

Rapport de projet : Conception d'un système autonome de
surveillance du port de Marseille

MIG « Systèmes embarqués »

Romain CAMPILLO, Nicolas CHABRIAC, Evann COURDIER, Thibault FORGET,
Charles FOUCAULT, Simon JONCHÈRE, Paul LE FLOC'H, Samuel MACKEOWN,
Charles MASSON, François PACAUD, Arthur ROLLAND, Guillaume SCHNEIDER-MAUNOURY
Avec la participation du CMA

20 décembre 2012



Nous souhaitons avant tout remercier pour leurs interventions :

Xavier Rival de l'INRIA

Guillaume Foeillet de la SNCF

Salma Zouaoui du CMA

Patrice Mistretta de Thales Underwater Systems

Pierre Deroi du GPMM

Khodor AbouTaha, Antoine Morel, Coralie Chouman et Marc Saisset d'Eurocopter

Nicole Mary et Miginiac Roland de Dassault-Aviation

Laurent Thery de l'INRIA

Ainsi que pour leur accueil chaleureux au CMA :

Nadia Maïzi

Catherine Auguet-Chadaj

Table des matières

I Buts et objectifs de notre projet	6
1 Définition de l'approche client	6
1.1 Présentation du contexte	6
1.2 Objectif de notre projet	6
2 Systèmes existants	7
3 Analyse économique	8
3.1 Équipements retenus	8
3.2 Évaluation des coûts d'étude	9
3.3 Budget de fonctionnement du système	9
4 Cahier des charges	10
4.1 Choix algorithmique	10
4.2 Interface graphique homme-machine	10
II Développement du projet	11
1 Organisation du projet	11
2 Spécifications	12
3 Préparation de la zone de travail	12
3.1 Définition de normes de travail communes au groupe	12
3.2 Choix du système de coordonnées approprié	12
3.2.1 Définition d'une zone de travail commune à tout le groupe	13
3.2.2 Création d'états : discréétisation de la carte	14
3.3 Exploiter les données pour assurer la surveillance dans le port	14
3.3.1 Définition des zones	14
3.3.2 Application à la sécurité du port	15
4 Bases de données	15
4.1 Nécessité d'utiliser une base de données	15
4.1.1 Données disponibles	15
4.1.2 Utilité de la base de données	16
4.2 Création de la base de données	16
4.3 Utilisation de la base de données	17
4.3.1 Récupération des données	17
4.3.2 Normalisation des positions	17
4.4 Défauts et possibilités d'amélioration	18
5 Prédiction des trajectoires grâce aux Chaînes de Markov Cachés	18
5.1 Présentations des chaînes de Markov	18
5.2 Adaptation à la prédiction de trajectoire	19
5.3 Adaptation au projet	21
6 Prédiction des trajectoires selon une approche statistique	21
6.1 Introduction	21
6.2 Construction de l'arbre de probabilités	22
6.3 Prédiction de la trajectoire la plus probable	23
6.4 Probabilité de présence	23
6.5 Résultats	24

7 Déviation des trajectoires	24
7.1 Maneuvre d'évitement d'obstacles statiques	25
7.1.1 Définition de la zone impraticable	26
7.1.2 Calcul de la déviation par contournement	26
7.1.3 Une autre approche : l'algorithme de <i>Dijkstra</i>	26
7.1.4 Ré-incorporation de la composante temporelle	27
7.2 Manœuvre d'évitement d'obstacles dynamiques	28
7.2.1 Établissement de tables de priorité	29
7.2.2 Traitement des différentes trajectoires	29
7.2.3 Utilisation de la déviation <i>statique</i>	30
8 Interface graphique	30
8.1 Fonctionnalités du logiciel	31
8.2 Difficultés rencontrées	33
Conclusion	35

Introduction

Aujourd’hui, les systèmes embarqués sont utilisés partout, de l’aéronautique à la défense en passant par la construction automobile. Ils doivent être très stables : le moindre problème peut causer des dégâts énormes, tant d’un point de vue humain qu’économique. Le crash de la fusée Ariane en 1996 dû à une panne logicielle s’est chiffré en centaines de millions de dollars. Pourtant, ces systèmes sont nécessaires pour renforcer la sécurité. Par exemple, les ingénieurs d’Eurocopter réussissent à créer des systèmes qui préviennent et corrigent les erreurs de pilotage.

L’amélioration de la sécurité et l’optimisation des systèmes passent souvent par une meilleure intégration logicielle. De lourds investissements doivent alors être faits pour prouver la correction des systèmes embarqués, que ce soit du côté logiciel comme du côté matériel. Pour ce faire, des systèmes logiciels de vérification et de preuve de programmes ont été implémentés, notamment par l’INRIA (Coq[?], Esterel[?], etc).

Dans le cadre de notre MIG, nous avons étudié plus en détails ces systèmes embarqués. Nous nous sommes rendus au *Centre de Mathématiques Appliquées (CMA)* de l’*École des Mines de Paris (MINES ParisTech)*. Nous avons pu visiter les usines d’Eurocopter à Marignane et de Dassault Aviation à Istres, où sont prouvés et testés les systèmes embarqués à bord des hélicoptères ou des avions. Pour avoir un meilleur point de vue sur les difficultés techniques induites par la sécurisation de ces systèmes, nous avons assisté à quelques cours sur la preuve de programme. Ainsi, nous avons abordé l’analyse statique de programme, la preuve de programme par Coq et les langages impératifs comme Esterel.

Notre projet s'est inscrit dans l'idée de concevoir un système autonome de surveillance multi-capteurs du port de Marseille ; le CMA avait eu un rôle actif il y a quelques années pour créer un tel produit. Nous avons choisi de créer par nous-mêmes un système qui prend en entrée les données traitées de l'ensemble des capteurs présents dans le port de Marseille, qui analyse ces données dans le but de surveiller le comportements des bateaux présents dans la zone et qui fournit une interface affichant les déplacements des bateaux dans l'enceinte du port et en montre éventuellement les mauvais comportements. Afin de mieux cerner les besoins et les attentes de nos hypothétiques clients, nous avons discuté avec Pierre Deroi, responsable de la sécurité du port de Marseille et Patrice Mistretta de *Thales Underwater Systems* afin de dresser un cahier des charges. Une fois ce cahier établi, nous avons pu passer à l'implémentation même de notre système logiciel. Pour ce faire, nous avons dû travailler en équipe et diviser les tâches. À la fin du MIG, nous avons pu prendre un peu de recul par rapport à nos réalisations, et faire le point sur notre projet.

C'est l'ensemble de notre démarche que nous souhaitons vous présenter dans ce rapport.

Buts et objectifs de notre projet

1 Définition de l'approche client

1.1 Présentation du contexte

Notre projet répond à une attente de la part du port de Marseille. Contrairement à l'accès terrestre, l'accès maritime reste peu surveillé. Depuis le 11 septembre 2001, et dans le cadre de la lutte contre le terrorisme, les autorités portuaires sont tenues de surveiller les marchandises qui embarquent dans leur port. Le but est d'éviter la circulation d'armes et de passagers clandestins. De plus, la situation de Marseille fait que les fraudes et autres trafics sont très fréquents dans le port : Marseille est en effet un carrefour entre la France (et donc l'Europe), les pays du Nord de l'Afrique et les pays de l'Est. Les trafics d'armes et de drogues sont donc très actifs. Sans parler des vols à l'intérieur même de l'enceinte du port. Pour lutter contre cet état de fait, l'accent a été mis sur la sécurité de l'accès terrestre au port : construction d'une grille pour fermer l'accès terrestre, mise en place de contrôles à l'intérieur de l'enceinte du port et d'un système de vidéosurveillance performant. Cette politique commence aujourd'hui à porter ses fruits, le port étant bien plus sûr qu'au début des années 2000¹.

Cependant, l'accès maritime au port de Marseille reste quant à lui peu surveillé. Ce qui, en soi, est problématique au vu des enjeux économiques, commerciaux et stratégiques. Le Grand Port Maritime de Marseille (GPMM) concentre en effet en son sein des raffineries, des terminaux pétroliers et gaziers, ainsi que des usines pétrochimiques. Pour donner un ordre de grandeur, près de 40% du pétrole consommé en France passe par le port de Marseille-Fos². Un attentat contre une raffinerie pourrait paralyser tout le pays. De même, si une attaque est menée à bien contre une usine chimique, les conséquences sur les populations avoisinantes peuvent être désastreuses : certaines usines retraitent des produits très dangereux comme le chlore. Dans tous les cas, ce sont la sûreté des personnes, le respect de l'environnement et les intérêts économiques qui sont en jeu.

Un système de surveillance aurait l'avantage de pouvoir détecter les menaces éventuelles et permettre ainsi aux autorités (sémaphores, douaniers, gendarmes maritimes) d'intervenir le plus vite possible. Le but d'un tel système est aussi de détecter des petites menaces comme celles représentées par les trafiquants de drogue, qui utilisent souvent des petits bateaux (jet ski, *go-fast*) pour transporter leurs marchandises.

Les partenaires du projet **SECMAR**³ (SÉCurité MARitime), du pôle de compétitivité Mer-PACA, dont *Thales Underwater Systems* a été le leader, ont réalisé le prototype d'un système de surveillance multi-capteurs. Un ensemble de capteurs innovants (SONAR, RADAR, caméras optroniques, ...) avait été mis en place pour surveiller la zone maritime du port (qui représente environ 10.000 hectares, l'étang de Berre mis à part). Ce prototype, fonctionnel au début de l'année 2011, est aujourd'hui démantelé mais nous avons pu disposer d'enregistrements des données issues de ses capteurs ainsi que de son module de fusion de données.

1.2 Objectif de notre projet

Notre projet reprend donc cet objectif de sécuriser le port de Marseille via une approche logicielle. On va chercher à utiliser les données issues des capteurs du port de Marseille pour créer une interface de surveillance de la surface maritime du Port de Marseille. Cette interface doit être ergonomique pour que les opérateurs puissent facilement l'utiliser. Le programme final doit prendre en entrée les données traitées issues des capteurs pour pouvoir afficher les bateaux présents dans le port et détecter les menaces éventuelles. Les algorithmes utilisés doivent être suffisamment performants pour éviter la multiplication des erreurs et des fausses alertes tout en détectant efficacement les menaces.

1. Source : entrevue avec M. Pierre Deroi le 27 novembre 2012

2. Source : <http://archive.wikiwix.com/cache/?url=http://www.unctad.org/Templates/WebFlyer.asp?intItemID=4398%26lang=2&ttitle=%C3%88>

3. Une présentation du projet est disponible sur <http://www.polemerpaca.com/Securite-et-surete-maritime/Protection-maritime-et-endeue-et-rapprochee/SECMAR>



FIGURE 1 – Plan du port de Marseille (*source : GoogleMaps*)

2 Systèmes existants

Bien que l'accès maritime reste peu sécurisé, certains systèmes existent déjà et doivent être pris en compte.

Aujourd’hui, pour savoir ce qu’il se passe en mer, le port est muni de trois vigies d’observation qui permettent d’avoir une vue d’ensemble sur la partie maritime. De plus, le dispositif VTS (*Vessel Traffic Services*) oblige les navires d’une certaine taille ou transportant des passagers ou des produits dangereux à se faire reconnaître par radiotéléphonie auprès d’un centre de contrôle et le cas échéant, à recevoir des recommandations sur leur route et leur vitesse pour éviter les accidents.

Ces bateaux sont obligatoirement munis d'un système nommé **AIS** (*Automatic Identification System*) qui émet en permanence différentes informations sur le bateau comme sa position, sa cargaison, sa destination, et aussi son numéro **MMSI** (*Maritime Mobile Service Identity*) qui est le numéro d'immatriculation du bateau. Ils sont donc suivis tout le long des côtes françaises par les autorités, bien avant leur arrivée au port, et tout comportement suspect est signalé. Ils doivent de plus prévenir 24 heures avant leur arrivée et de nombreux contrôles sont effectués avant de les laisser entrer dans le port. Un pilote doit nécessairement monter à bord du bateau pour l'aider à accoster, ce qui limite les accidents et permet de prévenir la vigie en cas de comportement suspect.

Enfin, le port suit le **code ISPS** (*International Ship and Port Facility Security*) qui l'oblige à vérifier le contenu de chaque bateau qui sort du port. Cela ne protège pas le port en lui-même mais plutôt le bateau (donc le trafic international) et le port d'arrivée. Ainsi, un bateau qui arrive à Marseille depuis un port satisfaisant la norme ISPS a normalement eu sa cargaison contrôlée lors de son départ.

Toutes ces mesures permettent donc de sécuriser le port, mais ce dernier reste très vulnérable à un accident ou à une attaque mettant en jeu un petit bateau qui n'est pas tenu d'avoir un AIS.

Pour récupérer davantage d'informations sur les bateaux navigant dans le port, il faudrait donc implanter d'autres types de senseurs :

- des radars qui émettent une onde électromagnétique qui va se réfléchir sur les bateaux et revenir en écho, permettant ainsi de connaître leur position. Des radars à effet Doppler peuvent aussi mesurer la vitesse de ces

- bateaux ;
- des sonars (soit actifs qui émettent une onde sonore sous l'eau, soit passifs qui écoutent les sons sans en émettre) permettant de la même manière de connaître l'azimuth des bateaux et des sous-marins.

Une des difficultés majeures est alors le traitement des données reçues pour éliminer le bruit dû aux vagues et à de mauvaises conditions météorologiques pour pouvoir suivre efficacement un bateau⁴. Certaines difficultés sont aussi dûes à l'implantation même de ces senseurs. Ils ne doivent par exemple pas perturber les appareils de mesure des avions qui atterrissent à la base militaire d'Istres ou à l'aéroport de Marseille. Les senseurs placés en hauteur doivent aussi être stables par grand vent, d'où la nécessité de les placer sur des pylônes pouvant s'enfoncer à plusieurs dizaines de mètres dans le sol. Enfin, leur positionnement doit être étudié : il faut par exemple qu'il y ait des recoulements dans les zones de couverture des capteurs pour qu'un endroit du port ne devienne pas invisible dès qu'un pétrolier passe devant un radar.

3 Analyse économique

Introduction

Après toutes les considérations théoriques et techniques de l'élaboration du projet, il est nécessaire de s'intéresser aux coûts et d'évaluer le budget nécessaire à la réalisation d'un tel système automatique de surveillance multi-capteurs. En effet, le budget est au cœur des préoccupations de l'ingénieur et bien souvent mis au premier plan. C'est lui qui limite le développement du projet répondant au cahier des charges. Un système automatique de surveillance nécessite en effet l'installation de capteurs de natures diverses (sonars, caméras, radars,...), et le coût de ces équipements affecte directement le projet. En effet, à la question « combien de capteurs place-t-on dans la baie du port ? » vient peser le facteur coût, à savoir : « chaque caméra rajoutée augmente le budget. » Il va donc falloir optimiser le placement des capteurs pour éviter une explosion des coûts tout en assurant une surveillance efficace. C'est en ce sens qu'une analyse économique est pertinente, pour offrir le meilleur système possible avec un coût minimal.

Nous tâcherons également d'évaluer les coûts de l'étude que nous avons réalisée en tant que travail mené par des ingénieurs et donner le budget total d'un tel projet (de l'étude à la mise en place des équipements) pour avoir ainsi une idée du poids économique que pourrait avoir une implémentation réelle de notre projet.

3.1 Équipements retenus

Voici les équipements nécessaires à la récupération des données. Dans le cadre de cette étude économique, nous avons choisi la nature, le type et le nombre des équipements :

- **Senseurs**
 - 2 sonar (prix à l'unité : 150 000 €)
 - 4 radars (prix à l'unité : 150 000 €)
 - Coûts d'installation des radars (prix : 200 000 €)
 - 1 caméra de poursuite et 1 pylône de maintien pour la stabiliser⁵. (prix à l'unité : 200 000 €)
 - 20 caméras (prix à l'unité : 30 000 €)

4. Source : entrevue avec M. Patrice Mistretta le 23 novembre 2012

5. la caméra doit être très stable pour avoir une bonne qualité d'image. Même en cas de vents forts, elle ne doit pas bouger sur son axe de plus de 2cm

- **Équipements informatiques**

- 10 serveurs (prix à l'unité : 2500 €)
- 10 racks⁶ (prix à l'unité : 4000 €)
- 10 licences logicielles (prix à l'unité : 10 000 €)
- 1 équipement Wimax : 5 antennes émettrices placées sur des points hauts.⁷ (prix à l'unité : 60 000 €)

Il faut également prendre en compte les aléas lors de l'installation (augmentation des délais du développement, solution qui ne marche pas, tempête qui annule les essais, structure qui coûte plus cher que prévue, etc). On doit donc se donner une marge supplémentaire de 10 % pour prendre en compte les risques.

Il est bien sûr nécessaire de prendre en compte les coûts de maintenance qui nous ont été indiqués à environ 10% du coût total par an. De surcroît, l'environnement marin nécessite une maintenance plus importante du fait de la corrosion et de toutes les salissures des équipements (par exemple des mollusques peuvent s'accrocher au sonar qui doit donc être nettoyé régulièrement).

Coûts de maintenance compris, cela représente un total de :

2,5 millions d'euros à l'installation + 250 000 €par an.

3.2 Évaluation des coûts d'étude

Il s'agit désormais d'évaluer le coût de notre travail, à savoir le développement d'algorithmes et d'une interface graphique permettant d'interpréter les données recueillies par les senseurs et de faciliter la surveillance du port.

Nous fixons le coût horaire d'un ingénieur sur ce projet à 100 euros de l'heure. Au nombre de 12, nous avons travaillé pendant 15 jours à raison de 9 heures par jour.

Cela représente donc un total de :

162 000 €.

3.3 Budget de fonctionnement du système

Une fois le système mis en place, la question se pose des coûts de fonctionnement. L'un des critères du cahier des charges est d'obtenir un système de surveillance automatique du port, donc limitant au maximum une surveillance humaine. Il faut néanmoins prendre en compte le coût du personnel devant mettre en œuvre le PC de sûreté maritime au niveau de la vigie de la capitainerie, à savoir un opérateur présent 24h/24. On peut fixer ce budget à une somme indicative de :

96 000 €par an.

Conclusion

Ces évaluations donnent une idée de la démarche utilisée par l'ingénieur pour évaluer un tel projet et des éléments principaux qu'il faut recenser dans notre projet. Le prix résultant total est évidemment faussé par tous les coûts annexes que nous n'avons pas pris en compte comme par exemple les coûts d'installation.

6. Bâties métalliques contenant les serveurs, les machines électroniques, les câbles, etc.

7. L'équipement Wimax permet de transmettre des données vidéos en temps réel aux différents postes de contrôle du port ainsi qu'aux éventuelles vedettes d'intervention sur mer.

4 Cahier des charges

Maintenant qu'une mise au point a été faite sur les systèmes existants et sur les contraintes économiques, on peut commencer la conception de notre projet. Comme expliqué précédemment, le but de ce dernier est de proposer un système de surveillance autonome qui assiste les techniciens de la régie dans leur prise de décision. Ce système doit être à la fois **ergonomique** et **simple d'utilisation**.

4.1 Choix algorithmique

On peut maintenant définir ce que va effectuer notre système.

Prédiction de trajectoire Ce dernier doit pouvoir prévoir les trajectoires des bateaux pour anticiper les accidents et permettre aux autorités d'intervenir rapidement. Pour améliorer la sécurité, nous avons choisi deux approches algorithmiques complémentaires pour prévoir les trajectoires des bateaux :

- une approche basée sur les **Modèles de Markov Cachés** (MMC). On utilise pour ce faire l'algorithme de *Baum-Welch*⁸ ;
- l'autre algorithme utilise un découpage en zones du port de Marseille pour aborder d'un point de vue purement statistique les trajectoires des bateaux.

Perturbations Enfin, un dernier algorithme a été proposé pour pouvoir déterminer automatiquement la trajectoire d'un bateau en cas de perturbation (accident, obstacle imprévu sur la trajectoire initiale, risque de collision avec un autre bateau). Cet algorithme permet de déterminer automatiquement la nouvelle trajectoire du bateau à partir de sa position initiale, de son point d'arrivée et de la position de la perturbation.

4.2 Interface graphique homme-machine

Une interface graphique sera ensuite créée pour pouvoir afficher de manière simple les résultats obtenus par les différents algorithmes. Cette interface devra être ergonomique. Elle affichera la position des bateaux en temps réel en fonction des données reçues et affichera les différentes trajectoires possibles grâce aux algorithmes. Elle devra aussi afficher les différents senseurs disponibles sur le port et les bateaux détectés par ces derniers. Cette option permet de vérifier si les capteurs surveillent bien l'ensemble de la surface maritime sans avoir d'angle mort.

8. cf. Article de Lawrence Rabiner présenté dans la Bibliographie

Développement du projet

1 Organisation du projet

Afin d'aboutir au programme final, nous avons constitué plusieurs groupes travaillant chacun sur une partie spécifique du programme. Toutefois, chaque groupe utilise au sein de ses algorithmes les résultats des algorithmes des autres groupes, ce qui nourrit une certaine interdépendance. C'est pourquoi il a fallu lors de l'implémentation de nos algorithmes, d'une part, prendre en compte les besoins spécifiques des autres groupes et d'autre part, respecter certains standards de programmation que nous avions préalablement établis (typage des variables, répartition en classes et en fonctions...).

Trois pôles de développement se distinguent :

- Le premier pôle est chargé du traitement des données. Son rôle est de rendre les données dont nous disposons exploitables pour les autres groupes, notamment grâce à la constitution d'une base de données *MySQL*.
- Le deuxième pôle implémente le cœur du programme, c'est-à-dire les algorithmes qui servent à la gestion de la sûreté et de la sécurité du port. D'une part, la base de données *MySQL* est utilisée pour prédire la trajectoire à long terme des bateaux en utilisant les modèles de Markov cachés. D'autre part, la trajectoire à court terme des bateaux est prédite grâce à une méthode statistique. Cette prédiction est ensuite utilisée pour gérer la présence de perturbations sur le trajet prévu pour le bateau.
- Enfin, le troisième pôle programme l'interface graphique. Cette dernière donne la position en temps réel des bateaux dans le port, affiche les informations sur les bateaux dans le port ainsi que les trajectoires calculées par les algorithmes susmentionnés, les alertes de sûreté ou de sécurité ainsi que la position et la portée des senseurs.

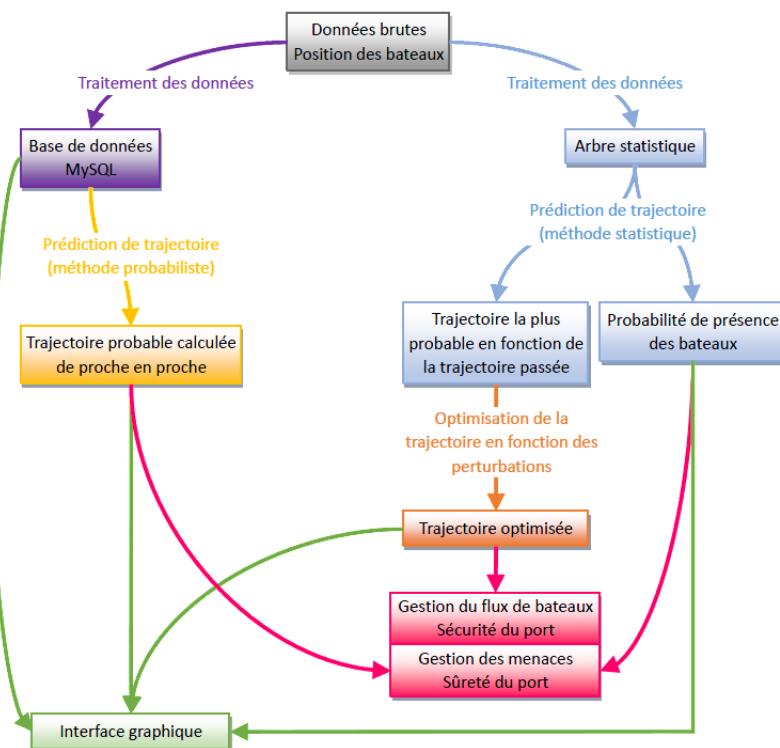


FIGURE 2 – Organigramme du projet

2 Spécifications

Comme expliqué dans la partie précédente, certains choix ont été effectués pour améliorer la coordination entre les groupes. Cette partie explique ainsi la logique de nos démarches, et les moyens qui ont été mis en œuvre pour les réaliser.

Compte tenu de la nécessité d'avoir un suivi des bateaux, le choix d'un logiciel possédant une interface graphique claire et lisible qui puisse être facilement utilisée s'est imposé. Nous nous sommes servis d'un logiciel indépendant de Java du nom de **Processing** qui nous a permis de créer une interface graphique. D'autre part, il était indispensable de retraiter les données fournies par les capteurs afin de pouvoir les exploiter. Pour ce faire, nous avons eu recours à un système de gestion de base de données du nom de **MySQL** qui permet un accès rapide et un classement des données selon des critères pertinents.

Nous avons tous travaillé sur l'environnement de programmation *Eclipse* muni du langage *Java*, respectivement logiciel et langage avec lesquels nous sommes tous familiers, ce qui a facilité le travail en équipe et l'implémentation des différents modèles.

D'autre part, il a été nécessaire de définir des conventions de nommage qui permettent d'avoir une cohérence et une compatibilité entre les différentes parties du projet, qui ont souvent dû échanger des données à l'intérieur du programme. Cela a aussi permis d'extraire les données traitées et de les utiliser sans conversion vers un format propre à chaque groupe, ce qui constitue un gain en performance et en temps d'exécution, et supprime une source potentielle d'erreurs de programmation. Ces conventions de nommage ont été établies très tôt et ont permis d'éviter de nombreux problèmes en aval.

Néanmoins, un tel logiciel se bornant au simple affichage des trajectoires ne permet pas à lui seul de garantir la sécurité maritime du port de Marseille. En effet, compte tenu du grand nombre de bateaux pouvant être en mouvement à un instant donné dans le port⁹, une anomalie dans la trajectoire d'un bateau (panne, comportement suspect, etc.) sera difficilement repérable à la simple observation de l'affichage du logiciel. Dans une logique d'automatisation du traitement des données, nous avons choisi de concevoir et/ou d'implémenter des modèles de prédiction de trajectoire des bateaux. À cela, nous avons jugé intéressant d'ajouter un modèle de gestion des perturbations dans le trafic (obstacle, croisement, etc), notamment par le calcul d'itinéraires de contournement. Un logiciel comportant ces fonctionnalités permettrait donc de repérer seul les anomalies de comportement et de proposer de manière autonome des corrections de trajectoires.

Toutefois, le logiciel développé fait office de prototype. Les améliorations possibles restent nombreuses. L'objectif à terme serait d'exploiter les données envoyées en temps réel par les capteurs. Les modèles que nous avons implantés ont tous leurs limites, comme nous le verrons ultérieurement. Cependant, cette version permet d'évaluer la faisabilité et les exigences d'un projet futur, et par ailleurs de voir si les modèles développés sont applicables et efficaces en pratique et satisfont les opérateurs du port.

3 Préparation de la zone de travail

3.1 Définition de normes de travail communes au groupe

Pour améliorer la coordination entre les différents groupes, il a fallu choisir des données et des méthodes de travail communes, notamment le système de repérage nécessaire pour localiser un bateau sur le port.

3.2 Choix du système de coordonnées approprié

Les senseurs, autrement dit les radars, sonars et autres systèmes de mesure et de détection installés dans le port, fournissent des données sous la forme d'un couple latitude-longitude. Toutefois, on préfère convertir ces coordonnées

⁹. Le port de Fos compte plus de 5000 accostages de bateaux commerciaux par an, soit une moyenne de 14 accostages par jour, sans parler des bateaux au mouillage et des plus petits bateaux. En tout, une centaine de bateau circule dans l'enceinte du port (navires commerciaux, navires pilotes, etc) et en été ce nombre peut monter jusqu'à 500 - Source : Pierre Deroi, *op.cit*

géographiques en coordonnées de projection, qui s'expriment en kilomètres. Le choix d'un système métrique offre deux avantages majeurs pour l'exploitation des données :

- Le système de projection est décimal, contrairement à la latitude et à la longitude qui s'expriment en degrés, minutes et secondes, soit en système hexadécimal, lequel s'avère relativement difficile à manipuler dans les calculs, et qui risque de poser des problèmes de conversion.
- Le système de projection est rectangulaire, mesuré en kilomètres. Il permet notamment de calculer des distances approximatives entre deux points de la surface de la Terre. Cette approximation est légitime dans le cas que nous étudions car nous travaillons dans une zone restreinte de quelques dizaines de kilomètres.

On choisit d'utiliser la projection de *Mercator* qui fournit une très bonne approximation des distances à cette latitude. Cette conversion est faite dès l'extraction des données de la base de données afin de permettre à chacun d'exploiter les mesures déjà converties, évitant ainsi de perdre inutilement du temps en convertissant à chaque utilisation les coordonnées géographiques en coordonnées métriques.

3.2.1 Définition d'une zone de travail commune à tout le groupe

Le système commun de coordonnées étant fixé, il semble nécessaire de délimiter la zone que nous allons étudier. Nous définissons par ailleurs sur la carte une origine commune à l'ensemble du groupe. La zone d'étude est la suivante :



FIGURE 3 – Zone commune de travail : Le port de Marseille

Deux raisons majeures nous permettent d'expliquer ce choix :

- Les eaux se trouvant au-delà de cette carte ne se situent plus dans les limites administratives du port, ce qui induit que les autorités portuaires n'y ont aucun moyen d'action. Il n'est donc pas utile de s'intéresser aux bateaux se trouvant en dehors de ces limites.
- Les senseurs ayant une portée limitée¹⁰, ils ne parviennent généralement plus à détecter les bateaux en dehors de la carte ci-dessus.

10. Bien que les radars peuvent détecter des bateaux à des dizaines de kilomètres de distance

3.2.2 Création d'états : discréétisation de la carte

Les données normalisées précédemment peuvent être exploitées de la manière suivante : on choisit de discréétiser les coordonnées des bateaux en divisant la carte en un certain nombre de cases, que nous avons choisies avec une dimension de 250 mètres de côté, afin de répondre aux attentes des groupes traitant du modèle statistique et de l'évitement de collisions¹¹.

Cette discréétisation permet également de simplifier les calculs, en considérant alors qu'un bateau ne peut se trouver sur la carte que dans un nombre fini d'états possibles : il n'y a plus une infinité de positions dans lesquelles un bateau peut se trouver. Pour simplifier, on dira que le bateau est désormais repérable sur la carte en coordonnées dites "bataille navale".

3.3 Exploiter les données pour assurer la surveillance dans le port

3.3.1 Définition des zones

Dans le but d'assurer la surveillance et la sécurité du port, on définit des zones particulières au niveau de la carte. On peut le faire grâce aux coordonnées des bouées présentes dans le bassin, que l'on peut trouver dans un arrêté préfectoral relatif au Port de Marseille¹². La carte suivante représente les différentes zones :

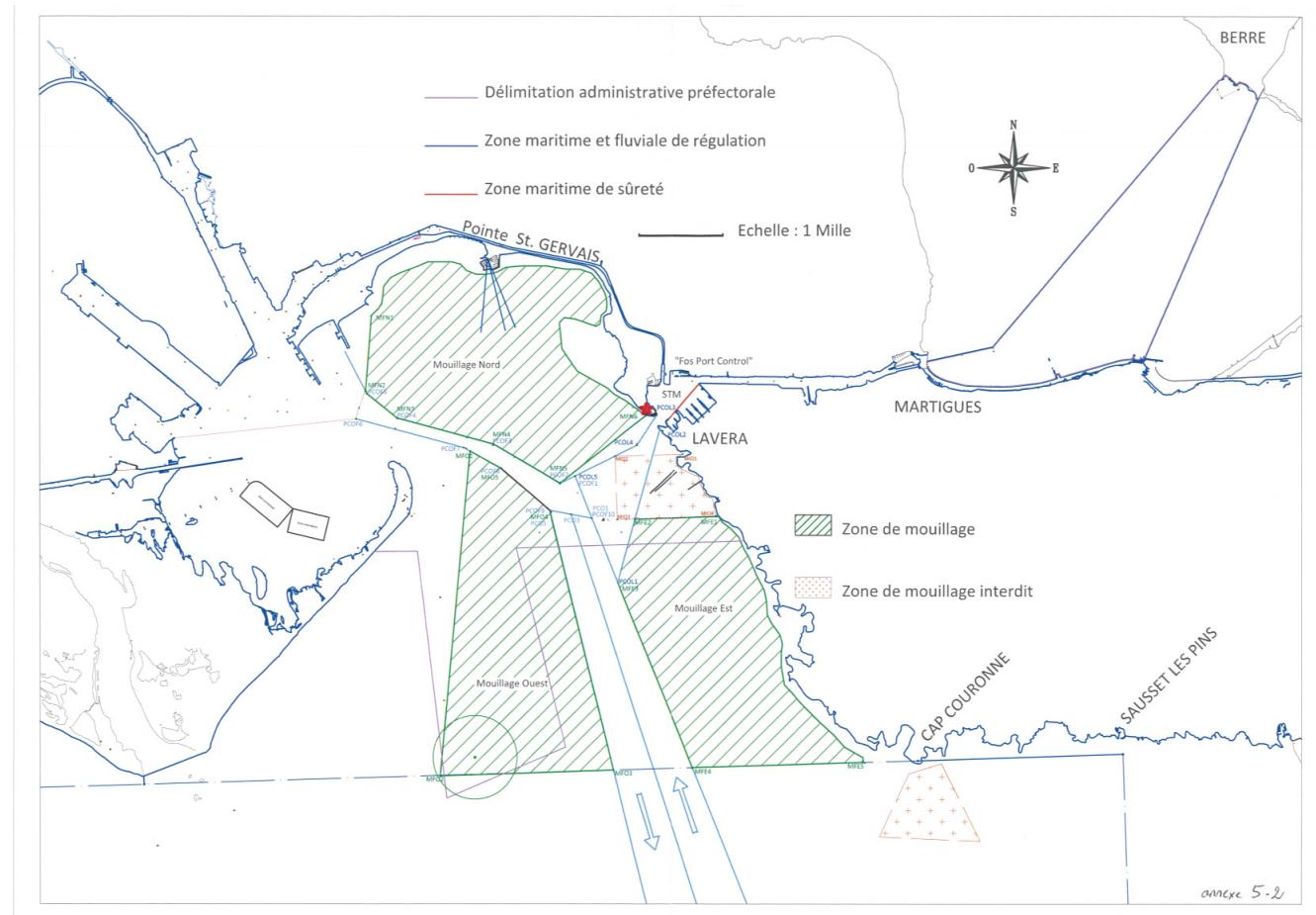


FIGURE 4 – Zones définies sur le Port de Marseille

11. En choisissant de telles cases, on s'assure qu'entre deux pas de temps un bateau se déplace au maximum de 250m. Les bateaux ne peuvent donc pas sauter de case.

