TOWER DEFENSE : Bee Defense

Majda BENMALEK Amenah MUSHTAQ L2 INFO 1

Introduction :

Le sujet de ce rapport est la réalisation d'un Tower Defense (Bee Defense). Un Tower Defense est un jeu dont le but est de protéger sa base en plaçant stratégiquement des tours afin d'éliminer l'armée d'ennemis. Un jeu Tower Defense a deux styles un où les ennemis avancent en lignes droites et le joueur doit placer ses tours sur leurs chemins et un autre où les ennemis avancent le long d'un chemin sinueux, le joueur doit placer ses tours en dehors du chemin. Nous avons choisi le premier style. Dans notre jeu, le joueur doit donc protéger la ruche des chenilles, des araignées et des frelons qui sont friands de miel. Pour cela il peut placer différents types d'abeilles : les abeilles snipers, les abeilles venimeuses et les abeilles butineuses. Le miel permet « d'acheter » les abeilles. Chaque abeille possède une attaque normale et certaines abeilles ont des atouts en plus. Enfin le joueur possède d'un dernier atout : les filets. Les filets sont en quantités limités et permet au joueur d'éliminer tous l'ennemis qui se trouve à côté de la ruche ainsi que les ennemis présents sur la ligne où se trouve le filet.

Voici des tableaux présentant le rôle de toutes les entités présentes dans le jeu.

Abeille	Prix	Rôles	Attaque normal Dégâts	Attaque Spécial +dégâts	Point de vie
Sniper	25	Lance des projectiles	20	0	100
Butineuse	30	Donne du miel	0	Donner du miel 5	150
Venimeuse	50	Pique l'ennemi s'il est proche de lui	10	L'adversaire est blessé s'il est piqué et perds des points de vie pendant plusieurs tours	50

Ennemis	Rôles	Attaque normal dégâts	Attaque spéciale +dégâts	Point de vie
Chenille	Attaque si une abeille est proche d'elle	25	X	100
Frelons	Pique l'abeille si elle est proche de lui	50	X	100
Araignée	Attaque et lance des toiles	10	Ses toiles immobilisent les Abeilles qui ne peuvent ni attaquer ni donner du miel	100

Nous verrons par la suite les parties du cahier des charges qui ont été traités, une description détaillée de la structure de notre projet pour les modes terminal et graphique (comportant un schéma modélisant la structure de notre jeu, des tableaux présentant chaque classe avec leurs champs leurs méthodes accompagnés d'un texte explicatif de nos choix et les difficultés connus lors de la conception) et pour finir les pistes d'extensions que nous n'avons pas encore implémentés.

Respect du cahier des charges :

Dans cette partie nous verrons en détails les différents aspects du jeu imposés par le cahier des charges.

Notre jeu contient un menu qui explique les règles du jeu en plus, qui permet de paramétrer le jeu. En effet le jeu contient un mode normal où les vagues d'ennemis sont limités et connus par le joueur et un mode marathon où les vagues d'ennemis ne sont pas connus par le joueur, il devra donc protéger sa ruche le plus longtemps possible.

Le mode normal contient différents niveaux de difficultés : le niveau facile, le niveau moyen et le niveau difficile.

Voici un tableau présentant les niveaux de difficultés en détails :

Difficultés	Nombre de vague	Type d'abeilles que le joueur peut utiliser	Type d'ennemis	Taille du terrain	Nombre de filets	Quantité de miel au début de la partie
Facile	3	Sniper	Chenilles	12x3	3	100
Moyen	4	Sniper Butineuse	Chenilles Frelons	12x5	5	150
Difficile	5	Sniper Butineuse Venimeuse	Chenille Frelons Araignée	12x7	7	200

