Année 2022 - 2023 Semestre 3
TE38XI

Compétences informatique, spécialisation projet

Rapport de projet
Mi-semestre

Projet réalisé par :
Lucas Triozon
Adrian Guilhem
Elisa Cezard
Khalil Makhloufi

Projet encadré par :
Jérôme Pasquet
Stathis Delivorias

Introduction

Le projet à rendre en ce semestre 3 pour l'UE TE38XI est une machine SMA, simulation multi-agent. Le sujet est libre du moment où sont réunies deux conditions : faire une simulation avec une interface graphique et que nos agents soient des Threads indépendants.

C'est un travail de groupe où chacun a réalisé des tâches parmi une liste établie et mise à jour en fonction de notre avancée.

Après concertation, nous nous sommes orientés vers la simulation d'une plage, et plus précisément des actions et événements entre les individus qui se déroulent sur une plage.

Le projet fonctionnera donc avec deux composantes essentielles : une plage et des personnes. Chaque individu a comme caractéristiques son âge, sa vitesse, sa capacité à sauver des gens, sa probabilité à se noyer. La plage a comme caractéristiques sa longueur, sa largeur, la taille de la mer, la température et la vitesse du vent.

En plus des caractéristiques, chaque classe possède des attributs plus particuliers pour faire fonctionner la simulation, pour lesquels on reviendra dessus dans l'approche.

Les personnes vont arriver progressivement sur la plage, placer leurs affaires où elles le souhaitent, puis faire des aller et retours dans la mer pour se baigner, puis ultimement partir. Elles pourront se noyer, être sauvées et acheter des choses auprès d'un vendeur itinérant.

Avant la simulation, l'utilisateur aura la possibilité de choisir ses paramètres, pour avoir une simulation sur mesure.

Tout sera affiché dans une interface graphique faite par nos soins, avec des points qui se déplacent, représentant les individus sur la plage.

Ce qui suit est le détail de notre approche dans la programmation et les rapports individuels.

Sommaire

Approche

Introduction

Notre projet est réalisé dans le langage Java, conçu comme étant intégralement orienté objet.

Au fur et à mesure de nos avancées dans le code, s'est dégagé pour notre simulation plusieurs axes, formant des familles de classes, qui interagissent entre eux et avec les autres.

Bien que parti à la base avec seulement une Plage et des Personne, nous arrivons sur un programme qui cumule 15 classes différentes, chacun ayant leur rôle dans la réussite de la simulation.

- Les composants d'interfaces : InteractionMenu, Interface, BarreVitesse
- La nature : Plage
- Les agents : Personne, Sauveteur, Vendeur
- Les vecteurs de déplacement : Vector, VecteurVertical, VecteurOblique
- Des énumérateurs pour faciliter les actions des agents : Etat et Objectif
- D'autres classes pour aider le code : Coeff, Rectangle, Coordonnees, Meteo
- Et enfin la classe Main, permettant de lancer le programme

La difficulté se trouve dans l'organisation de toutes ces classes, la gestion de leurs interdépendances et la garantie d'indépendance des agents dans leurs actions.

Ce qui suit détaille le fonctionnement profond de la simulation, en partant des composants pour aller vers les événements qui peuvent se produire.

I - Les composantes majeures

1) Main.java

La base du projet est le fichier *`Main.java`*. Dedans sont instanciés les interfaces et la Plage, en fonction des valeurs entrées par l'utilisateur dans l'interface de début, puis démarre une boucle infinie qui fait le tour de la plage (`plage.turn()`) puis rafraîchit l'affichage pour répercuter les changements.

Ce "tour de garde" en quelque sorte vérifie l'état de chaque personne une par une, effectue les modifications internes nécessaires en fonction des actions effectuées, et donne des indications aux personnes.

C'est le fichier qu'il faut lancer pour démarrer la simulation.

2) Rectangle, Coeff, Meteo, Coordonnées

Ces quatres classes sont de petits objets utilisés à différents moments du programme, avec des rôles minimes mais nécessaires au fonctionnement correct de notre programme. Ils seront évoqués à plusieurs reprises dans la suite du rapport.

