Projet Java III - 2022-2023
HELBAquarium

POPADIUC CLAUDIU

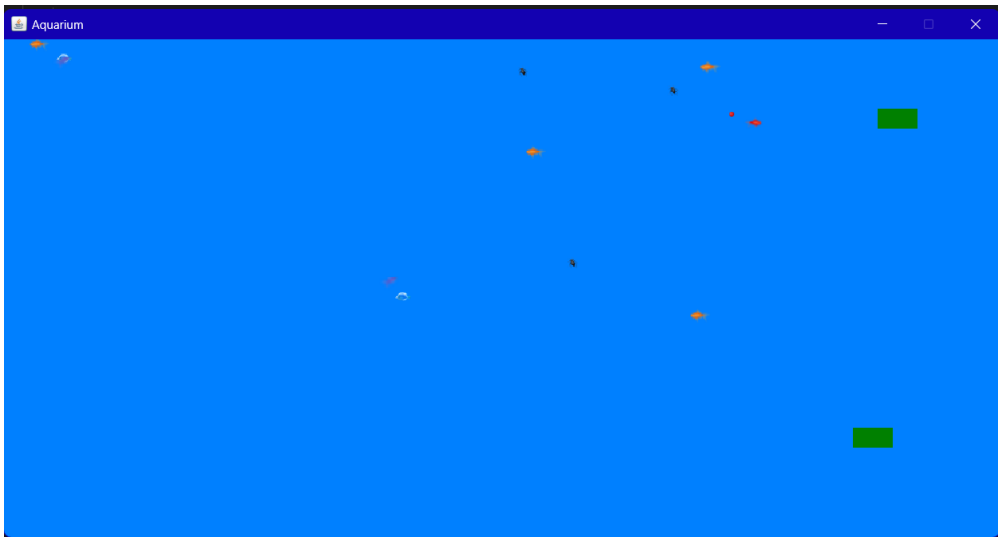
Monsieur Riggio, Jonathan

Table des matières

Introduction

Dans le cadre du cours de JAVA Q III, il nous a été demandé de réaliser un aquarium. Pour avoir un certain début, monsieur Riggio nous a proposés de partir du « Snake » comme base fonctionnelle, c'est ce que j'ai fait personnellement. Le projet a été réalisé en oriente objet, c'est une condition obligatoire.

Fonctionnalités de base



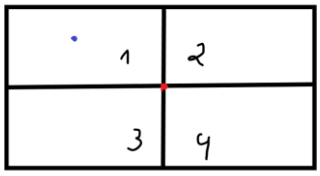
Les fonctionnalités suivantes sont fonctionnelles dans le projet :

Mouvements des 4 types de poissons :

Pour les déplacements de tous les poissons, j'ai créé une classe parent « Fish ». Tous les déplacements viennent de cette classe. Dedans Il y a une fonction « move », cette fonction permet de donner une Target position à tous les poissons.
Ainsi ils sauront tous ou aller tout le temps car chaque poisson aura une Target position à chaque instant.
Dans « Fish », je crée aussi une méthode « positionFish » qui fera en sorte d'ajouter tous les poissons dans des position aleatoire au lancement de la partie.
Chaque poisson peut se déplacer dans l'aquarium, soit verticalement, soit horizontalement, soit en diagonale grâce au système que je viens d'expliquer, leurs Target dépendra de leur comportement, et pour cela il y aura une classe pour chaque type de comportement, ils hériteront tous de la class « Fish ». Ils auront tous leurs positions aléatoires et leur déplacement vers une Target. Chaque poisson a aussi un ID car c'est nécessaire lors de leurs morts, le poisson rouge doit savoir quel poisson il devra tuer exactement, ceci sera mieux expliqueé lors de son comportement.
Toutes les variables qui sont utilisées plusieurs fois dans les comportements de poisson ont été ajouter une seule fois dans Fish, il y aura un get et parfois un set en fonction du besoin de la variable dans les autres classes, leur utilisation sera expliquée.

1. Les poissons orange se déplacent constamment aléatoirement jusqu'à rencontrer un bord de l'aquarium, à ce moment précis, ils prendront une nouvelle position aléatoire. Pour cela j'ai créé une class « FishOrange ».
   A l'intérieur j'ai ajouté une méthode « randomEdgePosition » qui donnera un bord aléatoire au poisson orange comme Target position.
   Dans cette méthode je crée un random de 1 à 4 car il y a 4 bords, et chaque random donne les coordonner d'un des bords mais tout à fait aléatoirement que ce soit le bord gauche au milieu, plus bas ou plus haut, etc…
   Une fois cette Target définie, le poisson orange ira jusqu'à cette Target grâce à « super.update », cette ligne de code est place dans d'autre méthodes mais pour comprendre pourquoi, il va falloir comprendre le fonctionnement des Pellets et des insectes, cela viendra plus tard.
   Il faut aussi savoir que la Target a une hitbox, car le poisson avance par plus d'un pixel en x et y (il avance en +6). Si ce poisson touche un obstacle il prendra aussi un nouveau bord random en vérifiant si le nom du poisson qui touche l'obstacle est celui de l'orange.