12. Arrêté préfectoral n° 2012016-002 du 16 janvier 2012

On définit ainsi :

- Trois zones de mouillage (Nord, Est et Ouest) permettant aux bateaux de stationner en attendant une autorisation d'entrée vers les docks.
- Une zone de mouillage interdite.
- Un chenal montant et un chenal descendant, permettant l'accès des bateaux aux différentes zones stratégiques du port. Il est creusé avec une profondeur suffisante pour permettre le passage des plus gros bateaux tels que les supertankers ou les cargos.
- Des accès particuliers (sortes de routes navales).
- Des zones sécurisées correspondant aux sites sensibles du port, dans lesquels se trouvent notamment les usines pétrochimiques et les raffineries de pétrole.

3.3.2 Application à la sécurité du port

Un contrôle des autorisations. Puisque les bateaux doivent posséder des autorisations particulières pour se trouver dans les zones de sécurité, la définition précédente de ces zones permet de contrôler en temps réel si les bateaux sont en situation régulière ou non. On peut éventuellement afficher des alertes lorsque ce n'est pas le cas pour les intercepter. Le programme et l'interface graphique permettent ce contrôle en temps réel.

Anticiper le contrôle des autorisations. Pour assurer une sécurité du port bien plus efficace, il suffit d'utiliser les résultats obtenus par les groupes travaillant sur la prédiction de trajectoire. En effet, si les statistiques annoncent qu'un bateau va probablement se rendre dans une zone à accès sécurisé, les programmes peuvent vérifier à l'avance si ce bateau possède les bonnes autorisations. Si ce n'est pas le cas, il suffit de surveiller ce bateau particulier de manière accrue, et si sa trajectoire se poursuit comme prévu, il pourra être intercepté dès son entrée dans la zone.

4 Bases de données

Une fois la zone d'étude bien définie, il a fallu ensuite exploiter les données disponibles pour les rendre utilisables par les différents groupes.

Présentation

Pour étudier ou tracer la trajectoire d'un bateau, certaines données sont nécessaires, comme : sa longitude, sa latitude, sa vitesse, son cap, etc.

Comme expliqué précédemment, les bateaux commerciaux sont tenus au-dessus d'une certaine taille d'avoir une **balise AIS** qui envoie régulièrement un certain nombre d'informations par ondes radio. Ces données renseignent ainsi les autorités portuaires sur la position du bateau, sa nature, le type de sa cargaison, le nombre de personnes à bord, etc. On identifie alors le bateau avec son **MMSI**¹³.

4.1 Nécessité d'utiliser une base de données

4.1.1 Données disponibles

Les fichiers sont des données issues des senseurs du projet SECMAR¹⁴. Les fichiers disponibles étaient de plusieurs types : les données enregistrées pour chacun des senseurs, les données AIS et les données fusionnées pour former des pistes de déplacements des bateaux. Toutefois, leur format ne permettait pas une utilisation simple. La construction d'une base de données dans laquelle il est possible d'exporter toutes les informations contenues dans ces fichiers permet d'avoir à la fois un enregistrement clair et propre des données strictement nécessaires ainsi que la possibilité d'y accéder rapidement.

13. cf Partie I.1

14. Ces données ont été enregistrées lors des phases de tests et d'intégration des différents sous-systèmes SECMAR, la qualité des données n'est pas optimale et très inégale.

4.1.2 Utilité de la base de données

L'utilisation d'un **langage à requête structurée** nous (*SQL*) autorise à construire des requêtes extrêmement claires et précises pour parvenir à sélectionner uniquement les données pertinentes, comme par exemple la liste des positions pour un bateau dont on connaît le MMSI. La flexibilité du langage SQL permet ainsi de gagner du temps, et de fournir un outil simple et efficace pour exploiter les données.

4.2 Crédation de la base de données

Une partie du programme Java a donc été spécialement développée pour permettre d'exporter l'ensemble des enregistrements SECMAR disponibles dans une base de données *MySQL* créée pour le projet. En raison du nombre conséquent de données disponibles (plus de 35 Go), seuls quelques jours d'enregistrements ont été exportés dans la base de données.

Une base de données relationnelle contient plusieurs tables qui peuvent interagir les unes avec les autres. Nous avons défini et créé trois tables pour nos données :

- Une première table contient l'ensemble des bateaux ayant circulé les jours étudiés dans le port. Chaque bateau est identifié par son MMSI, le nom étant rarement disponible¹⁵. Cette table contient en outre pour chaque bateau sa longueur, sa largeur, son heure d'arrivée estimée et la position de l'antenne AIS sur le pont.
- Une autre table contient toutes les positions des bateaux avec la longitude, la latitude et la date de la mesure. Le MMSI permet de relier une position à un bateau donné. Cette table contient, en plus, des données comme la vitesse par rapport au sol, le cap, ainsi qu'un couple (x,y) fourni par le programme de conversion en coordonnée métrique¹⁶.
- Enfin une dernière table contient la correspondance entre le type de chaque bateau (qui est un nombre entier compris entre 20 et 99)¹⁷ et sa fonction. Cela permet de savoir si c'est un pétrolier, un chimiquier, un gazier, etc. Cette table est nécessaire pour pouvoir distinguer les différents types de bateaux sur l'interface graphique.

MMSI	NOM	LONGUEUR	LARGEUR	DATE ARRIVÉE	TYPE
227571140	ESCANDAIL	16	5	2011-01-11 10:55:00	0
227016900	MARSEILLAIS 6	35	12	2011-01-11 10:54:55	45
...
304756000	RITA SIBUM	132	19	2011-01-11 10:55:00	22

MMSI	DATE MESURE	LONGITUDE	LATITUDE	VITESSE	CAP
227016900	2011-01-11 10:55:03	4.755516	43.42365	0	5
304756000	2011-01-11 10:55:00	4.916633	43.41111	52.6	1
...
227016900	2011-01-11 10:55:10	4.8555	43.405	0	1

FIGURE 5 – Schéma des différents tableaux présents dans la base de données

La constitution de la base de données est une étape assez longue. La piste d'un bateau peut contenir jusqu'à 15.000 positions en une seule journée et l'on recense une centaine de bateaux présents dans le port chaque jour. De plus, les communications entre Java¹⁸ et la base de données ne sont pas immédiates, ce qui induit un temps de

15. Les données AIS sont en effet rarement complètes, voire présentent parfois des informations erronées

16. cf partie précédente.

17. Cette table de correspondance est internationale, et est disponible sur Internet

18. Notons que les enregistrements contenaient des données serialisées à partir de Java.

communication supplémentaire. Pour donner un ordre de grandeur, l'exportation d'un seul fichier contenant 10.000 positions dans la base de données prend entre deux et trois minutes avec un ordinateur récent.

4.3 Utilisation de la base de données

4.3.1 Récupération des données

Une fois la base de données créée, le programme peut importer les données nécessaires aux différents algorithmes. L'interface **JDBC** (*Java Database Connectivity*) de Java permet au programme Java et à la base de données *MySQL* de communiquer. Une fonction permettant de récupérer l'ensemble des bateaux sous forme d'un tableau à partir de la base de données a ainsi été créée, et une autre fonction récupère les positions correspondantes dans un autre tableau.

4.3.2 Normalisation des positions

Une fonction de **normalisation** a dû aussi être implémentée afin d'avoir un pas de temps constant lors de l'affichage. En effet, chaque capteur envoie ses données, indépendamment des autres capteurs, avec un pas de temps différent. De manière à *synchroniser* les données à traiter, on doit dès lors introduire un temps initial puis une succession de temps séparés par une durée constante¹⁹ avec à chaque fois une position pour chacun des bateaux.

Pour construire un tableau de positions normalisées, il faut tout d'abord fixer un pas de temps qui définit l'espace-temps temporel que l'on veut avoir entre les différentes mesures. Ensuite, il faut calculer des valeurs pour les dates ainsi définies. Pour cela, on opère une interpolation linéaire entre les deux mesures les plus proches de la date voulue et qui l'encadrent. Par exemple, si on a les mesures du tableau ci-dessous et que l'on veut connaître la position du bateau de **MMSI 227170110** à $t = 15 : 07 : 30$, on utilise les mesures les plus proches encadrant cette date, soit celle de $15 : 07 : 23$ (*mesure 1*) et celle de $15 : 07 : 58$ (*mesure 2*). Pour calculer X, on calcule l'écart Δ entre les deux positions : $\Delta = x_2 - x_1$. On calcule le coefficient d'interpolation $c = \frac{t - t_1}{t_2 - t_1}$. On ajoute à la position x_1 la quantité : $\Delta * c$, et on trouve ainsi la valeur de x à $t : x = x_1 + c * \Delta$.

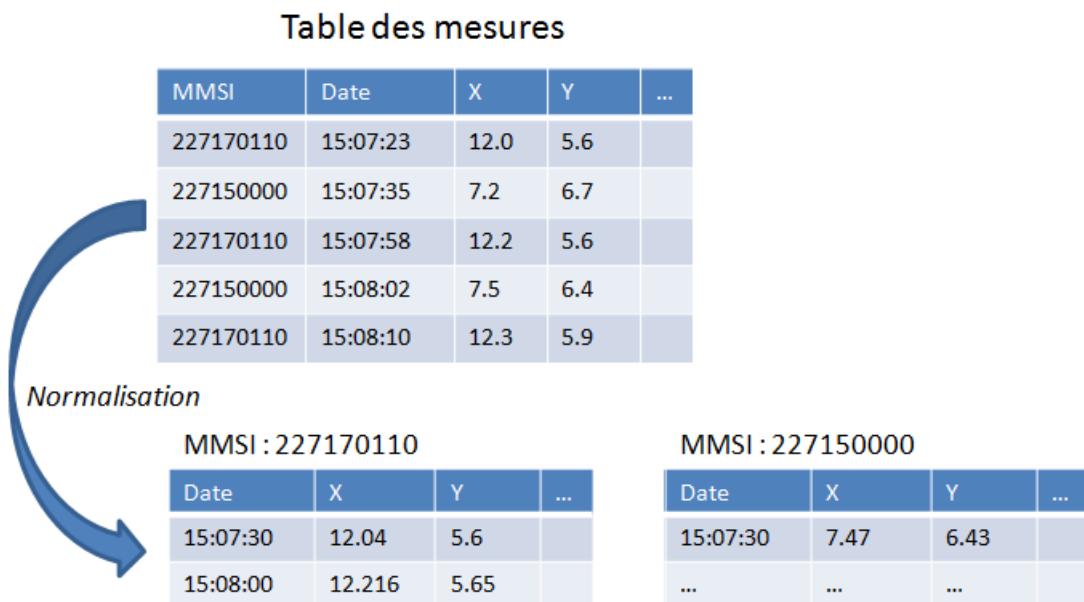


FIGURE 6 – Principe de la fonction de normalisation.

Les positions étant extrapolées, le risque est de voir apparaître des positions impossibles²⁰, notamment dans les chenaux et les zones étroites. Pour éviter ce cas de figure, si le délai entre deux positions est trop important, on n'affiche pas la trajectoire du bateau entre ces deux points.

19. Dans le cadre du projet, nous avons choisi un pas de temps de 30 secondes.

20. Certains bateaux apparaissaient sur la terre ferme ...

4.4 Défauts et possibilités d'amélioration

Un premier défaut déjà mentionné est le temps d'exécution important que prend l'exportation des données dans la base. D'autre part, le temps de chargement initial du programme qui récupère les positions des bateaux peut durer de une à deux minutes avec un ordinateur récent. Cependant, dans le cas d'une implémentation réelle, cela ne poserait pas de problème. En effet les données des senseurs arriveraient en temps réel et seraient directement utilisées par le programme.

Un autre défaut vient du fait que seules les données AIS ont été exploitées dans notre logiciel, car il est difficile d'associer les données d'un senseur à un bateau précis. En effet, les données collectées par les senseurs sont complexes à exploiter : un senseur n'envoie que des pistes (c'est-à-dire des bouts de trajectoires repérées) mais ne les associe pas à un bateau, ce qui fait qu'on ne peut pas suivre un bateau donné. Pour obtenir sa trajectoire réelle, il faudrait mettre bout à bout les différentes pistes : c'est le rôle de la *fusion*. Or, un radar ou un sonar détecte toutes sortes de choses, des plus gros tankers jusqu'au moindre débris, détection qui est de plus rendue difficile par le bruit électromagnétique. Il est alors assez difficile d'associer un objet détecté à un seul bateau. Dans certains cas, un bateau peut même être détecté comme deux entités séparées (notamment les supertankers et autres porte-containers). Exploiter les données des senseurs pour pouvoir les exporter de manière pertinente dans une base de données aurait donc demandé beaucoup de temps.

La non-utilisation des fichiers senseurs exclut d'emblée un certain nombre de données, et peut poser problème dans le cas d'un système réel pour détecter l'ensemble des menaces et des dangers.

5 Prédiction des trajectoires grâce aux Chaînes de Markov Cachés

Maintenant que les bases du programme ont été construites avec le système de repérage et la base de données, on peut passer à l'implémentation des algorithmes principaux.

5.1 Présentations des chaînes de Markov

Un **modèle de Markov caché** (*MMC* ou *HMM*) est un automate probabiliste à états finis. Le système possède un nombre fini d'états, et il passe à chaque instant d'un état à un autre selon des probabilités qui ne dépendent que de l'état présent. Ces probabilités sont regroupées dans une matrice de changement d'état. Toutefois on ne s'intéresse pas à l'état du système, que l'on ignore à priori (c'est ce qui justifie le terme cachée), mais à une observation, dont la loi de probabilité d'émission est fixée selon l'état du système lors de l'émission.

Un exemple d'un tel système serait un ensemble de trois boîtes opaques et identiques contenant des boules de couleur. On part d'une boîte au départ, on tire une boule, on observe sa couleur, puis on remet la boule dans la boîte. On passe ensuite à une autre boîte selon une probabilité définie à l'avance et on recommence. Les états correspondent donc aux boîtes et les observations sont la couleur de la boule tirée. La séquence de couleur des boules observées est appelée séquence d'observation. Il n'est cependant pas possible de connaître la séquence des états par lesquels on est passé juste en connaissant la couleur des boules tirées, d'où le nom de caché.

Les paramètres d'un modèle de Markov caché sont la *matrice de changement d'état* (le coefficient (i,j) est la probabilité d'être à l'instant $t+1$ à l'état j si on est à l'état i lors de l'instant t), un *vecteur* définissant les probabilités de présence initiales, et les *lois de probabilités d'émission* pour chaque état.

Lorsque tous ces paramètres sont définis, il est possible de calculer la probabilité d'occurrence d'une séquence d'observation donnée (en ignorant les états par lesquels transite l'automate). Ces paramètres peuvent aussi être ajustés selon l'*algorithme de Baum-Welch* pour qu'une séquence d'observation, ou, plus intéressant, une suite de séquence d'observation, soit rendue la plus probable possible.

5.2 Adaptation à la prédiction de trajectoire

Après que cet outil mathématiques nous a été présenté, nous nous sommes interrogés sur la possibilité de l'utiliser dans notre projet pour évaluer les trajectoires des bateaux. En effet l'*algorithme de Baum-Welch* permet de créer des modèles optimaux pour une séquence de trajectoire donnée. Ainsi en appliquant l'algorithme à partir de bateaux qui vont tous au même endroit, les trajectoires les plus probables pour l'automate obtenu seront les trajectoires qui vont à cet endroit. Nous avons donc créé trois automates à partir des trajectoires caractéristiques allant vers trois destinations différentes. Cette phase s'appelle l'**apprentissage**. Elle doit être effectuée une fois au début du projet et ensuite le programme s'appuie uniquement sur ses résultats. Une fois les trois automates obtenus, à partir d'un bout de trajectoire donné, en comparant la probabilité que chaque automate soit à l'origine de cette observation, on peut connaître sa destination la plus probable.

Les états que nous avons choisis pour nos modèles de Markov sont des zones du port modélisées par des ellipses. Les séquences d'observations sont les positions successives du bateau. La probabilité de présence d'un bateau dans un zone, c'est à dire la probabilité d'une observation donnée dans la zone, était modélisé par une gaussienne s'appuyant sur l'ellipse représentant la zone.

Techniquement les ellipses étaient représentées dans nos modèles de Markov par la position de leur centre et leur matrice de covariance, qui sont suffisant pour définir la gaussienne, et sur lesquels on peut appliquer l'*algorithme de Baum-Welch*. En effet une des difficultés a été d'adapter cet algorithme à des observations qui sont continues (positions de bateau) et non discrètes comme dans l'exemple de la partie précédente.

Pour l'apprentissage, nous avons observé grâce à l'interface graphique une journée de données dans le port, et nous avons choisi les bateaux ayant des trajectoires caractéristiques pour chaque destination. Il a ensuite fallu récupérer dans la base de donnée le bout de trajectoire que nous avions besoin. Une des difficulté que nous avons eu par la suite a été de normaliser les trajectoires. En effet, les données que nous avions dans la base de donnée étaient à intervalle de temps fixé (toutes les 30 secondes).

Or nos modèles ne prennent pas en compte la dimension temporelle du parcours, ils ne s'appuient que sur la trajectoire. Ainsi deux bateaux ayant exactement la même trajectoire mais ayant des vitesses de parcours très différentes donneront des automates différents (par exemple pour le plus rapide la probabilité d'un changement d'état sera beaucoup plus grande), et donc l'apprentissage ne sera pas cohérent. Il nous a donc fallu faire une normalisation spatiale des trajectoires, par exemple en prenant une position tous les 500 mètres. Il a ensuite fallu définir les états initiaux des modèles de Markov, que nous avons pris selon une trajectoire probable des bateaux pour limiter le nombre d'itération de l'*algorithme de Baum-Welch*. Voici les zones initiales que nous avons choisi :



FIGURE 7 – Initialisation des zones Ouest



FIGURE 8 – Initialisation des zones Est

Et voici le résultat de l'apprentissage :

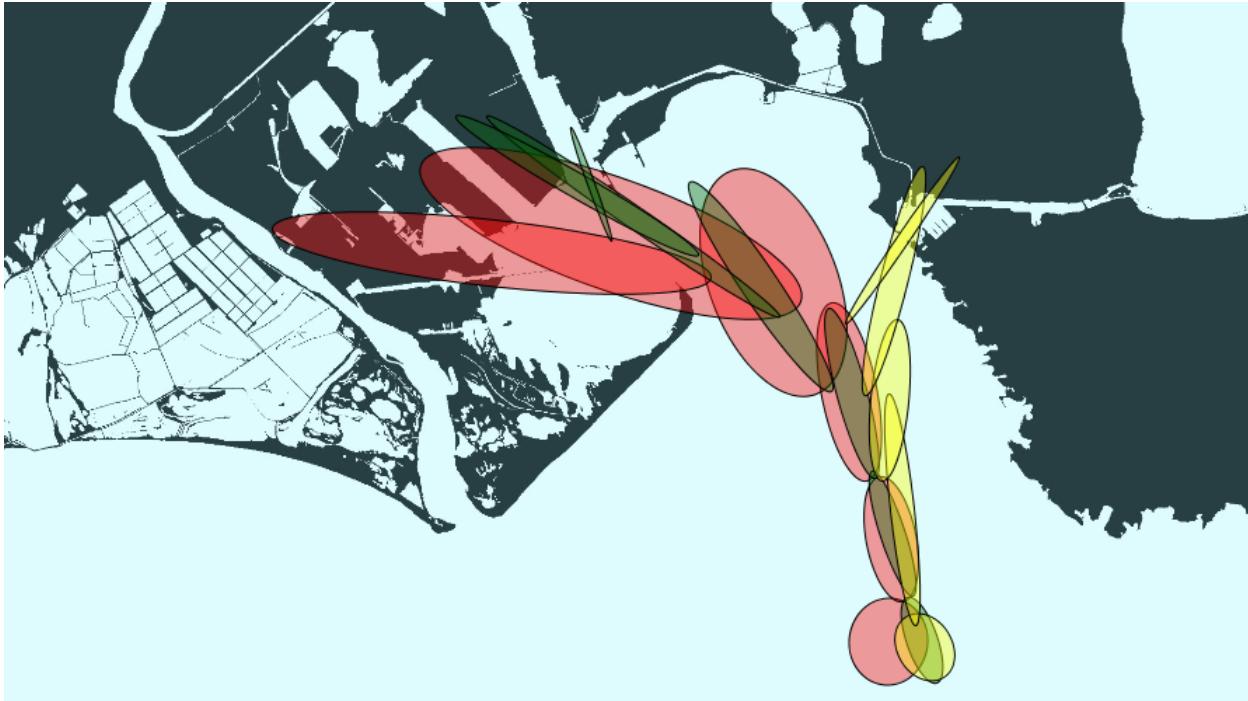


FIGURE 9 – Résultat de l'apprentissage

5.3 Adaptation au projet

Les modèles de Markov cachés nous ont donc permis d'évaluer la destination d'un bateau à partir d'un bout de sa trajectoire, mais aussi de connaître sa trajectoire la plus probable car comme on le voit sur le schéma précédent les zones modifiées par l'*algorithme de Baum-Welch* s'alignent sur la trajectoire généralement prise par les bateaux allant au même endroit que lui.

Ceci peut donc permettre de détecter de manière automatique différentes anomalies. En effet si un bateau n'a pas d'AIS, cela peut permettre de savoir où il va, et donc de savoir s'il va rentrer ou non dans une zone interdite pour lui. De plus si un bateau a un AIS, il émet au port sa destination, donc nos modèles permettent de savoir par où il devrait passer de manière automatique et donc de lever une alerte s'il s'écarte trop d'une trajectoire normale. Enfin, nos modèles pouvant prédire les trajectoires de différents bateaux, ils pourraient aussi lever une alerte si une collision est possible.

Toutes ces choses pourraient bien sûr être effectuées de manière plus efficace par des opérateurs s'ils pouvaient surveiller en permanence tous les bateaux, mais l'avantage de ce programme est qu'il peut détecter des anomalies de manière automatique, et permet donc idéalement de n'avoir qu'une seule personne qui va prendre en compte toutes les alertes lancées par le programme, éliminer les fausses alertes et prévenir en cas de danger les personnes concernées. Cela permet donc de réduire les effectifs.

6 Prédiction des trajectoires selon une approche statistique

6.1 Introduction

Dans cette partie, nous présentons une méthode alternative de prédiction de trajectoire que nous avons entièrement développée et implémentée. Il s'agit en fait d'une méthode statistique qui consiste à analyser les trajectoires suivies par les bateaux antérieurement présents dans le port. Ces trajectoires sont dénombrées et le nombre d'occurrences d'un même comportement est traduit en terme de probabilités afin de prédire la trajectoire d'un autre bateau en fonction de sa trajectoire passée. Concrètement, si parmi les trajectoires analysées, un grand nombre de bateaux ont suivi un même chemin (typiquement, lorsque les bateaux suivent des chenaux), alors, lorsqu'un bateau commence à suivre ce même chemin, notre programme pourra prévoir qu'il va continuer à le suivre jusqu'au bout.

Notre programme utilise un arbre de probabilités dont chaque noeud, correspondant à une trajectoire, est pondéré. Plus le poids est grand, plus la trajectoire correspondante est empruntée par des bateaux. Cet arbre est construit à partir de trajectoires connues, en l'occurrence les trajectoires des bateaux qui sont déjà passés dans le port. Puis, lorsqu'un bateau se déplace dans le port, en utilisant la trajectoire qu'il vient d'emprunter ainsi que l'arbre de probabilités, le programme peut déterminer à la fois la trajectoire la plus probable qu'il est susceptible de suivre, ainsi que la probabilité de présence du bateau dans une zone donnée de la carte à un instant ultérieur.

À la différence de la méthode de prédiction utilisant les modèles de Markov cachés, la méthode statistique est autonome. En effet, elle ne nécessite pas de sélectionner des trajectoires de bateau optimales lors de la phase d'apprentissage : les trajectoires les plus marginales, rarement empruntées par les bateaux, n'auront que peu d'influence lors de la prédiction de trajectoire. Cela présente l'avantage de ne pas nécessiter d'intervention humaine durant cette phase d'apprentissage. Même après avoir été installé sur le port, le système pourra donc de lui-même continuer à améliorer sa capacité de prédiction en analysant continuellement les nouvelles trajectoires, et ce de manière totalement autonome. En revanche, cela implique de disposer de suffisamment de données sur les trajectoires des bateaux au moment de l'implantation du système sur le site afin que les premières prédictions soient pertinentes.

6.2 Construction de l'arbre de probabilités

L'arbre recense toutes les trajectoires des bateaux qui ont été utilisées pour le construire. Il est construit en y ajoutant chaque trajectoire une par une. Avant d'être traitée, une trajectoire de bateau est discrétisée spatialement d'une part, c'est-à-dire que la carte est maillée d'un quadrillage et la position d'un bateau est représentée par les coordonnées de la case dans laquelle il se trouve, et temporellement d'autre part, c'est-à-dire que l'on utilise les positions des bateaux à intervalle de temps régulier, en l'occurrence toutes les trente secondes. Une trajectoire se trouve ainsi définie par une suite de coordonnées. Dans l'exemple ci-dessous, cette suite de coordonnées est : A4, B3, C2, D2, E2.

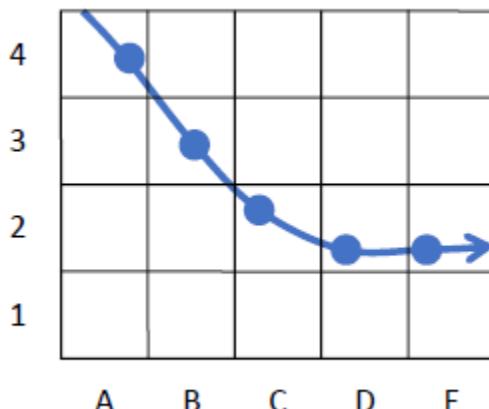


FIGURE 10 – Représentation d'une trajectoire

La trajectoire est ensuite insérée dans l'arbre de probabilités en partant de la racine. Ci-dessous sont représentés les arbres successivement obtenus en ajoutant les trajectoires [A4, B3, C2, D2, E2], [B3, C4] et [A4, B3, B2] en partant de l'arbre vide. Dans chaque noeud apparaît les coordonnées de la dernière case du chemin ainsi que le poids du nœud.

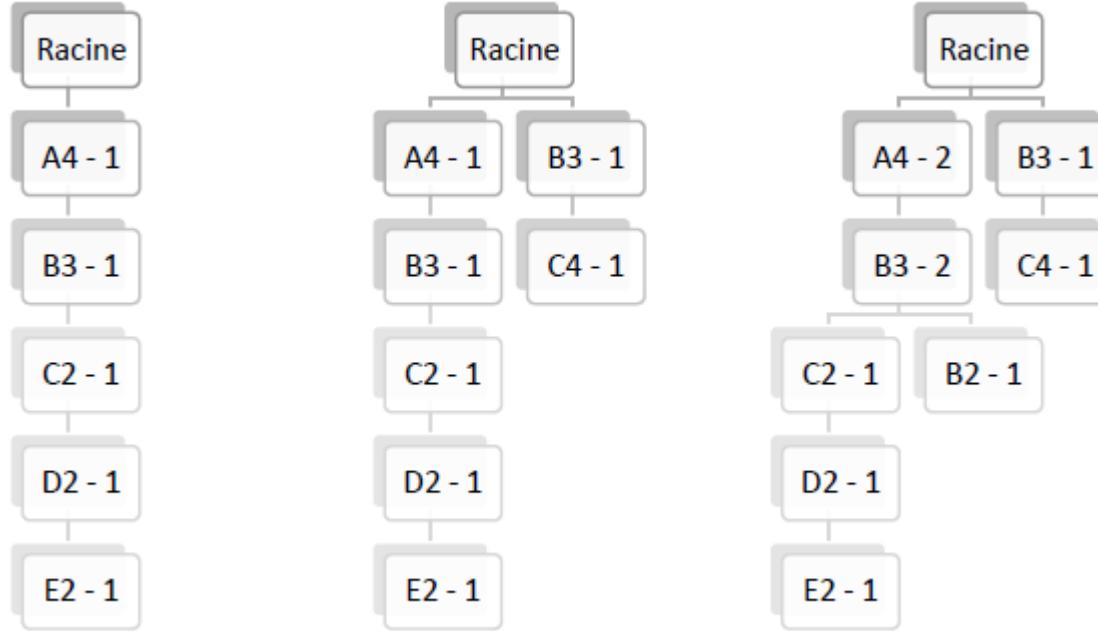


FIGURE 11 – Ajout successif des trajectoires [A4, B3, C2, D2, E2], [B3, C4] et [A4, B3, B2] à l'arbre vide.

Une fois l'arbre construit, il contient toutes les informations nécessaires aux algorithmes de prédiction de trajectoire. Les données nécessaires pour construire l'arbre ne sont donc plus utilisées.

6.3 Prédiction de la trajectoire la plus probable

Nous allons décrire notre algorithme de prédiction de la trajectoire la plus probable en l'illustrant sur un exemple concret. On suppose que l'arbre de probabilités a été construit (il est représenté ci-dessous) et que l'on cherche à déterminer la trajectoire future d'un bateau qui se déplace dans le port et dont la trajectoire passée, discrétisée de la même manière que précédemment, est [B4, C4].

Pour cela, on commence par récupérer le sous-arbre correspondant à cette trajectoire [B4, C4], en partant de la racine de l'arbre et en choisissant à chaque fois le fils correspondant au trajet suivi. Le sous-arbre ainsi récupéré est représenté en bleu ci-dessous. Enfin, on obtient la trajectoire la plus probable en parcourant l'arbre en choisissant à chaque fois le fils ayant le poids le plus élevé, ce qui nous donne ici [D4, E4]. La trajectoire complète comprenant la portion déjà empruntée et celle prédictée par l'algorithme est donc : [B4, C4, D4, E4].

6.4 Probabilité de présence

Nous avons également implémenté un second algorithme qui, comme le précédent, prend en paramètre la trajectoire passée d'un bateau, et renvoie un tableau contenant les probabilités de présence du bateau dans les cases du maillage aux instants ultérieurs (*id est* au bout de trente secondes, au bout d'une minute, etc.). Pour cela, comme précédemment, le sous-arbre correspondant au trajet passé est récupéré. On raisonne ensuite ligne par ligne, puisque chaque ligne correspond à un instant donné et deux lignes successives correspondent à deux instants séparés de trente secondes. L'algorithme dénombre alors les positions possibles et calcule leur probabilité en sommant les poids des nœuds correspondants.

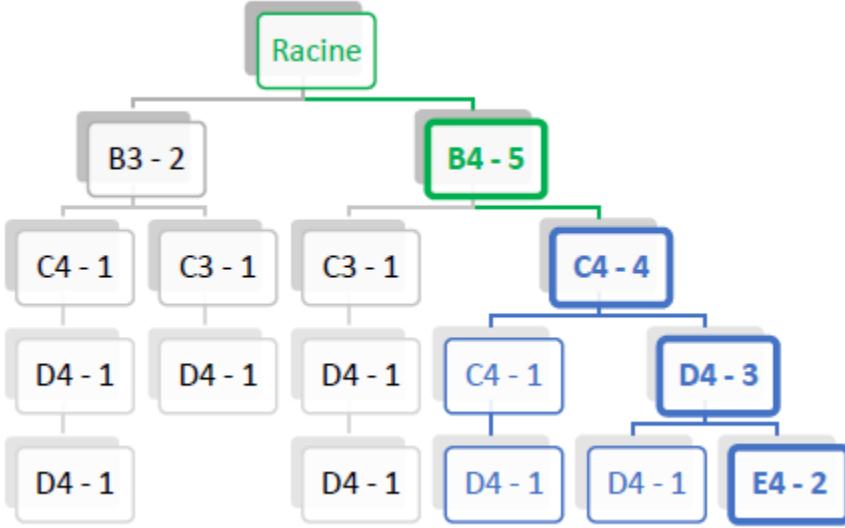


FIGURE 12 – Illustration de l’algorithme de prédiction de trajectoire

En reprenant l’arbre et le trajet utilisés précédemment, on obtient les résultats suivants :

Case	.	C4	D4	E4	.
t+30s	.	1	3	0	.
t+1min	.	0	2	2	.

