe tout ce qui doit évoluer en son sein. C'est pourquoi elle possède une liste comprenant les personnages et une autre comprenant des boules de feu, des dimensions et une matrice de collision afin de gérer si besoin, les personnages et les objets avec les éléments du décors.

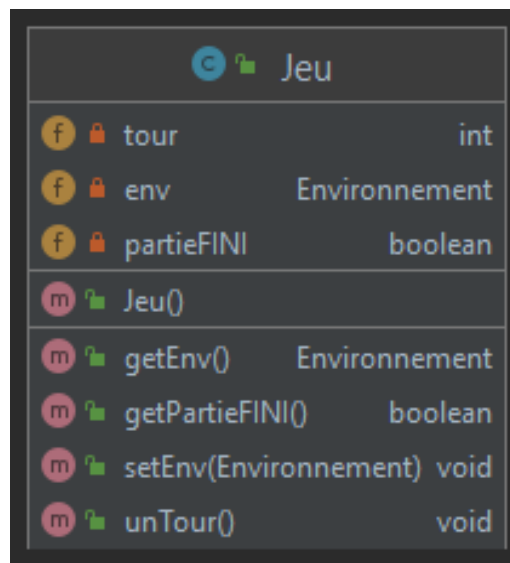
Elle possède plusieurs méthodes, dont certaines lui permettant d'ajouter ou retirer les éléments des listes, de rechercher un personnage via son nom, savoir si un personnage est mort/vaincu ou s'il entre en collision avec un élément du décor, faire évoluer les boules de feu dans l'environnement.



Environnement	
f	width int
f	height int
f	listePersonnages ObservableList<Personnage>
f	listeFireBall ObservableList<FireBall>
f	matriceCollision MatriceMap
m	Environnement(int, int)
m	addFireBall(FireBall) void
m	addPersonnage(Personnage) void
m	cibleTouche() void
m	collision(String) boolean
m	dansTerrain(int, int) boolean
m	ennemisVaincu(String) boolean
m	getHeight() int
m	getLink() Link
m	getListeFireBall() ObservableList<FireBall>
m	getListePersonnages() ObservableList<Personnage>
m	getMatriceCollision() MatriceMap
m	getPersonnage(String) Personnage
m	getWidth() int
m	initialiserJeu() void
m	removeFireBall() void
m	removePersonnage(String) void
m	setHeight(int) void
m	setListePersonnages(ObservableList<Personnage>) void
m	setWidth(int) void
m	viderEnvironnement() void

Powered by yFiles

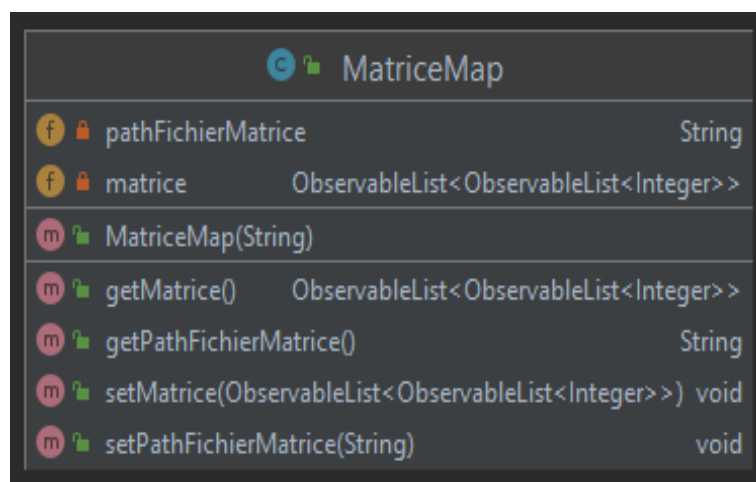
C'est la classe Jeu qui se charge de réaliser un tour de jeu et d'initialiser l'environnement dans son constructeur.



Jeu		
f	tour	int
f	env	Environnement
f	partieFINI	boolean
m		
m	Jeu()	
m	getEnv()	Environnement
m	getPartieFINI()	boolean
m	setEnv(Environnement)	void
m	unTour()	void

La classe MatriceMap :

Cette classe à pour but de reproduire la matrice d'un fichier ".txt" à partir de son "Path" et doit représenter les éléments de décors d'une Map. La "Vue" pourra alors se charger de faire une reconstruction de la map à partir de cette matrice et du tileSet adéquat de façon dynamique avec des tuiles de 16x16. Mais l'environnement pourra l'utiliser pour gérer les collisions, les collisions étant représentées par un entier différent de -1.