2. Les poissons mauves se déplacent toujours dans la direction la plus éloignée du poisson rouge le plus proche. Les poissons mauves sont également stimulés par les décorations de l'aquarium, ce qui a un effet sur leur vitesse de déplacement.
   Pour cela j'ai créé une class « FishPurple », dans cette class il y a une méthode « oppositeDirectionOfTheRedFish », cette méthode prendra en compte toute la liste des poissons, une condition vérifiera que si le poisson en cours, c'est un poisson rouge, et si c'est le cas, il calculera le poisson rouge le plus proche en utilisant ce calcul :
   (Distance = Math.sqrt(x*x + y*y)).
   Une fois que le poisson mauve aura calculé la distance de tous les poissons rouges, il prendra le plus proche, en comparant les distance, ensuite le poisson mauve ira dans la direction la plus éloignée du poisson rouge.
   Pour mieux expliquer ce qui va suivre je vous fais ce dessin :



   Admettons que le poisson rouge est au milieu, si le poisson mauve est dans la case n°1, il ira dans le coin supérieur gauche, si le mauve est dans la case n°3, il ira dans le coin inférieur gauche, etc…
   Ainsi le poisson mauve ira toujours dans la direction la plus éloigné du poisson rouge. C'est ce qui se passe dans la fin de la méthode à l'aide de plusieurs vérifications, la Target du poisson mauve sera toujours un des 4 coins de l'application.
   J'indique aussi que le poisson mauve a une vitesse de basse similaire aux autres poissons mais j'ajoute à sa vitesse, le nombre d'obstacles dans l'aquarium (6+n°Obstacle). Si ce poisson touche un obstacle il le contournera en faisant x--, c'est

vérifier en regardant si le nom du poisson qui a touché l'obstacle est le nom du poisson mauve, cela sera mieux expliqué lors des obstacles. Ce sera similaire pour le poisson bleu et rouge.

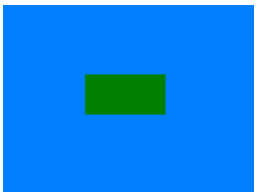
3. Les poissons bleus se déplacent toujours dans la direction du poisson mauve ou bleu le plus proche. Pour ce fait comme pour le poisson mauve, j'ai calculé le poisson le plus proche en utilisant le calcul de la distance, cette fois-ci, il ira chercher le poisson mauve ou bleu le plus proche et le donnera comme Target. Mais une des conditions en plus dans la méthode de direction du poisson bleu c'est qu'il ne peut pas se définir lui-même comme Target, cela arrivait et le code ne fonctionnait plus alors j'ai dû vérifier que dans la liste de poissons, le poisson en cours n'avait pas la même position que le poisson bleu pour lequel tout le calcul était fait.
   Ce poisson est aussi plus rapide que la vitesse de base, sa vitesse passe à 7 au lieu de 6.

4. Les poissons rouges se déplacent toujours dans la direction du poisson, non rouge, le plus proche. Quand un poisson rouge rencontre un poisson d'une autre couleur, celui-ci le mange, ce qui a pour effet de le faire disparaitre de l'aquarium.
   Les poissons rouges sont stimulés par la température de l'aquarium. Ce qui veut dire que si j'appuie sur la touche 1 ils iront moins vite, la touche 2 ce sera la vitesse de base et la touche 3 ils seront plus rapides que tous les poissons, cela sera mieux expliquer lors des touches du clavier.
   Comme pour le poisson mauve le poisson rouge fera le même calcul de distance pour trouver un poisson d'une autre couleur, il prendra le plus proche, une fois qu'il rentrera dans sa hitbox il le supprimera de la liste des poissons, cela est fait via la méthode « fishKilling », qui prendra en compte dans le Board, une méthode qui supprimera le poisson qui a été touché de la liste de poissons, cela en utilisant l'ID du poisson qui a été ajouté préalablement dans la classe « Fish ».

Déplacements et vitesse :

Sauf exception, les poissons n'arrêtent jamais leur déplacement dans l'aquarium.
Les poissons se déplacent avec une certaine vitesse de base. Cette vitesse de base je l'ai mise pour chaque poisson à 6.
Pour la vitesse de base, fait de mettre moins de 6 aurait causé un souci pour la contrainte que aucun bonus ne peut dépasser le double de la vitesse de base. Le mauve est stimulé par le nombre d'obstacles j'en ai alors mi qu'entre 1 et 3 afin d'ajouter 1,2,3 au poisson mauve en vitesse, il ira donc jusqu'à 9.
Avec les bonus d'insecte ça monte à 10 pour tous les poissons.
C'est pour cela que j'ai choisi 6 en vitesse de base, il fallait calculer le fait que le poisson mauve peut être stimulé et que le rouge doit être dans tous les cas le plus rapide des poissons dans le cas où la touche 3 est pressé, alors même si le mauve a 6+3 en vitesse, le rouge avec la touche 3 il passerait à 11 et il serait dans tous les dans le plus rapide même si un poisson venait à prendre un insecte.