TABLE 1 – Poids des positions

Case	.	C4	D4	E4	.
t+30s	.	0.25	0.75	0	.
t+1min	.	0	0.5	0.5	.

TABLE 2 – Probabilité des positions

Le résultat est ensuite donné graphiquement, en représentant successivement les probabilités des positions par une échelle de couleur (typiquement, une couleur est d’autant plus foncée que la probabilité est grande).

6.5 Résultats

Notre modèle s’est avéré très efficace pour prédire les trajectoires à court terme. En revanche, la prédiction à long terme est peu efficace car elle nécessite de construire un arbre plus gros, donc plus lourd en mémoire et plus long à construire. En effet, la hauteur de l’arbre est liée à la durée maximale de prédiction : si on note h la hauteur de l’arbre, Δt l’incrément de temps choisi (30 secondes dans notre cas) et T la durée de la trajectoire passée utilisée pour la prédiction de la trajectoire, la durée maximale de prédiction sera $h\Delta t - T$. On pourrait tout de fois résoudre ce problème en utilisant un système de données adapté, c’est-à-dire en enregistrant par exemple l’arbre sur un disque dur de telle sorte qu’on puisse le lire sans devoir préalablement le charger en mémoire. En somme, notre modèle est donc complémentaire au modèle de Markov, qui lui prédit les trajectoires à long terme.

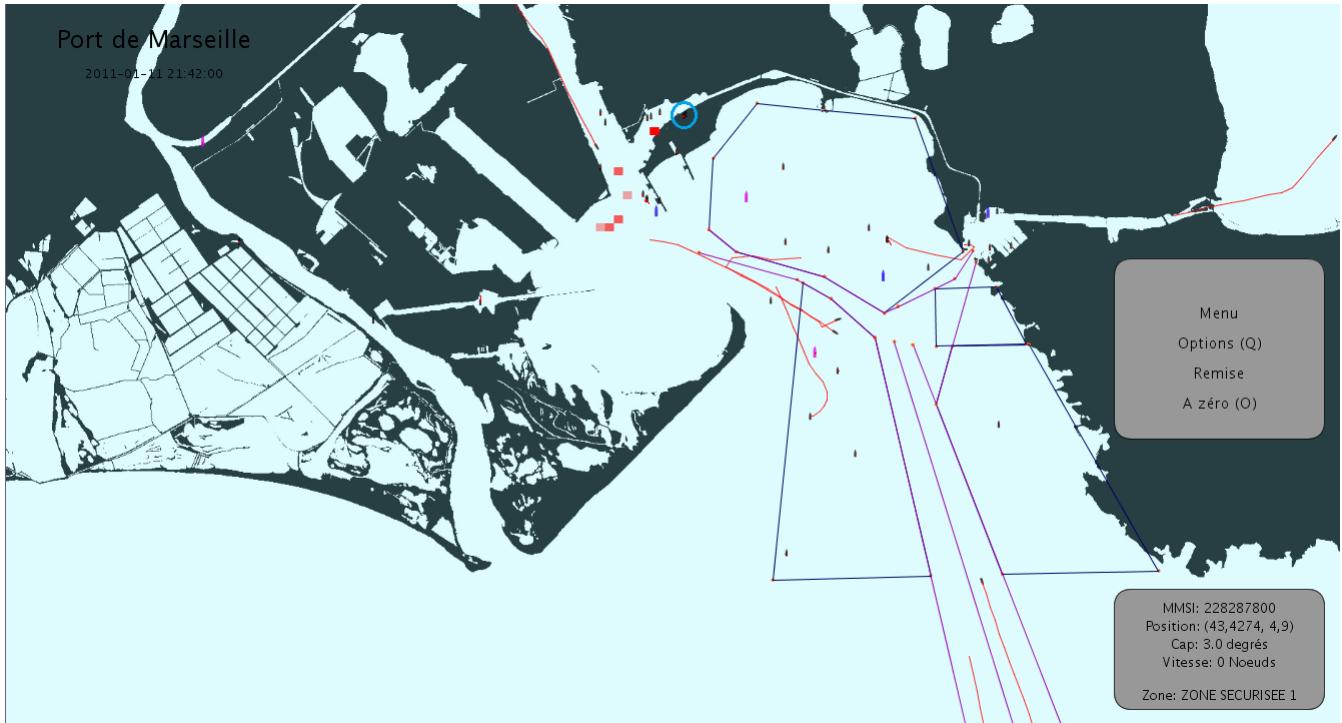


FIGURE 13 – Résultat de la prédiction faite sur le bateau entouré à $t + 3$ min

7 Déviation des trajectoires

Les algorithmes précédents servent avant tout à prédire la trajectoire d'un bateau. Si un problème est détecté (collision, naufrage), le programme devrait être en mesure de proposer une déviation automatique des trajectoires pour tous les bateaux présents sur le port et ainsi éviter une aggravation de la perturbation.

Présentation de la démarche

Lorsqu'un bateau entre dans la zone, les capteurs relèvent sa position. Après un certain nombre de relevés, on applique l'algorithme précédent pour déterminer la trajectoire future la plus probable pour le bateau. Notre but est de savoir comment modifier cette trajectoire pour éviter tous les obstacles qui pourraient se trouver au milieu du chemin.

Ces obstacles peuvent être **statiques**, comme dans le cas d'un bateau au mouillage ou en panne, ou bien **dynamiques**, comme des bateaux eux-mêmes en train d'éviter un obstacle.

L'intérêt de ce système est évident : il s'agit d'éviter les collisions en déterminant des **routes maritimes sécurisées** pour les navires. En choisissant la déviation la plus courte, il permet de limiter les retards en cas de perturbation du trafic.

Cependant, lesdites trajectoires ne sont qu'indicatives : contrairement au transport aérien, les capitaines des navires sont susceptibles de dévier des routes indiquées. Aussi, l'algorithme devra être relancé régulièrement pour tenir compte de ces éventuelles modifications, ainsi que de l'arrivée de nouveaux bateaux dans la zone.

Nous avons fait le choix d'imaginer et de développer une méthode originale de contournement d'obstacles. N'étant pas certains de son aboutissement au début du projet, nous avons implémenté en parallèle l'algorithme de Dijkstra. Ainsi, nous avons assuré que l'un des algorithmes était correct et nous avons pu comparer les deux méthodes a posteriori.

7.1 Manœuvre d'évitement d'obstacles statiques

L'objectif de cette partie est de définir les déviations à suivre pour éviter les obstacles statiques (ou possédant une vitesse négligeable par rapport à celle des navires) tels que des écueils, un navire en panne ou une fuite de produits dangereux.

7.1.1 Définition de la zone impraticable

Lorsqu'un danger est identifié, on définit autour de lui une zone interdite : le ou les navires n'auront pas le droit de s'y rendre.

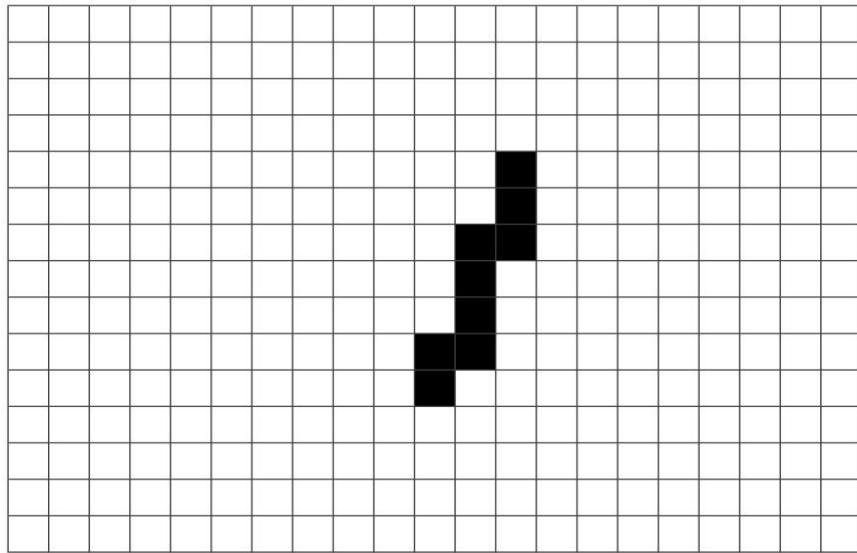


FIGURE 14 – La zone interdite est représentée par des cases noircies

Les zones situées hors des canaux aménagés spécialement pour les navires à grand tirant d'eau (les pétroliers par exemple) leur sont *par défaut* interdites.

7.1.2 Calcul de la déviation par contournement

Les zones interdites étant définies, il faut mettre en place une méthode de déviation.

Si le navire est censé passer par une case interdite, nous étudions par deux programmes de contournement les déviations par bâbord et par tribord de l'obstacle rencontré. En fait, on va longer la zone interdite et tenter de récupérer, derrière l'obstacle, la trajectoire initiale.

Dans le déroulement du processus, il faut tenir compte des obstacles qui ont une étendue infinie à bâbord ou à tribord et assurer la terminaison y compris dans ces cas-là.

On sélectionne la déviation la plus courte des deux, puis on lisse la trajectoire obtenue afin d'éviter au maximum les détours et les changements de cap inutiles.

7.1.3 Une autre approche : l'algorithme de *Dijkstra*

Un algorithme alternatif de contournement d'obstacles repose sur une version optimisée de l'algorithme de *Dijkstra*.

- On cherche à calculer la distance de chaque point de la carte au point initial.
On part du point initial. On le place dans une liste.

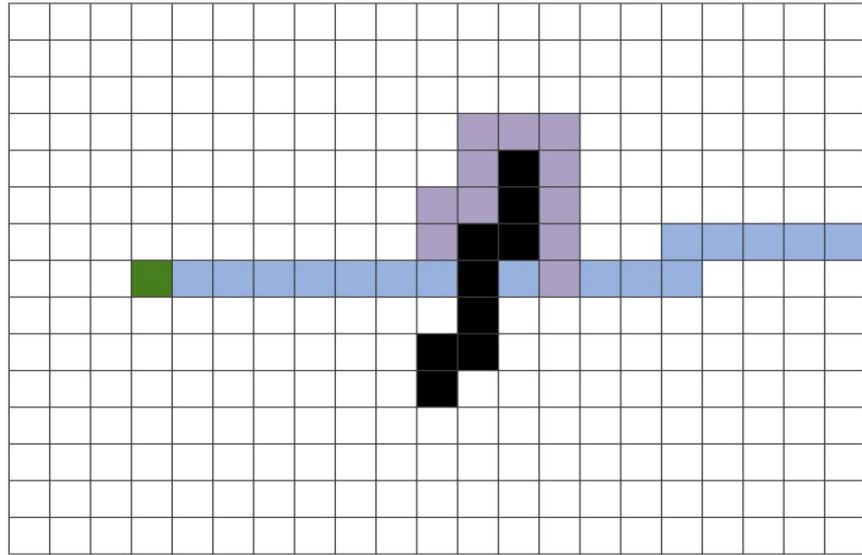


FIGURE 15 – On longe l’obstacle de manière à avoir le chemin le plus court

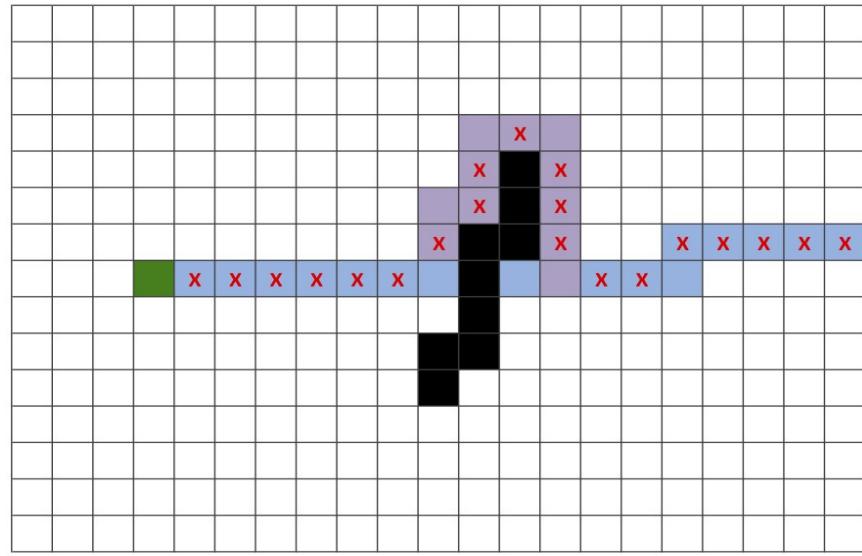


FIGURE 16 – Le lissage permet d’éviter les impasses, les détours et de réduire les changements de cap

- Tant qu’on n’a pas calculé la distance pour le point final :
 - on part du point de la liste qui a la distance la plus faible,
 - on calcule les distances des points qui lui sont adjacents, points que l’on ajoute dans la liste,
 - on supprime le point de la liste.
- On se place sur le point final.
- Tant qu’on n’est pas arrivé au point initial, on se déplace vers la case adjacente la plus proche de ce dernier.

Cette méthode peut générer des trajectoires d’évitement assez éloignées de la trajectoire initiale. Pour éviter cela, on peut appliquer l’algorithme à des sous-traj ectoires, puis recoller et lisser.

7.1.4 Ré-incorporation de la composante temporelle

Comme la position des bateaux est discrétisée, il peut y avoir une répétition de cases d’un instant sur l’autre : selon la vitesse du bateau, il peut avoir besoin de plusieurs pas de temps différents pour traverser une unique case.

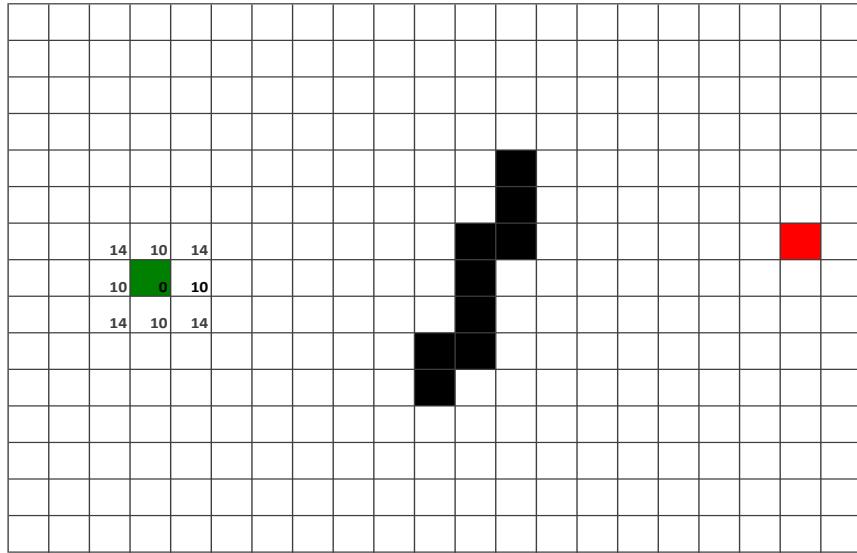


FIGURE 17 – Initialisation du Dijkstra

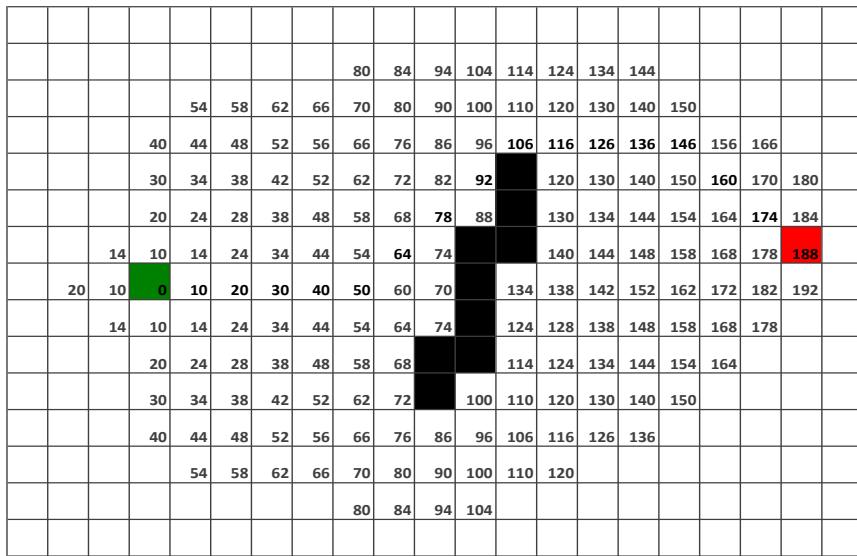


FIGURE 18 – Diffusion des indices

Il se trouve que l'algorithme ci-dessus ne définit qu'une trajectoire de déviation et ne détermine pas la vitesse de parcours.

Pour résoudre ce problème, il faut donc ré-introduire cette information en dédoublant certaines cases.

Pour ce faire, des mesures doivent être périodiquement effectuées sur la trajectoire sans déviation. Ensuite, on découpe la nouvelle trajectoire en tronçons sur lesquels la vitesse est considérée comme constante. Enfin, on duplique les cases de chaque tronçon selon les vitesses relevées précédemment.

7.2 Manœuvre d'évitement d'obstacles dynamiques

Pressés par le temps, nous avons décidé d'adapter l'évitement d'obstacles statiques aux cas où les obstacles sont en mouvement.

En présence d'obstacles *fixes* à éviter, certains bateaux vont dévier de leur trajectoire. Ainsi, ils deviennent des

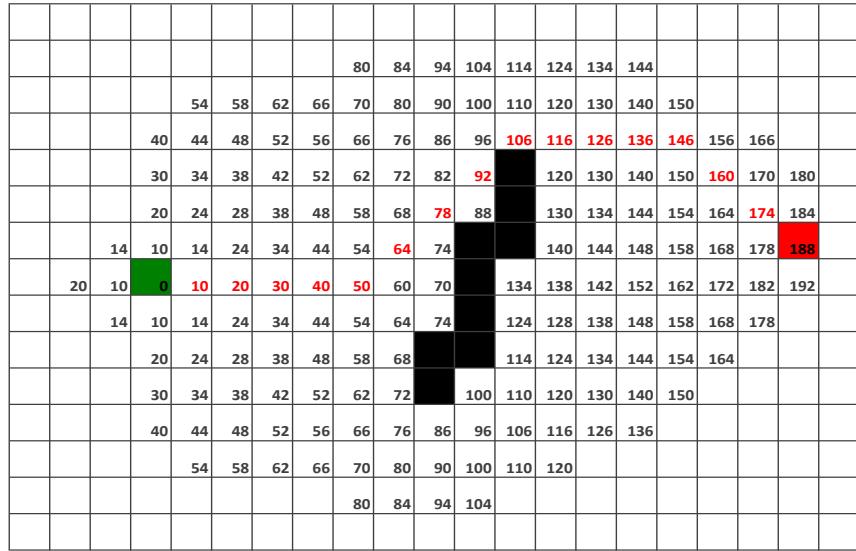


FIGURE 19 – Calcul de la déviation

obstacles *mobiles* potentiels pour les autres navires de la zone.

On s'appuie sur des règles de priorité afin de décider quel bateau doit entamer une manœuvre d'évitement.

7.2.1 Établissement de tables de priorité

Considérons les principaux paramètres qui caractérisent les bateaux :

- leur taille,
 - leur poids (en fonction de leur cargaison),
 - leur manœuvrabilité,
 - la dangerosité de la marchandise embarquée (par exemple : hydrocarbures, produits chimiques),
 - la méthode de propulsion (voile ou moteur).

Ces différents critères nous permettent d'établir des règles de priorité, afin de déterminer *a priori* quelle trajectoire sera déviée, et laquelle restera inchangée.²¹

7.2.2 Traitement des différentes trajectoires

À présent, la difficulté consiste à éviter l'**explosion du nombre d'appels** à la procédure d'évitement d'un obstacle. En effet, il faut s'assurer qu'en évitant un obstacle, on ne crée pas un grand nombre de zones interdites qu'il faudra elles-mêmes éviter par la suite.

Pour ce faire, nous parcourons notre tableau **par ordre décroissant de priorité**.

- Pour chaque navire $Navire[i]$ de priorité i donnée,
 - Pour chaque navire $Navire[k]$ de priorité k dans $[1; i - 1]$,
 - On effectue les déviations éventuelles de $Navire[i]$ pour éviter d'entrer en collision avec $Navire[k]$.

On définit ainsi un invariant de boucle, puisque à l'étape i de l'algorithme : **pour tout** k **dans** $[1; i - 1]$, $\text{Navire}[k]$ **possède sa trajectoire définitive** (puisque celle ne sera plus perturbée par les navires d'indice supérieur à k , car de priorité inférieure).

21. Ceci constitue une première approximation. Il est évident que les règles de priorités maritimes modifient ce modèle, mais nous ne pouvons pas en tenir compte de manière simple avec cet algorithme.

7.2.3 Utilisation de la déviation *statische*

Considérons ici deux bateaux, $N_i = \text{Navire}[i]$ et $N_k = \text{Navire}[k]$ avec $i > k$. Ainsi, N_i est dévié par N_k , d'après l'ordre de priorité.

On dispose de t_i et t_k les instants initiaux des trajectoires des navires, et de Dt_i et Dt_k les durées des trajets.

Pour chaque instant t tel que :

- $t_i < t < t_i + Dt_i$
- $t_k < t < t_k + Dt_k$

on vérifie que les positions $(x_i(t), y_i(t))$ et $(x_k(t), y_k(t))$ sont suffisamment distantes.

Une telle vérification n'est pas triviale : les bateaux n'arrivent pas (respectivement ne quittent pas) en même temps la zone, si bien que leur position au même instant n'est pas stockée au même indice dans le tableau de leurs positions.

bateaux	priorité	t=0	30s	1m	1m 30s	2m	2m 30s	...
Navire 1	2	-	B3	B4	C4	C5	D5	
Navire 2	4	A1	B1	B2	-	-	-	
Navire 3	1	D5	D5	D4	E4	E4	-	
Navire 4	3	-	-	F4	E4	D4	D4	
...								

FIGURE 20 – Exemple de trajectoires

Après avoir référencé toutes les positions successives de danger, on procède à la déviation de N_i .

Pour cela, on utilise la même procédure que pour les obstacles statiques, avec la donnée d'une zone fixe **temporellement impraticable** pour le navire N_i (ce sont des *zones d'évitement centrées sur les positions de N_k pendant la période où les deux navires sont trop proches l'un de l'autre*).

Une fois l'ensemble de la trajectoire déviée, on relance l'algorithme d'évitement jusqu'à ce que N_i ne soit plus dévié par **aucun** N_k (avec k dans $[1; i - 1]$).

On passe alors au traitement de $\text{Navire}[i + 1]$.

Grâce à ces deux méthodes, nous arrivons à proposer aux capitaines de bateaux des nouvelles trajectoires sans obstacle. Ainsi, on peut espérer éviter les accidents et résorber rapidement tout incident de trafic.

8 Interface graphique

Enfin, une équipe s'est consacrée à la réalisation de l'interface graphique. Cette dernière exploite les informations de la base de données et affiche les résultats des modèles développés dans les autres pôles du mini-projet.

Nous avions deux objectifs majeurs :

- créer un logiciel utilitaire pour les opérateurs de la tour de contrôle du port de Marseille, qui soit **pratique, efficace, et ergonomique**. Le principal défi fut de créer un affichage simulant un déroulement réel, c'est-à-

dire avec tous les bateaux placés correctement sur la carte du port de Marseille au bon moment. Nous avons ensuite dû intégrer les programmes des autres pôles du projet, ce qui a nécessité un travail en équipe intense basé sur la compréhension des travaux de l'ensemble du groupe.

- créer un programme qui permet de faire la simulation de positionnement des senseurs²² afin d'optimiser la détection des bateaux à l'entrée du port. Cela a nécessité la création de fonctionnalités permettant de créer/modifier/supprimer des capteurs pendant une simulation de l'activité du port simulant l'activité du port.

8.1 Fonctionnalités du logiciel

- Affichage d'une carte du secteur Ouest du Grand Port Maritime de Marseille en arrière-plan.
- Affichage de bateaux traversant le port de Marseille. Ceux-ci se déplacent selon les données récupérées de la base de données selon le temps courant (c'est-à-dire un point toutes les 30 secondes). La taille et la forme d'un bateau varient légèrement selon son type (cf **Figure 21**).

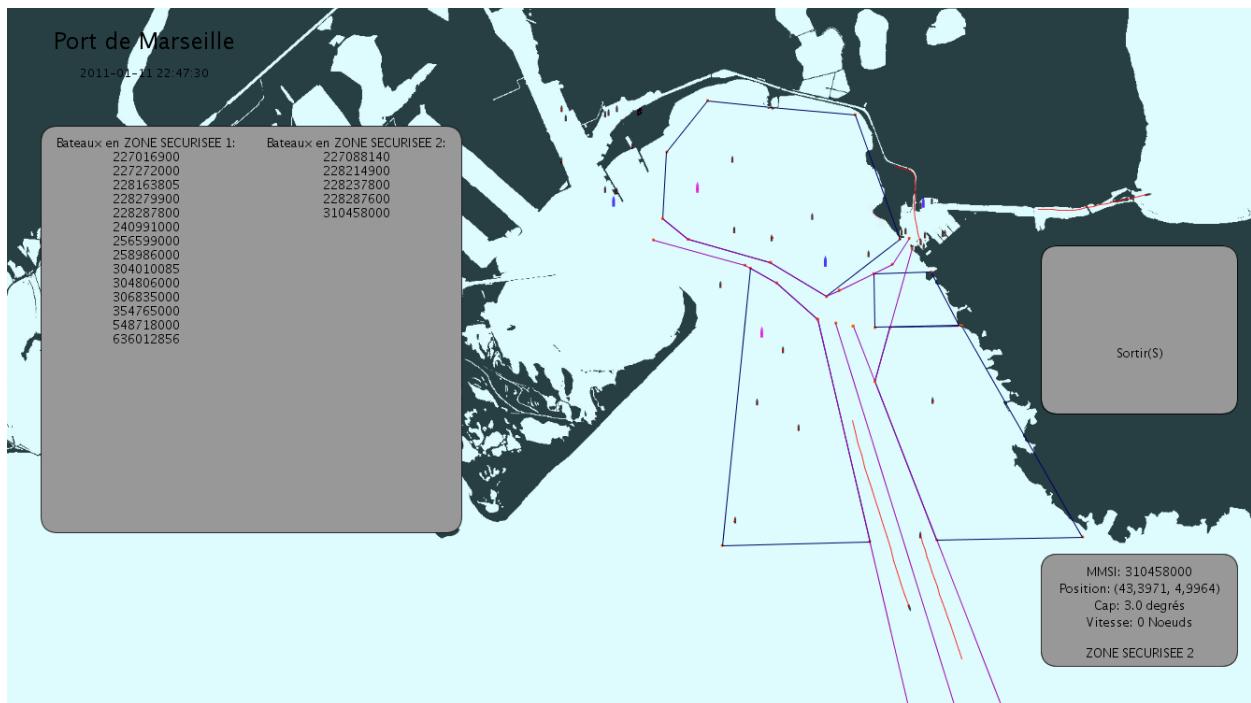


FIGURE 21 – Interface graphique avec trajectoire des bateaux

- Affichage de leur trajectoire passée durant la dernière demi-heure dans un souci de lisibilité. Cela correspond à la trajectoire rouge ou verte que laissent les bateaux derrière eux.
- Affichage des diverses zones du port de Marseille. On peut ainsi facilement observer des comportements non conformes à la zone dans laquelle un bateau se situe. Par exemple, le port comporte des chenaux principaux dans lequel un bateau en arrêt pendant une durée prolongée peut être considéré comme suspect.
- Affichage de senseurs. Si un bateau se trouve dans la zone de détection d'un senseur, la couleur de la traînée qu'il laisse passe du rouge au vert (cf **Figure 22**).
- Affichage des trajectoires prédites par les modèles statistiques et probabilistes.

22. radars, sonars, et caméras

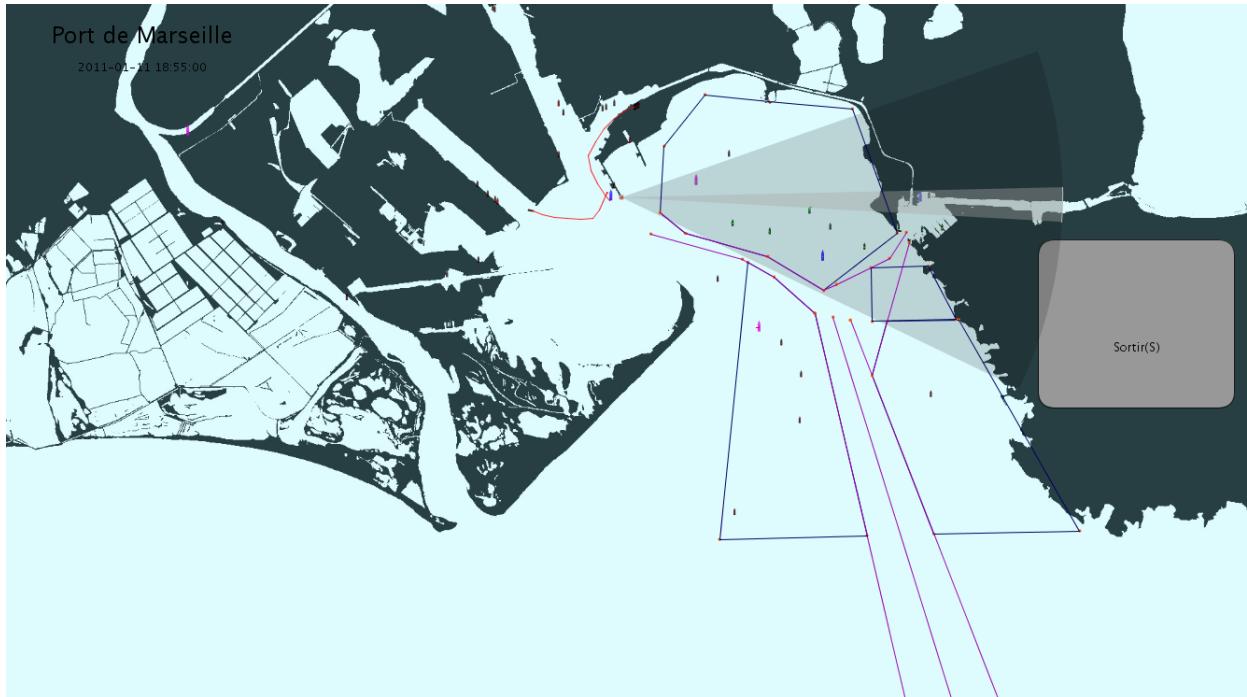


FIGURE 22 – Interface graphique avec affichage de l'aire de balaiement d'un senseur

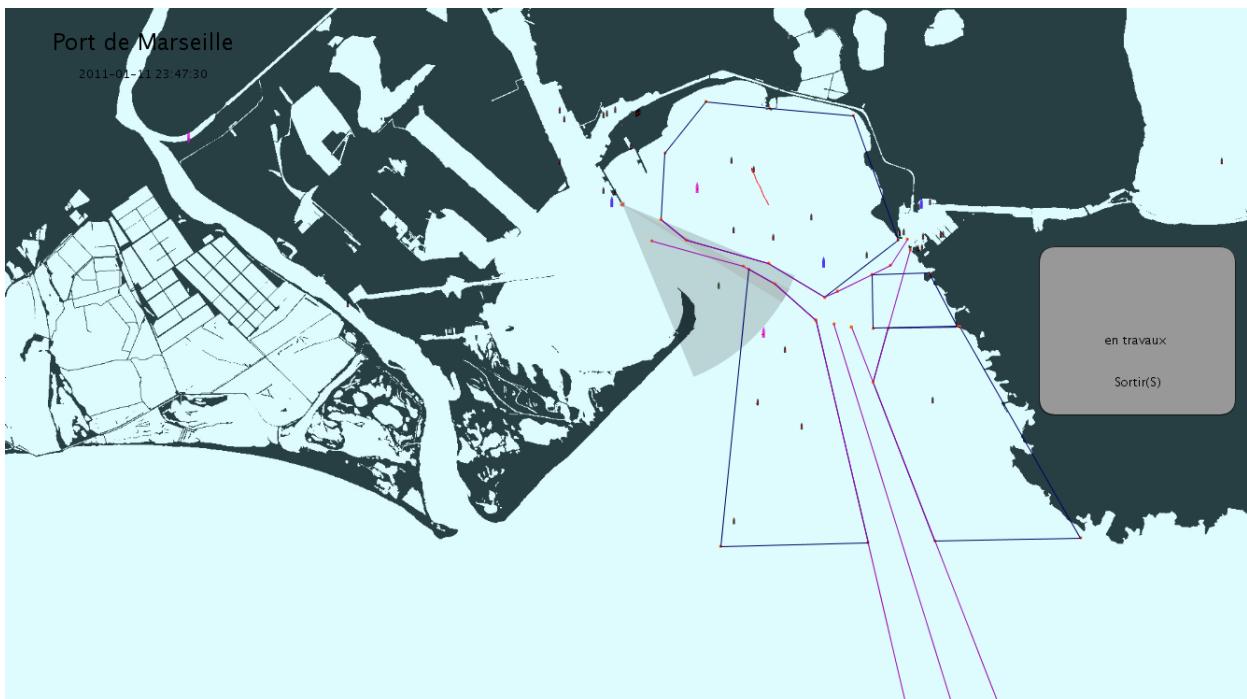


FIGURE 23 – La distance et la direction du balayage peuvent être changées par le logiciel

- Affichage d'informations sur les bateaux ou les senseurs au survol de la souris. Ces informations peuvent permettre de mieux identifier les causes d'un comportement anormal.
- Affichage d'un menu avec plusieurs options.