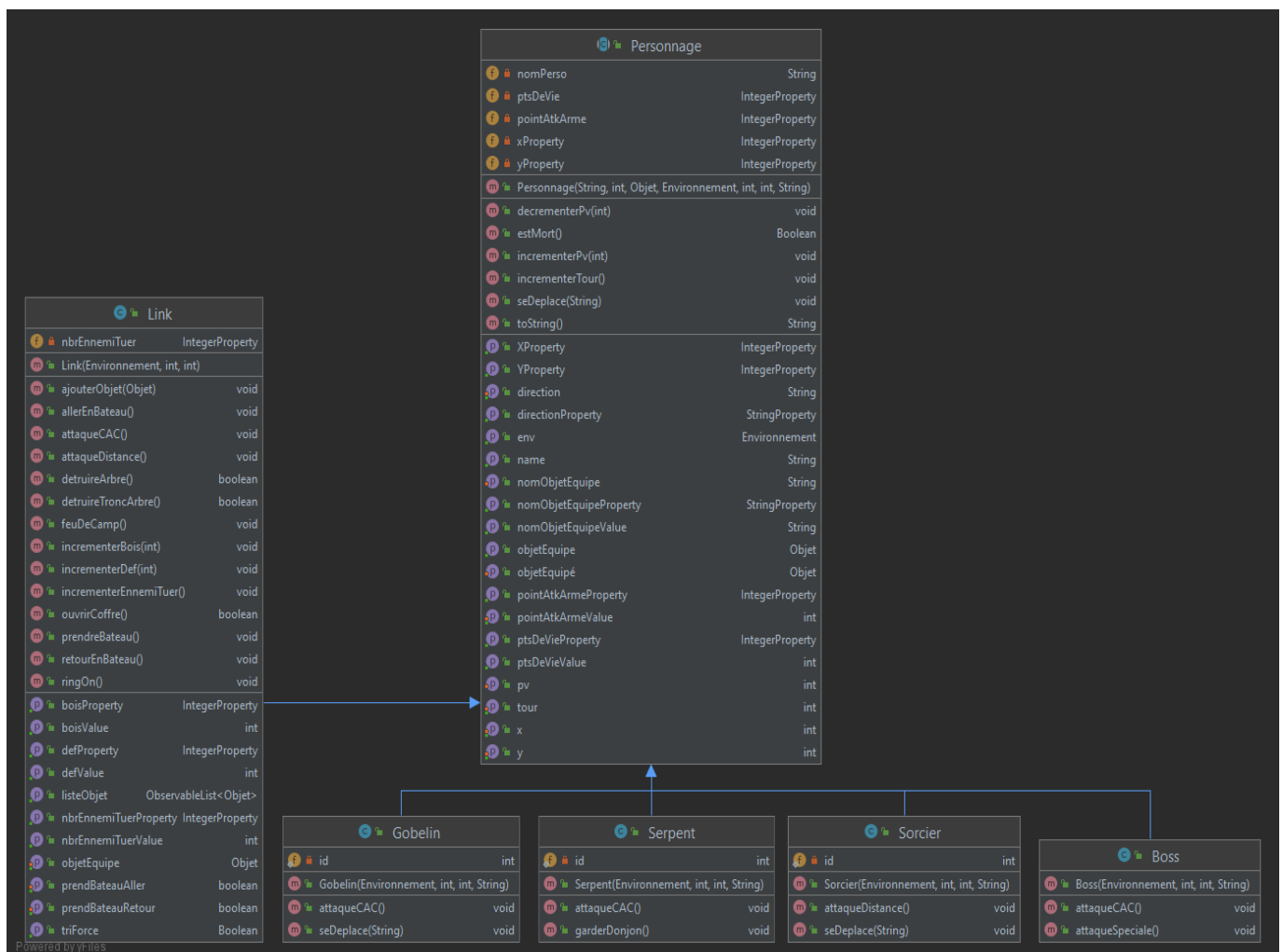


MatriceMap		
f	pathFichierMatrice	String
f	matrice	ObservableList<ObservableList<Integer>>
m		
m	MatriceMap(String)	
m	getMatrice()	ObservableList<ObservableList<Integer>>
m	getPathFichierMatrice()	String
m	setMatrice(ObservableList<ObservableList<Integer>>)	void
m	setPathFichierMatrice(String)	void