Obstacle :



Pour les obstacles, j'ai créé une class « Obstacle » dans lequel chaque obstacle aura une position random dans l'aquarium, ceci grâce à la méthode « positionRandomObstacle » qui donnera des coordonnés à l'obstacle.
A l'intérieur de cette classe, il y a une méthode qui vérifiera si un poisson rentre en collision avec la hitbox de l'obstacle. Quand cela arrivera une variable prendra le nom du poisson qui a touché la hitbox.
Dans chaque poisson une vérification vérifiera si le nom du poisson est égal au nom du poisson qui a touché l'obstacle ensuite il contournera l'obstacle si le poisson en question n'est pas dans la hitbox de l'obstacle, le poisson ira simplement dans un else, qui lui donnera un comportement normal. Pour le poisson orange il changera juste directement de direction et ne le contournera pas. Normalement tout est là pour que ça fonctionne si un poisson le touche de haut ou de bas mais Malheureusement cela ne fonctionne pas, alors je n'ai pas été plus loin en vérifiant si le poisson venait de gauche ou droite. (Pour l'orange ça fonctionne mieux que pour les autres mais pas tout le temps).

Pastilles :

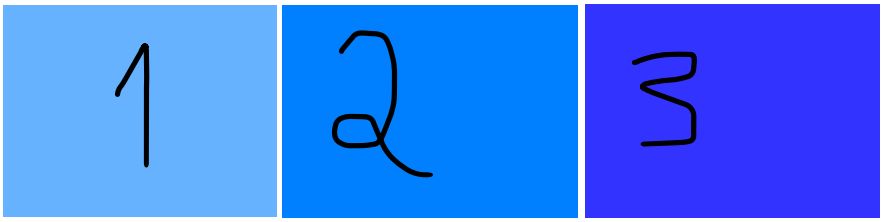
Au lancement de la simulation, l'aquarium peut contenir un certain nombre de pastilles. Ces éléments sont fixes et placés de façon aléatoire dans l'aquarium. Ceci est réalisé grâce à la class « EdiblePellet », comme pour les poissons les pellets prendront une position aléatoire grâce à la méthode « PositionRandomPellet », une fois place dans l'aquarium, les poissons pourront entrer dans la hitbox de la pastille, cela est vérifié dans la méthode qui prendra en compte la liste des poissons, quand c'est fait la pastille disparaitra, et une autre apparaitra ailleurs dans l'aquarium, car je rappelle la méthode « PositionRandomPellet ».
Je prends ensuite en compte, dans une variable la couleur du poisson qui a touché la pastille ainsi qu'un compteur de 10 secondes qui sera démarrée une fois une pastille touche.
Dans mon code je fais le calcul de 100 x le n° de poisson car j'ai remarqué lors de plusieurs tests qu'au plus il y avait de poissons au plus vite le compteur était décrémenter et cela ne faisait pas 10 secondes, mais grâce au n° de poisson cela vaudra les 10 secondes.
Dans chaque classe de poisson, une méthode « ifTheFishTouchThePellet » y sera et inséré, à l'intérieur il y aura une vérification qui vérifiera quel poisson a prit la pastille, comme dit précédemment, une variable prendra en compte le nom du poisson, ainsi si c'est l'orange alors l'orange continuera d'avancer et arrêtera les déplacements de tous les autres poissons, grâce au compteur qui décrémentera pour les autre poisson et ils ne rentreront plus dans la condition qui les fait bouger tant que le compteur n'est pas égal à 0, le compteur décrémentera de 1 toute les 60 millisecondes (Timmer), pendant ce temps-là les poissons ne feront plus rien.

Insectes :

Au lancement de la simulation, l'aquarium peut contenir un certain nombre d'insectes. Ces éléments sont fixes et placés de façon aléatoire dans l'aquarium. Ceci est réalisé grâce à la class « Insect », comme pour les poissons les insectes prendront une position aléatoire grâce à la méthode « PositionRandomInsect ».
Dans cette class j'ai ajouté une méthode avec 3 noms diffèrent, chaque insecte aura un nom diffèrent et cela est fait pour pouvoir savoir quel insecte des 3 a été touché, lorsqu'un poisson rentrera dans la hitbox d'un insecte, le nom du poisson sera enregistré dans une variable, la condition vérifiera aussi quel nom l'insecte porte, grâce à la liste de Fish qu'on traversera avec

un « for » et en fonction du nom, le temps de chaque insecte sera diffèrent mais donnera à tous le même bonus de vitesse qui sera 11.
Dans chaque classe poisson une méthode « ifTheFishTouchedTheInsect » sera ajoutée afin de vérifier si c'est ce poisson-là qui touche l'insecte grâce à la variable enregistrée si oui, il changera sa vitesse à 11 et fait avancer le poisson, un compteur commencera à décrémenter, lorsqu'il atteindra 0, le temps de l'insecte en question sera terminé et le poisson reprendra sa vitesse initiale et continuera d'avancer. C'est le même système que pour le pellet mais c'est plus simple car c'est le poisson qui a pris l'insecte qui aura un boost et pas les autres poissons.