Un objet Rectangle représente un rectangle géométrique, avec 4 sommets de coordonnées (x,y), la zone de la plage dans laquelle il est placé et son centre. Il peut être comparé à d'autres instances de Rectangle, et sert à placer les gens sur la plage et créer le poste de sauvetage. On peut savoir si un Rectangle croise un autre Rectangle avec la méthode isIn.

Coeff est une classe qui n'est jamais instanciée en tant que tel. Elle possède un attribut de classe : coeff, qui correspond à la vitesse de la simulation en cours. Le fait que ce soit un attribut de classe permet de diffuser ce coefficient dans toutes les classes sans avoir à créer d'objet et à les transmettre dans les constructeurs des autres classes. Sa valeur est modifiée par l'utilisateur lors de la simulation grâce à la barre de vitesse.

La Meteo est une variable choisie à l'initialisation de la plage, tout comme le vent. Les deux vont agir à la fois sur les personnes et sur les vagues. Suivant leurs valeurs, ils affectent la vitesse des personnes et des vagues ainsi que la hauteur et la force des vagues.

Les Coordonnees sont un couple de double (x,y), représentant un point, que l'on peut comparer à d'autres en termes de hauteur x. Toutes les positions utilisent cet objet.

3) La classe Plage

La `Plage` fonctionne principalement autour de son constructeur et de sa méthode `turn()`. Elle a en sa possession toutes les variables de la simulation et la liste des Threads. Au vue de ce qu'elle fait, on pourrait la considérer comme étant Dieu, mais puisque nous en avons le contrôle par notre code, ce n'est qu'un prophète qui guide les individus. Pour des raisons de conventions et de bon sens, on parlera plutôt de la nature de la simulation. Elle a la main sur à peu près tout, de la

vie à la “*mort*” des individus. Ici le décès correspond à l’arrêt des threads, quand un individu part de la plage.

Ses attributs sont la `longueur` et `largeur` de la plage, la largeur de la `mer`, la vitesse du vent, la liste des `threads`, le découpage de la plage avec 3 `zones` distinctes, les ids des `sauveteurs`, la `meteo` sur la plage, la position du `poste` de sauvetage et tous les `placements` des personnes. Si jamais un Vendeur se trouve sur la Plage, l’adresse est stockée, et on a toutes les informations sur les vagues : la liste des vagues, les attributs et leur coefficient.

Notre simulation fonctionne sur un repère orthonormé, avec deux axes x et y. Ils sont tous les deux bornés par 0 et respectivement la longueur et la largeur choisie pour la plage. Contrairement à la mi-semestre, la Plage ne possède pas le résumé direct de toutes les positions de chaque agent. Les coordonnées sont des réels, où 1 = 1m.

Elle s’instancie en donnant tous les paramètres de la simulation souhaités : longueur, largeur, nombre maximal de personnes, la météo, etc…

Lors de sa création la `Plage` va placer le poste de sauvetage en son centre, en créant un Rectangle qui y correspond, et déterminer les zones.

Elle va aussi générer tous les threads des Personnes, du Vendeur (s’il est présent) et des Sauveteurs qui vivront sur cette plage. Pour ne pas créer le chaos, même s’ils sont créés en même temps ils ne démarreront leurs actions qu’après un timing déterminé par un nombre de millisecondes multiplié par le numéro du thread.

Lors de l’appel de la méthode `turn()`, la plage va faire un tour de ce que sont en train de faire les threads. Elle va donc observer un à un les instances de `Personne` et en fonction de l’état qui est déclaré, la plage va faire des choses différentes :

- si un chemin est demandé : la Plage va trouver le meilleur chemin pour aller vers son objectif, en évitant les collisions avec les affaires posées sur le sable.
- en cas d’arrivée : donne un emplacement à la Personne, où elle devra se rendre pour se placer
- en cas de départ : supprime l’emplacement de la Personne
- en cas de placement : elle va confirmer le placement de la Personne et l’enregistrer pour que cela puisse être affiché sur l’interface
- si la Personne n’est plus sur la plage : interrompt le thread
- en cas de danger : appelle un Sauveteur
- si elle arrive/quitte l’eau : modifie la vitesse de ses vecteurs

4) La classe Personne

Les `Personne` sont des individus qui font leurs actions sur la plage. Chacune possède un `id` qui correspond à son placement dans la liste de tous les Threads qui sont lancés dans la classe `Plage`. Chaque instance a un comportement indépendant, prenant ses propres décisions et agissant dans son coin.