La classe Personnage et ses sous-classes :

Contenu dans le package personnages, la classe Personnage est la super classe de tous les types de personnages. Elle est rendue abstraite pour empêcher une instantiation de cette même classe, définir les attributs et fonctionnalités commune entre tous les sous-types comme être capable de se déplacer, la possibilité de rendre abstract une méthode que toutes les sous-classes auraient besoin de coder à leur manière. L'une des classes majeures qui étend la classe Personnage est Link.

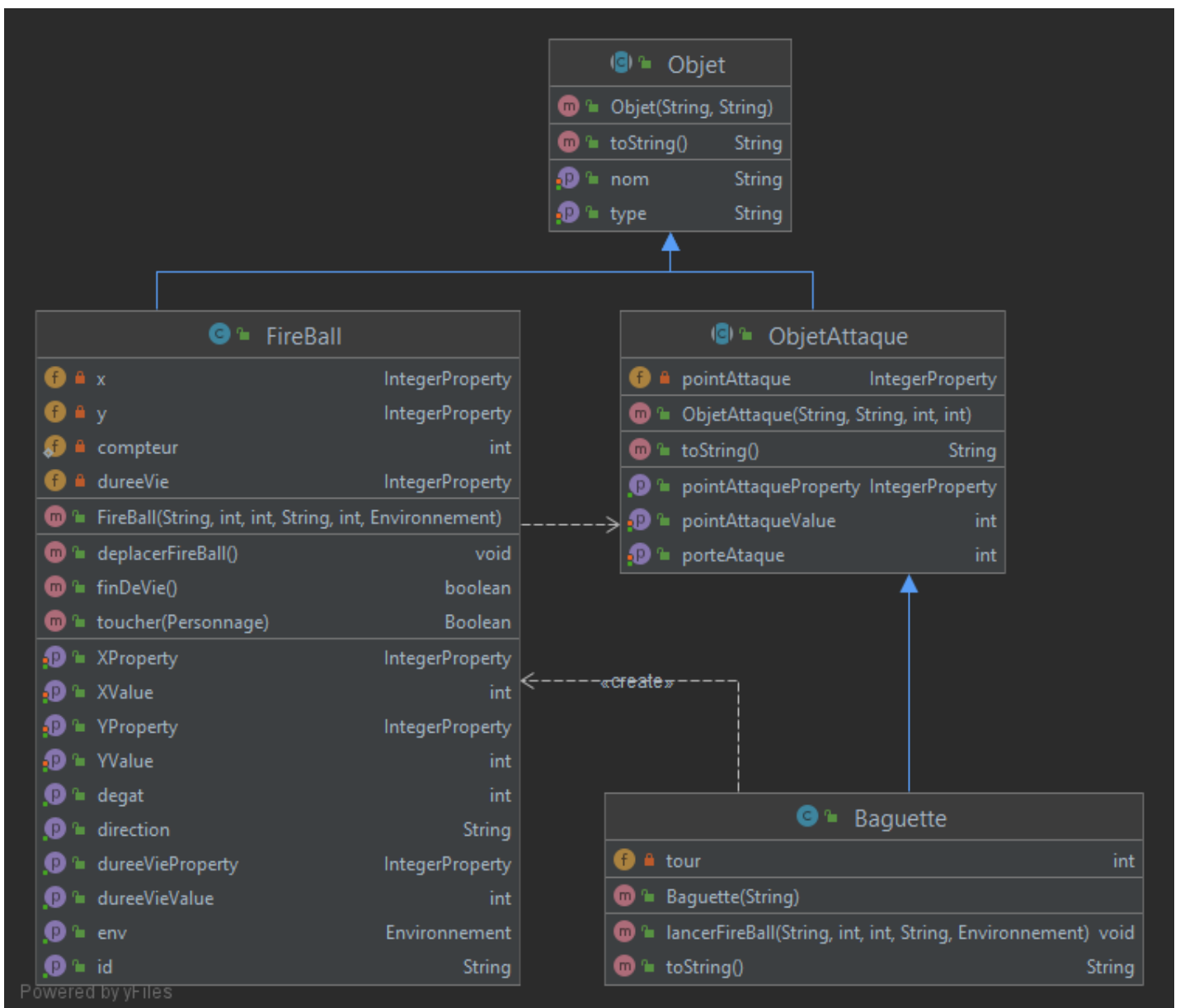
La classe Link étend la classe Personnage et possède plein de nouveaux attributs et de méthodes qui répondent aux responsabilités que le personnage a dans le jeu et par extension le joueur. Link doit être capable d'attaquer au corps à corps et à distance mais aussi de se protéger, pour cela, il possède une liste d'objets qui évoluera dynamiquement au fur et à mesure que la partie avance. Il doit aussi être en mesure d'ouvrir des coffres, détruire des obstacles, récupérer des matériaux comme du bois, se rendre d'un point à un autre grâce à un moyen de transport et régénérer ses points de vie. Toutes ces capacités regroupent au final la plupart des autres types de personnages en plus de celles qui lui sont propres.