Température de l'aquarium :



L'aquarium possède une température de base, qui par défaut est tiède. Pour ma part elle sera mise en un bleu normal (image 2), Il est possible de changer la température de l'aquarium sur trois configurations différentes : Froid, tiède et chaud. Pour les 3 changements il y aura du bleu mais diffèrent, bleu clair (image 1) pour le cold et bleu foncé (image 3) pour le hot.
Dans la configuration froide, les poissons rouges ont un malus de vitesse par rapport à leur vitesse de base, ce qui les rend plus lents que tous les autres types de poissons. Pour cela dans mes touches de clavier, je ferais un changement de vitesse du poisson rouge et du background. Il y a un setter qui changera la vitesse du poisson rouge de 6 à 10 pour la température chaude, et de 6 à 3 pour la température froide, pour la température tiède, cela reste à 6. D'autres explication seront donnée dans les touches 1,2 et 3.

Reproduction :

Quand deux poissons de la même couleur entrent en contact, ils peuvent s'accoupler. Les deux poissons qui sont entrés en contact disparaissent, et trois nouveaux poissons de la même couleur que les deux précédents sont placés, chacun, à une nouvelle position aléatoire dans l'aquarium.
Dans ma classe « Fish » j'ai ajouté une méthode « couplingFish » qui vérifiera si le poisson du for en question n'a pas la même position du poisson de la classe même, si ce n'est pas le cas il vérifiera si le poisson en cours entre dans en contact via une hitbox avec un autre poisson. Pour donner suite à cela il fera une dernière vérification si les 2 poissons sont de la même couleur. Si c'est le cas, il supprimera les 2 poissons et ajoutera 3 autres de la même couleur dans des positions aléatoires via les méthodes implémenter dans Board.
Vu que monsieur Riggio nous a demandés de faire en sorte de limiter la reproduction en fonction du nombre de poissons présents dans l'aquarium alors j'ai ajouté une variable qui prendra en compte la taille de la liste de poissons à chaque instant, avec cette variable j'en ajoute une autre qui fera un random du nombre de poissons, pour finir je vérifie que si ce random est égal à 0, ils se reproduise, sinon il ne font rien ainsi au plus il y aura de poissons au plus il y aura un grand nombre random, au moins il y aura de chance que ce soit égal à 0, donc moins de chance de se reproduire.

Commandes de l'aquarium :

Afin de permettre des interactions entre l'utilisateur et l'aquarium, il existe un système de commandes permettant d'influencer la simulation. Quand l'utilisateur appuie sur une touche, cela déclenche un évènement précis :

Ces touches ont été ajoutées dans la classe Board, avec une méthode « keyPressed », pour chaque touche une condition vérifiera si la touche a été pressée et il y aura ensuite à l'intérieur soit une fonction soit directement le code du comportement de la touche :

- 0 : Reset l'aquarium.

Une méthode fera en sorte de supprimer tous les éléments de l'aquarium, de mettre le compteur des pellets à 0 au cas où, avant que la touche soit pressée, un des poissons aurait pris un pellet et avait arrêté les autres poissons pour 10 secondes.
Le background reprendra sa couleur par défaut (tiède) et la méthode « initGame » sera appelée pour recommencer le jeu.
La vitesse du poisson rouge ira de nouveau à 6 au cas où elle aurait été modifier par les touches 1,2 ou 3.
Je redonne aussi des valeurs aléatoires pour les pellets, obstacles et insectes afin de ne pas avoir les mêmes valeurs qu'au démarrage de l'aquarium.
Je remets aussi le nom du poisson à « Default », qui avait été enregistrer si l'une des touches R, M, B, O avait été pressé, afin de reprendre leurs déplacements si la touche 0 est pressé.

- 1 : Définir la température de l'aquarium sur froid.

Diminue la vitesse du poisson rouge avec un set de vitesse qui est initialisé dans le poisson rouge, qui changera donc la vitesse du poisson rouge de 6 à 3, alors le poisson avancera par pas de 3 en x et en y au lieu de 6.
Il y aura aussi un changement du background en un bleu plus claire.

- 2 : Définir la température de l'aquarium sur tiède.

Similaire que pour la touche précédente mais le poisson rouge prend sa vitesse de base (6), avec la couleur par défaut du background

- 3 : Définir la température de l'aquarium sur chaud.