Bien qu'instanciés, les individus ne sont pas actifs (ou en vie) tout de suite, ils ne le deviendront qu'après un timing donné dans son constructeur. L'attribut `alive` stocke cette information. Quand `alive` vaut `false`, la `Personne` n'est pas considérée dans le tour de la `Plage` ni de l'interface. Le thread est vivant, l'objet est instancié, mais est ignoré puisque pas encore arrivée officiellement sur la plage.

Les `Personne` sont régies par deux attributs primordiaux : leur `etat` et leur `objectif`. Ce sont tous deux des Enum, qui permettent de déterminer ce que fait chaque individu, et les faire agir en fonction. Vous pouvez retrouver un détail de tous les Etat et Objectif ici. A la fin du mouvement de l'individu, son etat va changer en fonction de son objectif.

L'Etat qui sera le plus utilisé est le `MOUVEMENT`. Il assure le déplacement des personnes, vers une position objectif (`objPosition`), propre à chacune et qui évolue. Pour cela, elle va suivre une série de vecteurs qui lui sont propres, et qui ont été calculés au préalable. Une fois arrivé, l'individu change son état et effectue des actions en lien avec son objectif.

La série de vecteur est déterminée par un pathfinding, signé par l'Etat PATH.

La position de chaque individu est stockée dans l'attribut position. C'est cet attribut qui est utilisé par l'interface pour placer les threads sur la plage.

Si jamais la Personne est dans un état qui nécessite une action ou une validation de la Plage, un "drapeau" est mis en place pour s'assurer que la nature a pu confirmer les actions. C'est l'attribut `oath`.

En plus de ça elle possède des attributs pour la caractériser : son age, sa vitesse, sa vitesseNage, son énergie (stamina), son emplacement sur la plage (positionPlage), ainsi que des booléens pour savoir si l'individu est placé et dans l'eau.

Lors de son instanciation, on doit donner à la Personne son id, sa position de départ, et le timing à partir duquel il commencera à faire des actions. De manière automatique et aléatoire, elle va déterminer son âge, sa vitesse et son niveau d'énergie.

L'énergie sert à savoir si la personne va se noyer et si elle a besoin d'aide. A chaque mouvement, elle en perd, et au repos elle en reprend. Les collisions avec les vagues font aussi perdre de l'énergie.

5) Tableau des états

Etat	Application
ARRIVEE	Signifie que la Personne vient juste d'apparaitre sur la Plage. Elle attend l'attribution de son emplacement.
PLACEMENT	La Personne vient d'arriver à l'emplacement indiqué. Elle souhaite confirmer qu'elle s'y installe.
REPOS	Après s'être baigné, la Personne se repose pendant un laps de temps.
ATTENTE	Après s'être placé ou reposé, elle va décider ce qu'elle fait : aller à l'eau ou partir
BAIGNADE	La Personne est dans l'eau, en train de nager. Au bout de 10 minutes ou si son énergie devient trop faible, elle retourne se reposer.
AUSECOURS	La Personne est en difficulté, elle appelle un Sauveteur.
NOYADE	Une fois qu'un Sauveteur est appelé, la Personne ne bouge plus et perd de l'énergie.
DEPART	La Personne remballe ses affaires et se dirige vers la sortie.
PARTI	La Personne n'est plus sur la Plage.
SAUVETAGE	Un Sauveteur est en train de sauver quelqu'un.
PATH	Demande son chemin à la Plage.
MOUVEMENT	La Personne se déplace vers sa position objectif. Se déclenche si un vecteur a été assigné.
COMMERCE	Le Vendeur est sur le moment de vendre l'un de ses produits à l'une des personnes sur la plage.
ACHETER	Après s'être déplacé vers le vendeur, la personne achète des choses au vendeur.
ENFILE	La Personne attend dans la file pour acheter auprès du Vendeur

II Comportement des Personnes

1) Placement sur la plage

Lorsqu'une Personne est instanciée, son état est par défaut à ARRIVEE. Cela permet à la Plage d'attribuer un emplacement aux individus dès qu'ils passent au statut "en vie". Cet emplacement représente un rectangle de 1.2m de longueur pour 2m de largeur, qui sont les affaires d'un individu, et l'endroit où il ira pour se reposer.