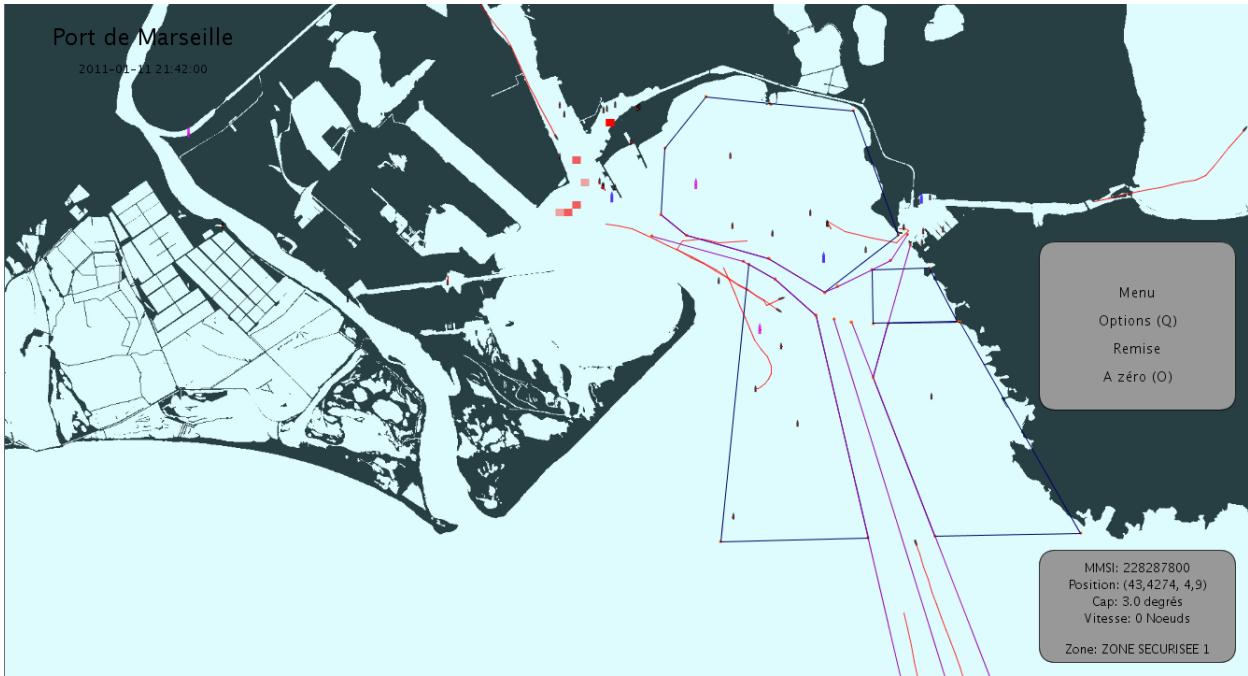


FIGURE 24 – Affichage des prédictions de trajectoire (petits carrés colorés)

Organisation du travail. Au début, nous avons commencé à faire l’interface graphique avec **Processing**, qui facilite l’affichage des éléments à l’écran. En parallèle, nous avons commencé à porter le programme sous Eclipse pour permettre une meilleure coordination avec les autres groupes.

Lorsque la première version de la base de données est arrivée, nous avons continué exclusivement sous Eclipse (mais avec des fonctionnalités de Processing). Pendant le développement des programmes des autres groupes, nous avons veillé à implémenter d’autres options (conversion des coordonnées, affichage des zones et des bateaux), à améliorer l’affichage et le confort d’utilisation du logiciel, tout en nous assurant que l’arrivée de ces programmes se fasse avec le moins d’accrocs possibles.

8.2 Difficultés rencontrées

Il a fallu concevoir à partir de zéro un logiciel qui permette de représenter les résultats des autres groupes travaillant sur les modèles de prévision et de perturbation de la trajectoire. Cela ne s’est bien évidemment pas réalisé sans quelques problèmes :

- Initialement, il a fallu commencer à programmer sans connaître les données d’affichage des bateaux. Il n’a pas toujours été aisément de travailler sous cette contrainte, en tâtonnant un peu, sans savoir exactement ce que l’on voulait obtenir. Toutefois, une fois la première version de la base de données construite, notre tâche est devenue beaucoup plus claire.
- Un autre problème qui a nécessité du temps et surtout beaucoup de recherches a été le portage sous Eclipse. Plusieurs réflexes, pas toujours immédiats, doivent être acquis pour pouvoir effectuer cette tâche. Il a fallu un peu de temps ainsi que la résolution de nombreux problèmes avant de faire tourner le logiciel correctement avec Java.
- Néanmoins, cette décision s’est avérée utile pour la suite du développement. Il en a été de même pour les conventions de nommage, qui ont permis de travailler de façon beaucoup plus cohérente avec les autres groupes et ainsi d’éviter de nombreux problèmes d’intégration de leurs programmes.
- Cette intégration ne s’est pas pour autant déroulée parfaitement, ce qui est naturel étant donnée l’ampleur du

projet. Cela a nécessité une bonne entente et une bonne coordination entre chaque groupe, ainsi que -comme pour tout le reste- un certain temps de déboguage des programmes.

- Il convient cependant d'insister sur le fait que la conception du logiciel ne s'est jamais déroulée de façon isolée. L'interaction avec les autres groupes s'est faite de manière continue tout au long du développement et ceux-ci ont également formulé de nombreuses suggestions pour améliorer l'interface.

Il a été vraiment formateur de travailler ainsi en équipe sur un projet d'une telle envergure. Il est très satisfaisant de constater que le projet a abouti et que le logiciel est fonctionnel.

Conclusion

Le projet a donc abouti à la création d'un logiciel fonctionnel. Une fois les différents algorithmes corrigés et rendus opérationnels, ceux-ci ont pu être incorporés dans l'interface graphique. Les images présentées précédemment montrent que les trajectoires des bateaux sont bien prédictes par le logiciel (cf **Figure 24**). L'interface graphique arrive à fonctionner de manière stable et affiche de manière autonome les trajectoires des bateaux stockées dans la base de données. Elle a aussi été munie de plusieurs options d'affichage pour remettre à zéro l'affichage ou encore changer les positions des senseurs définis dans le logiciel.

Pour vérifier la compatibilité du logiciel avec les algorithmes, nous avons choisi de le tester avec les données que nous avions. Nous avons affiché toutes les trajectoires des bateaux dans le port de Marseille qui ont été sauvegardées avec l'**AIS** le 11 Janvier 2011. Une fois la base de données créée pour ce jour-ci, nous avons pu lancer le logiciel. Après la correction de quelques erreurs, le logiciel a pu afficher correctement les trajectoires des bateaux ainsi que les prédictions réalisées. Le test a donc été probant. Un second test a été mené sur les données d'une autre journée, qui a été concluant lui aussi.

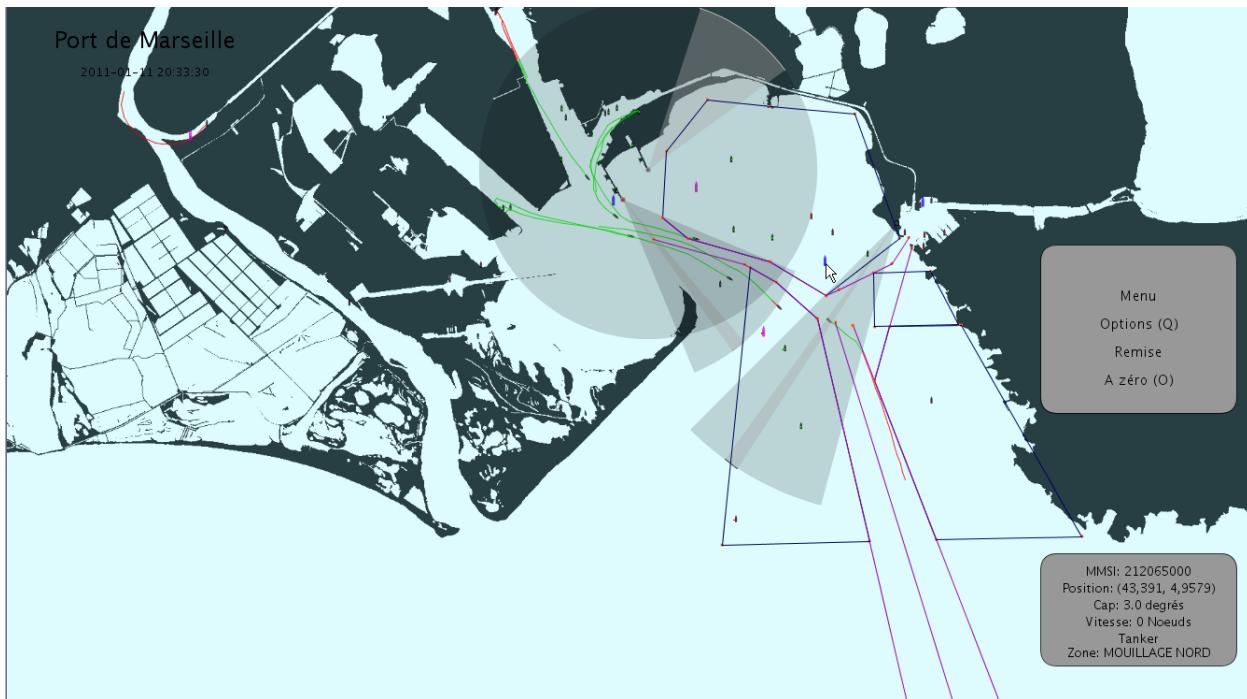


FIGURE 25 – Fenêtre graphique du logiciel final

Bien sûr, aboutir à un tel résultat a demandé beaucoup de temps. Tout d'abord, nous avons eu besoin de temps pour nous familiariser avec **MySQL** et **Processing** et pouvoir les utiliser pour construire le programme final. De nombreux essais ont dû être faits pour tester la validité des différents programmes implémentés. Comme toujours en informatique, le débogage a pris lui-aussi beaucoup de temps. La coordination des différents programmes, notamment au niveau de l'interface graphique, a elle-aussi posé beaucoup de problèmes.

Le logiciel créé a été développé en seulement trois semaines. Bien qu'il soit fonctionnel, il n'est pas exempt de défauts, et peut être de ce fait amélioré. Un logiciel complet devrait pouvoir traiter directement les données brutes issues des senseurs, et non seulement les données **AIS**. Or, cette intégration aurait demandé un temps conséquent, comme expliqué précédemment. Les algorithmes pourraient être optimisés pour gagner en temps et en espace mémoire. Les algorithmes utilisant les arbres de probabilités ont en effet une complexité exponentielle et ne pourraient donc pas utilisés pour traiter tous les bateaux. De même, des algorithmes plus performants que celui de *Baum-Welch* existent, et auraient pu être utilisés s'ils n'avaient pas demandé beaucoup plus de temps pour être implémentés. Des considérations physiques auraient pu aussi être prises en compte, comme l'inertie d'un bateau (un bateau ne peut évidemment pas s'arrêter en quelques secondes), ou comme l'utilisation de trajectoires plus réalistes. L'interface

graphique aurait pu aussi gagner en ergonomie, et d'autres options auraient pu être rajoutées.

Cependant, cela dépasse largement le cadre de notre projet. Notre programme n'est qu'un prototype, qui sert à démontrer la faisabilité d'un logiciel qui serait plus perfectionné. Comme l'ensemble des tests a été concluant, le prototype a parfaitement rempli son rôle, et il ne nous a manqué que du temps pour passer à une implémentation plus poussée du projet.

Bibliographie

Application Programming Interface de Processing. 23 Novembre 2012. <http://processing.org/reference/>

Brigaumont Chantal, *Administrez vos bases de données avec MySQL*. Site du Zero. 22 Novembre 2012. <http://www.siteduzero.com/3-464494-administrez-vos-bases-de-donnees-avec-mysql.html>.

Briquet Cyril, *Algorithmes de contournement de barrières*. Travail de fin d'études, Diplôme d'Etudes Approfondies en Informatique, Université de Liège. Juin 2003.

Maisonneuve Francis, *Probabilités*, Presse de l'École des Mines, 2012. 195 pages.

Préfecture des Bouches-du-Rhône. Arrêté n° 2012016-002 du 16 janvier 2012 portant création de la zone maritime et fluviale de régulation du grand port maritime de Marseille, réglementant le service de trafic maritime et de diverses mesures relatives à la sûreté du grand port maritime de Marseille.

Rabiner Lawrence. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, VOL.77. Février 1989. Page 258 à 286.

Systèmes embarqués critiques et sécurité de zones maritimes. CMA. 15 Décembre 2012. <http://www.cma.ensmp.fr/FR/recherche/securite/>

Systèmes de sécurité des biens, des personnes et des installations présents dans une zone maritime sensible. Pôle Mer PACA. 15 Décembre 2012. <http://www.polemerpacac.com/Securite-et-surete-maritime/Protection-maritime-etendue-et-rapprochee/SECMAR>

The ESTEREL Language. INRIA. 1^{er} Décembre 2012. <http://www-sop.inria.fr/meije/esterel/esterel-eng.html>

The Coq Proof Assistant. INRIA. 1^{er} Décembre 2012. <http://coq.inria.fr/>

Annexe

On donne ici le code source du programme implémenté durant le projet.

```
1 import processing.core.*;
2
3 // Affichage des ellipses pour le groupe travaillant sur les HMM.
4
5 public class AffichageHMM {
6     static PApplet parent;
7
8     public static void Affiche(int n) {
9
10        switch (n) {
11            case 1:
12                parent.fill(400, 0, 0, 100);
13                parent.pushMatrix();
14                parent.translate(632, 456);
15                parent.rotate((float) 0.19);
16                parent.ellipse(0, 0, 56, 62);
17                parent.popMatrix();
18                parent.pushMatrix();
19                parent.translate(634, 383);
20                parent.rotate((float) -0.19);
21                parent.ellipse(0, 0, 35, 91);
22                parent.popMatrix();
23                parent.pushMatrix();
24                parent.translate(604, 278);
25                parent.rotate((float) -0.19);
26                parent.ellipse(0, 0, 40, 129);
27                parent.popMatrix();
28                parent.pushMatrix();
29                parent.translate(550, 200);
30                parent.rotate((float) -0.37);
31                parent.ellipse(0, 0, 91, 171);
32                parent.popMatrix();
33                parent.pushMatrix();
34                parent.translate(434, 165);
35                parent.rotate((float) 0.34);
36                parent.ellipse(0, 0, 288, 80);
37                parent.popMatrix();
38                parent.pushMatrix();
39                parent.translate(349, 179);
40                parent.rotate((float) 0.11);
41                parent.ellipse(0, 0, 314, 48);
42                parent.popMatrix();
43                break;
44
45            case 2:
46                parent.fill(0, 100, 0, 100);
47                parent.pushMatrix();
48                parent.translate(656, 455);
49                parent.rotate((float) -0.33);
50                parent.ellipse(0, 0, 22, 65);
51                parent.popMatrix();
52                parent.pushMatrix();
53                parent.translate(635, 379);
54                parent.rotate((float) -0.32);
55                parent.ellipse(0, 0, 17, 94);
56                parent.popMatrix();
57                parent.pushMatrix();
58                parent.translate(606, 279);
59                parent.rotate((float) -0.26);
60                parent.ellipse(0, 0, 21, 125);
61                parent.popMatrix();
62                parent.pushMatrix();
63                parent.translate(541, 203);
64                parent.rotate((float) -0.59);
65                parent.ellipse(0, 0, 26, 180);
66                parent.popMatrix();
67        }
68    }
69}
```

```

67     parent.pushMatrix();
68     parent.translate(450, 153);
69     parent.rotate((float) 0.6);
70     parent.ellipse(0, 0, 254, 15);
71     parent.popMatrix();
72     parent.pushMatrix();
73     parent.translate(410, 131);
74     parent.rotate((float) 0.52);
75     parent.ellipse(0, 0, 200, 18);
76     parent.popMatrix();
77     parent.pushMatrix();
78     parent.translate(420, 130);
79     parent.rotate((float) -0.35);
80     parent.ellipse(0, 0, 5, 86);
81     parent.popMatrix();
82     break;
83
84 case 3:
85     parent.fill(400, 400, 0, 100);
86     parent.pushMatrix();
87     parent.translate(658, 460);
88     parent.rotate((float) -0.55);
89     parent.ellipse(0, 0, 39, 51);
90     parent.popMatrix();
91     parent.pushMatrix();
92     parent.translate(642, 362);
93     parent.rotate((float) -0.11);
94     parent.ellipse(0, 0, 17, 166);
95     parent.popMatrix();
96     parent.pushMatrix();
97     parent.translate(633, 284);
98     parent.rotate((float) 0.1);
99     parent.ellipse(0, 0, 27, 115);
100    parent.popMatrix();
101    parent.pushMatrix();
102    parent.translate(636, 199);
103    parent.rotate((float) 0.25);
104    parent.ellipse(0, 0, 20, 168);
105    parent.popMatrix();
106    parent.pushMatrix();
107    parent.translate(642, 170);
108    parent.rotate((float) 0.6);
109    parent.ellipse(0, 0, 8, 145);
110    parent.popMatrix();
111    break;
112 }
113 }
114 }
```

```

1 import java.io.File;
2 import java.io.FileInputStream;
3 import java.io.FileNotFoundException;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8 import java.io.Serializable;
9 import java.util.*;
10
11 /**
12  * Gestion de l'arbre de probabilit? pour le groupe travaillant sur la pr?ision de trajectoires
13  * ? aide de donn? statistiques.*/
14
15 public class ArbreProba implements Serializable {
16     private static final long serialVersionUID = 1L;
17     public static final int FACTEUR_LIMITE_ECART = 4;
18     public static final String CHEMIN_BASE_DE_DONNEES = "D:\\\\Documents\\\\MIG\\\\Base de données";
19
20     /**
21      * Coordonnées de la case */
22     public short x, y;
23
24     /**
25      * Nombre de bateaux ayant suivi la trajectoire correspondant au noeud */
26     public int nbPassages;
```

```

23  /** Ensemble des fils du noeud */
24  public Set<ArbreProba> ensembleFils;
25  /** Variable globale correspondant au nombre de noeuds dans l'arbre */
26  public static int nbNoeuds;
27
28  /** Constructeur */
29  public ArbreProba(short x, short y) {
30      this.x = x;
31      this.y = y;
32      nbPassages = 0;
33      ensembleFils = new HashSet<ArbreProba>();
34  }
35
36  /** Constructeur sans argument (pour la racine) */
37  public ArbreProba() {
38      this((short) -1, (short) -1);
39  }
40
41  /** Renvoie l'arbre lu dans le fichier chemin. */
42  public static ArbreProba lireDepuisFichier(String chemin)
43      throws IOException {
44      ArbreProba arbre = null;
45      try {
46          FileInputStream in = new FileInputStream(new File(chemin));
47          ObjectInputStream objIn = new ObjectInputStream(in);
48          arbre = (ArbreProba) objIn.readObject();
49          in.close();
50      } catch (ClassNotFoundException e) {
51          System.err.println("Classe introuvable");
52      }
53      return arbre;
54  }
55
56  /** Enregistre l'arbre dans le fichier chemin */
57  public void enregistrerDansFichier(String chemin) {
58      try {
59          FileOutputStream out = new FileOutputStream(chemin);
60          ObjectOutputStream objOut = new ObjectOutputStream(out);
61          objOut.writeObject(this);
62          out.close();
63      } catch (FileNotFoundException e) {
64          System.err.println("Dossier introuvable");
65      } catch (IOException e) {
66          System.err.println("Erreur lors de l'enregistrement");
67      }
68  }
69
70  /** Renvoie la chaîne de caractères correspondant à l'arbre */
71  public String toString() {
72      String s = "[" + x + "," + y + " : " + nbPassages + " | ";
73      for (ArbreProba a : this.ensembleFils)
74          s = s + a;
75      return s + "]";
76  }
77
78  /**
79   * Ajoute un fils à un arbre et renvoie l'arbre lui-même (avec son nouveau
80   * fils)
81   */
82  public ArbreProba ajouterFils(ArbreProba a) {
83      ensembleFils.add(a);
84      return this;
85  }
86
87  /**
88   * Ajoute à l'arbre un chemin (en ajoutant les noeuds correspondants si
89   * nécessaire) et renvoie l'arbre
90   */
91  public ArbreProba ajouterCheminUnique(Chemin c, int hauteurMax) {
92      if (c != null && c.x != -1 && hauteurMax != 0) {
93          ArbreProba fils = fils(c.x, c.y);

```

```

94     if (fils == null) {
95         ajouterFils((new ArbreProba(c.x, c.y)).ajouterCheminUnique(
96             c.suiteChemin, hauteurMax - 1));
97         nbNoeuds++;
98     } else
99         fils.ajouterCheminUnique(c.suiteChemin, hauteurMax - 1);
100    }
101    nbPassages++;
102    return this;
103}
104
105 /**
106 * Ajoute à l'arbre un chemin et tous ses sous-chemins associés et renvoie
107 * l'arbre
108 */
109 public ArbreProba ajouterChemin(Chemin c, int hauteurMax) {
110     ajouterCheminUnique(c, hauteurMax);
111     if (c != null) {
112         ajouterChemin(c.suiteChemin, hauteurMax);
113     }
114     return this;
115 }
116
117 /**
118 * Renvoie les indices où les écarts de temps entre deux pointages
119 * consécutifs sont trop grands.
120 */
121 public static int[] ecarts(long[] time, long max) {
122     LinkedList<Integer> l = new LinkedList<Integer>();
123     l.add(-1);
124     for (int i = 0; i < time.length - 1; i++) {
125         if (time[i + 1] - time[i] > max)
126             l.add(i);
127     l.add(time.length - 1);
128     int[] r = new int[l.size()];
129     int j = 0;
130     for (Integer i : l)
131         r[j++] = (int) i;
132     return r;
133 }
134
135 // Méthodes AIS
136 /**
137 * Ajoute à l'arbre tous les chemins enregistrés dans le fichier
138 * spécifié en argument et renvoie l'arbre
139 */
140 public ArbreProba ajouterFichierAIS(String chemin, long increment,
141     int hauteurMax) {
142     BateauAIS[] bateaux = BateauAIS.lireFichier(chemin);
143     for (int i = 0; i < bateaux.length; i++) {
144         System.out.print(" " + (i + 1) + "/" + bateaux.length
145             + " [taille initiale : " + bateaux[i].time.length);
146         System.out.flush();
147         Chemin c = Chemin.cheminNormalise(bateaux[i], increment);
148         System.out.print(", taille normalisée : " + c.size());
149         ajouterChemin(c, hauteurMax);
150         System.out.println(", nombre de noeuds dans l'arbre : " + nbNoeuds
151             + "]");
152     }
153     return this;
154 }
155
156 /**
157 * Ajoute à l'arbre les chemins de tous les fichiers du dossier spécifié
158 * en argument et renvoie l'arbre
159 */
160 public ArbreProba ajouterDossierAIS(String chemin, long increment,
161     int hauteurMax) {
162     String[] cheminsFichiers = GestionFichiers
163         .recupererCheminsfichiers(chemin);
164     for (int i = 0; i < cheminsFichiers.length; i++) {

```

```

165     System.out.println((i + 1) + "/" + cheminsFichiers.length + " : "
166         + cheminsFichiers[i]);
167     System.out.flush();
168     ajouterFichierAIS(cheminsFichiers[i], increment, hauteurMax);
169 }
170 return this;
171 }
172
173 /** Renvoie l'arbre construit à partir du dossier spécifié en argument */
174 public static ArbreProba genererArbreAIS(long increment, int hauteurMax,
175     int nX, int nY) {
176     Etat.N1 = nX;
177     Etat.N2 = nY;
178     System.out.println("Calcul de l'arbre statistique AIS");
179     System.out.println("Dossier : " + CHEMIN_BASE_DE_DONNEES + "\\AIS");
180     System.out.println("Incrément temporel : " + increment + "s");
181     System.out.println("Hauteur maximale de l'arbre : " + hauteurMax);
182
183     nbNoeuds = 0;
184     ArbreProba arbre = new ArbreProba();
185     arbre.ajouterDossierAIS(CHEMIN_BASE_DE_DONNEES + "\\AIS\\Données",
186         1000 * increment, hauteurMax);
187
188     System.out.println("Enregistrement de l'arbre... ");
189
190     arbre.enregistrerDansFichier(CHEMIN_BASE_DE_DONNEES
191         + "\\AIS\\Arbres statistiques\\" + increment + "_" + hauteurMax
192         + "_" + Etat.N1 + "_" + Etat.N2 + ".obj");
193
194     System.out.println();
195     System.out.println("Calcul de l'arbre statistique AIS terminé");
196     System.out.println("Arbre enregistré sous " + CHEMIN_BASE_DE_DONNEES
197         + "\\AIS\\Arbres statistiques\\" + increment + "_" + hauteurMax
198         + "_" + nX + "_" + nY + ".obj");
199     System.out.println("Incrément temporel : " + increment + "s");
200     System.out.println("Hauteur maximale de l'arbre : " + hauteurMax);
201     System.out.println("Longueur temporelle de l'arbre : "
202         + (increment * hauteurMax));
203     System.out.println("Nombre de noeuds dans l'arbre : " + nbNoeuds);
204
205     return arbre;
206 }
207
208 /**
209 * Renvoie l'arbre correspondant aux paramètres spécifié, en le calculant
210 * si nécessaire
211 */
212 public static ArbreProba arbreAIS(long increment, int hauteurMax, int nX,
213     int nY) {
214     try {
215         System.out.println("Lecture de l'arbre depuis le fichier... ");
216         return lireDepuisFichier(CHEMIN_BASE_DE_DONNEES
217             + "\\AIS\\Arbres statistiques\\" + increment + "_"
218             + hauteurMax + "_" + nX + "_" + nY + ".obj");
219     } catch (FileNotFoundException e) {
220         System.out.println("Arbre non enregistré");
221         return genererArbreAIS(increment, hauteurMax, nX, nY);
222     } catch (IOException e) {
223         System.out.println("Erreur de lecture");
224         return genererArbreAIS(increment, hauteurMax, nX, nY);
225     } finally {
226         System.out.println();
227         System.out.println("Arbre chargé");
228         System.out.println("Incrément temporel : " + increment);
229         System.out.println("Hauteur maximale de l'arbre : " + hauteurMax);
230         System.out.println("Maillage : " + nX + "x" + nY);
231     }
232 }
233
234 /**
235 * Renvoie l'arbre correspondant aux paramètres spécifiés, en le

```

```

236     * calculant si nécessaire
237     */
238     public static ArbreProba arbreAIS(long increment, int hauteurMax,
239         double taille) {
240         return arbreAIS(increment, hauteurMax, (int) (Etat.X / taille),
241             (int) (Etat.Y / taille));
242     }
243
244     // Méthode Sys
245     /**
246     * Ajoute à l'arbre tous les chemins enregistrés dans le fichier
247     * spécifié en argument et renvoie l'arbre
248     */
249     public ArbreProba ajouterFichierSys(String chemin, long increment,
250         int hauteurMax) {
251         BateauSys[] bateaux = BateauSys.lireDansFichier(chemin);
252         for (int i = 0; i < bateaux.length; i++) {
253             System.out.print(" " + (i + 1) + "/" + bateaux.length
254                 + " [taille initiale : " + bateaux[i].time.length);
255             System.out.flush();
256             int[] delimitateurs = ecarts(bateaux[i].time, FACTEUR_LIMITE_ECART
257                 * increment);
258             System.out.print(", nombre de chemins distincts : "
259                 + (delimitateurs.length - 1));
260             for (int j = 0; j < delimitateurs.length - 1; j++)
261                 ajouterChemin(Chemin.cheminNormalise(bateaux[i], increment,
262                     delimitateurs[j] + 1, delimitateurs[j + 1]), hauteurMax);
263             // Chemin c = Chemin.cheminNormalise(bateaux[i], increment);
264             // System.out.print(", taille normalisée : " + c.size());
265             // ajouterChemin(c, hauteurMax);
266             System.out.println(", nombre de noeuds dans l'arbre : " + nbNoeuds
267                 + "]");
268         }
269         return this;
270     }
271
272     /**
273     * Ajoute à l'arbre les chemins de tous les fichiers du dossier spécifié
274     * en argument et renvoie l'arbre
275     */
276     public ArbreProba ajouterDossierSys(String chemin, long increment,
277         int hauteurMax) {
278         String[] cheminsFichiers = GestionFichiers
279             .recupererCheminsfichiers(chemin);
280         for (int i = 0; i < cheminsFichiers.length; i++) {
281             System.out.println((i + 1) + "/" + cheminsFichiers.length + " : "
282                 + cheminsFichiers[i]);
283             System.out.flush();
284             ajouterFichierSys(cheminsFichiers[i], increment, hauteurMax);
285         }
286         return this;
287     }
288
289     /** Renvoie l'arbre construit à partir du dossier spécifié en argument */
290     public static ArbreProba genererArbreSys(long increment, int hauteurMax) {
291         System.out.println("Calcul de l'arbre statistique Sys");
292         System.out.println("Dossier : " + CHEMIN_BASE_DE_DONNEES + "\\Sys");
293         System.out.println("Incrément temporel : " + increment + "s");
294         System.out.println("Hauteur maximale de l'arbre : " + hauteurMax);
295
296         nbNoeuds = 0;
297         ArbreProba arbre = new ArbreProba();
298         arbre.ajouterDossierSys(CHEMIN_BASE_DE_DONNEES + "\\Sys\\Données",
299             1000 * increment, hauteurMax);
300
301         System.out.println("Enregistrement de l'arbre... ");
302
303         arbre.enregistrerDansFichier(CHEMIN_BASE_DE_DONNEES
304             + "\\Sys\\Arbres statistiques\\\" + increment + "_" + hauteurMax
305             + "_" + Etat.N1 + "_" + Etat.N2 + ".obj");

```

```

307     System.out.println();
308     System.out.println("Calcul de l'arbre statistique Sys terminé");
309     System.out.println("Arbre enregistré sous " + CHEMIN_BASE_DE_DONNEES
310         + "\\\\" + Sys + "\\Arbres statistiques\\" + increment + "_" + hauteurMax
311         + "_" + Etat.N1 + "_" + Etat.N2 + ".obj");
312     System.out.println("Incrément temporel : " + increment + "s");
313     System.out.println("Hauteur maximale de l'arbre : " + hauteurMax);
314     System.out.println("Longueur temporelle de l'arbre : "
315         + (increment * hauteurMax));
316     System.out.println("Nombre de noeuds dans l'arbre : " + nbNoeuds);
317
318     return arbre;
319 }
320
321 /**
322 * Renvoie l'arbre correspondant aux paramètres spécifié , en le calculant
323 * si nécessaire
324 */
325 public static ArbreProba arbreSys(long increment , int hauteurMax) {
326     try {
327         System.out.println("Lecture de l'arbre depuis le fichier...");
328         return lireDepuisFichier(CHEMIN_BASE_DE_DONNEES
329             + "\\\\" + Sys + "\\Arbres statistiques\\" + increment + "_"
330             + hauteurMax + "_" + Etat.N1 + "_" + Etat.N2 + ".obj");
331     } catch (FileNotFoundException e) {
332         System.out.println("Arbre non enregistré");
333         return genererArbreSys(increment , hauteurMax);
334     } catch (IOException e) {
335         System.out.println("Erreur de lecture");
336         return genererArbreSys(increment , hauteurMax);
337     } finally {
338         System.out.println();
339         System.out.println("Arbre chargé");
340         System.out.println("Incrément temporel : " + increment);
341         System.out.println("Hauteur maximale de l'arbre : " + hauteurMax);
342         System.out.println("Maillage : " + Etat.N1 + "x" + Etat.N2);
343     }
344 }
345
346 /**
347 * Renvoie le fils correspondant à l'état e , ou null si le fils n'existe
348 * pas .
349 */
350 public ArbreProba fils(short x, short y) {
351     for (ArbreProba fils : ensembleFils)
352         if (fils.x == x && fils.y == y)
353             return fils;
354     return null;
355 }
356
357 /**
358 * Renvoie le fils le plus probable (i.e. celui ayant le poids le plus
359 * grand)
360 */
361 public ArbreProba filsLePlusProbable() {
362     ArbreProba resultat = null;
363     int nbPassagesMax = -1;
364     for (ArbreProba fils : ensembleFils)
365         if (fils.nbPassages > nbPassagesMax) {
366             resultat = fils;
367             nbPassagesMax = fils.nbPassages;
368         }
369     return resultat;
370 }
371
372 /**
373 * Renvoie le chemin le plus probable , calculé de proche en proche */
374 public Chemin cheminLePlusProbable() {
375     ArbreProba fils = filsLePlusProbable();
376     if (fils == null)
377         return new Chemin(x, y, null);
378     else