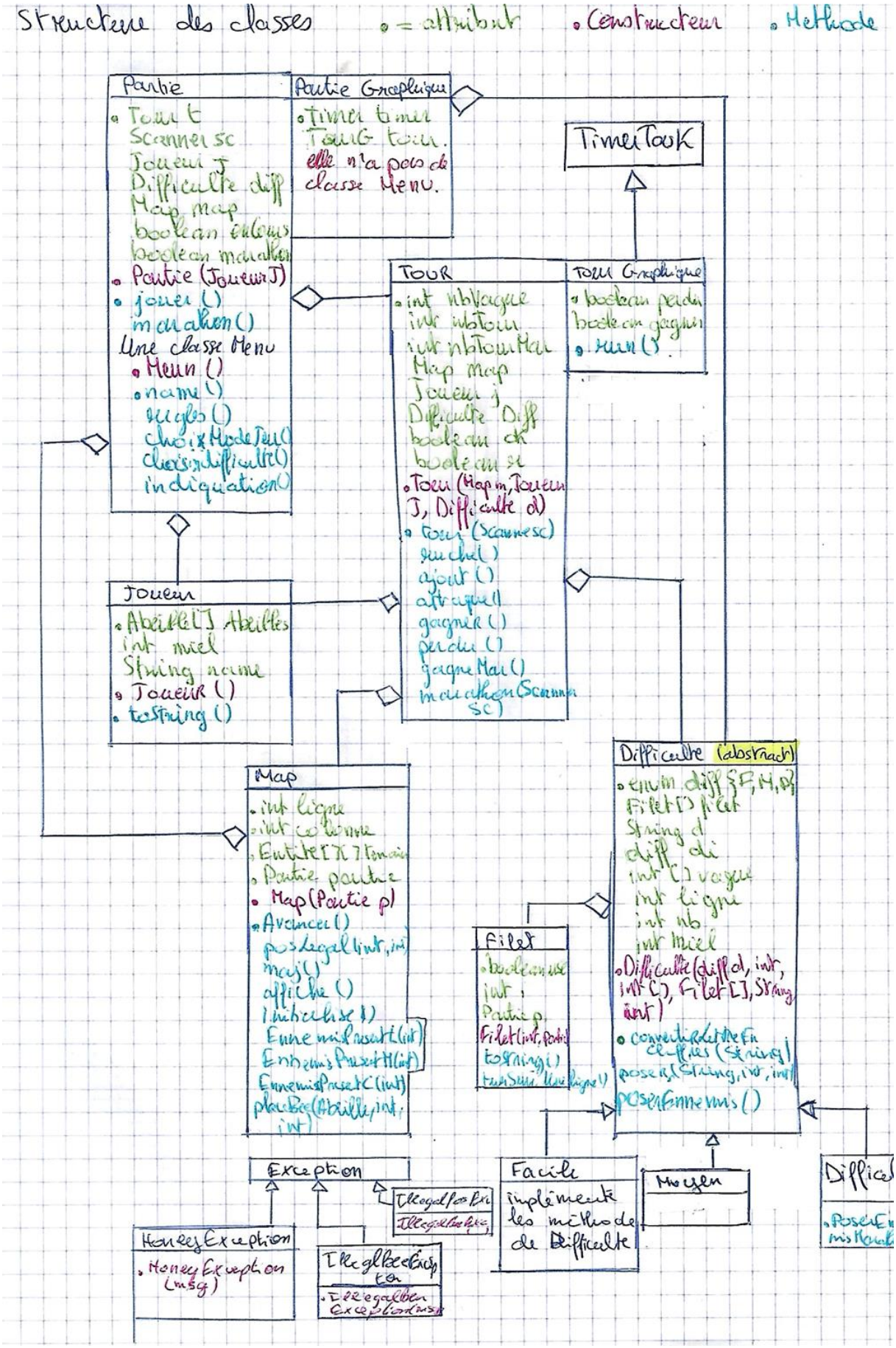
Dans le mode Marathon, la taille du terrain est le même que celle du mode difficile ainsi que le nombre de filets et toutes les abeilles et les ennemis sont présents dans la partie.