L'attribution de l'emplacement se fait par la Plage en analysant la place qu'il reste sur le sable. La Plage est découpée en 3 zones de taille égales. Elle essaiera d'abord de placer les personnes dans la zone la plus proche de la mer. Elle tire au hasard des coordonnées dans cette zone et regarde si quelqu'un est déjà installé. Si l'emplacement n'est pas libre, elle tire à nouveau des coordonnées maximum 15 fois avant de changer de zone. Si l'emplacement est correct, la personne stocke son emplacement, qui est un objet Rectangle, et s'y dirige. C'est une stratégie premier vu premier servi.

La Plage va aussi garder le Rectangle en mémoire, afin d'éviter les collisions dans son attribut placements.

Une fois arrivé, l'état de la personne va passer à PLACEMENT, et va passer son attribut placed à true. A partir de ce moment là, les affaires vont apparaitre sur l'interface, à l'aide d'un rectangle noir.

2) Déplacement sur la Plage : Vector et pathfinding

Pour pouvoir se déplacer sur la plage correctement, et en évitant les collisions on effectue un pathfinding ou "recherche de chemin". C'est là que les vecteurs interviennent.

Nous avons créé une classe abstraite Vector. Cette classe modélise un vecteur, d'un point A à un point B. Les coordonnées de A sont (x,y), et les coordonnées de B sont (objX, objY). L'objectif est de faire glisser le point A vers le point B sur une droite, à une vitesse constante.

A est la position de base de la Personne, B son objectif et vitesse sa vitesse de déplacement sur terre ou de nage si elle est dans l'eau.

Vector est abstraite car nous devons distinguer deux types de droites différentes, et donc créer deux nouvelles classes qui en héritent : les droites affines et les droites verticales.

Les droites affines sont d'équation y=mx+p. L'objectif est que la Personne, lors de son mouvement, se déplace de A vers B en suivant la droite, en suivant parfaitement la vitesse donnée. On trouve x avec la formule sqrt(vitesse^2/m^2+1), puis on trouve y avec notre fonction. Cette formule a été trouvée à la main en

cherchant une solution à ce problème. La vitesse ici est multipliée par le coefficient de vitesse de la simulation. Cela permet d'avancer à la vitesse que l'on veut, en ajustant parfaitement x et y.

Pour les droites verticales, c'est plus simple. Comme l'équation de la droite est x = a, a une constante, il suffit d'ajouter à y la vitesse de notre vecteur.

Dans les deux cas, il ne faut pas oublier le sens dans lequel on va.

Maintenant que l'on a introduit nos vecteurs, il faut entrer dans le détail du pathfinding. L'objectif est d'éviter les affaires posées sur le sable et trouver le chemin le plus optimal.

Déjà, si les Personnes ne peuvent pas déterminer eux même leur position objectif, la Plage le fait pour eux, par exemple pour aller se baigner, ils ne connaissent pas la limite entre le sable et l'eau.

De base, la meilleure solution est de faire une diagonale vers notre objectif. On crée un premier vecteur qui tire "tout droit" vers notre objectif. Ensuite, on va regarder toutes les affaires qui se trouvent entre la position de base et l'objectif. On regarde s' il y a un croisement, et si oui, où ?.

S'il y en a un, elle va ajouter le point de croisement dans une liste, ainsi que les coordonnées du lieu où il faut aller pour le contourner. Ensuite est créé un nouveau vecteur partant du point de contournement vers la position objectif, et on refait jusqu'à ne plus avoir d'emplacements à vérifier.

Cela nous donne une suite de points à suivre pour arriver à bon port, et on crée notre suite de vecteurs à partir de celle-ci.

3) Baignade / repos

Une fois placée, la `Personne` a pour objectif d'aller dans l'eau. Tous les individus vont forcément au moins une fois dans l'eau. Pour s'y rendre, les `Personne` reçoivent de la part de la `Plage` les informations sur où se trouve la mer (en termes de coordonnées), et choisissent un endroit aléatoire dans l'eau. Automatiquement, les mouvements s'enclenchent jusqu'à l'arrivée aux coordonnées objectif. L'état de la `Personne` passe à l'état `BAIGNADE`, et le thread dort. Au réveil, elle se dirige vers son emplacement, passe à `REPOS` à son arrivée et endort le thread à nouveau. Une fois réveillé, la `Plage` décide si elle doit retourner se baigner ou non, par un tirage aléatoire. C'est quelque chose que nous changerons la semaine prochaine.