Similaire, sauf avec un background en un bleu plus foncé, et une vitesse supérieure que tous les poissons, de 6 à 11.
Chose importante à savoir, il y avait certains bugs s'il prenait un insecte alors il aurait changé sa vitesse de 11 à 10 (10 la vitesse bonus d'un insecte) et ne serait plus, plus rapide que tous les autres poissons, alors dans ma classe « FishRed », j'ai indiqué soigneusement que si le poisson avait une vitesse = à 11 alors il ne prendrait pas en compte la vitesse bonus d'insecte qui est seulement de 10 car dans tous les cas le poisson rouge serait le plus rapide en 11.

- 4 : Ajouter à une position aléatoire un insecte.

Cela est fait via la commande add, pour la liste d'insectes, dans le Board, comme au démarrage de l'application quand il y a un certain nombre par défaut de poisson et d'éléments, rien de bien compliqué.

- 5 : Ajouter à une position aléatoire une pastille comestible.

Similaire que pour la touche 4 mais pour des pastilles.

- 6 : Passer en mode insectivore. Dans ce mode, tous les poissons qui ne sont pas

Stoppés, se dirigent vers l'insecte le plus proche, s'il y en a un dans l'aquarium.
Une méthode est appelée qui ira chercher d'abord si le poisson rouge est en mouvement et ensuite pour chaque poisson rouge, pour tous les insectes, la méthode calculera l'insecte le plus proche une fois qu'il là trouvé, la Target du poisson sera la position de l'insecte le plus proche. Malheureusement il ne va jamais vers l'insecte, car je ne peux pas mettre en statique les setters Target position, j'ai essayé et cela créait d'autres bugs. Alors je ne suis pas allé plus loin mais j'aurais appliqué le même principe pour tous les poissons car je l'ai fait que pour le rouge.

- 7 : Passer en mode pastille. Dans ce mode, tous les poissons qui ne sont pas stoppés, se dirigent vers la pastille la plus proche, s'il y en a une dans l'aquarium.

Similaire que pour l'insecte, cela ne fonctionne pas aussi.

- 8 : Passer en mode reproduction. Dans ce mode, tous les poissons qui ne sont pas stoppés, cherchent à rencontrer, le poisson de la même couleur, le plus proche.

Similaire que pour l'insecte, cela ne fonctionne pas aussi.

- 9 : Ajoute un poisson, de type aléatoire, à une position aléatoire dans l'aquarium.

Une méthode qui fera un random de 1 à 4, pour choisir une couleur de poisson aléatoire, une fois que c'est fait il l'ajoutera simplement dans la liste des poissons, comme pour la touche 4 et 5.

- R : Cela permet de stopper tous les poissons sauf les rouges.

Un string prendra le nom du poisson rouge, et dans la classe du poisson rouge, lorsqu'il est censé aller vers sa Target position, il vérifiera d'abord la variable du nom de la touche R, si c'est « FishRed » ou « Default » il continuera d'avancer.
En ce qui concerne « Default », c'est la valeur par défaut du nom de poisson dont la touche est appuyée, elle ne changera pas tant qu'une des 4 touches n'ont pas été enclencher, il est important donc de faire avancer tous les poissons tant que la variable reste à « Default », sinon ils ne bougeront plus.
Pour les autres class de poisson ils auront aussi cette vérification et pour eux comme ce sera « FishRed » ils entreront dans la condition et ce sera une condition vide qui ne les fera rien faire, ils n'avanceront donc plus jamais.

- B : Cela permet de stopper tous les poissons sauf les bleus.

Similaire que pour la touche R.

- M : Cela permet de stopper tous les poissons sauf les mauves.

Similaire que pour la touche R.

- O : Cela permet de stopper tous les poissons sauf les oranges.

Similaire que pour la touche R.

Fonctionnalités supplémentaires

Additionnement, il nous a été demandé d'implémenter au minimum une fonctionnalité
Supplémentaire originale, j'ai ajoutée alors le « FishBlack ».

Le poisson noir : l'ennemie jurée des poissons rouges, son seul et unique but sera d'exterminer tous les poissons rouges, avant que celui-ci n'ait tué tous les autres poissons. Ainsi il pourra protéger tous les autres poissons de leur prédateur, pour y arriver il est le seul poisson sur qui il n'y aura pas d'effets si un des autres poissons prend un pellet, il pourra continuer de traquer les poissons rouges, plus de cela le poisson noir pourra toujours prendre les insectes, donc il pourra avoir les bonus de vitesse des insectes afin de rattraper tous les poissons rouges. Cerise sur le gâteau, si lui prend un pellet il arrêtera les déplacements de tous les poissons rouges mais pas ceux des autres couleurs.

Malheureusement la seule façon pour le poisson noir d'exister c'est en pressant la touche « A » sûr le clavier, et cela est à double tranchant si l'utilisateur appuie sur cette touche alors un autre poisson rouge apparaitra aussi, il y aura donc 1 poisson noir mais un 2ème poisson rouge dans l'aquarium ! Ainsi de suite si la touche est encore pressée.