```

```

378     return new Chemin(x, y, fils.cheminLePlusProbable());
379 }
380
381 public Chemin cheminLePlusProbable(int n) {
382     ArbreProba fils = filsLePlusProbable();
383     if (fils == null || n == 0)
384         return new Chemin(x, y, null);
385     else
386         return new Chemin(x, y, fils.cheminLePlusProbable(--n));
387 }
388
389 /** Renvoie le sous-arbre associé à un chemin */
390 public ArbreProba sousArbre(Chemin chemin) {
391     if (chemin == null)
392         return this;
393     else {
394         ArbreProba fils = fils(chemin.x, chemin.y);
395         if (fils == null)
396             return new ArbreProba(chemin.x, chemin.y);
397         else
398             return fils.sousArbre(chemin.suiteChemin);
399     }
400 }
401
402 /** Renvoie la table de probabilités de présence */
403 public double[][][] tableProbabilitesPresence(int duree) {
404     double t2 = (double) nbPassages;
405     Set<ArbreProba> ensemble = new HashSet<ArbreProba>();
406     ensemble.add(this);
407     double[][][] table = new double[duree + 1][Etat.N1][Etat.N2];
408     for (int t = 0; t <= duree && !ensemble.isEmpty(); t++) {
409         Set<ArbreProba> tmp = new HashSet<ArbreProba>();
410         for (ArbreProba n : ensemble) {
411             if (n.x > 0 && n.x <= Etat.N1 && n.y > 0 && n.y <= Etat.N2)
412                 table[t][n.x - 1][n.y - 1] += n.nbPassages / t2;
413                 tmp.addAll(n.ensembleFils);
414         }
415         ensemble = tmp;
416     }
417     return table;
418 }
419
420 /** Normalise la table */
421 public static double[][][] normaliser(double[][][] table) {
422     for (int i = 0; i < table.length; i++) {
423         double max = 0;
424         for (int j = 0; j < table[0].length; j++)
425             for (int k = 0; k < table[0][0].length; k++)
426                 if (table[i][j][k] > max)
427                     max = table[i][j][k];
428         for (int j = 0; j < table[0].length; j++)
429             for (int k = 0; k < table[0][0].length; k++)
430                 table[i][j][k] /= max;
431     }
432     return table;
433 }
434
435 /** Renvoie le maximum de la table */
436 public static double max(double[][] table) {
437     double max = 0;
438     for (int i = 0; i < table.length; i++)
439         for (int j = 0; j < table[0].length; j++)
440             if (table[i][j] > max)
441                 max = table[i][j];
442     return max;
443 }
444
445 /** Renvoie le minimum de la table */
446 public static double min(double[][] table) {
447     double min = 1;
448     for (int i = 0; i < table.length; i++)

```

```

449     for (int j = 0; j < table[0].length; j++)
450         if (table[i][j] < min)
451             min = table[i][j];
452     return min;
453 }
454
455 /**
456 * Renvoie true si les éléments du tableau sont triés par ordre
457 * croissant, false sinon
458 */
459 public static boolean verifierChronologie(long[] l) {
460     boolean b = false;
461     for (int i = 0; i < l.length - 1; i++)
462         b = b || (l[i] > l[i + 1]);
463     return b;
464 }
465 }
```

```

1 import java.sql.Connection;
2 import java.sql.ResultSet;
3 import java.sql.SQLException;
4 import java.sql.Statement;
5 import java.util.Date;
6
7 /* classe Bateau. Chaque objet bateau contient des informations sur le bateau considéré,
8 ainsi que le tableau normalisé des positions qu'il a occupé. */
9
10 public class Bateau implements Comparable<Bateau> {
11
12     private Flotte flotte = null;
13     private Position[] position;
14     public int[][] traj;
15     private long MMSI;
16     private long minPosNorm;
17     private long maxPosNorm;
18     private String nomBateau;
19     private String nomUsage;
20     private int typeBateau;
21     private double longueur;
22     private double largeur;
23     private Date ETA;
24
25     public String getNomBateau() {
26         return nomBateau;
27     }
28
29     public String getNomUsage() {
30         return nomUsage;
31     }
32
33     public long getMMSI() {
34         return MMSI;
35     }
36
37     public int getNumeroTypeBateau() {
38         return typeBateau;
39     }
40
41     public String getTypeBateau() {
42         String s = null;
43
44         if (typeBateau == 0)
45             s = "Pas de Type";
46         else {
47             try {
48                 Connection connect = mySQL.ConnectionSQL();
49                 Statement state = connect.createStatement();
50                 ResultSet res = state
51                     .executeQuery("SELECT description FROM Type WHERE numero = "
52                         + typeBateau + ";");
53                 res.first();
```

```

54     s = res.getString("description");
55 } catch (SQLException e) {
56     e.printStackTrace();
57 }
58 }
59
60     return s;
61 }
62
63 public Position[] getPosition() {
64     return position;
65 }
66
67 public Date getETA() {
68     return ETA;
69 }
70
71 public int getMinPosNorm() {
72     return (int) (minPosNorm);
73 }
74
75 public int getMaxPosNorm() {
76     return (int) (maxPosNorm);
77 }
78
79 public void setMaxPosNorm(long maxPosNorm) {
80     this.maxPosNorm = maxPosNorm;
81 }
82
83 public void setMinPosNorm(long minPosNorm) {
84     this.minPosNorm = minPosNorm;
85 }
86
87 public void setFlotte(Flotte flotte2) {
88     flotte = flotte2;
89 }
90
91 public Bateau(long MMSI, String nomBateau, String nomUsage, int typeBateau,
92             Position[] pos, double longueur, double largeur, Date ETA,
93             Flotte flotte) {
94     super();
95     this.MMSI = MMSI;
96     this.nomBateau = nomBateau;
97     this.nomUsage = nomUsage;
98     this.typeBateau = typeBateau;
99     this.position = pos;
100    this.longueur = longueur;
101    this.largeur = largeur;
102    this.ETA = ETA; // Attention au pbm de conversion
103    this.flotte = flotte;
104 }
105
106 public Bateau(long mmsi, Position[] pos) {
107     this.MMSI = mmsi;
108     this.position = pos;
109 }
110
111 public Bateau(long mmsi, String nom, Position[] pos) {
112     this.position = pos;
113     MMSI = mmsi;
114     nomBateau = nom;
115 }
116
117 public Bateau(String nom, int[][] traj, double longueur, double largeur,
118               long minPosNorm, long maxPosNorm) {
119     this.nomBateau = nom;
120     this.traj = traj;
121     this.longueur = longueur;
122     this.largeur = largeur;
123     this.minPosNorm = minPosNorm;
124     this.maxPosNorm = maxPosNorm;

```

```

125 }
126
127 public void afficher() {
128     System.out.println("\n#Info sur le bateau : ");
129     System.out.println(" MMSI : " + MMSI);
130     System.out.println(" Nom : " + nomBateau);
131     System.out.println(" Nom d'usage : " + nomUsage);
132     System.out.println(" Type de bateau : " + typeBateau);
133     System.out.println(" LARGEUR : " + largeur + " LONGUEUR : " + longueur);
134     System.out.println(" Nb de positions : " + position.length);
135
136     System.out.println();
137 }
138
139 public String toString() {
140     return ("\n#Info sur le bateau : " + " MMSI : " + MMSI + " Nom : "
141         + nomBateau + " Nom d'usage : " + nomUsage
142         + " Type de bateau : " + typeBateau + " LARGEUR : " + largeur
143         + " LONGUEUR : " + longueur + " Nb de positions : "
144         + position.length + "\n");
145 }
146
147 public static Flotte normaliserPosition(Flotte flotte) {
148     flotte.setBateaux(Bateau.normaliserPosition(flotte.getBateaux(), 30));
149     return flotte;
150 }
151
152 public static Flotte normaliserPosition(Flotte flotte, long deltaT) {
153     flotte.setBateaux(Bateau.normaliserPosition(flotte.getBateaux(), deltaT));
154     return flotte;
155 }
156
157 public static Bateau[] normaliserPosition(Bateau[] b) {
158     return Bateau.normaliserPosition(b, 30);
159 }
160
161 public static Bateau[] normaliserPosition(Bateau[] b, long deltaT) {
162     if (b == null)
163         return null;
164     for (int i = 0; i < b.length; i++) {
165         if (b[i].normaliserPosition(deltaT) == null) {
166             Bateau[] b2 = new Bateau[b.length - 1];
167             for (int j = 0; j < b2.length; j++) {
168                 if (j < i)
169                     b2[j] = b[j];
170                 else
171                     b2[j] = b[j + 1];
172             }
173             System.out.println("Bateau supprimé : " + b[i].nomBateau + " (" +
174                 + b[i].MMSI + ")");
175             b = b2;
176             i--;
177         }
178     }
179     return b;
180 }
181
182
183 public static void normaliserPosition(Bateau b) {
184     if (b == null)
185         return;
186     b.normaliserPosition(30);
187 }
188
189 public static void normaliserPosition(Bateau b, long deltaT) {
190     if (b == null)
191         return;
192     b.normaliserPosition(deltaT);
193 }
194
195 public Position[] normaliserPosition(long deltaT) { // Cette fonction m'a

```

```

196                                     // tué ... des heures de
197                                     // debug ...
198 Position [] pos = this.position;
199 int n = pos.length;
200
201 if (pos.length <= 1)
202     return null;
203
204 Date t0 = pos[0].getTemps();
205 Date tf = pos[n - 1].getTemps();
206
207 long a = t0.getTime();
208 long b = tf.getTime();
209 long t = 0;
210
211 if (((a + deltaT * 1000) / (deltaT * 1000)) * deltaT * 1000 == (a + deltaT * 1000))
212     t = a;
213 else
214     t = ((a + deltaT * 1000) / (deltaT * 1000)) * deltaT * 1000;
215
216 long res = (b - t) / (deltaT * 1000);
217 res += 1;
218
219 Position [] tab;
220 tab = new Position[(int) res];
221
222 // On arrondit l'entier t comme on veut.
223
224 int i = 0, cap = 0;
225 double coef1 = 0, coef2 = 0;
226 double X, Y, longitude, latitude, sog = 0, cog = 0, tauxRot = 0;
227
228 for (int k = 0; k < res; k++) {
229     while (pos[i + 1].getTemps().getTime() <= t && (i + 2) < n) {
230         i++;
231     }
232
233     a = pos[i].getTemps().getTime();
234     b = pos[i + 1].getTemps().getTime();
235
236     if (a == t) {
237         tab[k] = pos[i];
238     } else if (((b - a) > 1000 * 300)
239                 && (Math.abs(pos[i].getX() - pos[i + 1].getX()) > 0.1)
240                 && (Math.abs(pos[i].getY() - pos[i + 1].getY()) > 0.1)) {
241         tab[k] = null;
242     } else {
243         double t1 = (double) t;
244         double a1 = (double) a;
245         double b1 = (double) b;
246
247         coef1 = (t1 - a1) / (b1 - a1);
248         coef2 = (b1 - t1) / (b1 - a1);
249
250         X = coef2 * pos[i].getX() + coef1 * pos[i + 1].getX();
251         Y = coef2 * pos[i].getY() + coef1 * pos[i + 1].getY();
252         longitude = coef2 * pos[i].getLongitude() + coef1
253             * pos[i + 1].getLongitude();
254         latitude = coef2 * pos[i].getLatitude() + coef1
255             * pos[i + 1].getLatitude();
256         cog = coef2 * pos[i].getCog() + coef1 * pos[i + 1].getCog();
257         sog = coef2 * pos[i].getSog() + coef1 * pos[i + 1].getSog();
258         cap = (int) (coef2 * pos[i].getCap() + coef1
259             * pos[i + 1].getCap());
260         tauxRot = coef2 * pos[i].getVitesseAng() + coef1
261             * pos[i + 1].getVitesseAng();
262
263         tab[k] = new Position(new Date(t), longitude, latitude, X, Y,
264             sog, cog, tauxRot, cap);
265     }
266 }
```

```

267     t += deltaT * 1000;
268 }
269
270 // On enlève les Positions nulles au début et à la fin
271 // Au début
272
273 int j = 0;
274 while (j < tab.length && tab[j] == null) {
275     j++;
276 }
277 if (j == tab.length) {
278     // System.out.println("Sans position : " + this.MMSI);
279     this.position = null;
280     return null;
281 }
282
283 // A la fin
284
285 int l = tab.length - 1;
286 while (l >= 0 && tab[l] == null) {
287     l--;
288 }
289 if (l < 0) {
290     // System.out.println("Sans position 2 : " + this.MMSI);
291     this.position = null;
292     return null;
293 }
294
295 Position[] tab2 = new Position[l + 1 - j];
296 for (int m = 0; m <= l - j; m++) {
297     tab2[m] = tab[m + j];
298 }
299
300 this.position = tab2;
301 return tab2;
302 }
303
304 public Position getPosNorm(int pos) { // A n'utiliser que si le bateau fait
305     // partie d'une flotte normalisée !
306     long indPos = (long) (pos);
307     if (!flotte.isNormalise())
308         return null;
309     if (indPos >= minPosNorm && indPos <= maxPosNorm) {
310         return this.position[(int) (indPos - minPosNorm)];
311     }
312
313     return null;
314 }
315
316 /**
317 * Permet de comparer l'ordre de priorité de deux bateaux
318 *
319 * @param b : instance de Bateau à comparer avec l'instance actuelle
320 *
321 * @return i : -1 si this est prioritaire face à b 0 si indifférent +1 si b
322 * est prioritaire face à this
323 *
324 * Amélioration : selon le type du bateau (importance tactique ou non) ?
325 */
326 public int compareTo(Bateau b) {
327
328     if (this.longueur * this.largeur > b.longueur * b.largeur) {
329         return -1;
330     } else if (this.longueur * this.largeur < b.longueur * b.largeur) {
331         return 1;
332     } else {
333         return 0;
334     }
335
336 }
337

```

```

1 import java.io.File;
2 import java.io.FileInputStream;
3 import java.io.FileNotFoundException;
4 import java.io.IOException;
5 import java.io.ObjectInputStream;
6 import java.io.Serializable;
7
8 // Vérification de l'existence d'un bateau.
9
10 public class BateauAIS implements Serializable {
11     private static final long serialVersionUID = 101L;
12
13     public long MMSI;
14     public String name;
15     public String callSign;
16     public int shipAndCargoType;
17     public long time[];
18     public double position[][];
19
20     public static BateauAIS[] lireFichier(String cheminFichier) {
21         BateauAIS[] bateaux = null;
22         try {
23             ObjectInputStream s = new ObjectInputStream(new FileInputStream(
24                 new File(cheminFichier)));
25             bateaux = (BateauAIS[]) s.readObject();
26             s.close();
27         } catch (FileNotFoundException e) {
28             System.err.println("Fichier non trouvé");
29         } catch (IOException e) {
30             System.err.println("Erreur de lecture");
31         } catch (ClassNotFoundException e) {
32             System.err.println("Classe introuvable");
33         }
34         return bateaux;
35     }
36 }
```

```

1 import java.io.*;
2
3 // Vérification de l'existence d'un bateau.
4
5 public class BateauSys implements Serializable {
6
7     private static final long serialVersionUID = 1L;
8
9     public long MMSI;
10    public long[] time;
11    public double[][] position;
12
13    public static BateauSys[] lireDansFichier(String cheminFichier) {
14        BateauSys[] bateaux = null;
15        try {
16            FileInputStream in = new FileInputStream(cheminFichier);
17            ObjectInputStream objIn = new ObjectInputStream(in);
18            bateaux = (BateauSys[]) objIn.readObject();
19            in.close();
20        } catch (FileNotFoundException e) {
21            System.err.println("Fichier non trouvé");
22        } catch (IOException e) {
23            System.err.println("Erreur lors de la lecture du fichier");
24        } catch (ClassNotFoundException e) {
25            System.err.println("Classe non trouvée");
26        }
27        return bateaux;
28    }
29
30    public void enregistrerDansFichier(String cheminFichier) {
31        try {
```

```

32     FileOutputStream out = new FileOutputStream(cheminFichier);
33     ObjectOutputStream objOut = new ObjectOutputStream(out);
34     objOut.writeObject(this);
35     out.close();
36 } catch (FileNotFoundException e) {
37     System.out.println("Le fichier ne peut pas être ouvert");
38 } catch (IOException e) {
39     System.out.println("Erreur lors de l'écriture du fichier");
40 }
41 }
42 }
43 }
```

```

1 import processing.core.*;
2
3 // Affichage des bouées délimitant les zones du port
4
5 public class Bouee {
6     public static PApplet parent;
7     public double[] coordonnees;
8     public String nomBouee;
9
10    public Bouee(double[] coord, String nom) {
11        coordonnees = coord;
12        nomBouee = nom;
13
14        if (parent == null) {
15            parent = interfacegraphique.truc;
16        }
17        parent.fill(252, 97, 0);
18        parent.noStroke();
19        double[] coordutm = ConversionMercator.conversion(coord);
20        double coord1pix = ConversionPixel
21            .conversionPixelX((float) coordutm[0]);
22        double coord2pix = ConversionPixel
23            .conversionPixelY((float) coordutm[1]);
24
25        parent.arc((float) coord1pix, (float) coord2pix, (float) 3, (float) 3,
26            (float) 0, (float) (2 * Math.PI));
27        parent.noFill();
28
29        if (parent.dist(parent.mouseX, parent.mouseY, (float) coord1pix,
30            (float) coord2pix) < 2) {
31            parent.textSize(8);
32            parent.rectMode(PConstants.CENTER);
33            parent.stroke(50, 50, 50);
34            parent.fill(252, 97, 0);
35            parent.rect((float) coord1pix + 30, (float) coord2pix + 30,
36                (float) 60, (float) 15, (float) 18, (float) 18, (float) 18,
37                (float) 18);
38            parent.fill(0, 0, 0, 200);
39            parent.textAlign(PConstants.CENTER);
40            parent.text("Bouée " + nom, (float) coord1pix + 30,
41                (float) coord2pix + 30);
42
43            parent.noFill();
44            parent.noStroke();
45        }
46    }
47
48    public static void dessinerLesBouees() {
49        // MOUILLAGE_NORD:
50        Bouee MFN0 = new Bouee(Zone.MFN0, "MFN0");
51        Bouee MFN1 = new Bouee(Zone.MFN1, "MFN1");
52        Bouee MFN2 = new Bouee(Zone.MFN2, "MFN2");
53        Bouee MFN3 = new Bouee(Zone.MFN3, "MFN3");
54        Bouee MFN4 = new Bouee(Zone.MFN4, "MFN4");
55        Bouee MFN5 = new Bouee(Zone.MFN5, "MFN5");
56        Bouee MFN6 = new Bouee(Zone.MFN6, "MFN6");
57        Bouee MFN7 = new Bouee(Zone.MFN7, "MFN7");
58        Bouee MFN8 = new Bouee(Zone.MFN8, "MFN8");
```

```

59
60    relierBouees(MFN0, MFN1, 0, 0, 100);
61    relierBouees(MFN1, MFN2, 0, 0, 100);
62    relierBouees(MFN2, MFN3, 0, 0, 100);
63    relierBouees(MFN3, MFN4, 0, 0, 100);
64    relierBouees(MFN4, MFN5, 0, 0, 100);
65    relierBouees(MFN5, MFN6, 0, 0, 100);
66    relierBouees(MFN6, MFN7, 0, 0, 100);
67    relierBouees(MFN7, MFN8, 0, 0, 100);
68    relierBouees(MFN8, MFN0, 0, 0, 100);
69
70 // MOUILLAGE_EST:
71
72 Bouee MFE1 = new Bouee(Zone.MFE1, "MFE1");
73 Bouee MFE2 = new Bouee(Zone.MFE2, "MFE2");
74 Bouee MFE3 = new Bouee(Zone.MFE3, "MFE3");
75 Bouee MFE4 = new Bouee(Zone.MFE4, "MFE4");
76 Bouee MFE5 = new Bouee(Zone.MFE5, "MFE5");
77
78 relierBouees(MFE1, MFE2, 0, 0, 100);
79 relierBouees(MFE2, MFE3, 0, 0, 100);
80 relierBouees(MFE3, MFE4, 0, 0, 100);
81 relierBouees(MFE4, MFE5, 0, 0, 100);
82 relierBouees(MFE5, MFE1, 0, 0, 100);
83
84 // MOUILLAGE_OUEST:
85
86 Bouee MFO1 = new Bouee(Zone.MFO1, "MFO1");
87 Bouee MFO2 = new Bouee(Zone.MFO2, "MFO2");
88 Bouee MFO3 = new Bouee(Zone.MFO3, "MFO3");
89 Bouee MFO4 = new Bouee(Zone.MFO4, "MFO4");
90 Bouee MFO5 = new Bouee(Zone.MFO5, "MFO5");
91
92 relierBouees(MFO1, MFO2, 0, 0, 100);
93 relierBouees(MFO2, MFO3, 0, 0, 100);
94 relierBouees(MFO3, MFO4, 0, 0, 100);
95 relierBouees(MFO4, MFO5, 0, 0, 100);
96 relierBouees(MFO5, MFO1, 0, 0, 100);
97
98 // MOUILLAGE_INTERDIT
99 Bouee MIO1 = new Bouee(Zone.MIO1, "MIO1");
100 Bouee MIO2 = new Bouee(Zone.MIO2, "MIO2");
101 Bouee MIO3 = new Bouee(Zone.MIO3, "MIO3");
102 Bouee MIO4 = new Bouee(Zone.MIO4, "MIO4");
103
104 relierBouees(MIO1, MIO2, 0, 0, 100);
105 relierBouees(MIO2, MIO3, 0, 0, 100);
106 relierBouees(MIO3, MIO4, 0, 0, 100);
107 relierBouees(MIO4, MIO1, 0, 0, 100);
108
109 // CHENAL_SUD // Bordure-Est (Rail Montant)
110 Bouee PCO1 = new Bouee(Zone.PCO1, "PCO1");
111 Bouee PCO2 = new Bouee(Zone.PCO2, "PCO2");
112
113 relierBouees(PCO1, PCO2, 157, 2, 178);
114
115 // Séparation
116 Bouee PCO3 = new Bouee(Zone.PCO3, "PCO3");
117 Bouee PCO4 = new Bouee(Zone.PCO4, "PCO4");
118
119 relierBouees(PCO3, PCO4, 157, 2, 178);
120
121 // Bordure-Ouest (Rail Descendant)
122 Bouee PCO5 = new Bouee(Zone.PCO5, "PCO5");
123 Bouee PCO6 = new Bouee(Zone.PCO6, "PCO6");
124
125 relierBouees(PCO5, PCO6, 157, 2, 178);
126
127 // ACCES_LAVERA
128 Bouee PCOL1 = new Bouee(Zone.PCOL1, "PCOL1");
129 Bouee PCOL2 = new Bouee(Zone.PCOL2, "PCOL2");

```

```

130 Bouee PCOL3 = new Bouee(Zone.PCOL3, "PCOL3");
131 Bouee PCOL4 = new Bouee(Zone.PCOL4, "PCOL4");
132 Bouee PCOL5 = new Bouee(Zone.PCOL5, "PCOL5");
133
134 relierBouees(PCOL1, PCOL2, 157, 2, 178);
135 relierBouees(PCOL3, PCOL4, 157, 2, 178);
136 relierBouees(PCOL4, PCOL5, 157, 2, 178);
137
138 // ACCES_GOLFE_FOS
139 Bouee PCOF1 = new Bouee(Zone.PCOF1, "PCOF1");
140 Bouee PCOF2 = new Bouee(Zone.PCOF2, "PCOF2");
141 Bouee PCOF3 = new Bouee(Zone.PCOF3, "PCOF3");
142 Bouee PCOF4 = new Bouee(Zone.PCOF4, "PCOF4");
143 Bouee PCOF5 = new Bouee(Zone.PCOF5, "PCOF5");
144 Bouee PCOF6 = new Bouee(Zone.PCOF6, "PCOF6");
145 Bouee PCOF7 = new Bouee(Zone.PCOF7, "PCOF7");
146 Bouee PCOF8 = new Bouee(Zone.PCOF8, "PCOF8");
147 Bouee PCOF9 = new Bouee(Zone.PCOF9, "PCOF9");
148 Bouee PCOF10 = new Bouee(Zone.PCOF10, "PCOF10");
149
150 relierBouees(PCOF1, PCOF2, 157, 2, 178);
151 relierBouees(PCOF2, PCOF3, 157, 2, 178);
152 relierBouees(PCOF3, PCOF4, 157, 2, 178);
153 relierBouees(PCOF4, PCOF5, 157, 2, 178);
154
155 relierBouees(PCOF6, PCOF7, 157, 2, 178);
156 relierBouees(PCOF7, PCOF8, 157, 2, 178);
157 relierBouees(PCOF8, PCOF9, 157, 2, 178);
158
159 parent.noStroke();
160
161 }
162
163 public static void relierBouees(Bouee b1, Bouee b2, int x, int y, int z) {
164     parent.stroke(x, y, z);
165     parent.line(ConversionPixel.conversionPixelX((float) ConversionMercator
166         .conversion(b1.coordonnees)[0]), ConversionPixel
167         .conversionPixelY((float) ConversionMercator
168             .conversion(b1.coordonnees)[1]), ConversionPixel
169         .conversionPixelX((float) ConversionMercator
170             .conversion(b2.coordonnees)[0]), ConversionPixel
171         .conversionPixelY((float) ConversionMercator
172             .conversion(b2.coordonnees)[1]));
173     parent.noStroke();
174 }
175 }
```

```

1 public class Chemin {
2
3     /* Classe utilisée par le groupe travaillant sur les statistiques .
4      * elle détermine le chemin le plus probable pour un bateau donné .
5      */
6
7     /** Coordonnées de la case */
8     public short x, y;
9     /** Chemin représentant la suite du chemin */
10    public Chemin suiteChemin;
11
12    /** Constructeurs */
13    public Chemin(double[][][] cheminTableau, int indicePremierEtat) {
14        // premierEtat = new Etat(cheminTableau[indicePremierEtat]);
15        x = Etat.coordX(cheminTableau[indicePremierEtat][0]);
16        y = Etat.coordY(cheminTableau[indicePremierEtat][1]);
17        if (indicePremierEtat < cheminTableau.length - 1)
18            suiteChemin = new Chemin(cheminTableau, ++indicePremierEtat);
19        else
20            suiteChemin = null;
21    }
22
23    public Chemin(double[][][] cheminTableau) {
24        this(cheminTableau, 0);
```

```

25 }
26
27 public Chemin(int [][] cheminTableau, int indicePremierEtat) {
28     x = (short) cheminTableau[indicePremierEtat][0];
29     y = (short) cheminTableau[indicePremierEtat][1];
30     if (indicePremierEtat < cheminTableau.length - 1)
31         suiteChemin = new Chemin(cheminTableau, ++indicePremierEtat);
32     else
33         suiteChemin = null;
34 }
35
36 public Chemin(int [][] cheminTableau) {
37     this(cheminTableau, 0);
38 }
39
40 public Chemin() {
41     x = -1;
42     y = -1;
43     suiteChemin = null;
44 }
45
46 public Chemin(short x, short y, Chemin suiteChemin) {
47     this.x = x;
48     this.y = y;
49     this.suiteChemin = suiteChemin;
50 }
51
52 /**
53  * Fonction auxiliaire
54 */
55 public static Chemin cheminNormalise(double [][] cheminTableau, long[] temps, long increment,
56                                     long instantInitial,
57                                     int indicePremierEtat) {
58     while (indicePremierEtat + 1 < temps.length
59           && (temps[indicePremierEtat + 1] < instantInitial || temps[indicePremierEtat] >= temps[
60               indicePremierEtat + 1]))
61         indicePremierEtat++;
62     if (indicePremierEtat + 1 < temps.length) {
63         Chemin chemin = new Chemin();
64         long r = (instantInitial - temps[indicePremierEtat])
65                 / (temps[indicePremierEtat + 1] - temps[indicePremierEtat]);
66         chemin.x = Etat.coordX(cheminTableau[indicePremierEtat][0] + r
67                               * (cheminTableau[indicePremierEtat + 1][0] - cheminTableau[indicePremierEtat][0]));
68         chemin.y = Etat.coordY(cheminTableau[indicePremierEtat][1] + r
69                               * (cheminTableau[indicePremierEtat + 1][1] - cheminTableau[indicePremierEtat][1]));
70         chemin.suiteChemin = cheminNormalise(cheminTableau, temps, increment, instantInitial +
71                                             increment,
72                                             indicePremierEtat);
73     }
74     return chemin;
75 } else if (temps[indicePremierEtat] == instantInitial) {
76     Chemin chemin = new Chemin();
77     chemin.x = Etat.coordX(cheminTableau[indicePremierEtat][0]);
78     chemin.y = Etat.coordY(cheminTableau[indicePremierEtat][1]);
79     return chemin;
80 } else
81     return null;
82 }
83
84 public static Chemin cheminNormalise(double [][] cheminTableau, long[] temps, long increment,
85                                     long instantInitial,
86                                     int indicePremierEtat, int indiceDernierEtat) {
87     while (indicePremierEtat + 1 < indiceDernierEtat
88           && (temps[indicePremierEtat + 1] < instantInitial || temps[indicePremierEtat] >= temps[
89               indicePremierEtat + 1]))
90         indicePremierEtat++;

```

```

91     chemin.suiteChemin = cheminNormalise(cheminTableau, temps, increment, instantInitial +
92         increment,
93         indicePremierEtat);
94     return chemin;
95 } else if (temps[indicePremierEtat] == instantInitial) {
96     Chemin chemin = new Chemin();
97     chemin.x = Etat.coordX(cheminTableau[indicePremierEtat][0]);
98     chemin.y = Etat.coordY(cheminTableau[indicePremierEtat][1]);
99     return chemin;
100 } else
101     return null;
102 }
103 /**
104  * Méthode statique qui renvoie le chemin du bateau donné en argument mais avec un incrément de
105  * temps prédefini */
106 public static Chemin cheminNormalise(BateauAIS bateau, long increment) {
107     return cheminNormalise(bateau.position, bateau.time, increment, bateau.time[0], 0);
108 }
109 public static Chemin cheminNormalise(BateauSys bateau, long increment) {
110     return cheminNormalise(bateau.position, bateau.time, increment, bateau.time[0], 0);
111 }
112 /**
113  * Méthode renvoyant la taille du chemin */
114 public int size() {
115     if (suiteChemin == null)
116         return 1;
117     else if (x == -1)
118         return 0;
119     else
120         return 1 + suiteChemin.size();
121 }
122 /**
123  * Méthode renvoyant la représentation sous forme d'une chaîne de caractères */
124 public String toString() {
125     return "(" + x + "," + y + ")" + (suiteChemin != null ? " - " + suiteChemin : "");
126 }
127 public static Chemin cheminNormalise(BateauSys bateau, long increment, int indicePremierEtat,
128                                         int indiceDernierEtat) {
129     return cheminNormalise(bateau.position, bateau.time, increment, bateau.time[indicePremierEtat
130                                         ],
131                                         indicePremierEtat, indiceDernierEtat);
132 }
133 public int[][] toArray() {
134     int n = size();
135     int[][] array = new int[n][2];
136     Chemin chemin = this;
137     for (int i = 0; i < n; i++) {
138         array[i][0] = chemin.x;
139         array[i][1] = chemin.y;
140         chemin = chemin.suiteChemin;
141     }
142     return array;
143 }

```

```

1
2
3 public class contour {
4
5     public final static int HAUT = 0;
6     public final static int GAUCHE = 1;
7     public final static int BAS = 2;
8     public final static int DROITE = 3;
9
10    public final static int X = perturbation.X;
11    public final static int Y = perturbation.Y;
12    public final static int T = 24 * 60 / 2;
13    public final static int base = 20;