Une fois dans l'eau, la vitesse de la personne se retrouve ralentie, et ses vecteurs prennent la vitesse de nage de notre individu.

III Événements

1. Vagues

A l'initialisation de la plage, une liste de vagues va être créée avec des caractéristiques qui dépendent des conditions. Plus le temps est mauvais et le vent fort, plus vagues seront rapides, fortes et hautes, ce qui bien sûr affecte plus ou moins les personnes. Une vague va se rapprocher de la côte et quand elle l'aura atteint, repartir du bout de la mer qui est au bas de la fenêtre. De cette manière, après la création initiale, il n'y a pas besoin de créer d'autres objets de type vague.

2. Noyade

Malheureusement, cette partie n'est pas terminée. Voici ce qui devrait avoir lieu :

Si dans l'eau, l'énergie de la personne passe en dessous d'un certain cap, elle revient vers la terre. Si par malheur des vagues trop fortes la frappe, son énergie va arriver à 5 avant qu'elle ne puisse atteindre la rive. Dans ce cas, elle appelle un sauveteur.

Si son énergie est nulle lors du sauvetage, elle est inconsciente et nécessite une aide supplémentaire, et est exfiltrée de la plage.

3. Sauvetage

Malheureusement, cette partie n'est pas terminée. Voici ce qui devrait avoir lieu :

Les personnes capables d'effectuer des sauvetages sont les Sauveteurs. Ils héritent des Personne, mais n'ont pas le même comportement. Leur emplacement est le poste de sauvetage, fixé au début de la simulation. Elles patrouillent sur la plage ou restent au poste, jusqu'à ce que quelqu'un demande une assistance. Si c'est le cas, le plus proche sauveteur court vers la personne, en bousculant tout le monde, et l'extrait de l'eau.

4. Vendeur

Après une durée déterminée, un **vendeur** apparaît sur la Plage et a pour objectif de vendre et de faire du commerce si l'occasion s'y présente. Contrairement aux personnes, le vendeur apparaît devant la Mer. De plus, il reçoit de la part de la Plage les coordonnées par rapport à l'endroit où il doit se rendre, à savoir la fin de la plage. Si pendant son déplacement, une personne est en état d'acheter, le thread correspondant au vendeur (tout comme celui de la personne) sera alors immobilisé pendant un certain temps afin de vendre des éléments à la personne et son état passera à commerce. Dans le cas contraire, le vendeur poursuit son mouvement et lorsqu'il atteint les coordonnées données par la Plage il disparaît de la simulation.

Une file d'attente se crée derrière le vendeur si plusieurs personnes veulent acheter en même temps. Une fois vidée, il reprend son mouvement.

Ca ne marche pas très bien….

IV Les interfaces

1) InteractionMenu

Dès le lancement de la méthode main, l'interface `InterfaceMenu` se lance et apparaît. Elle propose des paramètres déjà rentrés pour le lancement de la simulation que l'utilisateur peut modifier si voulu:

- soit en important ses propres paramètres grâce à un volet déroulant proposant ses documents txt sauvegarder dans le dossier `Patterns`.
- soit en remplaçant les données directement dans les cases spécifiques en saisissant les paramètres voulus.
- soit en utilisant les préréglages implémentés dans des volets déroulants

Une fois les paramètres choisis, l'utilisateur à le choix avant de lancer la simulation de sauvegarder ses données saisies grâce au bouton `Sauvegarder` qui va lancer une autre interface nommée `frameFichier` qui propose de rentrer le nom du fichier txt qu'il veut sauvegarder. Une fois saisi le nom, le fichier va être enregistré dans le dossier Sauvegardes et être réutilisable par la suite.

Lorsque l'utilisateur à choisi ses paramètres pour la simulation, il va cliquer sur le bouton `Lancement !` et l'`interfaceMenu` va récupérer les valeurs saisies et se fermer.