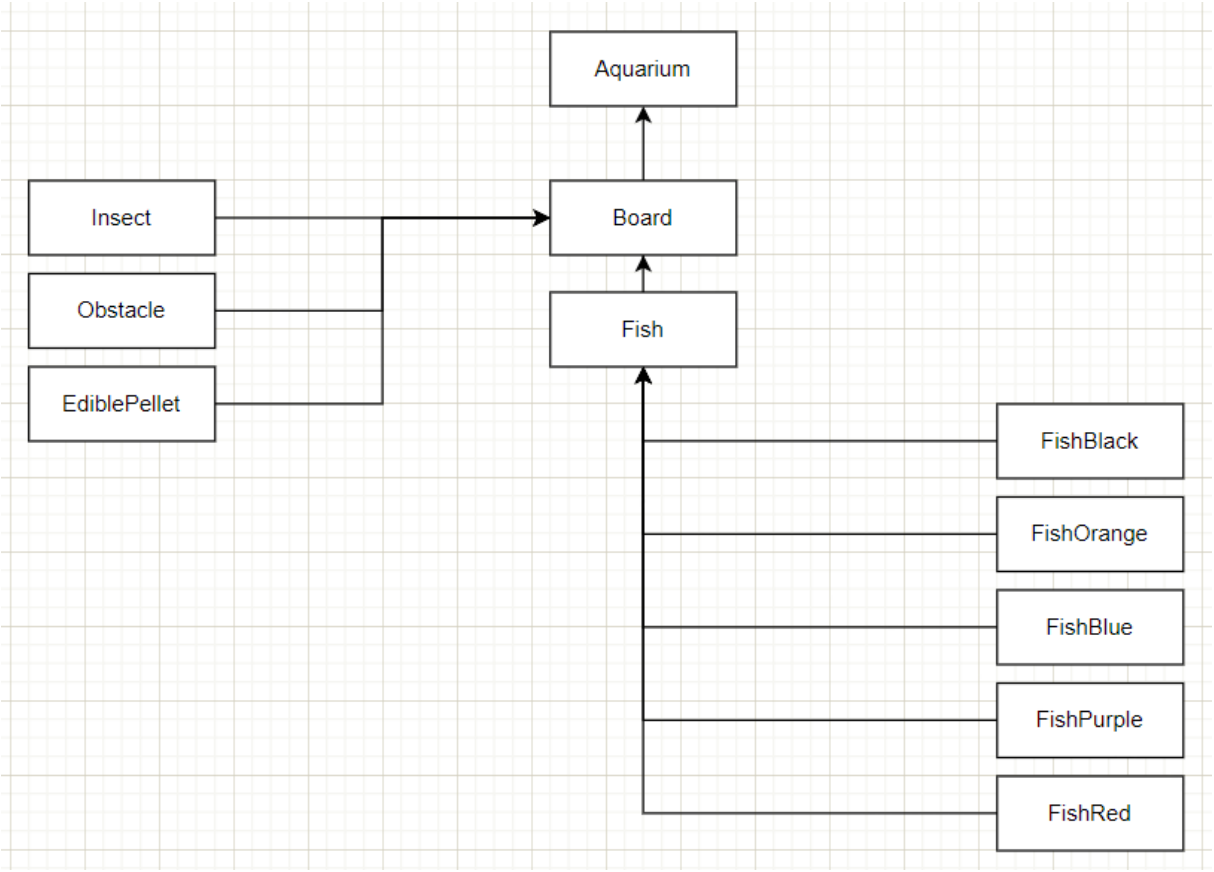
Pour faire tout cela, j'ai donné comme Target au poisson noir, le poisson rouge le plus proche, une fois qu'il atteindra sa hitbox, il le supprimera de la liste des poissons. Comme déjà expliquer dans le poisson rouge.

J'ai fait en sorte d'ajouter dans la classe du poisson rouge, dans les conditions d'arrêt si un poisson touche un pellet, il devra s'arrêter, il s'arrêtera aussi si c'est un poisson noir qui prend un pellet, cependant les autres poissons n'auront pas cette condition alors ils ne s'arrêteront pas.

Pour que le poisson noir ne s'arrête pas si un autre poisson prend un pellet, je n'ai juste pas ajouté le fait d'arrêter le poisson noir si un pellet a été pris par un autre poisson. Ceci se passe dans la classe même du poisson noir, du coup il n'y aura pas de méthode qui arête le poisson noir en cas de pellet pris. Pour ce qui est des insectes, ce sera pareil que pour tous les autres poissons, il n'y a pas de changements là-dessus. Pour ce qui est de la touche pressée, c'est pareil que pour la touche 9, ajout de poisson dans la liste mais là ce sera précisément un poisson noir et un poisson rouge.

Le poisson noir est aussi censé contourner un obstacle, dès qu'il touche sa hitbox mais malheureusement comme pour les autres poissons l'obstacle ne fonctionne pas.