```

```

14
15 /**
16 * Initialise toutes les cases de la matrice à -1
17 *
18 * @param mat : matrice de taille quelconque
19 *
20 * @return void
21 */
22 public static void initialiserMatrice (int [][] mat) {
23     for (int i = 0; i < mat.length; i++) {
24         for (int j = 0; j < mat [0].length; j++) {
25             mat [i] [j] = -1;
26         }
27     }
28 }
29
30 /**
31 * Vérifie si le point est dans la zone (existe dans la matrice carte)
32 */
33 public static boolean existe (int x, int y) {
34     return (x >= 0 && x < X) && (y >= 0 && y < Y);
35 }
36
37
38 /**
39 * Indique le nombre de pas de temps pendant lesquels le bateau reste sur une même case
40 *
41 * @param traj : trajectoire
42 * @param ind0 : indice courant
43 * @param nbPositions : compte pour nbCases positions différentes
44 *
45 * @return res : nombre de cases pour nbPositions différentes autour de ind0
46 */
47 public static int tempsParCase (int [][] traj, int ind0, int nbPositions) {
48     if (nbPositions <= 0 || ind0 >= traj.length) {return 0;}
49     else {
50         int i = 0;
51         int ind = ind0;
52         int x = traj [ind] [0];
53         int y = traj [ind] [1];
54         while (ind >= 1 && traj [ind] [0] == x && traj [ind] [1] == y) {
55             i++;
56             ind--;
57         }
58
59         ind = ind0 + 1;
60         while (ind < traj.length && traj [ind] [0] == x && traj [ind] [1] == y) {
61             i++;
62             ind++;
63         }
64         return i + tempsParCase (traj, ind++, nbPositions -1);
65     }
66 }
67
68 /**
69 * Indique le nombre de pas de temps pendant lesquels le bateau reste sur une même case
70 *
71 * @param traj : trajectoire
72 *
73 * @return res : chaque élément du tableau représente le nombre de cases pour nbPositions
74 * différentes
75 */
76 public static int [] nbDoublons (int [][] traj) {
77     int [] doublons = new int [(traj [0] [0]) / base + 1];
78
79     for (int i = 0; i < (traj [0] [0]) / base + 1; i++) {
80         int k = tempsParCase (traj, (i*base) + 1, 1);
81         doublons [i] = k;
82     }
83     return doublons;
84 }
```

```

84
85 /**
86 * Duplique un élément du tableau
87 *
88 * @param cible : endroit où on copie
89 * @param ind : indice où on commence à copier
90 * @param valeur :
91 *
92 */
93 public static void duplique(int[][] cible, int ind, int[] valeur, int nbFois) {
94     for (int k = 0; k < nbFois; k++) {
95         cible[ind+k] = valeur;
96     }
97 }
98 public static int[][] duplique (int[][] cible, int nbFois) {
99     int [][] res = new int [cible[0][0] * nbFois + 1][2];
100    res [0][0] = cible[0][0] * nbFois;
101    for (int i = 0; i < cible[0][0]; i++) {
102        for (int j = 1; j <= nbFois; j++) {
103            res [i*nbFois+j][0] = cible[i+1][0];
104            res [i*nbFois+j][1] = cible[i+1][1];
105        }
106    }
107    return res;
108 }
109
110 /**
111 * Recalque la trajectoire d'évitement sur la vitesse en trajectoire normale
112 *
113 * @param traj : trajectoire d'évitement
114 * @param doublons : nombre de duplications de case, en fonction de la position dans traj
115 *
116 * @return traj : trajectoire d'évitement prenant en compte la vitesse
117 */
118 public static int[][] accordTemp (int[][] traj, int[] doublons) {
119     int l = traj[0][0];
120     int newL = 1;
121
122     for (int i = 0; i < doublons.length; i++) {
123         if (i != doublons.length-1) {
124             for (int j = 1; j <= l/doublons.length; j++) {
125                 int n = doublons[i];
126                 if (n - (3 * (n/3)) == 0) {
127                     newL += (n/3);
128                 }
129                 else if (n - (3 * (n/3)) == 1) {
130                     if (j%3 == 0) { newL += (n/3); }
131                     else if (j%3 == 1) { newL += (n/3)+1; }
132                     else { newL += (n/3); }
133                 }
134             }
135             else {
136                 if (j%3 == 0) { newL += (n/3)+1; }
137                 else if (j%3 == 1) { newL += (n/3); }
138                 else { newL += (n/3)+1; }
139             }
140         }
141     }
142 }
143 else {
144     for (int j = 1; i*l/doublons.length + j <= l; j++) {
145         int n = doublons[i];
146         if (n - (3 * (n/3)) == 0) {
147             newL += (n/3);
148         }
149         else if (n - (3 * (n/3)) == 1) {
150             if (j%3 == 0) { newL += (n/3); }
151             else if (j%3 == 1) { newL += (n/3)+1; }
152             else { newL += (n/3); }
153         }
154     }
155 }
```

```

155         if (j%3 == 0) { newL += (n/3)+1; }
156     else if (j%3 == 1) { newL += (n/3); }
157     else { newL += (n/3)+1; }
158   }
159 }
160 }
161 }
162 }
163 int [][] trajTemp = new int [newL][2];
164 trajTemp[0][0] = newL - 1;
165 int compteur = 1;
166 for (int i = 0; i < doublons.length; i++) {
167   if (i != doublons.length-1) {
168     for (int j = 1; j <= 1/doublons.length; j++ ) {
169       int n = doublons[i];
170       int ind = i*1/doublons.length + j;
171
172       if (n - (3 * (n/3)) == 0) {
173         duplique(trajTemp, compteur, traj[ind], n/3);
174         compteur += n/3;
175       }
176     else if (n - (3 * (n/3)) == 1) {
177       if (j%3 == 0) { duplique(trajTemp, compteur, traj[ind], n/3); compteur += n/3; }
178       else if (j%3 == 1) { duplique(trajTemp, compteur, traj[ind], (n/3)+1); compteur += (n/3)+1; }
179       else { duplique(trajTemp, compteur, traj[ind], n/3); compteur += n/3; }
180     }
181     else {
182       if (j%3 == 0) { duplique(trajTemp, compteur, traj[ind], (n/3)+1); compteur += (n/3)+1; }
183       else if (j%3 == 1) { duplique(trajTemp, compteur, traj[ind], n/3); compteur += n/3; }
184       else { duplique(trajTemp, compteur, traj[ind], (n/3)+1); compteur += (n/3)+1; }
185     }
186   }
187 }
188 else {
189   for (int j = 1; i*1/doublons.length + j <= 1; j++ ) {
190     int n = doublons[i];
191     int ind = i*1/doublons.length + j;
192
193     if (n - (3 * (n/3)) == 0) {
194       duplique(trajTemp, compteur, traj[ind], n/3);
195       compteur += n/3;
196     }
197     else if (n - (3 * (n/3)) == 1) {
198       if (j%3 == 0) { duplique(trajTemp, compteur, traj[ind], n/3); compteur += n/3; }
199       else if (j%3 == 1) { duplique(trajTemp, compteur, traj[ind], (n/3)+1); compteur += (n/3)+1; }
200       else { duplique(trajTemp, compteur, traj[ind], n/3); compteur += n/3; }
201     }
202     else {
203       if (j%3 == 0) { duplique(trajTemp, compteur, traj[ind], (n/3)+1); compteur += (n/3)+1; }
204       else if (j%3 == 1) { duplique(trajTemp, compteur, traj[ind], n/3); compteur += n/3; }
205       else { duplique(trajTemp, compteur, traj[ind], (n/3)+1); compteur += (n/3)+1; }
206     }
207   }
208 }
209 }
210 return trajTemp;
211 }
212 }
213 }
214 /**
215 * Renvoie le chemin le plus court pour contourner l'obstacle
216 *
217 * @param compteur : matrice créée par calculer_cout_gauche OU par calculer_cout_droite
218 * @param x0 : position initiale (en x)
219 * @param y0 : position initiale (en y)
220 * @param lg : valeur renvoyée par calculer_cout_gauche/droite
221 */

```

```

222 * @return chemin : chemin[0] contient la liste des abscisses des cases visitées (démarre à 1)
223 *         chemin[1] contient la liste des ordonnées des cases visitées (démarre à 1)
224 *         chemin[0][0] contient la longueur utile du tableau chemin (positions entre l'indice
225 *         i et chemin[0][0])
226 */
227 public static int[][] compteurToChemin (int [][] compteur, int x0, int y0, int lg, int arrivee,
228     boolean diagonales) {
229     int [][] chemin = new int [lg+2][2];
230     initialiserMatrice(chemin);
231     int pas = 2;
232     int x = x0, y = y0;
233     boolean continuer = true;
234
235     while (pas < lg+2 && continuer) {
236         int jMax = -1;
237         int kMax = -1;
238         int max = -1;
239         for (int j = -1; j < 2; j++) {
240             for (int k = -1; k < 2; k++) {
241                 if (j == 0 && k == 0) {}
242                 else if (!diagonales && ((j== -1 && (k== -1 || k== 1)) || (j== 1 && (k== -1 || k== 1)))) {}
243                 else {
244                     if (existe(x+j, y+k) && compteur[x+j][y+k] > max) { jMax = j; kMax = k; max = compteur
245                         [x+j][y+k]; }
246                 }
247             }
248             if (pas > 2 && chemin[pas-2][0] == x && chemin[pas-2][1] == y) { continuer = false; }
249             else {
250                 chemin[pas][0] = x;
251                 chemin[pas][1] = y;
252                 pas++;
253             }
254         }
255         chemin[0][0] = pas-1;
256         chemin[0][1] = arrivee;
257         chemin[1][0] = x0;
258         chemin[1][1] = y0;
259         return chemin;
260     }
261 }
262 /**
263 * Vérifie si le tableau {x, y} se trouve après l'indice i dans le tableau tab
264 *
265 * @param tab : tableau de tableaux de longueur 2
266 * @param i : indice à partir duquel commencer la recherche
267 * @param x : première valeur du tableau à rechercher
268 * @param y : deuxième valeur du tableau à rechercher
269 *
270 * @return l'indice de {x, y} dans tab ou -1 si non trouvé
271 */
272 static public int estApres(int [][] tab, int i, int x, int y) {
273     int n = tab.length;
274     int j = i+1;
275
276     while (j < n) {
277         if (tab[j][0] == x && tab[j][1] == y) {
278             return j;
279         }
280         j++;
281     }
282     return -1;
283 }
284
285 /**
286 * Lisse la trajectoire en fin de traitement (minimise les changements de cap)
287 *
288 * @param traj : trajectoire actuelle
289 */

```

```

290 * @return traj : trajectoire lissée
291 */
292 public static int[][] lissageTraj (int[][] traj) {
293     int lg = traj[0][0]; // longueur du trajet
294     int[][] compteur = new int[X][Y];
295     initialiserMatrice(compteur);
296     int k = 0;
297
298     // Remplissage de la matrice compteur
299     for (int i = 1; i < lg; i++) {
300         compteur[traj[i][0]][traj[i][1]] = k;
301         k++;
302     }
303
304     // Nouvelle trajectoire
305     return compteurToChemin(compteur, traj[1][0], traj[1][1], k, 0, false);
306 }
307
308 /**
309 * Calcule la trajectoire modifiée pour éviter l'obstacle le plus efficacement possible
310 *
311 * @param carte : true si pas d'obstacles
312 * @param compteurGauche : matrice qui stocke l'ordre des appels à la fonction
313 *                         calculer_cout_gauche selon le couple (x,y)
314 * @param compteurDroite : matrice qui stocke l'ordre des appels à la fonction
315 *                         calculer_cout_droite selon le couple (x,y)
316 * @param compteurC : valeur actuelle du compteur
317 * @param trajPrevue : trajectoire prévue en l'absence de perturbations
318 * @param indC : indice de la position actuelle dans le tableau trajPrevue
319 */
320 public static int[][] minimiser (boolean[][][] carte, int[][][] compteurGauche, int[][][]
321 compteurDroite, int compteurC, int[][][] trajPrevue, int indC) {
322     // Détermination de la direction normale
323     compteurGauche[trajPrevue[indC][0]][trajPrevue[indC][1]] = compteurC + 1;
324     compteurDroite[trajPrevue[indC][0]][trajPrevue[indC][1]] = compteurC + 1;
325     compteurC += 1;
326
327     int[] retGauche, retDroite;
328     if (trajPrevue[indC+1][0] - trajPrevue[indC][0] > 0) { // l'obstacle est à droite
329         retGauche = calculer_cout_gauche(carte, compteurGauche, compteurC, trajPrevue[indC][0],
330                                         trajPrevue[indC][1], HAUT, trajPrevue, indC, 0);
331         retDroite = calculer_cout_droite(carte, compteurDroite, compteurC, trajPrevue[indC][0],
332                                         trajPrevue[indC][1], BAS, trajPrevue, indC, 0);
333     } else if (trajPrevue[indC+1][0] - trajPrevue[indC][0] < 0) { // l'obstacle est à gauche
334         retGauche = calculer_cout_gauche(carte, compteurGauche, compteurC, trajPrevue[indC][0],
335                                         trajPrevue[indC][1], BAS, trajPrevue, indC, 0);
336         retDroite = calculer_cout_droite(carte, compteurDroite, compteurC, trajPrevue[indC][0],
337                                         trajPrevue[indC][1], HAUT, trajPrevue, indC, 0);
338     } else if (trajPrevue[indC+1][1] - trajPrevue[indC][1] > 0) { // l'obstacle est en haut
339         retGauche = calculer_cout_gauche(carte, compteurGauche, compteurC, trajPrevue[indC][0],
340                                         trajPrevue[indC][1], GAUCHE, trajPrevue, indC, 0);
341         retDroite = calculer_cout_droite(carte, compteurDroite, compteurC, trajPrevue[indC][0],
342                                         trajPrevue[indC][1], DROITE, trajPrevue, indC, 0);
343     } else { // l'obstacle est en bas
344         retGauche = calculer_cout_gauche(carte, compteurGauche, compteurC, trajPrevue[indC][0],
345                                         trajPrevue[indC][1], DROITE, trajPrevue, indC, 0);
346         retDroite = calculer_cout_droite(carte, compteurDroite, compteurC, trajPrevue[indC][0],
347                                         trajPrevue[indC][1], GAUCHE, trajPrevue, indC, 0);
348     }
349     int lgGauche = retGauche[0];
350     int lgDroite = retDroite[0];
351     compteurC += Math.max(lgGauche, lgDroite);
352
353     // Sélection de la meilleure
354     int[][] traj;
355     int arrivee;

```

```

350     if (lgGauche < lgDroite) {
351         arrivee = retGauche[1];
352         traj = compteurToChemin(compteurGauche, trajPrevue[indC][0], trajPrevue[indC][1], lgGauche,
353                                     arrivee, false);
354     }
355     else {
356         arrivee = retDroite[1];
357         traj = compteurToChemin(compteurDroite, trajPrevue[indC][0], trajPrevue[indC][1], lgDroite,
358                                     arrivee, false);
359     }
360     // Duplication pour prendre en compte la vitesse
361     traj = duplique(traj, tempsParCase(trajPrevue, indC, 1));
362
363     /* Fusion de traj et trajPrevue
364      *
365      * 0 -> indC : recopie de trajPrevue
366      * indC+1 -> indC-1 + lg : insertion de traj (évitement de l'obstacle)
367      * indC+1 + lg -> fin : recopie de trajPrevue à partir du point commun à traj et trajPrevue
368      */
369     int i;
370     int[][] newTraj = new int[traj[0][0] + trajPrevue[0][0] - arrivee + indC + 1][2];
371     for (i=1 ; i <= indC; i++) { newTraj[i] = trajPrevue[i]; }
372     for ( ; i <= indC+traj[0][0]; i++) { newTraj[i] = traj[i-indC]; }
373     for ( ; i - indC-traj[0][0] + arrivee <= trajPrevue[0][0]; i++) { newTraj[i] = trajPrevue[i - indC - traj[0][0] + arrivee]; }
374
375     newTraj[0][0] = newTraj.length - 1 ;
376     newTraj[0][1] = indC+traj[0][0];
377
378     return newTraj;
379 }
380
381 /**
382  * Corrige la trajectoire en fonction des obstacles présents sur la carte
383  *
384  * @param carte : true si pas d'obstacle
385  * @param traj : trajectoire envisagée
386  *
387  * @return traj : trajectoire corrigée
388  */
389 public static int[][] parcours (boolean[][][] carte, int[][] traj) {
390     int i = 2;
391     int compteurC = 1;
392     int[][] compteurGauche = new int[X][Y];
393     int[][] compteurDroite = new int[X][Y];
394     initialiserMatrice(compteurGauche);
395     initialiserMatrice(compteurDroite);
396
397     while (i < traj.length) {
398         if (!carte[traj[i][0]][traj[i][1]]) {
399             traj = minimiser(carte, compteurGauche, compteurDroite, compteurC, traj, i-1);
400             compteurC += traj[0][0];
401             i = traj[0][1];
402         }
403         else { i++; }
404     }
405
406     return lissageTraj(traj); //accordTemp(lissageTraj(traj), doublons); // Ici, accordTemp déjà fait
407 }
408
409 /**
410  * Calcule la longueur du contour de l'obstacle, vu lors du contournement gauche/droit
411  *
412  * @param carte : true si pas d'obstacles
413  * @param compteur : matrice qui stocke l'ordre des appels de la fonction selon le couple (x,y)
414  * @param compteurC : valeur courante du compteur
415  * @param x0 : abscisse courante
416  * @param y0 : ordonnée courante

```

```

417 * @param direction : direction courante
418 * @param trajPrevue : tableau des couples position prévus en l'absence de perturbation
419 * @param indC : indice dans le tableau précédent du couple x0,y0
420 * @param lg : longueur du chemin
421 *
422 * @return {lg , arrivee} : longueur du chemin de contournement gauche/droit ; indice de
423 * trajPrevue sur lequel on récupère la trajectoire
424 */
425 public static int[] calculer_cout_gauche (boolean[][] carte , int[][] compteur , int compteurC ,
426     int x0 , int y0 , int direction , int[][] trajPrevue , int indC , int lg) {
427     System.out.printf("%d ; %d\n" , x0 , y0);
428     int arrivee = estApres(trajPrevue , indC , x0 , y0);
429     int[] returned = {lg , arrivee};
430     if (!carte[x0][y0]) { System.err.printf("Appel de calcul_cout_gauche sur une case avec un
431         obstacle"); return new int[0]; }
432     else if (arrivee != -1) { return returned; }
433     else {
434         // Direction du déplacement actuel
435         switch (direction) {
436             case HAUT:
437                 // A droite
438                 if (existe(x0+1, y0) && carte[x0+1][y0]) {
439                     if (compteur[x0+1][y0] != -1) {
440                         return calculer_cout_gauche(carte , compteur , compteurC + 1 , x0+1 , y0 , DROITE,
441                             trajPrevue , indC , compteur[x0+1][y0]);
442                     }
443                 }
444                 // En haut
445                 else if (existe(x0 , y0+1) && carte[x0 ][y0+1]) {
446                     if (compteur[x0 ][y0+1] != -1) {
447                         return calculer_cout_gauche(carte , compteur , compteurC + 1 , x0 , y0+1 , HAUT,
448                             trajPrevue , indC , compteur[x0 ][y0+1]);
449                     }
450                     else {
451                         compteur[x0 ][y0+1] = compteurC + 1;
452                         return calculer_cout_gauche(carte , compteur , compteurC + 1 , x0 , y0+1 , HAUT,
453                             trajPrevue , indC , lg + 1);
454                     }
455                 }
456                 // A gauche
457                 else if (existe(x0-1, y0) && carte[x0-1][y0]) {
458                     if (compteur[x0-1][y0] != -1) {
459                         return calculer_cout_gauche(carte , compteur , compteurC + 1 , x0-1 , y0 , GAUCHE,
460                             trajPrevue , indC , compteur[x0-1][y0]);
461                     }
462                 }
463                 // En bas
464                 else {
465                     return calculer_cout_gauche(carte , compteur , compteurC + 1 , x0 , y0-1 , BAS,
466                         trajPrevue , indC , compteur[x0 ][y0-1]);
467                 }
468             case GAUCHE:
469                 // En haut
470                 if (existe(x0 , y0+1) && carte[x0 ][y0+1]) {
471                     if (compteur[x0 ][y0+1] != -1) {
472                         return calculer_cout_gauche(carte , compteur , compteurC + 1 , x0 , y0+1 , HAUT,
473                             trajPrevue , indC , compteur[x0 ][y0+1]);
474                     }
475                 }

```

```

476         return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue
477             , indC, lg + 1);
478     }
479     // A gauche
480     else if (existe(x0-1, y0) && carte[x0-1][y0]) {
481         if (compteur[x0-1][y0] != -1) {
482             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
483                 trajPrevue, indC, compteur[x0-1][y0]);
484         }
485         else {
486             compteur[x0-1][y0] = compteurC + 1;
487             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
488                 trajPrevue, indC, lg + 1);
489         }
490     // En bas
491     else if (existe(x0, y0-1) && carte[x0][y0-1]) {
492         if (compteur[x0][y0-1] != -1) {
493             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0-1, BAS,
494                 trajPrevue, indC, compteur[x0][y0-1]);
495         }
496         else {
497             compteur[x0][y0-1] = compteurC + 1;
498             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0-1, BAS,
499                 trajPrevue, indC, lg + 1);
500         }
501     // A droite
502     else {
503         return calculer_cout_gauche(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
504             trajPrevue, indC, lg + 1);
505     }
506 case BAS:
507     // A gauche
508     if (existe(x0-1, y0) && carte[x0-1][y0]) {
509         if (compteur[x0-1][y0] != -1) {
510             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
511                 trajPrevue, indC, compteur[x0-1][y0]);
512         }
513         else {
514             compteur[x0-1][y0] = compteurC + 1;
515             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
516                 trajPrevue, indC, lg + 1);
517         }
518     // En bas
519     else if (existe(x0, y0-1) && carte[x0][y0-1]) {
520         if (compteur[x0][y0-1] != -1) {
521             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0-1, BAS,
522                 trajPrevue, indC, compteur[x0][y0-1]);
523         }
524     // En droite
525     else if (existe(x0+1, y0) && carte[x0+1][y0]) {
526         if (compteur[x0+1][y0] != -1) {
527             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
528                 trajPrevue, indC, compteur[x0+1][y0]);
529         }
530         else {
531             compteur[x0+1][y0] = compteurC + 1;
532             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
533                 trajPrevue, indC, lg + 1);
534     }
535 // En haut

```

```

535     else {
536         return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue,
537             indC, compteur[x0][y0+1]);
538     }
539     default: // case DROITE:
540     // En bas
541     if (existe(x0, y0-1) && carte[x0][y0-1]) {
542         if (compteur[x0][y0-1] != -1) {
543             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
544                 indC, compteur[x0][y0-1]);
545         }
546         else {
547             compteur[x0][y0-1] = compteurC + 1;
548             return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
549                 indC, lg + 1);
550         }
551         // A droite
552         else if (existe(x0+1, y0) && carte[x0+1][y0]) {
553             if (compteur[x0+1][y0] != -1) {
554                 return calculer_cout_gauche(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
555                     trajPrevue, indC, compteur[x0+1][y0]);
556             }
557             else {
558                 compteur[x0+1][y0] = compteurC + 1;
559                 return calculer_cout_gauche(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
560                     trajPrevue, indC, lg + 1);
561             }
562             // En haut
563             else if (existe(x0, y0+1) && carte[x0][y0+1]) {
564                 if (compteur[x0][y0+1] != -1) {
565                     return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue
566                         , indC, compteur[x0][y0+1]);
567                 }
568                 else {
569                     compteur[x0][y0+1] = compteurC + 1;
570                     return calculer_cout_gauche(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue
571                         , indC, lg + 1);
572                 }
573             }
574         }
575     }
576
577 /**
578 * Calcule la longueur du contour de l'obstacle, vu lors du contournement gauche/droit
579 *
580 * @param carte : true si pas d'obstacles
581 * @param compteur : matrice qui stocke l'ordre des appels de la fonction selon le couple (x,y)
582 * @param compteurC : valeur courante du compteur
583 * @param x0 : abscisse courante
584 * @param y0 : ordonnée courante
585 * @param direction : direction courante
586 * @param trajPrevue : tableau des couples position prévus en l'absence de perturbation
587 * @param indC : indice dans le tableau précédent du couple x0,y0
588 * @param lg : longueur du chemin
589 *
590 * @return {lg, arrivee} : longueur du chemin de contournement gauche/droit ; indice de
591 *         trajPrevue sur lequel on récupère la trajectoire
592 */
593 public static int[] calculer_cout_droite (boolean[][][] carte, int[][][] compteur, int compteurC,
594     int x0, int y0, int direction, int[][][] trajPrevue, int indC, int lg) {
595     System.out.printf("cd : %d ; %d\n", x0, y0);
596     int arrivee = estApres(trajPrevue, indC, x0, y0);
597     int[] returned = {lg, arrivee};

```

```

596     if (!carte[x0][y0]) { System.out.printf("Appel de calcul_cout_droite sur une case avec un
597         obstacle"); return new int[0]; }
598     else if (arrivee != -1) { return returned; }
599     else {
600         // Direction du déplacement actuel
601         switch (direction) {
602             case HAUT:
603                 // A gauche
604                 if (existe(x0-1, y0) && carte[x0-1][y0]) {
605                     if (compteur[x0-1][y0] != -1) {
606                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
607                                         trajPrevue, indC, compteur[x0-1][y0]);
608                     }
609                     else {
610                         compteur[x0-1][y0] = compteurC + 1;
611                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
612                                         trajPrevue, indC, lg + 1);
613                     }
614                 }
615                 // En haut
616                 else if (existe(x0, y0+1) && carte[x0][y0+1]) {
617                     if (compteur[x0][y0+1] != -1) {
618                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue
619                                         , indC, compteur[x0][y0+1]);
620                     }
621                 }
622                 // A droite
623                 else if (existe(x0+1, y0) && carte[x0+1][y0]) {
624                     if (compteur[x0+1][y0] != -1) {
625                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
626                                         trajPrevue, indC, compteur[x0+1][y0]);
627                     }
628                     else {
629                         compteur[x0+1][y0] = compteurC + 1;
630                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue
631                                         , indC, lg + 1);
632                     }
633                 }
634                 // En bas
635                 else {
636                     return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
637                                         indC, compteur[x0][y0-1]);
638                 }
639             case GAUCHE:
640                 // En bas
641                 if (existe(x0, y0-1) && carte[x0][y0-1]) {
642                     if (compteur[x0][y0-1] != -1) {
643                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
644                                         indC, compteur[x0][y0-1]);
645                     }
646                     else {
647                         compteur[x0][y0-1] = compteurC + 1;
648                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
649                                         indC, lg + 1);
650                     }
651                 }
652                 // A gauche
653                 else if (existe(x0-1, y0) && carte[x0-1][y0]) {
654                     if (compteur[x0-1][y0] != -1) {
655                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
656                                         trajPrevue, indC, compteur[x0-1][y0]);
657                     }
658                     else {
659                         compteur[x0-1][y0] = compteurC + 1;
660                         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
661                                         trajPrevue, indC, lg + 1);
662                 }
663             }
664         }
665     }

```

```

655     }
656 }
// En haut
658 else if (existe(x0, y0+1) && carte[x0][y0+1]) {
659     if (compteur[x0][y0+1] != -1) {
660         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue,
661             , indC, compteur[x0][y0+1]);
662     }
663     else {
664         compteur[x0][y0+1] = compteurC + 1;
665         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue,
666             , indC, lg + 1);
667     }
668 }
// A droite
669 else {
670     return calculer_cout_droite(carte, compteur, compteurC + 1, x0+1, y0, DROITE, trajPrevue,
671         , indC, compteur[x0][y0+1]);
672 }
case BAS:
// En droite
673 if (existe(x0+1, y0) && carte[x0+1][y0]) {
674     if (compteur[x0+1][y0] != -1) {
675         return calculer_cout_droite(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
676             trajPrevue, indC, compteur[x0+1][y0]);
677     }
678     else {
679         compteur[x0+1][y0] = compteurC + 1;
680         return calculer_cout_droite(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
681             trajPrevue, indC, lg + 1);
682     }
683 }
// En bas
684 else if (existe(x0, y0-1) && carte[x0][y0-1]) {
685     if (compteur[x0][y0-1] != -1) {
686         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
687             indC, compteur[x0][y0-1]);
688     }
689     else {
690         compteur[x0][y0-1] = compteurC + 1;
691         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
692             indC, lg + 1);
693     }
694 }
// A gauche
695 else if (existe(x0-1, y0) && carte[x0-1][y0]) {
696     if (compteur[x0-1][y0] != -1) {
697         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
698             trajPrevue, indC, compteur[x0-1][y0]);
699     }
700     else {
701         compteur[x0-1][y0] = compteurC + 1;
702         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE,
703             trajPrevue, indC, lg + 1);
704     }
705 }
// En haut
706 else {
707     return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue,
708         , indC, compteur[x0][y0+1]);
709 }
default: // case DROITE:
// En haut
710 if (existe(x0, y0+1) && carte[x0][y0+1]) {
711     if (compteur[x0][y0+1] != -1) {
712         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue,
713             , indC, compteur[x0][y0+1]);
714     }
715     else {
716         compteur[x0][y0+1] = compteurC + 1;

```

```

714         return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0+1, HAUT, trajPrevue
715             , indC, lg + 1);
716     }
717     // A droite
718     else if (existe(x0+1, y0) && carte[x0+1][y0]) {
719         if (compteur[x0+1][y0] != -1) {
720             return calculer_cout_droite(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
721                 trajPrevue, indC, compteur[x0+1][y0]);
722         }
723         else {
724             compteur[x0+1][y0] = compteurC + 1;
725             return calculer_cout_droite(carte, compteur, compteurC + 1, x0+1, y0, DROITE,
726                 trajPrevue, indC, lg + 1);
727         }
728     // En bas
729     else if (existe(x0, y0-1) && carte[x0][y0-1]) {
730         if (compteur[x0][y0-1] != -1) {
731             return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
732                 indC, compteur[x0][y0-1]);
733         }
734         else {
735             compteur[x0][y0-1] = compteurC + 1;
736             return calculer_cout_droite(carte, compteur, compteurC + 1, x0, y0-1, BAS, trajPrevue,
737                 indC, lg + 1);
738     // A gauche
739     else {
740         return calculer_cout_droite(carte, compteur, compteurC + 1, x0-1, y0, GAUCHE, trajPrevue
741             , indC, compteur[x0-1][y0]);
742     }
743 }
744 /**
745 * Affiche la carte (int) en repère cartésien
746 */
747 public static void afficher(int [][] a) {
748     int n = a[0].length;
749     int m = a.length;
750
751     for (int i=n-1; i >= 0; i--) {
752         for (int j=0; j < m; j++) {
753             System.out.printf("%2d | ", a[j][i]);
754         }
755         System.out.println();
756     }
757 }
758 /**
759 * Affiche la carte (boolean) en repère cartésien
760 */
761 public static void afficher(boolean [][] a) {
762     int n = a[0].length;
763     int m = a.length;
764
765     for (int i=n-1; i >= 0; i--) {
766         for (int j=0; j < m; j++) {
767             if (a[j][i]) { System.out.printf(" 1 |"); }
768             else { System.out.printf(" 0 |"); }
769         }
770         System.out.println();
771     }
772 }
773 /**
774 * Affiche le tableau des positions dans l'ordre
775 */
776 public static void afficher_traj(int [][] a) {
777     int n = a.length;