2) Interface

L'interface commence par initialiser la fenêtre où la plage sera affichée, qui si nécessaire suivant les dimensions de la plage, sera affectée d'un multiplicateur afin qu'elle ne soit ni trop petite ni trop grande.

La première chose affichée sera la plage (avec la mer), proportionnellement aux dimensions de la plage entrées en paramètre. Après on effectue une boucle où on vérifie tous les paramètres que l'on veut afficher pour chaque individu sur la plage. Ces paramètres sont la position des personnes et leur état. Suivant le statut (noyé ou non) et la nature des personnes (sauveteur, vendeur ou normale), la couleur dans laquelle la personne sera affichée est différente.

Ensuite, l'interface parcourt la plage afin de détecter les cases sur lesquelles sont posées des affaires et les colore en noir plutot qu'en jaune comme les cases de plage normales, puis la liste des vagues et les colores en blanc transparent, le poste de secours en rouge.

Bien sûr, l'interface se redessine régulièrement selon le même principe.

3) BarreVitesse

L'interface `BarreVitesse` se lance dès la fermeture de l'interface `InteractionMenu` donc en même temps que l'interface Interface. Elle permet de pouvoir changer la vitesse de la simulation grâce au composant Slider qui permet de changer la vitesse de la simulation en déplaçant un bouton de gauche à droite et de faire apparaître la vitesse actuelle.

Conclusion

Malheureusement, nous n'avons pas réussi à fournir ce que nous voulions à temps. Nous nous en excusons et essaieront de finir ce qu'il faut avant l'oral.

Néanmoins, ce projet nous a permis de voir le fonctionnement des Threads en Java et de nous exercer sur un projet conséquent demandant de l'investissement.

Rapports individuels

Rapport individuel - Lucas Triozon

Semaine 1 :

- Réflexion sur le sujet du projet, répertorisation des idées et répartition en catégories (par classe et par importance). Travail effectué en groupe par tout le monde, pour donner une direction au projet.

Semaine 2 :

- Mise en cohérence des premières classes écrites par le reste du groupe, correction des erreurs, pour commencer à mettre en relation la Plage, les Personnes et la fonction Main.

Semaine 3 :

- Création des méthodes turn() de Plage et run() de Personne, permettant aux personnes de savoir quoi faire, se déplacer de case en case vers un objectif, être sûr que leur déplacement est autorisé, puis le confirmer et mettre à jour les matrices qui observent ce qui se passe.
- Test de tout et correction des bugs : les Personne se déplacent, l'interface fonctionne, et la Plage écoute

Semaine 4 :

- Modification des constructeurs de Plage et Personne pour avoir une arrivée progressive des Personne sur la plage, à un intervalle donné. Toutes les instances sont créées en même temps, mais apparaissent au fur et à mesure du temps qui passe.
- Amélioration de la méthode modifVision() de la classe Plage, pour qu'elle fonctionne vraiment
- Correction de bugs

Semaine 5 :

- Rédaction de l'approche de notre projet pour le rapport
- Correction de bugs

Semaine 6 :

- Correction de bugs
- Faire en sorte que les Personnes soient totalement indépendantes dans leurs actions, ce n'est plus la Plage qui détermine leurs états. Son rôle sur les Personnes se limite à donner des coordonnées si besoin.
- Début de l'écriture des Sauveteurs, mis en suspens par la suite.

Semaine 7 :

- Changement total du système de déplacements. Retrait de la matrice géante dans Plage, des Cases et des matrices de vision des Personnes. Passage sur un système de Vecteurs, qui permet des mouvements plus fluides dans

un vrai plan et des coordonnées (x,y) en R^2. Revue des collisions entre personnes, du placement, et du pathfinding pour correspondre à ce nouveau système. “Ceci est une révolution”.

Semaine 8 :

- Reprise et mise en place des Sauveteurs. C’est une classe dérivée de Personne, ayant pour but de patrouiller sur la Plage et de sauver les Personnes de la noyade.

Semaine 9 :

- Stamina et sauvetages (pas fini)
- Rédaction finale rapport
- Collisions avec les affaires
- Correction de bugs

Semaine 10 :

- Sauvetage des meubles pour avoir un rendu pas trop horrible (c’est raté)

Rapport Individuel - Adrian Guilhem

Semaine 1 : Réflexion sur le sujet du projet, répertorisation des idées et répartition en catégories (par classe et par importance). Travail effectué en groupe par tout le monde, pour donner une direction au projet.