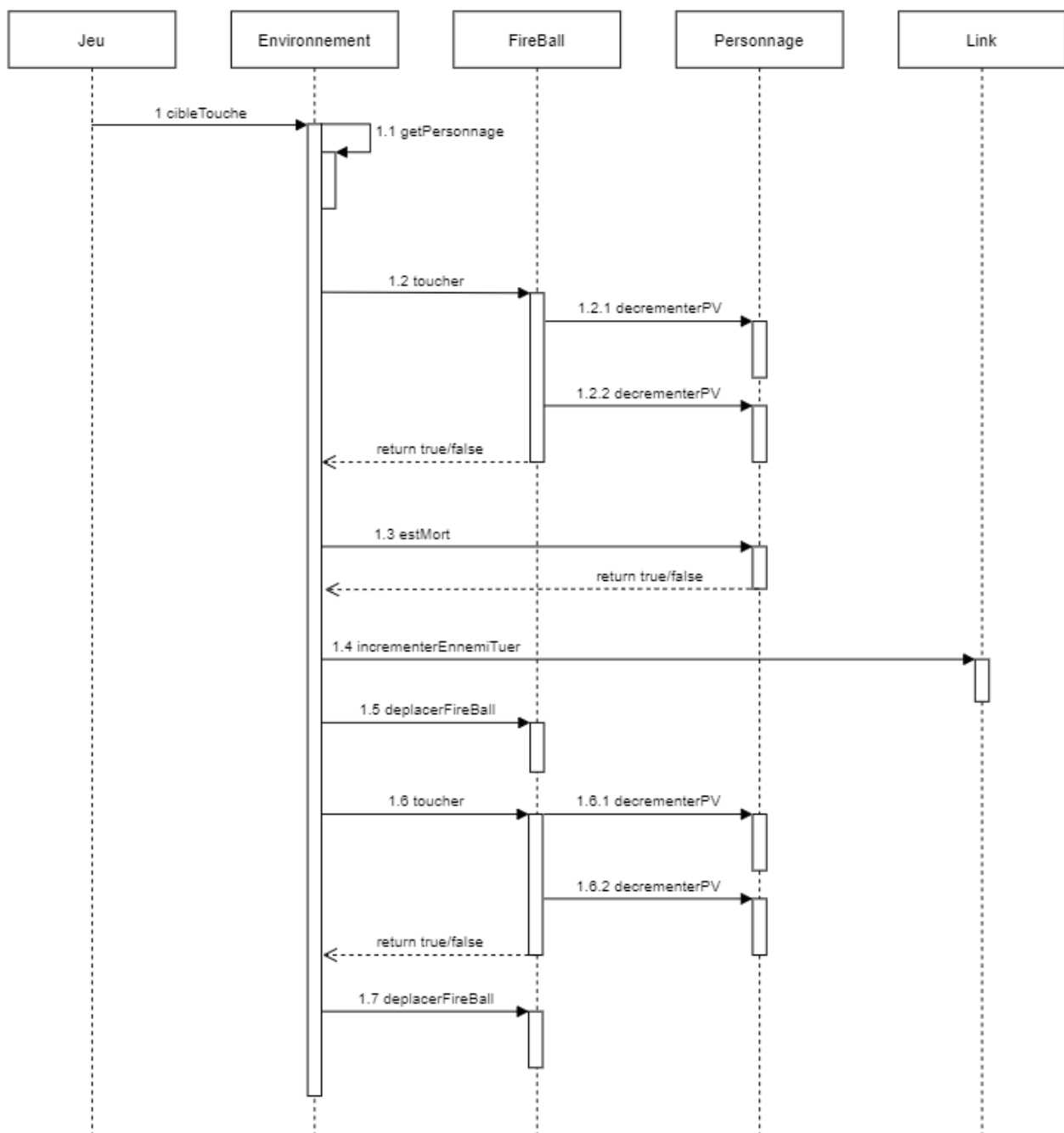
Les classes d'objet :