```

```

779 int m = a[0].length;
780
781 for (int i=0; i < n; i++) {
782     for (int j=0; j < m; j++) {
783         System.out.printf("%2d | ", a[i][j]);
784     }
785     System.out.println();
786 }
787 }
788 /**
789 * Affiche le tableau
790 */
791 public static void afficher(int[] a) {
792     int n = a.length;
793     for (int i=0; i < n; i++) {
794         System.out.printf("%2d | ", a[i]);
795     }
796 }
797 }
798 }
```

```

1 public class ConversionMercator {
2     // Conversion des coordonnées géographiques en coordonnées métriques.
3     // !!!!!! CONVERSION PREND EN ARGUMENT UN TABLEAU [LATITUDE,LONGITUDE].
4
5     /**
6      * Les valeurs de E et N sont en kilomètres, u0 correspond à la longitude du
7      * Port de Marseille (longitude du méridien de référence)
8      *
9      * Le tableau Z doit contenir Latitude , Longitude en degré
10     */
11
12     public static double[] conversion(double[] Z) {
13
14         // Définition des constantes :
15         final double E0 = 635.1225650830115; // E origine
16         final double P0 = 4791.737721766792; // N origine
17         // final double E0 = 658.2674789424228; // E origine
18         // final double P0 = 4804.078096325402; // N origine
19         final double u0 = 3 * 3.14159265 / 180;
20         final double N0 = 0; // 0 dans l'hémisphère Nord, 10000km dans le sud
21         final double a = 6378.137; // Rayon à l'équateur , km
22         final double e2 = 0.0818192; // Excentricité
23         final double k0 = 0.9996;
24         final double Pi = 3.14159265;
25
26         // Définition des paramètres :
27         double i = Z[0]; // Latitude géodésique en degré
28         double j = Z[1]; // Longitude en degré
29
30         double v = j * Pi / 180; // Latitude géodésique en radian
31         double u = i * Pi / 180; // Longitude en radian
32
33         double e; // Coordonnées vers l'Est, en kilomètres
34         double n; // Coordonnées vers le Nord, en kilomètres
35
36         // Calculs :
37         double vPhi = 1 / (Math.sqrt(1 - (e2 * e2 * Math.sin(u) * Math.sin(u))));
38         double A = (v - u0) * Math.cos(u);
39         double T = Math.tan(u) * Math.tan(u);
40         double C = Math.cos(u) * Math.cos(u) * (e2 * e2) / (1 - e2 * e2);
41         double sPhi = ((1 - (e2 * e2 / 4) - 3 * Math.pow(e2, 4) / 64 - 5 * Math
42             .pow(e2, 6) / 256) * u)
43             - ((3 * Math.pow(e2, 2) / 8 + 3 * Math.pow(e2, 4) / 32 + 45 * Math
44                 .pow(e2, 6) / 1024))
45             * Math.sin(2 * u)
46             + (15 * Math.pow(e2, 4) / 256)
47             + (45 * Math.pow(e2, 6) / 1024)
48             * Math.sin(4 * u)
49             - 35
50             * Math.pow(e2, 6)
```

```

51     * Math.sin(6 * u)
52     / 3072;
53
54     e = 500
55     + k0
56     * a
57     * vPhi
58     * (A + (1 - T + C) * Math.pow(A, 3) / 6 + (5 - 18 * T + T * T)
59     * Math.pow(A, 5) / 120);
60     n = N0
61     + k0
62     * a
63     * (sPhi + vPhi
64     * Math.tan(u)
65     * (A * A / 2 + (5 - T + 9 * C + 4 * C * C)
66     * Math.pow(A, 4) / 24 + (61 - 58 * T + T * T)
67     * Math.pow(A, 6) / 720));
68
69     double[] R = {e - E0, n - P0};
70     return R;
71 }
72 }
73 }
74 }
```

```

1 public class ConversionPixel {
2     //Conversion des coordonnées métriques en coordonnées "pixels" sur le canvas.
3
4     //Pour la carte d'Hassan.
5     public static float conversionPixelX(float x){
6         x=x*(float)(interfacegraphique.SIZE_X/32.315)-(float)(61.05/889*interfacegraphique.SIZE_X);
7         return x;
8     }
9
10    public static float conversionPixelY(float y){
11        y=-y*(float)(interfacegraphique.SIZE_Y/18.175)+(float)(578.20/500*interfacegraphique.SIZE_Y);
12        return y;
13    }
14
15    //Fonction inverse
16
17    public static float deconversionPixelX(float x){
18        x=((float)(x+61.05))/((float)27.51);
19        return x;
20    }
21
22    public static float deconversionPixelY(float y){
23        y=((float)(y-578.20))/((float)27.51);
24        return y;
25    }
26
27
28 }
29 }
```

```

1 import processing.core.*;
2
3     /* Groupe des prévisions "statistiques": Affichage des
4      résultats sous forme de carrés colorés sur la carte en
5      fonction de la probabilité de présence*/
6
7     public class DegradeStatistique {
8
9         public static PApplet parent;
10
11         public static double[][][] mat() {
12             double[][][] mat = new double[10][25][10];
13             for (int i = 0; i < 10; i++)
14                 for (int j = 0; j < 25; j++)
15                     mat[i][j][i] = .1 * i;
```

```

16     return mat;
17 }
18
19 public static void animer(double[][][] mat) {
20     if (parent == null) {
21         parent = interfacegraphique.truc;
22     }
23     float xcase = interfacegraphique.SIZE_X / mat[0].length;
24     float ycase = interfacegraphique.SIZE_Y / mat[0][0].length;
25     for (int i = 0; i < mat[0].length; i++) {
26         for (int j = 0; j < mat[0][0].length; j++) {
27             if (mat[interfacegraphique.TIME % mat.length][i][j] != 0) {
28                 parent.noStroke();
29                 parent.fill((float) 255, (float) 0, (float) 0,
30                             (float) (255 * mat[interfacegraphique.TIME
31                                         % mat.length][i][j]));
32                 parent.rect(i * xcase, (Etat.N2 - j - 1) * ycase, xcase,
33                             ycase);
34             }
35         }
36     }
37 }
38 }
39 }
40 }
```

```

1 import processing.core.*;
2
3 // Affichage des bateaux, de leur trajectoire et des informations qui leurs sont relatives.
4 public class Dessiner {
5     public static PApplet parent;
6
7     // méthode principale à appliquer à flotte []:
8
9     public static void dessiner() {
10
11        for (int i = 0; i < GestionPort.flotte.getBateaux().length; i++) {
12            Position position = GestionPort.flotte.getBateau(i).getPosNorm(
13                interfacegraphique.TIME);
14            if (position == null) {
15            } else {
16
17                float x = ConversionPixel
18                    .conversionPixelX((float) GestionPort.flotte
19                        .getBateau(i)
20                        .getPosNorm(interfacegraphique.TIME).getX());
21                float y = ConversionPixel
22                    .conversionPixelY((float) GestionPort.flotte
23                        .getBateau(i)
24                        .getPosNorm(interfacegraphique.TIME).getY());
25                if (GestionPort.flotte.getBateau(i).getPosNorm(
26                    interfacegraphique.TIME + 1) == null) {
27
28                    float sens = (float) (GestionPort.flotte.getBateau(i)
29                        .getPosNorm(interfacegraphique.TIME).getCap());
30
31                    if (parent == null) {
32                        parent = interfacegraphique.truc;
33                    }
34
35                    if (GestionPort.flotte.getBateau(i).getNumeroTypeBateau() <= 49
36                        && GestionPort.flotte.getBateau(i)
37                            .getNumeroTypeBateau() >= 40) {
38                        parent.noStroke();
39                        parent.fill(250, 66, 132, 200);
40                        parent.pushMatrix();
41                        parent.translate(x, y);
42                        parent.rotate((float) (sens));
43                        parent.rectMode(PConstants.CENTER);
44                        parent.rect((float) 0.0, (float) 0.0, (float) 5,
45                                    (float) 2.2);
```

```

46     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
47                     (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
48     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
49                     (float) -5 / 2, (float) -2.2 / 2, (float) -5, 0);
50     parent.popMatrix();
51     parent.stroke(200);
52 } else if (GestionPort.flotte.getBateau(i)
53             .getNumeroTypeBateau() == 50) {
54     parent.noStroke();
55     parent.fill(255, 114, 9, 200);
56     parent.pushMatrix();
57     parent.translate(x, y);
58     parent.rotate((float) (sens));
59     parent.rectMode(PConstants.CENTER);
60     parent.rect((float) 0.0, (float) 0.0, (float) 3,
61                 (float) 2);
62     parent.triangle((float) 3 / 2, (float) 1,
63                     (float) 3 / 2, (float) -1, (float) 3, 0);
64     parent.triangle((float) -3 / 2, (float) -1,
65                     (float) -3 / 2, (float) -1, (float) -3, 0);
66     parent.popMatrix();
67     parent.stroke(200);
68 } else if (GestionPort.flotte.getBateau(i)
69             .getNumeroTypeBateau() <= 69
70             && GestionPort.flotte.getBateau(i)
71             .getNumeroTypeBateau() >= 60) {
72     parent.noStroke();
73     parent.fill(226, 234, 0, 200);
74     parent.pushMatrix();
75     parent.translate(x, y);
76     parent.rotate((float) (sens));
77     parent.rectMode(PConstants.CENTER);
78     parent.rect((float) 0.0, (float) 0.0, (float) 8,
79                 (float) 3);
80     parent.triangle((float) 8 / 2, (float) 3 / 2,
81                     (float) 8 / 2, (float) -3 / 2, (float) 8, 0);
82     parent.triangle((float) -8 / 2, (float) -3 / 2,
83                     (float) -8 / 2, (float) -3 / 2, (float) -8, 0);
84     parent.popMatrix();
85     parent.stroke(200);
86 } else if (GestionPort.flotte.getBateau(i)
87             .getNumeroTypeBateau() == 30) {
88     parent.noStroke();
89     parent.fill(35, 232, 0, 200);
90     parent.pushMatrix();
91     parent.translate(x, y);
92     parent.rotate((float) (sens));
93     parent.rectMode(PConstants.CENTER);
94     parent.rect((float) 0.0, (float) 0.0, (float) 5,
95                 (float) 2.2);
96     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
97                     (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
98     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
99                     (float) -5 / 2, (float) -2.2 / 2, (float) -5, 0);
100    parent.popMatrix();
101    parent.stroke(200);
102 } else if (GestionPort.flotte.getBateau(i)
103             .getNumeroTypeBateau() <= 79
104             && GestionPort.flotte.getBateau(i)
105             .getNumeroTypeBateau() >= 70) {
106     parent.noStroke();
107     parent.fill(218, 0, 236, 200);
108     parent.pushMatrix();
109     parent.translate(x, y);
110     parent.rotate((float) (sens));
111     parent.rectMode(PConstants.CENTER);
112     parent.rect((float) 0.0, (float) 0.0, (float) 8,
113                 (float) 3);
114     parent.triangle((float) 8 / 2, (float) 3 / 2,
115                     (float) 8 / 2, (float) -3 / 2, (float) 8, 0);
116     parent.triangle((float) -8 / 2, (float) -3 / 2,

```

```

117         (float) -8 / 2, (float) -3 / 2, (float) -8, 0);
118     parent.popMatrix();
119     parent.stroke(200);
120 } else if (GestionPort.flotte.getBateau(i)
121     .getNumeroTypeBateau() <= 89
122     && GestionPort.flotte.getBateau(i)
123     .getNumeroTypeBateau() >= 80) {
124     parent.noStroke();
125     parent.fill(26, 6, 255, 200);
126     parent.pushMatrix();
127     parent.translate(x, y);
128     parent.rotate((float) (sens));
129     parent.rectMode(PConstants.CENTER);
130     parent.rect((float) 0.0, (float) 0.0, (float) 8,
131     (float) 3);
132     parent.triangle((float) 8 / 2, (float) 3 / 2,
133     (float) 8 / 2, (float) -3 / 2, (float) 8, 0);
134     parent.triangle((float) -8 / 2, (float) -3 / 2,
135     (float) -8 / 2, (float) -3 / 2, (float) -8, 0);
136     parent.popMatrix();
137     parent.stroke(200);
138 } else if (GestionPort.flotte.getBateau(i)
139     .getNumeroTypeBateau() == 35
140     || GestionPort.flotte.getBateau(i)
141     .getNumeroTypeBateau() == 51
142     || GestionPort.flotte.getBateau(i)
143     .getNumeroTypeBateau() == 55
144     || GestionPort.flotte.getBateau(i)
145     .getNumeroTypeBateau() == 58) {
146     parent.noStroke();
147     parent.fill(255, 0, 0, 200);
148     parent.pushMatrix();
149     parent.translate(x, y);
150     parent.rotate((float) (sens));
151     parent.rectMode(PConstants.CENTER);
152     parent.rect((float) 0.0, (float) 0.0, (float) 5,
153     (float) 2.2);
154     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
155     (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
156     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
157     (float) -5 / 2, (float) -2.2 / 2, (float) -5, 0);
158     parent.popMatrix();
159     parent.stroke(200);
160 } else if (GestionPort.flotte.getBateau(i)
161     .getNumeroTypeBateau() == 53) {
162     parent.noStroke();
163     parent.fill(255, 114, 9, 200);
164     parent.pushMatrix();
165     parent.translate(x, y);
166     parent.rotate((float) (sens));
167     parent.rectMode(PConstants.CENTER);
168     parent.rect((float) 0.0, (float) 0.0, (float) 3,
169     (float) 2);
170     parent.triangle((float) 3 / 2, (float) 1,
171     (float) 3 / 2, (float) -1, (float) 3, 0);
172     parent.triangle((float) -3 / 2, (float) -1,
173     (float) -3 / 2, (float) -1, (float) -3, 0);
174     parent.popMatrix();
175     parent.stroke(200);
176 } else {
177     parent.noStroke();
178     parent.fill(0, 0, 0, 200);
179     parent.pushMatrix();
180     parent.translate(x, y);
181     parent.rotate((float) (sens));
182     parent.rectMode(PConstants.CENTER);
183     parent.rect((float) 0.0, (float) 0.0, (float) 5,
184     (float) 2.2);
185     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
186     (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
187     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
```

```

188         (float) -5 / 2, (float) -2.2 / 2, (float) -5, 0);
189     parent.popMatrix();
190     parent.stroke(200);
191 }
192
193 } else {
194     float x1 = ConversionPixel
195         .conversionPixelX((float) GestionPort.flotte
196             .getBateau(i)
197             .getPosNorm(interfacegraphique.TIME + 1)
198             .getX());
199     float y1 = ConversionPixel
200         .conversionPixelY((float) GestionPort.flotte
201             .getBateau(i)
202             .getPosNorm(interfacegraphique.TIME + 1)
203             .getY());
204
205     if (parent == null) {
206         parent = interfacegraphique.truc;
207     }
208
209     if (GestionPort.flotte.getBateau(i).getNumeroTypeBateau() <= 49
210         && GestionPort.flotte.getBateau(i)
211             .getNumeroTypeBateau() >= 40) {
212         parent.noStroke();
213         parent.fill(250, 66, 132, 200);
214         parent.pushMatrix();
215         parent.translate(x, y);
216         if (Math.abs(x - x1) < 2) {
217             parent.rotate((float) -Math.PI / 2);
218         } else {
219             parent.rotate((float) (Math
220                 .atan((float) ((y1 - y) / (x1 - x)))));
221         }
222         parent.rectMode(PConstants.CENTER);
223         parent.rect((float) 0.0, (float) 0.0, (float) 5,
224             (float) 2.2);
225         parent.triangle((float) 5 / 2, (float) 2.2 / 2,
226             (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
227         parent.triangle((float) -5 / 2, (float) -2.2 / 2,
228             (float) -5 / 2, (float) -2.2 / 2, (float) -5, 0);
229         parent.popMatrix();
230         parent.stroke(200);
231     } else if (GestionPort.flotte.getBateau(i)
232         .getNumeroTypeBateau() == 50) {
233         parent.noStroke();
234         parent.fill(255, 114, 9, 200);
235         parent.pushMatrix();
236         parent.translate(x, y);
237         if (Math.abs(x - x1) < 2) {
238             parent.rotate((float) -Math.PI / 2);
239         } else {
240             parent.rotate((float) (Math
241                 .atan((float) ((y1 - y) / (x1 - x)))));
242         }
243         parent.rectMode(PConstants.CENTER);
244         parent.rect((float) 0.0, (float) 0.0, (float) 3,
245             (float) 2);
246         parent.triangle((float) 3 / 2, (float) 1,
247             (float) 3 / 2, (float) -1, (float) 3, 0);
248         parent.triangle((float) -3 / 2, (float) -1,
249             (float) -3 / 2, (float) -1, (float) -3, 0);
250         parent.popMatrix();
251         parent.stroke(200);
252     } else if (GestionPort.flotte.getBateau(i)
253         .getNumeroTypeBateau() <= 69
254         && GestionPort.flotte.getBateau(i)
255             .getNumeroTypeBateau() >= 60) {
256         parent.noStroke();
257         parent.fill(226, 234, 0, 200);
258         parent.pushMatrix();

```

```

259     parent.translate(x, y);
260     if (Math.abs(x - x1) < 2) {
261         parent.rotate((float) -Math.PI / 2);
262     } else {
263         parent.rotate((float) (Math
264             .atan((float) ((y1 - y) / (x1 - x)))));
265     }
266     parent.rectMode(PConstants.CENTER);
267     parent.rect((float) 0.0, (float) 0.0, (float) 8,
268             (float) 3);
269     parent.triangle((float) 8 / 2, (float) 3 / 2,
270             (float) 8 / 2, (float) -3 / 2, (float) 8, 0);
271     parent.triangle((float) -8 / 2, (float) -3 / 2,
272             (float) -8 / 2, (float) 3 / 2, (float) -8, 0);
273     parent.popMatrix();
274     parent.stroke(200);
275 } else if (GestionPort.flotte.getBateau(i)
276     .getNumeroTypeBateau() == 30) {
277     parent.noStroke();
278     parent.fill(35, 232, 0, 200);
279     parent.pushMatrix();
280     parent.translate(x, y);
281     if (Math.abs(x - x1) < 2) {
282         parent.rotate((float) -Math.PI / 2);
283     } else {
284         parent.rotate((float) (Math
285             .atan((float) ((y1 - y) / (x1 - x)))));
286     }
287     parent.rectMode(PConstants.CENTER);
288     parent.rect((float) 0.0, (float) 0.0, (float) 5,
289             (float) 2.2);
290     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
291             (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
292     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
293             (float) -5 / 2, (float) 2.2 / 2, (float) -5, 0);
294     parent.popMatrix();
295     parent.stroke(200);
296 } else if (GestionPort.flotte.getBateau(i)
297     .getNumeroTypeBateau() <= 79
298     && GestionPort.flotte.getBateau(i)
299     .getNumeroTypeBateau() >= 70) {
300     parent.noStroke();
301     parent.fill(218, 0, 236, 200);
302     parent.pushMatrix();
303     parent.translate(x, y);
304     if (Math.abs(x - x1) < 2) {
305         parent.rotate((float) -Math.PI / 2);
306     } else {
307         parent.rotate((float) (Math
308             .atan((float) ((y1 - y) / (x1 - x)))));
309     }
310     parent.rectMode(PConstants.CENTER);
311     parent.rect((float) 0.0, (float) 0.0, (float) 8,
312             (float) 3);
313     parent.triangle((float) 8 / 2, (float) 3 / 2,
314             (float) 8 / 2, (float) -3 / 2, (float) 8, 0);
315     parent.triangle((float) -8 / 2, (float) -3 / 2,
316             (float) -8 / 2, (float) 3 / 2, (float) -8, 0);
317     parent.popMatrix();
318     parent.stroke(200);
319 } else if (GestionPort.flotte.getBateau(i)
320     .getNumeroTypeBateau() <= 89
321     && GestionPort.flotte.getBateau(i)
322     .getNumeroTypeBateau() >= 80) {
323     parent.noStroke();
324     parent.fill(26, 6, 255, 200);
325     parent.pushMatrix();
326     parent.translate(x, y);
327     if (Math.abs(x - x1) < 2) {
328         parent.rotate((float) -Math.PI / 2);
329     } else {

```

```

330     parent.rotate((float) (Math
331         .atan((float) ((y1 - y) / (x1 - x))))) ;
332 }
333 parent.rectMode(PConstants.CENTER);
334 parent.rect((float) 0.0, (float) 0.0, (float) 8,
335     (float) 3);
336 parent.triangle((float) 8 / 2, (float) 3 / 2,
337     (float) 8 / 2, (float) -3 / 2, (float) 8, 0);
338 parent.triangle((float) -8 / 2, (float) -3 / 2,
339     (float) -8 / 2, (float) -3 / 2, (float) -8, 0);
340 parent.popMatrix();
341 parent.stroke(200);
342 } else if (GestionPort.flotte.getBateau(i)
343     .getNumeroTypeBateau() == 35
344     || GestionPort.flotte.getBateau(i)
345     .getNumeroTypeBateau() == 51
346     || GestionPort.flotte.getBateau(i)
347     .getNumeroTypeBateau() == 55
348     || GestionPort.flotte.getBateau(i)
349     .getNumeroTypeBateau() == 58) {
350     parent.noStroke();
351     parent.fill(255, 0, 0, 200);
352     parent.pushMatrix();
353     parent.translate(x, y);
354     if (Math.abs(x - x1) < 2) {
355         parent.rotate((float) -Math.PI / 2);
356     } else {
357         parent.rotate((float) (Math
358             .atan((float) ((y1 - y) / (x1 - x))))) ;
359     }
360     parent.rectMode(PConstants.CENTER);
361     parent.rect((float) 0.0, (float) 0.0, (float) 5,
362     (float) 2.2);
363     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
364     (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
365     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
366     (float) -5 / 2, (float) -2.2 / 2, (float) -5, 0);
367     parent.popMatrix();
368     parent.stroke(200);
369 } else if (GestionPort.flotte.getBateau(i)
370     .getNumeroTypeBateau() == 53) {
371     parent.noStroke();
372     parent.fill(255, 114, 9, 200);
373     parent.pushMatrix();
374     parent.translate(x, y);
375     if (Math.abs(x - x1) < 2) {
376         parent.rotate((float) -Math.PI / 2);
377     } else {
378         parent.rotate((float) (Math
379             .atan((float) ((y1 - y) / (x1 - x))))) ;
380     }
381     parent.rectMode(PConstants.CENTER);
382     parent.rect((float) 0.0, (float) 0.0, (float) 3,
383     (float) 2);
384     parent.triangle((float) 3 / 2, (float) 1,
385     (float) 3 / 2, (float) -1, (float) 3, 0);
386     parent.triangle((float) -3 / 2, (float) -1,
387     (float) -3 / 2, (float) -1, (float) -3, 0);
388     parent.popMatrix();
389     parent.stroke(200);
390 } else {
391     parent.noStroke();
392     parent.fill(0, 0, 0, 200);
393     parent.pushMatrix();
394     parent.translate(x, y);
395     if (Math.abs(x - x1) < 2) {
396         parent.rotate((float) -Math.PI / 2);
397     } else {
398         parent.rotate((float) (Math
399             .atan((float) ((y1 - y) / (x1 - x))))) ;
400     }

```

```

401     parent.rectMode(PConstants.CENTER);
402     parent.rect((float) 0.0, (float) 0.0, (float) 5,
403                 (float) 2.2);
404     parent.triangle((float) 5 / 2, (float) 2.2 / 2,
405                     (float) 5 / 2, (float) -2.2 / 2, (float) 5, 0);
406     parent.triangle((float) -5 / 2, (float) -2.2 / 2,
407                     (float) -5 / 2, (float) 2.2 / 2, (float) -5, 0);
408     parent.popMatrix();
409     parent.stroke(200);
410   }
411
412 }
413 // affichage des infos en posant la souris sur le bateau
414
415 if (PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
416                   x, y) <= 6) {
417
418   parent.textSize(13);
419   parent.rectMode(PConstants.CENTER);
420   parent.stroke(50, 50, 50);
421   parent.fill(152, 152, 152);
422   parent.rect((float) (float) interfacegraphique.SIZE_X
423             * (float) 10 / 11,
424             (float) (float) interfacegraphique.SIZE_Y
425             * (float) 43 / 50, (float) 140 / 889
426             * (float) interfacegraphique.SIZE_X,
427             (float) 4 / 25 * (float) interfacegraphique.SIZE_Y,
428             18, 18, 18, 18);
429   parent.fill(0, 0, 0, 255);
430   parent.textAlign(PConstants.CENTER);
431   parent.text("MMSI: "
432             + GestionPort.flotte.getBateau(i).getMMSI(),
433             (float) (float) interfacegraphique.SIZE_X
434             * (float) 10 / 11,
435             (float) (float) interfacegraphique.SIZE_Y
436             - (float) 94 / 500
437             * (float) interfacegraphique.SIZE_Y);
438   parent.text(
439     "Position: (" +
440     + Truncate
441       .TruncateDouble(
442         GestionPort.flotte
443           .getBateau(i)
444             .getPosNorm(
445               interfacegraphique.TIME)
446             .getLatitude(), 4)
447     + ", "
448     + " "
449     + Truncate
450       .TruncateDouble(
451         GestionPort.flotte
452           .getBateau(i)
453             .getPosNorm(
454               interfacegraphique.TIME)
455             .getLongitude(), 4)
456     + ")",
457     (float) (float) interfacegraphique.SIZE_X
458     * (float) 10 / 11,
459     (float) (float) interfacegraphique.SIZE_Y
460     - (float) 82 / 500
461     * (float) interfacegraphique.SIZE_Y);
462   parent.text(
463     "Vitesse: "
464     + " "
465     + Truncate.TruncateDouble(
466       position.getSog(), 1) + " "
467     + "Noeuds",
468     (float) (float) interfacegraphique.SIZE_X
469     * (float) 10 / 11,
470     (float) (float) interfacegraphique.SIZE_Y
471     - (float) 58 / 500

```

```

472     * (float) interfacegraphique.SIZE_Y);
473 parent.text("Cap:" + " " + position.getCap() + " "
474     + "degrés",
475     (float) (float) interfacegraphique.SIZE_X
476         * (float) 10 / 11,
477     (float) (float) interfacegraphique.SIZE_Y
478         - (float) 70 / 500
479         * (float) interfacegraphique.SIZE_Y);
480
481 if (GestionPort.flotte.getBateau(i).getNumeroTypeBateau() <= 49
482     && GestionPort.flotte.getBateau(i)
483         .getNumeroTypeBateau() >= 40) {
484     parent.text("High Speed Craft",
485         (float) (float) interfacegraphique.SIZE_X
486             * (float) 10 / 11,
487         (float) (float) interfacegraphique.SIZE_Y
488             - (float) 46 / 500
489             * (float) interfacegraphique.SIZE_Y);
490 } else if (GestionPort.flotte.getBateau(i)
491     .getNumeroTypeBateau() == 50) {
492     parent.text("Bateau pilote",
493         (float) (float) interfacegraphique.SIZE_X
494             * (float) 10 / 11,
495         (float) (float) interfacegraphique.SIZE_Y
496             - (float) 46 / 500
497             * (float) interfacegraphique.SIZE_Y);
498 } else if (GestionPort.flotte.getBateau(i)
499     .getNumeroTypeBateau() <= 69
500     && GestionPort.flotte.getBateau(i)
501         .getNumeroTypeBateau() >= 60) {
502     parent.text("Bateau de croisière",
503         (float) (float) interfacegraphique.SIZE_X
504             * (float) 10 / 11,
505         (float) (float) interfacegraphique.SIZE_Y
506             - (float) 46 / 500
507             * (float) interfacegraphique.SIZE_Y);
508 } else if (GestionPort.flotte.getBateau(i)
509     .getNumeroTypeBateau() == 30) {
510     parent.text("Bateau de pêche",
511         (float) (float) interfacegraphique.SIZE_X
512             * (float) 10 / 11,
513         (float) (float) interfacegraphique.SIZE_Y
514             - (float) 46 / 500
515             * (float) interfacegraphique.SIZE_Y);
516 } else if (GestionPort.flotte.getBateau(i)
517     .getNumeroTypeBateau() <= 79
518     && GestionPort.flotte.getBateau(i)
519         .getNumeroTypeBateau() >= 70) {
520     parent.text("Cargo",
521         (float) (float) interfacegraphique.SIZE_X
522             * (float) 10 / 11,
523         (float) (float) interfacegraphique.SIZE_Y
524             - (float) 46 / 500
525             * (float) interfacegraphique.SIZE_Y);
526 } else if (GestionPort.flotte.getBateau(i)
527     .getNumeroTypeBateau() <= 89
528     && GestionPort.flotte.getBateau(i)
529         .getNumeroTypeBateau() >= 80) {
530     parent.text("Tanker",
531         (float) (float) interfacegraphique.SIZE_X
532             * (float) 10 / 11,
533         (float) (float) interfacegraphique.SIZE_Y
534             - (float) 46 / 500
535             * (float) interfacegraphique.SIZE_Y);
536 } else if (GestionPort.flotte.getBateau(i)
537     .getNumeroTypeBateau() == 35
538     || GestionPort.flotte.getBateau(i)
539         .getNumeroTypeBateau() == 51
540     || GestionPort.flotte.getBateau(i)
541         .getNumeroTypeBateau() == 55
542     || GestionPort.flotte.getBateau(i)

```

```

543     .getNumeroTypeBateau() == 58) {
544         parent.text("Police/Armée/Méd.", 
545             (float) (float) interfacegraphique.SIZE_X
546                 * (float) 10 / 11,
547             (float) (float) interfacegraphique.SIZE_Y
548                 - (float) 46 / 500
549                 * (float) interfacegraphique.SIZE_Y);
550     } else if (GestionPort.flotte.getBateau(i)
551         .getNumeroTypeBateau() == 53) {
552         parent.text("Port Tender",
553             (float) (float) interfacegraphique.SIZE_X
554                 * (float) 10 / 11,
555             (float) (float) interfacegraphique.SIZE_Y
556                 - (float) 46 / 500
557                 * (float) interfacegraphique.SIZE_Y);
558     } else {
559     }
560
561 // affichage du nom de la zone dans laquelle est le bateau
562
563 double point[] = {
564     GestionPort.flotte.getBateau(i)
565         .getPosNorm(interfacegraphique.TIME).getX(),
566     GestionPort.flotte.getBateau(i)
567         .getPosNorm(interfacegraphique.TIME).getY() };
568 if (Zone.MOUILLAGE_OUEST(point)) {
569     parent.text("Zone: MOUILLAGE OUEST",
570         (float) interfacegraphique.SIZE_X * (float) 10
571             / 11, (float) interfacegraphique.SIZE_Y
572                 - (float) 36 / 500
573                 * (float) interfacegraphique.SIZE_Y);
574 }
575 if (Zone.MOUILLAGE_EST(point)) {
576     parent.text("Zone: MOUILLAGE EST",
577         (float) interfacegraphique.SIZE_X * (float) 10
578             / 11, (float) interfacegraphique.SIZE_Y
579                 - (float) 36 / 500
580                 * (float) interfacegraphique.SIZE_Y);
581 }
582 if (Zone.MOUILLAGE_NORD(point)) {
583     parent.text("Zone: MOUILLAGE NORD",
584         (float) interfacegraphique.SIZE_X * (float) 10
585             / 11, (float) interfacegraphique.SIZE_Y
586                 - (float) 36 / 500
587                 * (float) interfacegraphique.SIZE_Y);
588 }
589 if (Zone.MOUILLAGE_INTERDIT(point)) {
590     parent.text("Zone: MOUILLAGE INTERDIT",
591         (float) interfacegraphique.SIZE_X * (float) 10
592             / 11, (float) interfacegraphique.SIZE_Y
593                 - (float) 36 / 500
594                 * (float) interfacegraphique.SIZE_Y);
595 }
596 if (Zone.CHENAL_SUD_MONTANT(point)) {
597     parent.text("Zone: CHENAL SUD MONTANT",
598         (float) interfacegraphique.SIZE_X * (float) 10
599             / 11, (float) interfacegraphique.SIZE_Y
600                 - (float) 36 / 500
601                 * (float) interfacegraphique.SIZE_Y);
602 }
603 if (Zone.CHENAL_SUD_DESCENDANT(point)) {
604     parent.text("Zone: CHENAL SUD DESCENDANT",
605         (float) interfacegraphique.SIZE_X * (float) 10
606             / 11, (float) interfacegraphique.SIZE_Y
607                 - (float) 36 / 500
608                 * (float) interfacegraphique.SIZE_Y);
609 }
610
611
612
613

```

```

614         - (float) 36 / 500
615         * (float) interfacegraphique.SIZE_Y);
616     }
617     if (Zone.ACCEES_LAVERA(point)) {
618         parent.text("Zone: ACCES LAVERA",
619             (float) interfacegraphique.SIZE_X * (float) 10
620             / 11, (float) interfacegraphique.SIZE_Y
621             - (float) 36 / 500
622             * (float) interfacegraphique.SIZE_Y);
623     }
624     if (Zone.ZONE_SECURISE1(point)) {
625         parent.text("Zone: ZONE SECURISEE 1",
626             (float) interfacegraphique.SIZE_X * (float) 10
627             / 11, (float) interfacegraphique.SIZE_Y
628             - (float) 36 / 500
629             * (float) interfacegraphique.SIZE_Y);
630     }
631     if (Zone.ACCEES_GOLFE_FOS(point)) {
632         parent.text("Zone: ACCES AU GOLF DE FOS",
633             (float) interfacegraphique.SIZE_X * (float) 10
634             / 11, (float) interfacegraphique.SIZE_Y
635             - (float) 36 / 500
636             * (float) interfacegraphique.SIZE_Y);
637     }
638     if (Zone.ZONE_SECURISE2(point)) {
639         parent.text("ZONE SECURISEE 2",
640             (float) interfacegraphique.SIZE_X * (float) 10
641             / 11, (float) interfacegraphique.SIZE_Y
642             - (float) 36 / 500
643             * (float) interfacegraphique.SIZE_Y);
644     }
645     }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 // affichage de la trajectoire
654 parent.noStroke();
655
656 TrajRec.trajRec(i, interfacegraphique.TIME);
657
658 // affichage Charles
659
660 if (PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
661     x, y) <= 15) {
662     // TrajStat.dessiner(PredicTraj
663     // .trajectoireLaPlusProbable((PauseEvents
664     // .creerTableauCoord(GestionPort.flotte
665     // .getBateau(i)))); */
666     /*
667     * int k=0; while (k<PredicTraj
668     * .trajectoireLaPlusProbable((PauseEvents
669     * .creerTableauCoord(GestionPort.flotte
670     * .getBateau(i)))).length){ System.out.println(
671     * ConversionPixel.conversionPixelX((float) Etat
672     * .toMetrique(PredicTraj
673     * .trajectoireLaPlusProbable((PauseEvents
674     * .creerTableauCoord(GestionPort.flotte
675     * .getBateau(i))))[k])[0])); System.out.println( " "
676     * +ConversionPixel.conversionPixelX((float) Etat
677     * .toMetrique(PredicTraj
678     * .trajectoireLaPlusProbable((PauseEvents
679     * .creerTableauCoord(GestionPort.flotte
680     * .getBateau(i))))[k])[1])); k++; }
681     */
682 }
683 }
684 DegradeStatistique.animer(PredicTraj.probabilitePresence(