Mode Terminal

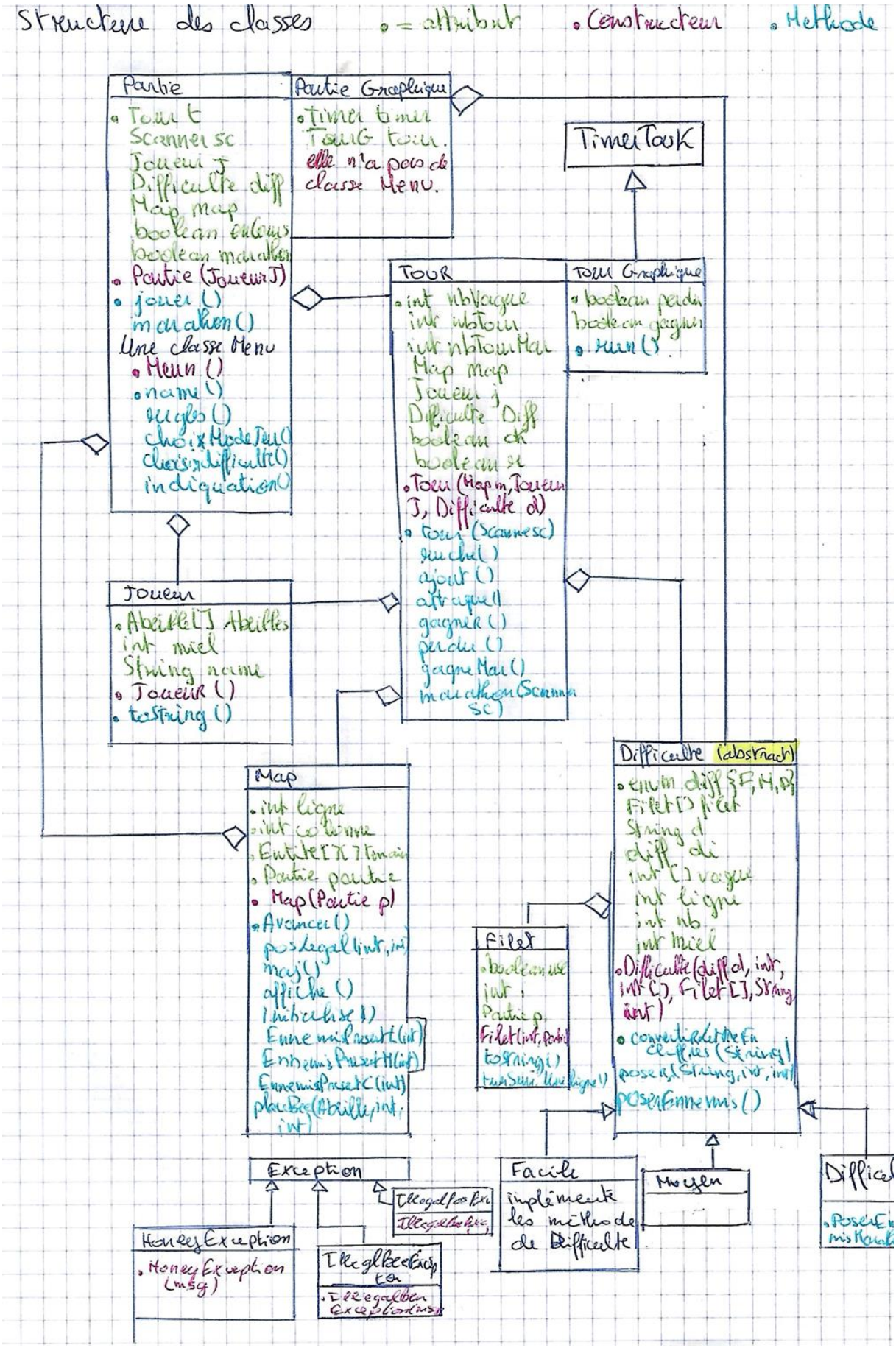
Pour le mode terminal nous avons séparé les entités (ennemis et abeilles) du déroulement de la partie (une partie et un tour le terrain etc…). De ce fait, Nous avons un package Perso contenant toutes les entités du jeu et un package …… qui contient le déroulement de la partie.

Voici la structure de notre jeu pour le mode Terminal :

Structure des classes ° = attribut ° Constructeur ° Methode



Structure des classes ° = attribut ° Constructeur ° Methode



Pour le mode graphique la stucture du code reste exactement la meme, sauf quelque modification sur les classes Tour et Partie.

Car dans la version terminale, Partie comme Tour avait besoin de Scanner pour recupere les entrées du joueur ce qui n'est plus le cas dans la version graphique.

Vue et Controller dans la version graphique :

Pour la vue, on a choisi d'utiliser un CardLayout pour faciliter la transition entre les differents Jpanel.

On a un Jpanel pour chaque vue :

Le plateau, le choix du mode de jeu, le choix de la difficulté, les cases du plateau,le menu principal, les filets…

Ainsi chaque vu s'accompagne de son Controller, qu'on initialise a chque fois qu'on crée la vue. Donc chaque vue a un controller associés.

Et chaque controller a en attribut un objet model, car on a besoin d'acceder au model a chaque etape, et mettre l'attribut Partie en static aurait été une erreur.

Package Perso :

La classe Entité est la classe mère d'Abeille et d'Ennemis. La classe Abeille est la classe mère de Butineuse, de Sniper et de Venimeuse tandis que la classe Ennemis est la classe mère de Chenille, de Frelons et de d'Araignée. Chaque entités a son attaque en plus de ses atouts.