Tous les personnages du jeu possèdent un unique objet (sauf Link, il peut en avoir plusieurs), les objets étendent tous au minimum la classe Objet. Certains peuvent étendre la classe ObjetAttaque ou ObjetDivers qui eux même étendent la super classe Objet.

Par exemple la classe Baguette étend la classe objetAttaque qui définit les propriétés communes à tous les objets d'attaque qui elle-même étend la super classe Objet définissant pour n'importe quels sous types un nom et son type sous forme de String. La classe Baguette peut créer un autre objet tel qu'une FireBall via sa méthode lancerFireBall, Elle s'ajoutera dans la liste de boules de feu de l'environnement. L'objet FireBall étend la classe Objet et possède une durée de vie qui diminuera à chaque tour de jeu.



1.2) Diagramme de séquence :



Pour notre diagramme de séquence, nous avons décidé de mettre en avant le fonctionnement de la méthode « cibleTouche » de la classe « Environnement ». Cette méthode a pour but de parcourir la liste des boules de feu que l'environnement possède et de vérifier, pour chacune des boules de feu, si elle touche un personnage en appelant la méthode « touche » de chaque boules de feu pour chaque personnage de la liste des personnages. Cette méthode distingue les boules de feu de Link de celles de ses ennemis, La méthode « touche » de chacune des boules de feu renvoie un boolean. En fonction de la valeur retournée elle se charge de déplacer les boules de feu et d'incrémenter le compteur d'ennemis vaincus de link ou non. La méthode « touche » permet d'examiner si les coordonnées de la boule de feu sont

comprises dans celle du personnage. Auquel cas, elle appelle la méthode « decrementsPV » du personnage. Elle se charge pareillement de vérifier si Link possède un bouclier afin de savoir si la boule de feu est en mesure de lui faire de dégât.

1.3) Présentation d'algorithmes

Parmi les algorithmes intéressants que nous avons implémentés dans le jeu il y a :

- **La méthode attaqueCAC du Boss** : Le Boss de notre jeu, l'Ogre, possède une attaque dont la zone d'action est un cercle et dont le rayon correspond à la portée d'attaque de son arme + la moitié de sa taille; le point central du cercle correspondant à $x = \text{moitié de la largeur du personnage}$ et $y = \text{moitié de la longueur du personnage}$. Pour ce faire nous avons utilisé la formule permettant de calculer la distance entre deux points $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$. Nous calculons la distance entre Link et le Boss et si la distance est inférieure ou égale au rayon d'action prévu, l'ennemi se fait attaquer (sauf s'il possède un bouclier).
- **La méthode seDeplace de la classe Personnage** : Cette méthode définit une vitesse de base, positive, et prend une direction en paramètre. Le principe, ici, est de savoir si l'on se déplace sur l'axe des x ou des y, et dans quel sens. On sait que si la direction est négative, nous allons soit à gauche, soit vers le haut. On teste donc si cette condition est vérifiée. Auquel cas, la vitesse vaut $\rightarrow -\text{vitesse}$. Ensuite il suffit de tester si l'on se déplace sur l'axe des x ou des y donc l'une des deux conditions possibles : direction = Gauche ou Droite (on se déplace sur l'axe des x sinon sur l'axe des y).
- **La méthode chargerMap de la classe AffichageMap** : Pour afficher nos différents calques réalisés avec l'éditeur "Tiled" nous faisons en sorte de parcourir notre matrice de calque. Elle est contenue dans une double ArrayList d'entier afin de construire pour chaque numéro de tuile, un rectangle dont les dimensions sont 16x16 et la position x, y correspondant à cette même tuile sur notre tileSet, afin de faire un crop de l'image et l'ajouter à notre TilePane.
- **La méthode Toucher de la classe FireBall** : Cette méthode a pour but de vérifier si une boule de feu touche un ennemi. Auquel cas elle lui fait perdre des points de vie. Pour cela il faut vérifier si les coordonnées x et y de la boule de feu sont comprises entre les coordonnées x et y du personnage sur qui est testé la méthode. La méthode teste aussi le