Analyse



Aquarium est la classe qui démarrera l'aquarium, c'est là qu'il y aura le main. Le Board quant à lui, c'est là qu'énormément de chose ce passera comme les ajouts des poissons dans leurs listes, leur image etc... Et Board sera utilisé lors du démarrage de l'aquarium.

Les class Insect, Obstacle ainsi que EdiblePellet seront utilisé dans Board pour ajouter des éléments dans leur liste respective, tout viendra de ces 3 classes-là, chaque élément aura une position aléatoire différente des autres via ces 3 classes, chacune des class aura son propre comportement, qui est demandé pour la réalisation du projet.

Il y a aussi la class Fish, ou chaque poisson y trouvera sa position aléatoire avec leurs déplacements et leurs fonctions réalisées lors de ce projet.

Enfin chaque couleur de poisson héritera des positions aléatoires de la classe Fish, ainsi que les mouvements avec Target, et chaque type de poisson aura son propre comportement dans sa classe respective, la plupart des variables similaires dans les comportements de poisson ont été ajouté une fois dans « Fish » afin de les réutiliser dans les autres classes.

Limitations

Pour la plupart des limitations ici, je les ai testés et vérifiés moi-même.

Nombre d'éléments dans l'aquarium :

J'ai choisi de mettre un nombre assez faible d'obstacle, de pellet ainsi que d'insecte.

Ce choix est stratégique car il y aura des problèmes à mettre 100 pellets, cela donnera trop vite la possibilité au poisson de prendre un pellet ce qui arrêtera toujours les déplacements des autres poissons. Similaires que pour les insectes, leur vitesse sera toujours augmentée, plus il y aura de pellets et d'insectes plus l'aquarium ne ressemblera plus à rien, déjà à partir de 20, 30 ça commence, même si en termes de fait ce n'est pas problématique si les poissons sont plus rapides mais ce sera trop fréquent, le plus gros souci sera pour les pellets, plus aucun poisson ne bougera appart celui qui là pris et on ne verra aucun comportement appart d'un seul poisson.

Pour ce qui est d'obstacle si on en met déjà 50 on ne verra presque plus l'aquarium car il sera submergé par les obstacles. Plus de cela on ne verra plus les poissons leurs déplacements seront étranges ils ne sauront plus où aller, car ils sont censés les contourner ou pour l'orange prendre un autre bord. Ce problème sera amplement plus grand avec un plus grand nombre d'obstacles.

On peut parler aussi du nombre de poisson, si on en mettait 1000, de couleurs au hasard, les poissons rouges tueraient très vite tous les autres poissons (testé et approuvés : moins de 1 seconde). Cependant vu le nombre de poissons même s'il y a peu de pellet, il y en aura toujours 1 qui touchera un pellet et si ce n'est pas un rouge alors les poissons tueront peut-être 800 poissons en 1 milliseconde (sans exagérer vu qu'il y a au moins 200 poissons rouges), du coup il y aura le même problème que s'il y avait 1000 pellets, in ne verrait que le comportement d'un seul poisson.

Vitesse du poisson Mauve :

La vitesse du poisson mauve augmente en fonction du nombre d'obstacles présents dans l'aquarium.

Pour cela j'ai fait en sorte que ce soit un random entre 1 et 3 obstacles mais si on en met déjà un de plus, la vitesse du poisson mauve de base qui est de 6, montrera à 10, et c'est la vitesse qui est augmentée lorsqu'un poisson prend un insecte alors celui-ci n'aura plus d'effet sur le poisson mauve, s'il y a 10 obstacles, la vitesse du poisson mauve augmentera considérablement et dépassera le double de la vitesse de base à partir du 6^{e} obstacle et compromettra les restrictions du projet, qui est qu'aucun poisson ne peut dépasser le double de la vitesse de base. Si on donne la possibilité d'ajouter 1000 obstacles, le poisson mauve se déplacera tellement vite qu'il sera impossible de le voir bouger.

Reproduction :

La reproduction fonctionne mais à partir du moment où ils sont beaucoup plus nombreux dans l'aquarium, suite aux reproductions, ils ont évidemment beaucoup plus de chance de se reproduire, et à partir d'un certain moment l'aquarium crash tellement il y a de reproduction. C'est un fait que j'ai pu remarquer.