Package Model :

Classe Difficultés et ses enfants

La difficulté et ses enfants (Facile, Moyen, Difficile) sont des classes abstraites qui permettent de paramétrer le jeu. En effet on créé le terrain de jeu, les filets, on informe le Joueur des types d'abeille dont il dispose ainsi que leurs prix.

Voici un tableau détaillé qui présentent les méthodes de la classe :

Méthode	Type d'Argument	Type de Retour	Rôle
convertirLettreEnChiffre	String	Int	Converti la coordonné de la colonne entrée par le joueur en un entier
Poser	Scanner	Void	Méthode abstraite
PoserEnnemis	X	Void	Méthode abstraire
PoserEnnemisMarathon	X	Void	Méthode abstraite

Les méthodes abstraites seront implémentées par les classes Enfant :

Méthode	Type d'Argument	Type de Retour	Rôle
Poser	Scanner	Void	Demande au joueur quelle abeille il

			souhaite poser et pose l'abeille dans le terrain
PoserEnnemis	X	Void	Pose des ennemis à chaque tour tant que les vagues ne sont pas terminés
Prix	X	Void	Affiche le prix des abeilles
validerTypeAbeille	String	Boolean	Teste si le joueur place une abeille qu'il peut placer

Classe Partie et Tour

La classe Partie permet de lancer une partie tant que le joueur n'a pas décidé de quitter la partie ou que la partie n'est pas terminé les tours s'enchaîne mais avant cela sa classe interne Menu affiche les règles du jeu et permet au joueur de paramétrer la difficulté et le mode de jeu.

Voici un tableau détaillé qui présentent les méthodes de la classe Partie :

Méthode	Type d'Argument	Type de Retour	Rôle
Jouer	X	Void	Permet de lancer une partie
Marathon	X	Void	Permet de lancer une partie d'un mode marathon

La classe Tour permet de faire des tours de jeu. Dans un tour de jeu on teste si le joueur a gagné ou perdu la partie pour arrêter la partie, puis selon le choix du joueur on augmente le nombre de miel si le joueur a voulu récolter du miel ou bien on pose une abeille ou bien on arrête la partie s'il ne veut plus jouer s'il continue de jouer on met à jour le terrain (on enlève les entités mortes, on avance les ennemies et on décrémente le point de vie des entités attaquées)

Voici un tableau détaillé qui présentent les méthodes de la classe Tour :

Méthode	Type d'Argument	Type de Retour	Rôle
tour	Scanner	Void	Permet de lancer un tour
ruche	X	Void	Augmente le nombre de miel
ajout	X	Void	Ajout des ennemis dans le terrain à chaque tour
gagner	X	Void	Test si le joueur a gagné pour arrêter la partie
perdu	X	Void	Test si le joueur a perdu pour arrêter la partie

gagnerMar	X	Void	Test si le joueur a gagner une partie d'un mode Marathon
Marathon	Scanner	Void	Lance un tour d'une partie d'un mode marathon

Classe Map

La classe Map permet de modéliser le terrain. De ce fait, elle gère les déplacements des ennemis, et les attaques des entités.

Méthode	Type d'Argument	Type de Retour	Rôle
rowLegal	Int	Boolean	Teste si la coordonné pour la ligne est correcte
columnLegal	Int	Boolean	Teste si la coordonné pour la colonne est correcte
posLegal	Int int	Boolean	Teste si la position est correcte
avancer	X	Void	Avance les ennemis ou attaquent les abeilles si c'est possible
Maj	X	Void	Met à jour la map en enlevant les entités mortes et en avançant les ennemis
Affiche	X	Void	Affiche la map
Initialise	Entité	Void	Initialise aléatoirement la position de l'entité passé en argument
EnnemisPresentL	Int	Ennemis	Teste si des ennemis sont présent sur la ligne donnée en argument
EnnemisPresentM	X	Boolean	Test si des ennemis sont présent dans la map
ennemisPresentC	Int	Entite	Vérifie si un ennemi est présent sur la première colonne et retourne l'ennemis
placeBee	Abeille int int	Void	Place les abeilles sur le terrain

Attaque	X	Void	Les abeilles attaquent les ennemis si elles peuvent les attaquer

Piste d'extension qu'on n'a pas encore implémentée

- Pouvoir voir
- Sauvegarder des parties
- Récompenser le joueur lui donner des badges s'il a réussi à tuer un certain nombre d'ennemis etc…
-