cas où il ne s'agit pas de la boule de feu de link, dans ce cas précis le personnage sur qui est testé la méthode est Link et il faut s'assurer qu'il n'a pas de Bouclier pour lui faire des dégâts.

2. Information pour le code :

2.1) Structures de données :

Nous utilisons dans notre projet des ObservableList et des Property dans les classes « Environnement », « MatriceMap », « Link », « Personnage », « FireBall », « ObjetAttaque »

Ces structures de données nous permettent d'observer les différents changements ayant lieu dans le Modèle et de pouvoir agir en conséquence sur la vue.

2.2) Exception :

Aucune classe ne contient de réelle gestion d'exception intéressante.

2.3) Junits :

Aucune classe Junits n'a été réalisée durant ce projet.

3. Document utilisateurs :

3.1) Description du jeu (son objectif, son univers...)

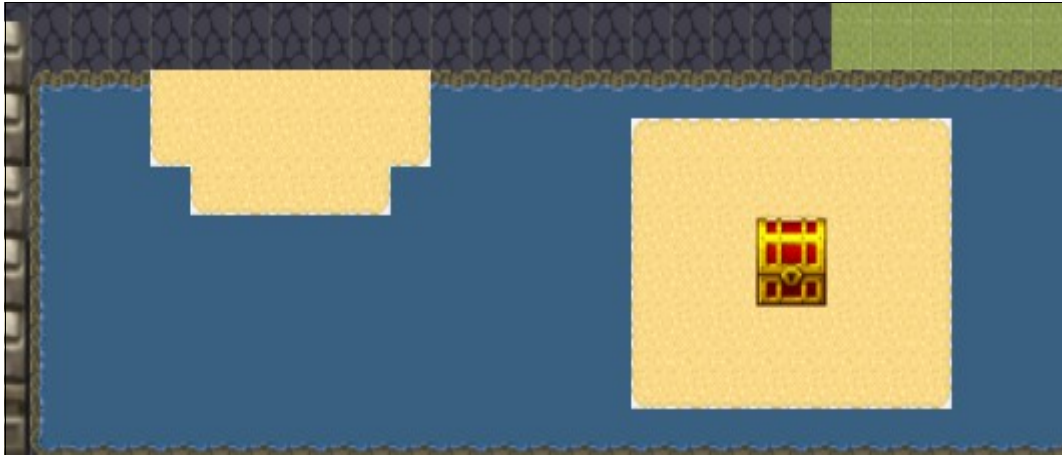
Vous êtes Link, vous apparaissez près d'un feu de camp et devez évoluer dans un univers volcanique. Là-bas, le sol est couvert de roche, les arbres sont, pour la plupart, vétustés, et les ennemis sont assoiffés de sang. Vous devez vous rendre au sud-ouest où se logent deux gobelins. Si vous parvenez à les tuer, vous obtiendrez dans le coffre une baguette magique !