Semaine 2 : Création basique de la classe Personne, basée sur les idées répertoriées la semaine précédente, et de la plupart de ses getters et setters, ainsi que de la plupart des méthodes (certaines conservées, certaines supprimées plus tard et d'autres adaptées)

Semaine 3 : Recherches sur les interfaces graphiques et création de toute la classe interface graphique du projet.

Semaine 4 : Création de la méthode PlacementPlage() de la classe Plage et implémentation dans la méthode turn() et division de la plage en zones. Tout cela permet aux personnes d'avoir chacune une place sur la plage pour poser leurs affaires, le plus proche de la mer possible et avec un certain espacement avec les autres personnes et de leur donner le moyen de s'y rendre et d'y poser leurs affaires à leur arrivée.
Modifications de l'interface graphique pour afficher correctement les affaires sur la plage.

Semaine 5 : Ajout d'un curseur dans une seconde fenêtre de type interface qui permet de changer la vitesse de la simulation en agissant sur la vitesse des personnes.

Semaine 6 : Modification de l'interface afin qu'elle prenne à peu près tout l'écran et que tous les composants soient toujours bien affichés. Cela nous permettra de mieux voir les collisions ou autres problèmes s'il y en a. On a un problème sur l'affichage des affaires qui affiche trois rectangles avec un espace entre eux au lieu d'un seul grand rectangle.

Semaine 7: Ajout d'un zoom à la création de l'interface afin que la fenêtre ne soit pas trop petite lorsque la plage est petite. Il conserve le ratio longueur/largeur de base de la plage et n'est pas trop important afin de ne pas créer de problèmes sur l'affichage comme nous en avions la semaine dernière. J'ai aussi réglé les problèmes causés par la création d'une plage plus grande que les dimensions de l'écran, maintenant elle sera réduite afin d'être entièrement visible sur l'écran, toujours en conservant son ratio de base.

Semaine 8: Création de la météo, de manière à ce qu'elle change aléatoirement régulièrement (par soucis de réalisme, elle ne peut pas passer de "soleil" à "pluie" et inversement mais doit passer par "nuageux" entre les deux) et implémentation de ses conséquences sur les personnes. Création du vent qui change aussi aléatoirement (entre plusieurs paliers afin de ne pas changer trop brusquement, comme la météo). Il a aussi des effets plus ou moins forts sur les personnes.

Semaine 9: Changement du vent et de la météo afin qu'ils ne soient plus aléatoires mais plutôt choisis à l'initialisation. Réécriture de la majeure partie de ces méthodes pour celà mais aussi pour l'adaptation après les changements sur la baignade, passée de la classe plage à la classe personne, et résolutions de quelques petites erreurs. Implémentation des "vagues", qui dépendent de la météo et du vent et qui agissent directement sur les chances de noyade des personnes.

Semaine 10: Création de la classe vague (et incorporation dans Plage et Personne) afin que les vagues ne soient plus juste un un concept invisible qui change les attributs des personnes dès qu'elles sont dans l'eau, mais des threads qui sont visibles, se déplacent, et qui n'affectent les personnes que lorsqu'elles passent dessus. Leur nombre et leurs caractéristiques changent en fonction de certains paramètres de la plage. Résolution de problèmes de taille de la fenêtre de l'interface présents dans certains cas, et enfin, affichage des vagues.

Rapport individuel - Elisa Cezard

Semaine 1 : Réflexion sur le sujet du projet, répertorisation des idées et répartition en catégories (par classe et par importance). Travail effectué en groupe par tout le monde, pour donner une direction au projet.

Semaine 2 : Création du début de la classe Plage avec les setters et getters et réflexion sur comment elle va marcher.

Semaine 3 : Création de l'interface InteractionMenu() qui permet à l'utilisateur de choisir les préréglages de l'interface de la plage avec : sa longueur, sa largeur, la vitesse du vent, la température et le nombre d'individus.

Semaine 4 : Réflexion sur comment pouvoir gérer la vitesse de déplacement des individus, de la simulation et rédaction de l'introduction du projet dans le rapport.