Difficulté rencontrée

Dans cette section je vais parler des difficultés que j'ai pu rencontrer lors du projet afin de mieux vous expliquer certains de mes choix.
Pour commencer je pense qu'il y a de la duplication de code dans mon projet, et s'il s'avère que cela est vrai pour vous, alors sachez bien que j'ai tout essayé pour modifier cela, d'améliorer, de supprimer ou d'ajouter, mais tout le temps un problème faisait surface.
Par exemple les poissons, obstacles, insectes et pastilles apparaissent dans une position aléatoire dans l'aquarium alors j'ai voulu créer une class « GameElement » qui fera une fois la fonction qui donne des positions random, mais cela n'a pas fonctionné du tout dans mon code car il mettait tout le temps tout en position (0, Max Height) sans aucune raison valable ou encore, pour les obstacles quoi que je changeais il y en avait toujours 1 seul qui apparaissait dans l'aquarium même si j'en ajoutais plusieurs dans la liste. Plusieurs soucis de ce genre-là ont fait que j'ai pris la décision de faire fonctionner tout cela même si je devais répéter la méthode de position aléatoire, sinon le projet aurait ressemblé à rien et je n'aurais jamais pu continuer ou faire tout ce que j'ai fait.
Il y a aussi le code qui permet au poisson d'aller plus vite s'ils prennent un insecte, le comportement du pellet, le comportement des obstacles ou encore les lignes de code qui permet à un poisson de choisir le plus proche poisson, sont répété dans les poissons, mais si j'ai choisi de le faire ainsi, c'est car je trouvais cela juste impossible de mettre tout ça dans la classe mère « Fish » et simplement tout appeler avec un super car en effet il y avait chaque comportement de poisson à prendre en compte ce qui faisait énormément de petit changement, surtout pour les petits détails qui faisait énormément la différence. J'ai quand même essayé d'arranger et j'ai fait tout ce que j'ai pu à l'aide des getters et des setters. Si j'avais eu plus de temps j'aurais surement arrangé ces soucis et aurais fini le projet à 100%.

Les fonctionnalités que je n'ai pas réussi à régler sont : les obstacles et les touches 6,7,8.
En ce qui concerne les obstacles normalement tout est là pour que ça fonctionne, le code a l'air bon, je ne vois pas mon erreur, mais malheureusement les poissons traversent quand même l'obstacle la plupart du temps. Pour l'orange ça fonctionne mais il prend un comportement anormal en allant un peu dans tous les sens pendant 1 seconde.
Pour les 3 touches, comme expliqué précédemment le problème venait du fait que je ne pouvais pas mettre les Target des poissons en statique, si ça j'avais pu le faire, les poissons qui sont en mouvement auraient pris comme Target le pellet, insectes, poisson le plus proche.

Conclusion

Je ne mentirais pas si je disais que j'ai passé plus d'une centaine d'heures sur ce projet, et je peux dire haut et fort que je ne regrette rien, j'ai tellement appris en faisant ce projet. J'en suis absolument fière, je pense que c'est la chose la plus incroyable intellectuellement que j'ai faite de ma vie ! Alors un grand MERCI monsieur Riggio pour m'avoir donné la possibilité de me surpasser lors de ce projet. J'ai pu m'améliorer énormément en tout ce qui est programmation, java, orienter objet, etc…

Au début c'était très difficile, rien que d'y penser j'en avais peur, j'ai beaucoup galéré, mais à l'aide de votre cours de ce que j'ai appris l'année passée, j'ai pu réussir à surmonter les moments difficiles et passer plusieurs journées entières sur ce projet pour finalement réussir (Même si à mon avis c'est réussi à 90%).

Il y avait des moments où je me disais : « bon c'est que 3 crédits », « laisse tomber », je pensais très sincèrement ne jamais y arriver, j'étais assez pessimiste, je ne voyais pas de progrès je n'arrivais pas à avancer, je me souviens lors d'un cours en JAVA, vous aviez posé comme question dans votre questionnaire sur l'avancement du projet, j'ai commencé à dire que c'était en d'autres mots que c'était impossible pour moi car je n'avais pas assez de compétence pour y arriver que ce n'était pas le temps le problème. Et devinez quoi, j'en suis là maintenant, j'écris ces mots en sachant que j'ai pratiquement fait fonctionner tout ce qui a été demandé lors de ce projet.

Finalement ceci a pu être réalisé, car j'étais seul, et même si je n'étais pas du tout content de cette manière de fonctionner, je me dis que je n'aurais jamais fait tout ça si jetais avec quelqu'un. Je ne me serais jamais surpassé à ce point-là, alors je voudrais vous dire que vous prenez une excellente décision en nous laissant tout seul pour vos projets !

Cependant une chose est sûre même si je ne regrette rien, j'ai dû faire certains sacrifices en termes de choix de travail que ce soit pour l'école ou la vie réelle, et ce dû au nombre de choses que vous demandiez à faire pour le projet.

Mon seul conseil serait peut-être de diminuer la charge du projet, peut-être mettre moins de chose à faire, moins de condition, restrictions etc… car ces petits détails prennent énormément de temps je sais que pour certains, cela ne pose pas de problème et ils peuvent faire ce projet en y travaillant quelques heures par jour en 1 semaine mais pour d'autres comme moi, cela m'a fait faire certains sacrifices. Peut-être que d'autres ne feront pas ces sacrifices et choisiront de rater le projet juste parce que c'est un peu trop de travail, j'espère que vous avez pu comprendre mon avis sur la longueur du projet.

Je pense très certainement garder ce projet tout ma vie, un jour je relierais ce rapport et je me souviendrais où tout a commencé.