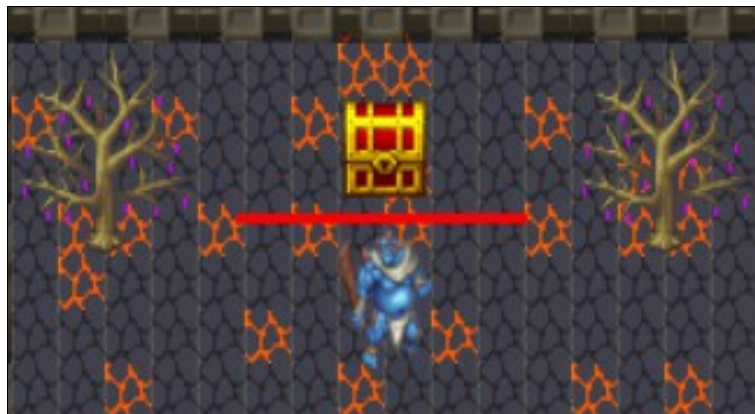
Des rondins de bois se sont éparpillés à plusieurs endroits sur la map. Certains vous obstruent donc le passage. Il vous suffit de les casser avec votre épée, de tuer les ennemis et d'obtenir le célèbre bouclier d'hyrule.



Vous remarquerez très rapidement deux arbres a feuilles dorée en parfait état sur la carte. Cassez-les grâce a l'épée pour obtenir du bois. Vous pourrez alors vous rendre sur l'île au coffre de la bague.



Malgré les obstacles que représentent l'environnement et les ennemis, vous devrez marcher jusqu'au boss final, et essayez de le vaincre afin de récupérer l'une des Tri-force d'Hyrule, celle du courage.



3.2) Description des armes et accessoires de Link, des types d'éléments de la map, des ennemis (tableau de relation ...).

Armes et équipements :

- Vous débutez avec une épée en métal, celle-ci fait 20 points de dégâts par coup.
- Votre 2e arme sera une baguette magique propulsant des boules de feu infligeant 10 points de dégâts par boule de feu.
- Une troisième armes peut être acquise, il s'agit du bouclier celle-ci bloque les attaques



reçue et permet au joueur de se sortir indemne de toutes attaques.

- Le dernier équipement trouvable est une bague permettant d'augmenter vos capacités. En effet, elle augmente les points d'attaque de l'épée à 60 dégâts par coup et 50 pour la baguette magique. Elle met aussi vos points de défense à 25. Et augmente vos points de vie jusqu'à 500.



Ennemis :

Le premier ennemi que vous croiserez sera un Gobelin Vert attaquant à la batte en bois. Il inflige jusqu'à 15 points de dégâts par coup et possède 50 points de vie.



Il peut être accompagné de Sorciers, possédant quant à eux 30 points de vie, lançant eux aussi des boules de feu réduisant vos points de vie de 10 par coup.



Enfin, il y a le boss final, un Ogre avec 300 points de vie qui se voit infliger 15 points de dégâts par coup avec sa hache.

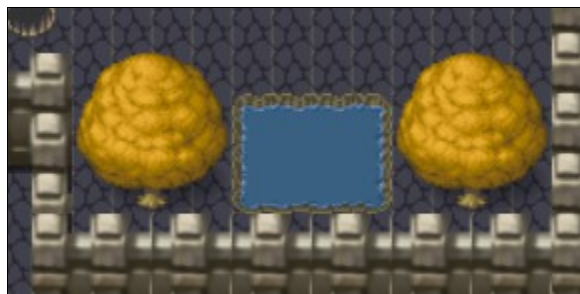


Il existe aussi les Serpents, ils ont 1 millions de points de vie et vous mord si vous les touchez. Vous perdrez alors 10 points de vie.



Types d'éléments sur la map :

Dans la zone au nord-est de la map, vous trouverez deux immenses arbres en parfait état. Il vous suffit de les casser avec votre épée afin d'avoir 20 unités de bois par arbre.



Ceux-ci détruits, vous pourrez vous rendre au sud afin de construire le navire fantôme nécessitant 40 unités de bois. Ce bateau vous emmènera sur l'île au coffre de la Bague.

Le feu de camp peut s'avérer aussi très utile. Après un combat acharné, si vous revenez près de celui-ci, il restaurera vos points de vie de 10 en 10 jusqu'à 300. Attention, il ne fonctionne pas si vous avez la bague en main !



Ensuite, les rondins de bois gênant votre passage ne sont pas très solides. Ils peuvent être détruits à l'épée.