Semaine 5 : Ajout à l'interface InteractionMenu, de volets déroulants permettant de choisir des paramètres par défaut pour les pré-réglages.

Semaine 6 : Création d'une nouvelle interface pour pouvoir contrôler la vitesse de la simulation.

Semaine 7 : Ajout dans InteractionMenu du choix du type de météo lors de la simulation. Ajout d'un bouton sauvegarde à l'interface InteractionMenu permettant de pourvoir enregistrer les données des paramètres saisies par un utilisateur dans un fichier .txt contenu dans un dossier Sauvegardes.

Semaine 8 : Ajout d'un volet déroulant permettant de récupérer les fichiers .txt contenus dans le dossiers Patterns et de pouvoir en choisir un et utiliser son contenu pour changer les paramètres de la simulation.

Semaine 9 : Rédaction de la partie InteractionMenu dans le rapport final.

Rapport individuel - Khalil Makhloufi

Semaine 1 : Réflexion sur le sujet du projet, répertorisation des idées et répartition en catégories (par classe et par importance). Travail effectué en groupe par tout le monde, pour donner une direction au projet.

Semaine 2 : Création de la méthode Main qui consiste à lancer la simulation en faisant appelle aux classes Plages et Personnes tout en considérant que chaque personnes soit représentée par un Thread chacun.

Semaine 3 : Création d'une nouvelle méthode intitulée Placement(). Cette méthode permet aux personnes qui entrent dans la plage d'avoir une position en objectif, de l'emmener, de la placer et de faire en sorte que la Plage enregistre le fait que la place est occupée.

Semaine 4 : - Début de réflexions sur le déplacement des personnes afin de les guider vers la mer pour qu'ils puissent se baigner. L'objectif est donc de déterminer les nouvelles coordonnées (dans la mer), tout en maintenant les bonnes transition des états et objectif, afin que la personne se dirige vers l'eau. Le thread sera suspendu pour une période x secondes pour bien représenter le temps de baignade (une personne ne va pas rester une demi seconde dans l'eau).

Semaine 5 : - Finalisation de la méthode gobaignade(). L'objectif de position est bien déterminé et correct. D'après plusieurs test, les personnes se place bien à une position x,y sur la plage puis vont à la mer et reste un peu de temps pour se baigner.

- Début de réflexion sur les mécanismes à mettre en place afin de permettre à la personne de partir de la plage. L'objectif de position devra à nouveau changer et les places libérées devront être considérées comme « vide » sur la Plage. (Le thread va donc être stoppé)

Semaine 6: Finalisation de processus permettant à la personne de disparaître complètement de la plage. S' ils ne veulent pas acheter, ne sont jamais allé à la Mer ou ne veulent plus y retourner (aléatoire), alors il devront se diriger vers l'endroit où ils sont entrés en ayant un nouvel objectif de position puis disparaître. C'est bien la classe Plage qui décide de supprimer le Thread de la personne concernée et non la classe Personne.

Semaine 7: (Actuellement les personnes vont dans l'eau et restent immobile pendant un certain temps). Mise en place de plusieurs instructions les permettant de faire de la nage sur de longues ou courtes distances (aléatoires). Dans un seconds temps il retourne à leurs place se reposer puis ont une multitude de choix (acheter, se baigner, partir).

Semaine 8: Début de réflexion de la classe Vendeur qui hérite de la Personnne. Création des constructeurs, setters, getters. Modification dans la méthode run de la classe Personne:

- dans la condition du Repos, mise en place d'une condition consistant à déterminer de manière aléatoire si la personne veut acheter ou pas.
- ajout d'une nouvelle Condition (else if) qui consiste à voir si la personne est en Etat d'acheter.
- si oui alors la personne aura un nouvel itinéraire (dans la classe Plage: Turn) pour se diriger vers le vendeur.

Semaine 9: Apparition de la méthode Run() concernant le vendeur avec plusieurs enchaînements d'état et d'objectif. Dans la plage, pour un nombre x de personnes, le vendeur apparaîtra à la moitié de ce nombre après un temps déterminé (au début du run).
Les personnes ne peuvent plus aller se placer devant la Mer (5 mètres environ) pour laisser la libre circulation au vendeur.
Dans la classe Interface, le faible espace se situant devant le vendeur sera colorié en noir afin de représenter la charrette.