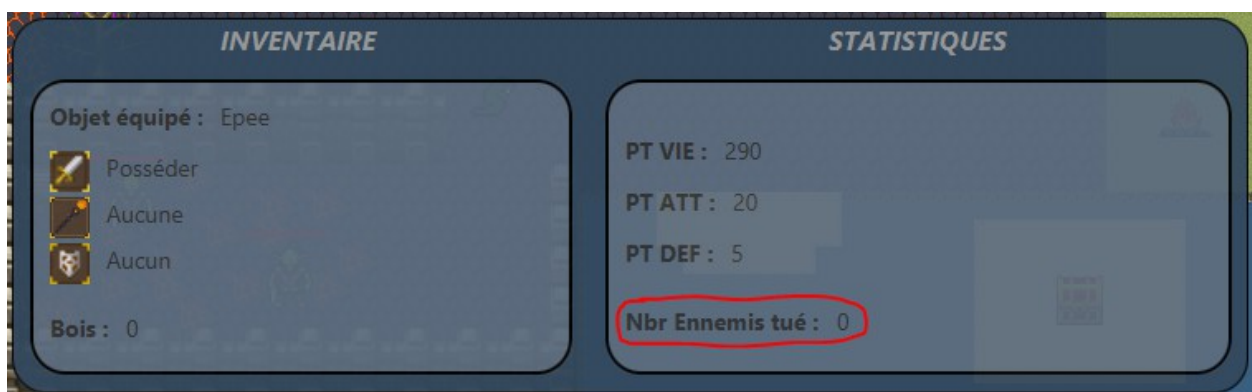


Enfin, nous avons placé des coffres contenant des armes et des équipements pour aider Link dans sa quête. Évidemment, vous pouvez les ouvrir.



3.3) Scoring.

Le principe de "scoring" dans notre jeu se résume à un compteur de mort qui s'incrémente de 1 à chaque ennemi éliminé. Link possède une property qui indique le nombre d'ennemis tués. A chaque fois qu'un personnage est tué cette property s'incrémente de 1 et cette property est bind avec le label "Nbr Ennemi Tué" via le contrôleur.



3.3) Fonctionnalités

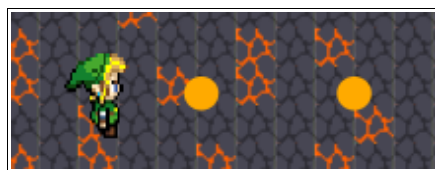
- Il existe plusieurs fonctionnalités dans notre jeu. La première est d'ouvrir les différents coffres présents dans la zone permettant de récupérer des armes.

- Nous pouvons en outre emprunter le magnifique navire fantôme présent en bas de la zone, qui est empruntable seulement si un certain nombre de bois coupé est présent dans le sac du joueur. Le bateau amène le joueur sur une île, où un coffre y est situé.



- La suivante est celle de pouvoir détruire les troncs d'arbres + les arbres en bonne santé. Détruire les troncs d'arbre permet notamment d'accéder à de nouveaux endroits de la map. Casser les arbres en bonne santé permet de récupérer le bois suffisant à la construction du navire.

- Le joueur dispose de 3 armes différentes: épée, baguette magique et bouclier. Chacune apporte une manière de jouer différente. L'épée, pour les dégâts au corps à corps, permet d'infliger le plus de dégâts, la baguette magique, pour blesser à distance, permet d'avoir une approche moins risquée. Idéale si vous ne voulez pas vous faire toucher par les monstres. Et enfin, le bouclier permet d'être invincible, mais empêche d'infliger des dégâts à l'ennemi.



(Link lançant des boules de feu avec sa baguette magique)

- Nous avons eu recours au déplacement vertical et horizontal, à l'inverse d'un classique MarioBros. Link peut se déplacer de haut en bas et de gauche à droite. Il ne peut cependant pas sauter.

- En pleine partie, le joueur peut jeter un œil à ses points de vie, son arme équipée, son nombre de bois récolté ou le nombre de monstres tué en appuyant sur la touche i. Il s'agit de notre inventaire.

- Pour finir, nous avons créé un menu de début de partie pour que le joueur puisse se

repérer plus facilement au lancement du jeu. L'action ne commence pas tout de suite, il peut regarder les commandes, s'instruire de la quête du héros ou bien quitter le jeu si l'envie lui en prend.



(Menu principal)

Nous nous sommes donnés à cœur joie dans la réalisation de ce jeu, en espérant qu'il vous plaise, bonne partie Héros !