```

```

685     PauseEvents.creerTableauCoord(GestionPort.flotte
686         .getBateau(i)), true);
687
688     // affichage Arthur
689
690     // PauseEvents.afficherEllipses(PauseEvents.creerTableauCoord(GestionPort.flotte.
691     // getBateau(i)));
692
693 }
694 }//
695
696 }
697 }
698 }
```

```

1 import java.util.*;
2
3
4 public class Dijkstra {
5
6     static final public int OBSTACLE = -2;
7     static final public int NONVISITE = -1;
8
9     public final int x0;
10    public final int y0;
11    public final int xf;
12    public final int yf;
13    public final int X;
14    public final int Y;
15    public int [][] carte;
16    public double pas;
17    public int [][] trajectoire;
18    public int longueur;
19
20    public Dijkstra(int [][] carte,int [][] trajectoire){
21        this.carte=carte;
22        X=carte.length;
23        Y=carte[0].length;
24        this.trajectoire=trajectoire;
25        longueur=trajectoire.length;
26        this.x0 = trajectoire[1][0];
27        this.y0 = trajectoire[1][1];
28        this.xf = trajectoire[trajectoire[0][0]][0];
29        this.yf = trajectoire[trajectoire[0][0]][1];
30    }
31
32    public int [][] parcourir (int x0, int y0, int xf, int yf) {
33        ArrayList<Point> openList=new ArrayList();
34        ArrayList<Point> closedList= new ArrayList();
35        Point init = new Point(x0,y0, 0);
36        carte[x0][y0] = 0;
37        Point fin = new Point (xf,yf);
38        openList.add(init);
39        while(!openList.contains(fin)&&!openList.isEmpty()) {
40            Collections.sort(openList);
41            Point current=openList.get(0);
42            openList.remove(0);
43            closedList.add(current);
44            for(int i=-1;i<2;i++){
45                for(int j=-1; j<2; j++){
46                    if (estDansLaCarte(new Point(current.x+i, current.y+j))) {
47                        int dist = (j == 0 || i == 0) ? 10 : 14;
48                        if(carte[current.x+i][current.y+j]==OBSTACLE) {}
49                        else if (carte[current.x+i][current.y+j]==NONVISITE) {
50                            carte[current.x+i][current.y+j]=dist+current.valeur;
51                            openList.add(new Point(current.x+i,current.y+j,dist+current.valeur));
52                        }
53                    else {
54                        carte[current.x+i][current.y+j]=Math.min(dist+current.valeur,carte[current.x+i][current.y+j]);
55                    }
56                }
57            }
58        }
59    }
60}
```

```

55         }
56     }
57   }
58 }
59 return cheminPlusCourt(trajectoire, carte, x0, y0, xf, yf);
60
61 }
62 }
63
64 public int [][] cheminPlusCourt (int [][] trajectoire, int [][] mat, int x0, int y0, int x, int y)
65 {
66   int longueur=trajectoire [0] [0];
67
68   if( mat [x] [y]==NONVISITE) {System. err. println("inatteignable"); System. exit (1);}
69
70   Point arrivee= new Point (x,y, mat [x] [y]);
71   ArrayList<Point> chemin=new ArrayList ();
72   chemin.add(arrivee);
73   while(x != x0 || y!=y0) {
74     int min =Integer.MAX_VALUE;
75     int imin=-1;
76     int jmin=-1;
77     for(int i=-1; i<2; i++) {
78       for(int j=-1; j<2; j++) {
79
80       if ( estDansLaCarte(new Point (x+i , y+j ))&&mat [x+i ] [y+j ]!= -2&&!(j==0&&i==0)&&mat [x+i ] [y+j ]
81           ]>=0&&mat [x+i ] [y+j ]<min) {
82         min=mat [x+i ] [y+j ];
83         imin=i ;
84         jmin=j ;
85       }
86       x=x+imin;
87       y=y+jmin;
88       chemin.add(new Point (x ,y ,min));
89     }
90   Collections .sort(chemin);
91   Object [] cheminBis=chemin .toArray ();
92
93   int [] [] nouvelleTraj= new int [cheminBis .length +1] [2];
94   nouvelleTraj [0] [0]=cheminBis .length ;
95   for (int i=1; i<cheminBis .length +1; i++) {
96     nouvelleTraj [i] [0]=(( Point )(cheminBis [i-1])).x;
97     nouvelleTraj [i] [1]=(( Point )(cheminBis [i-1])).y;
98   }
99   return nouvelleTraj;
100 }
101
102 public boolean existe (int x, int y) {
103   return (x >= 0 && x < X) && (y >= 0 && y < Y);
104 }
105 public boolean estDansLaCarte(Point p) {
106   return ! (p.x<0||p.x>=X||p.y<0||p.y>=Y);
107 }
108
109 /*
110  * Affiche la carte (int) en repère cartésien
111  */
112 public static void afficher(int [][] a) {
113   int n = a [0].length;
114   int m = a.length;
115
116   for (int i=n-1; i >= 0; i--) {
117     for (int j=0; j < m; j++) {
118       System.out.printf("%3d | ", a [j] [i]);
119     }
120     System.out.println();
121   }
122 }
123 */

```

```

124 * Affiche la carte (boolean) en repère cartésien
125 */
126 public static void afficher(boolean[][] a) {
127     int n = a[0].length;
128     int m = a.length;
129
130     for (int i=n-1; i >= 0; i--) {
131         for (int j=0; j < m; j++) {
132             if (a[j][i]) { System.out.printf(" 1 |"); }
133             else { System.out.printf(" 0 |"); }
134         }
135         System.out.println();
136     }
137 }
138 /*
139 * Affiche le tableau des positions dans l'ordre
140 */
141 public static void afficher_traj(int[][] a) {
142     int n = a.length;
143     int m = a[0].length;
144
145     for (int i=0; i < n; i++) {
146         for (int j=0; j < m; j++) {
147             System.out.printf("%3d | ", a[i][j]);
148         }
149         System.out.println();
150     }
151 }
152 /*
153 * Affiche le tableau
154 */
155 public static void afficher(int[] a) {
156     int n = a.length;
157     for (int i=0; i < n; i++) {
158         System.out.printf("%3d | ", a[i]);
159     }
160 }
161
162 public int detecterPlusProche(int [][] carte, int [][] traj, int i, int N) { //le point de la
163     // trajectoire le plus proche qui n'est pas un obstacle.
164     int j=0;
165     while (this.existe(traj[i+j][0], traj[i+j][1])&&carte[traj[i+j][0]][traj[i+j][1]]== -2&&j<N)
166     {
167         j++;
168     }
169     int k=0;
170     while (this.existe(traj[i+k][0], traj[i+k][1])&&carte[traj[i+k][0]][traj[i+k][1]]== -2&&k>-N)
171     {
172         k--;
173     }
174     int l=Math.min(j, -k);
175     if(l==j || j== -k) return i+l;
176     return i+k;
177 }
178
179 public int [][] fractionnerTrajectoire (int [][] trajectoire , int N) {
180     int longueurT= trajectoire [0][0];
181     int [][] trajFrac;
182     if (longueurT%N==0) {
183         trajFrac= new int [longueurT/N+2][2];
184         trajFrac[0][0] = longueurT/N+1;
185         for (int i=0; i<=longueurT/N-1; i++) {
186             trajFrac[i+1][0]= trajectoire[ detecterPlusProche(carte , trajectoire ,1+i*N,N)][0];
187             trajFrac[i+1][1]= trajectoire[ detecterPlusProche(carte , trajectoire ,1+i*N,N)][1];
188             trajFrac[longueurT/N+1][0]= trajectoire [longueurT ][0];
189             trajFrac[longueurT/N+1][1]= trajectoire [longueurT ][1];
190         }
191     }
192     else {
193         trajFrac= new int [longueurT/N+3][2];
194         trajFrac[0][0] = longueurT/N+2;

```

```

192     for (int i=0; i<=longueurT/N; i++) {
193         trajFrac[i+1][0] = trajectoire[detecterPlusProche(carte, trajectoire, 1+i*N, N)][0];
194         trajFrac[i+1][1] = trajectoire[detecterPlusProche(carte, trajectoire, 1+i*N, N)][1];
195         trajFrac[longueurT/N+2][0] = trajectoire[longueurT][0];
196         trajFrac[longueurT/N+2][1] = trajectoire[longueurT][1];
197     }
198 }
199 return trajFrac;
200 }
201
202 public static int[][] carteToMap(boolean[][] carte) {
203     int[][] map = new int[carte.length][carte[0].length];
204     for(int i=0; i < carte.length; i++) {
205         for(int j=0; j < carte[0].length; j++) {
206             if (carte[i][j]) { map[i][j] = -1; }
207             else { map[i][j] = -2; }
208         }
209     }
210     return map;
211 }
212
213 public static int[][] fusionner(int[][][] trajDecom) {
214     int l = 1;
215     for (int i = 0; i < trajDecom.length; i++) {
216         l += trajDecom[i][0][0];
217     }
218     int[][] trajFinale = new int[l][2];
219     trajFinale[0][0] = l-1;
220     int k = 1;
221     for (int i = 0; i < trajDecom.length; i++) {
222         for (int j = 1; j <= trajDecom[i][0][0]; j++) {
223             trajFinale[k][0] = trajDecom[i][j][0];
224             trajFinale[k][1] = trajDecom[i][j][1];
225             k++;
226         }
227     }
228     return trajFinale;
229 }
230
231 public static int[][] parcours (boolean[][] carte, int[][] trajPrevue) {
232     Dijkstra dijkstra=new Dijkstra(carteToMap(carte), trajPrevue);
233     int[][] trajFrac = dijkstra.fractionnerTrajectoire(trajPrevue, 10);
234     afficher_traj(trajFrac);
235     int[][][] trajDecom = new int[trajFrac.length - 2][][];
236
237     for (int i = 1; i < trajFrac.length-1; i++) {
238         trajDecom[i-1] = dijkstra.parcourir(trajFrac[i][0], trajFrac[i][1], trajFrac[i+1][0],
239                                         trajFrac[i+1][1]);
240     }
241     return (fusionner(trajDecom));
242 }
243 }
```

```

1 public class Etat {
2 /*
3 * Détermination ,en fonction des coordonnées du bateau, de la zone
4 * portuaire dans laquelle il se trouve (zones déterminées par l'arrêté
5 * préfectoral).
6 */
7
8 // _____ NE PAS MODIFIER _____
9 // public final static double [] COORDMERCATOR0 = {43.266206, 4.664967}; //
10 // Coordonnées du (0,0) en Mercator
11 public final static double[] COORDMERCATOR0 = { 43.372755, 4.953562 };
12 public final static double X = 35; // Taille de la zone
13 public final static double Y = 22; // Taille de la zone
14 // _____ FIN _____
15 public static int N1 = 140; // Nombre de cases suivant x et suivant y (Cases
16 // de taille 250m / 250m)
17 public static int N2 = 88;
```

```

18 public final static double A = X / ((double) (N1)); // Taille suivant x
19 // d'une case // case
20 public final static double B = Y / ((double) (N2)); // Taille suivant y
21 // d'une case
22
23 public int x;
24 public int y;
25
26 // !!!!!!! ATTENTION / IES CONSTRUCTEURS QUI NE PRENNENT PAS LE MEME TYPE
27 // D'ARGUMENTS !!!!
28
29 // Constructeur 1: Etat (LATITUDE, LONGITUDE)
30 public Etat(double u, double v) {
31     double[] coordMercator = new double[2];
32     double[] coordUTM = new double[2];
33     coordMercator[0] = u;
34     coordMercator[1] = v;
35     coordUTM = ConversionMercator.conversion(coordMercator);
36
37     x = (int) Math.floor((coordUTM[0]) / A);
38     y = (int) Math.floor((coordUTM[1]) / B);
39 }
40
41 // Constructeur 2: Etat ({CoordUTM_X, CoordUTM_Y})
42 public Etat(double[] s) {
43     x = (int) Math.floor((s[0]) / A);
44     y = (int) Math.floor((s[1]) / B);
45 }
46
47 // Fonction renvoyant les coordonnées métriques du centre de la case par
48 // rapport à l'origine
49 public double[] toMetrique() {
50     double[] coordMetriques = new double[2];
51     coordMetriques[0] = A * ((double) this.x + 1 / 2);
52     coordMetriques[1] = B * ((double) this.y + 1 / 2);
53     return coordMetriques;
54 }
55
56 public static double[] toMetrique(int[] t) {
57     double[] coordMetriques = new double[2];
58     coordMetriques[0] = A * ((double) t[0] + 1 / 2);
59     coordMetriques[1] = B * ((double) t[1] + 1 / 2);
60     return coordMetriques;
61 }
62
63 // Fonction prenant (xUTM) et renvoyant le numéro x de la case
64 public static short coordX(double a) {
65     short t = (short) (((a) / A));
66     return t;
67 }
68
69 // Fonction prenant (yUTM) et renvoyant le numéro y de la case
70 public static short coordY(double a) {
71     short t = (short) (((a) / B));
72     return t;
73 }
74
75 // Comparer l'égalité de deux états
76 public boolean equals(Etat e) {
77     if (this.x == e.x && this.y == e.y) {
78         return (true);
79     } else {
80         return (false);
81     }
82 }
83
84 // Modification du toString pour le System.out.println
85 public String toString() {
86     return ("[" + x + "," + y + "]");
87 }

```

```

89 // Savoir dans quelle(s) Zone se trouve le bateau
90 public void dansQuelleZone() {
91     Zone z1 = Zone.MOUILLAGE_OUEST();
92     Zone z2 = Zone.MOUILLAGE_EST();
93     Zone z3 = Zone.MOUILLAGE_NORD();
94     Zone z4 = Zone.MOUILLAGE_INTERDIT();
95     Zone z5 = Zone.CHENAL_SUD_MONTANT();
96     Zone z6 = Zone.CHENAL_SUD_DESCENDANT();
97     Zone z7 = Zone.ACCEES_LAVERA();
98     Zone z8 = Zone.ZONE_SECURISE1();
99     Zone z9 = Zone.ACCEES_GOLFE_FOS();
100    Zone z10 = Zone.ZONE_SECURISE2();
101
102    int e = 0;
103
104    if (z1.contains(this)) {
105        System.out.println("Bateau dans la zone: MOUILLAGE OUEST");
106        e = 1;
107    }
108    if (z2.contains(this)) {
109        System.out.println("Bateau dans la zone: MOUILLAGE EST");
110        e = 1;
111    }
112    if (z3.contains(this)) {
113        System.out.println("Bateau dans la zone: MOUILLAGE NORD");
114        e = 1;
115    }
116    if (z4.contains(this)) {
117        System.out.println("Bateau dans la zone: MOUILLAGE INTERDIT");
118        e = 1;
119    }
120    if (z5.contains(this)) {
121        System.out.println("Bateau dans la zone: CHENAL SUD, SENS MONTANT");
122        e = 1;
123    }
124    if (z6.contains(this)) {
125        System.out
126            .println("Bateau dans la zone: CHENAL SUD, SENS DESCENDANT");
127        e = 1;
128    }
129    if (z7.contains(this)) {
130        System.out.println("Bateau dans la zone: ACCES LAVERA");
131        e = 1;
132    }
133    if (z8.contains(this)) {
134        System.out.println("Bateau dans la zone: ZONE SECURISE 1");
135        e = 1;
136    }
137    if (z9.contains(this)) {
138        System.out.println("Bateau dans la zone: ACCES GOLF FOS");
139        e = 1;
140    }
141    if (z10.contains(this)) {
142        System.out.println("Bateau dans la zone: ZONE SECURISE 2");
143        e = 1;
144    }
145
146    if (e == 0) {
147        System.out.println("Bateau dans aucune des zones definies");
148    }
149
150 }
151
152 }
```

```

1 import java.util.Date;
2
3 // un Flotte est un tableau de bateaux.
4
5 public class Flotte {
6     private long tempsMin = 0;
```

```

7  private long tempsMax = 0;
8  private long nombrePositionNormalise = 0;
9  private long deltaT = 30;
10 private boolean normalise;
11 private Bateau[] bateaux;
12
13 public Flotte(Bateau[] bateaux) {
14     this.bateaux = bateaux;
15     for (int i = 0; i < bateaux.length; i++) {
16         this.bateaux[i].setFlotte(this);
17     }
18     setNormalise(false);
19 }
20
21 public Date getDate(int i) {
22     return new Date(i * deltaT * 1000 + tempsMin);
23 }
24
25 public Bateau getBateau(int i) {
26     if (i < bateaux.length && i >= 0)
27         return bateaux[i];
28     else {
29         System.out.println("Index incorrect : " + i);
30         return null;
31     }
32 }
33
34 public Bateau[] getBateaux() {
35     if (bateaux.length > 0)
36         return bateaux;
37     else
38         return null;
39 }
40
41 public void setBateaux(Bateau[] b) {
42     this.bateaux = b;
43 }
44
45 public int getNombreBateaux() {
46     return bateaux.length;
47 }
48
49 public long getNombrePositionNormalise() {
50     return nombrePositionNormalise;
51 }
52
53 public boolean isNormalise() {
54     return normalise;
55 }
56
57 public void setNormalise(boolean normalise) {
58     this.normalise = normalise;
59 }
60
61 public void afficher() {
62     for (int i = 0; i < bateaux.length; i++)
63         bateaux[i].afficher();
64 }
65
66 public String toString() {
67     String s = "";
68     for (int i = 0; i < bateaux.length; i++)
69         s += bateaux[i].toString();
70     return s;
71 }
72
73 public void normaliserPosition() {
74     this.normaliserPosition(30);
75 }
76
77 public void normaliserPosition(long deltaT) {

```

```

78 System.out.println("\nNormalisation ...");
79 this.deltaT = deltaT;
80 Bateau.normaliserPosition(this, deltaT);
81 setNormalise(true);
82 extrNormalise();
83 }
84
85 private void extrNormalise() {
86     tempsMin = bateaux[0].getPosition()[0].getTemps().getTime();
87     tempsMax = bateaux[0].getPosition()[bateaux[0].getPosition().length - 1]
88         .getTemps().getTime();
89
90     for (int i = 1; i < bateaux.length; i++) {
91         long tempsmin = bateaux[i].getPosition()[0].getTemps().getTime();
92         long tempsmax = bateaux[i].getPosition()[bateaux[i].getPosition().length - 1]
93             .getTemps().getTime();
94
95         if (tempsmax > tempsMax)
96             tempsMax = tempsmax;
97         if (tempsmin < tempsMin)
98             tempsMin = tempsmin;
99     }
100
101     long ecart = (tempsMax - tempsMin) / 1000;
102     nombrePositionNormalise = (ecart / deltaT);
103
104     for (int i = 0; i < bateaux.length; i++) {
105         bateaux[i].setMinPosNorm(((bateaux[i].getPosition()[0].getTemps()
106             .getTime()) - (tempsMin)) / (deltaT * 1000));
107         bateaux[i].setMaxPosNorm((bateaux[i].getPosition()[bateaux[i]
108             .getPosition().length - 1].getTemps().getTime() - tempsMin)
109             / (deltaT * 1000));
110     }
111 }
112 }
113 } // Fin classe Flotte

```

```

1 import java.io.File;
2
3 public class GestionFichiers {
4     public static String[] recupererCheminsfichiers(String dossier) {
5         File fichier = new File(dossier);
6         File[] fichiers = fichier.listFiles();
7         String[] chemins = new String[fichiers.length];
8         for (int i = 0; i < fichiers.length; i++)
9             chemins[i] = fichiers[i].getAbsolutePath();
10        return chemins;
11    }
12 }

```

```

1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4
5 //Chargement d'un morceau de la base de données.
6
7 public class GestionPort {
8     static Flotte flotte;
9
10    public static void creationFlotte() {
11
12        // Renvoyer le centre
13
14        System.out.println("### Test du programme ###");
15
16        long temps = System.currentTimeMillis();
17
18        String s1 = "C:\\\\Users\\\\Evann\\\\workspace\\\\MIG SE\\\\mesures\\\\sauvegarde1.sql";
19        String s2 = "C:\\\\Users\\\\Evann\\\\mig11012011-v2.6.sql";
20

```

```

21 Date date1 = null;
22 Date date2 = null;
23
24 try {
25     date1 = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
26         .parse("2011-01-11 11:00:00");
27     date2 = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
28         .parse("2011-01-11 13:00:15");
29 } catch (ParseException e) {
30     e.printStackTrace();
31 }
32
33 // !!!! Preferer l'utilisation de la classe Flotte !!!!
34
35 long[] mmsi = { 636090524, 304932000, 321556545,
36     LectureDB.revoieMMSI("RITA SIBUM") };
37 flotte = LectureDB.recupererFlotteF();
38 flotte.normaliserPosition();
39 mySQL.FermertConnection();
40
41 interfacegraphique.i = 1;
42 }
43
44 }

```

```

1 import java.util.Arrays;
2 import java.util.LinkedList;
3 import java.util.ListIterator;
4
5 /*Algorithme de calcul de la trajectoire
6 la plus probable pour le groupe travaillant sur les HMM*/
7
8 public class hmmcontinus2 {
9     int N;
10    double[] initialisation;
11    double[][] chgtéstat;
12    double[][] moyenne;
13    Matrice[] covariance;
14
15    public hmmcontinus2(int n) {
16        N = n;
17        initialisation = new double[n];
18        chgtéstat = new double[N][N];
19        moyenne = new double[N][2];
20        covariance = new Matrice[N];
21    }
22
23    public hmmcontinus2(int n, double[] in, double[][] chgt, double[][] moy,
24        Matrice[] cov) {
25        N = n;
26        initialisation = in;
27        chgtéstat = chgt;
28        moyenne = moy;
29        covariance = cov;
30    }
31
32    public void cons(int n, double[] in, double[][] chgt, double[][] moy,
33        Matrice[] cov) {
34        N = n;
35        initialisation = in;
36        chgtéstat = chgt;
37        moyenne = moy;
38        covariance = cov;
39    }
40
41    public double b(int j, double[] O) {
42        double d = Math.PI * 2 * Math.sqrt(covariance[j].det());
43        double[] aux = { O[0] - moyenne[j][0], O[1] - moyenne[j][1] };
44        Matrice m = covariance[j].inverse();
45        double n = aux[0] * aux[0] * m.a11 + aux[0] * aux[1] * (m.a12 + m.a21)
46            + aux[1] * aux[1] * m.a22;

```

```

47     return (1 / d * Math.exp(-1. / 2 * n));
48 }
49
50 public double[] forward(int t, double[][] obs) {
51     double[] alpha = new double[N];
52     double[] aux = new double[N];
53     for (int i = 0; i <= N - 1; i++) {
54         alpha[i] = initialisation[i] * b(i, obs[0]);
55     }
56     for (int i = 1; i <= t - 1; i++) {
57         for (int j = 0; j <= N - 1; j++) {
58             double a = 0.;
59             for (int k = 0; k < N; k++) {
60                 a += alpha[k] * chgtéat[k][j];
61             }
62             aux[j] = a * b(j, obs[i]);
63         }
64         for (int j = 0; j <= N - 1; j++) {
65             alpha[j] = aux[j];
66         }
67     }
68
69     return alpha;
70 }
71
72 public double proba(double[][] obs) {
73     int l = obs.length;
74     double[] alpha = new double[N];
75     double[] aux = new double[N];
76     for (int i = 0; i <= N - 1; i++) {
77         alpha[i] = initialisation[i] * b(i, obs[0]);
78     }
79     for (int i = 1; i <= l - 1; i++) {
80         for (int j = 0; j <= N - 1; j++) {
81             double a = 0.;
82             for (int k = 0; k < N; k++) {
83                 a += alpha[k] * chgtéat[k][j];
84             }
85             aux[j] = a * b(j, obs[i]);
86         }
87         for (int j = 0; j <= N - 1; j++) {
88             alpha[j] = aux[j];
89         }
90     }
91
92     double res = 0.;
93     for (int i = 0; i < N; i++) {
94         res += alpha[i];
95     }
96     return res;
97 }
98
99 public double[] backward(int t, double[][] obs) {
100    double[] alpha = new double[N];
101    double[] aux = new double[N];
102    for (int i = 0; i <= N - 1; i++) {
103        alpha[i] = 1;
104    }
105    for (int i = 1; i <= t - 1; i++) {
106        for (int j = 0; j <= N - 1; j++) {
107            double a = 0.;
108            for (int k = 0; k < N; k++) {
109                a += alpha[k] * chgtéat[j][k] * b(k, obs[i]);
110            }
111            aux[j] = a;
112        }
113        for (int j = 0; j <= N - 1; j++) {
114            alpha[j] = aux[j];
115        }
116    }
117    return alpha;

```

```

118 }
119
120 public double gamma(int t, int j, double[][][] obs) {
121     double alpha = this.forward(t, obs)[j];
122     double beta = this.backward(t, obs)[j];
123     double d = 0;
124     for (int i = 0; i < N; i++) {
125         d += this.forward(t, obs)[i] * this.backward(t, obs)[i];
126     }
127     return alpha * beta / d;
128 }
129
130 public double[][] nouvellemoyenne(double[][][] obs) {
131     int K = obs.length;
132     double[][] res = new double[N][2];
133     for (int j = 0; j < N; j++) {
134         double n1 = 0;
135         double d = 0;
136         double n2 = 0;
137         for (int k = 0; k < K; k++) {
138             int T = obs[k].length;
139             for (int i = 1; i <= T; i++) {
140                 n1 += this.gamma(i, j, obs[k]) * obs[k][i - 1][0];
141             }
142
143             for (int i = 1; i <= T; i++) {
144                 d += this.gamma(i, j, obs[k]);
145             }
146             for (int i = 1; i <= T; i++) {
147                 n2 += this.gamma(i, j, obs[k]) * obs[k][i - 1][1];
148             }
149         }
150         res[j][0] = n1 / d;
151         res[j][1] = n2 / d;
152     }
153     return res;
154 }
155
156 public Matrice[] nouvellecovariance(double[][][] obs) {
157     int K = obs.length;
158     Matrice[] res = new Matrice[N];
159     for (int j = 0; j < N; j++) {
160         double d1 = 0;
161         double u11 = 0;
162         double u12 = 0;
163         double u22 = 0;
164         for (int k = 0; k < K; k++) {
165             int T = obs[k].length;
166             for (int i = 1; i <= T; i++) {
167                 d1 += this.gamma(i, j, obs[k]);
168             }
169             for (int t = 1; t <= T; t++) {
170                 u11 += 1 / d1 * this.gamma(t, j, obs[k])
171                     * (obs[k][t - 1][0] - this.moyenne[j][0])
172                     * (obs[k][t - 1][0] - this.moyenne[j][0]);
173             }
174             for (int t = 1; t <= T; t++) {
175                 u12 += 1 / d1 * this.gamma(t, j, obs[k])
176                     * (obs[k][t - 1][1] - this.moyenne[j][1])
177                     * (obs[k][t - 1][0] - this.moyenne[j][0]);
178             }
179             for (int t = 1; t <= T; t++) {
180                 u22 += 1 / d1 * this.gamma(t, j, obs[k])
181                     * (obs[k][t - 1][1] - this.moyenne[j][1])
182                     * (obs[k][t - 1][1] - this.moyenne[j][1]);
183             }
184         }
185         res[j] = new Matrice(u11, u12, u12, u22);
186     }
187     return res;
188 }

```

```

189
190     public double epsilon(int i, int j, int t, double[][] obs) {
191         double n = this.forward(t, obs)[i] * this.b(j, obs[t])
192             * this.backward(t + 1, obs)[j] * chgtéat[i][j];
193         double d = 0;
194         for (int p = 0; p < N; p++) {
195             for (int q = 0; q < N; q++) {
196                 d += this.forward(t, obs)[p] * this.b(q, obs[t])
197                     * this.backward(t + 1, obs)[q] * chgtéat[p][q];
198             }
199         }
200         return n / d;
201     }
202
203     public double gamma2(int t, int i, double[][][] obs) {
204         double res = 0;
205         for (int j = 0; j < N; j++) {
206             res += epsilon(i, j, t, obs);
207         }
208         return res;
209     }
210
211     public hmmcontinus2 approx(double[][][] obs) {
212         int K = obs.length;
213         hmmcontinus2 ap = new hmmcontinus2(N);
214         for (int i = 0; i < N; i++) {
215             ap.initialisation[i] = this.gamma2(1, i, obs[0]);
216         }
217         for (int n = 0; n < N; n++) {
218             for (int m = 0; m < N; m++) {
219                 double a = 0;
220                 double b = 0;
221                 for (int k = 0; k < K; k++) {
222                     int T = obs[k].length;
223                     for (int t = 1; t < T; t++) {
224                         a += this.epsilon(n, m, t, obs[k]);
225                     }
226                     for (int t = 1; t < T; t++) {
227                         b += this.gamma2(t, n, obs[k]);
228                     }
229                 }
230                 ap.chgtéat[n][m] = a / b;
231             }
232         }
233         ap.moyenne = this.nouvellemoyenne(obs);
234         ap.covariance = this.nouvellecovariance(obs);
235         return ap;
236     }
237
238     public double proba2(double[][][] obs) {
239         int K = obs.length;
240         int res = 1;
241         for (int k = 0; k < K; k++) {
242             res *= proba(obs[k]);
243         }
244         return res;
245     }
246
247     public hmmcontinus2 baumwelch(double[][][] obs) {
248         hmmcontinus2 ap = this.approx(obs);
249         double epsilon = ap.proba2(obs) - this.proba2(obs);
250         hmmcontinus2 ap2;
251         while (epsilon > Math.pow(10, -300)) {
252             ap2 = ap.approx(obs);
253             epsilon = ap2.proba2(obs) - ap.proba2(obs);
254             ap = ap2;
255         }
256         return ap;
257     }
258
259     public static double[][] traitemenDonnées(double[][] position) {

```

```

260     double pas = 1.5;
261     LinkedList<double[]> pos = new LinkedList<double[]>();
262     pos.add(position[0]);
263     remplissagePosition(pas, pas, pos, position);
264     int l = 0;
265     ListIterator<double[]> iter = pos.listIterator();
266     while (iter.hasNext()) {
267         l++;
268         iter.next();
269     }
270     ListIterator<double[]> iter2 = pos.listIterator();
271     double[][] res = new double[l][];
272     for (int i = 0; i < l; i++) {
273         res[i] = iter2.next();
274     }
275     return res;
276 }
277
278 public static double distance(double[] a, double[] b) {
279     return Math.sqrt((a[0] - b[0]) * (a[0] - b[0]) + (a[1] - b[1])
280                     * (a[1] - b[1]));
281 }
282
283 public static void remplissagePosition(double pas, double d,
284                                         LinkedList<double[]> pos, double[][] position) {
285     int l = position.length;
286     if ((l > 1)) {
287         if (distance(position[0], position[1]) < d) {
288             remplissagePosition(pas,
289                                 d - distance(position[0], position[1]), pos,
290                                 Arrays.copyOfRange(position, 1, l));
291         } else {
292             double d0 = distance(position[0], position[1]);
293             double x = position[0][0] + (position[1][0] - position[0][0])
294                     * d / d0;
295             double y = position[0][1] + (position[1][1] - position[0][1])
296                     * d / d0;
297             double[] tab = {x, y};
298             position[0] = tab;
299             pos.add(tab);
300             remplissagePosition(pas, pas, pos, position);
301         }
302     }
303 }
304
305 public static void afficher(double[] tab) {
306     for (int i = 0; i < tab.length; i++) {
307         System.out.print(tab[i] + " ");
308     }
309 }
310
311 public static void afficher(double[][] tab) {
312     for (int i = 0; i < tab.length; i++) {
313         System.out.print(" | ");
314         afficher(tab[i]);
315         System.out.println(" | ");
316     }
317 }
318
319 }

```

```

1 import processing.core.*;
2
3 // Initialisation du canvas dans setup() puis exécution des instructions dans draw()
4
5
6 public class interfacegraphique extends PApplet {
7     // Variables de structure et importations
8     public static interfacegraphique truc;
9     int[][] tab1={{98,41},{98,40}};
10    public static int j=1;

```

```

11 public static int i;
12 public static int bateauchoisi = 0;
13 private static final long serialVersionUID = 1L;
14 // Paramètres fenêtre
15 PImage img;
16 public static int SIZE_X = 1334;
17 public static int SIZE_Y = 750;
18 public static int FACTEUR_DE_DIVISION = 50;
19 public static int state = 0;
20 public static int TIME = 0;
21 public static int START;
22 public static int PAUSE = 0;
23 public static int TEMPSPAUSSETOT = 0;
24
25 // pour les senseurs initiaux :
26
27 public static double xpos1 = 637;
28 public static double ypos1 = 124;
29 public static double portee1 = 210;
30 public static double angle1 = PI / 6;
31 public static double sens1 = 2 * PI / 3;
32 public static int sensor_type1 = 1;
33
34 public static double xpos2 = 461;
35 public static double ypos2 = 117;
36 public static double portee2 = 200;
37 public static double angle2 = PI / 4;
38 public static double sens2 = PI / 4;
39 public static int sensor_type2 = 2;
40
41 public static double xpos3 = 442;
42 public static double ypos3 = 134;
43 public static double portee3 = 150;
44 public static double angle3 = 2 * PI;
45 public static double sens3 = 0;
46 public static int sensor_type3 = 3;
47
48 public static void main(String args[]) {
49     PApplet.main(new String[] { "--present", "interfacegraphique" });
50 }
51
52 // Initialisation de la fenêtre
53 public void setup() {
54
55     START = millis();
56     size(SIZE_X, SIZE_Y);
57     background(250);
58
59     img=loadImage("carte_vhd_5240_MODIFIEE2.jpg");
60     img.resize(width, height);
61     image(img, 0, 0);
62
63     interfacegraphique.truc = this;
64
65     System.out.println("temps écoulé");
66     START = millis();
67
68     GestionPort.creationFlotte();
69     PredicTraj.gerer_arbre();
70 }
71
72 // Fonction principale (le main)
73 public void draw() {
74     background(img);
75
76     if ((int) ((millis() - START) / 100) != ((int) (millis() - START)) / 100)
77         truc = this;
78
79     Dessiner.dessiner();
80     Menu.Menu();
81     Timer.timer();

```

```

82     Timer.afficherDate(GestionPort.flotte);
83
84     Senseur camera1 = new Senseur(truc, xpos1, ypos1, portee1, angle1,
85         sens1, sensor_type1);
86     Senseur radar2 = new Senseur(truc, xpos2, ypos2, portee2, angle2,
87         sens2, sensor_type2);
88     Senseur radar3 = new Senseur(truc, xpos3, ypos3, portee3, angle3,
89         sens3, sensor_type3);
90     camera1.dessinerSenseur();
91     radar2.dessinerSenseur();
92     radar3.dessinerSenseur();
93     Bouee.dessinerLesBouees();
94     if(bateauxchoisi == 1){
95         DegradeStatistique.animer(DegradeStatistique.mat());
96
97         TrajStat.dessiner(PauseEvents.creerTableauCoord(PauseEvents.getBateau(GestionPort.flotte)));
98
99         DegradeStatistique.animer(PredicTraj.probabilitePresence(tab1,true));
100    }
101
102
103    public void keyPressed() {
104
105        switch (interfacegraphique.state) {
106            case 0:
107                switch (key) {
108                    case 'q':
109                    case 'Q':
110                        interfacegraphique.state = 1;
111                        break;
112                    case 'o':
113                    case 'O':
114                        interfacegraphique.state = 4;
115                        break;
116                }
117            case 1:
118                switch (key) {
119                    case 'w':
120                    case 'W':
121                        interfacegraphique.state = 2;
122                        break;
123                    case 's':
124                    case 'S':
125                        interfacegraphique.state = 0;
126                        break;
127                    case 'm':
128                    case 'M':
129                        interfacegraphique.state = 5;
130                        break;
131                }
132            case 2:
133                switch (key) {
134                    case 's':
135                    case 'S':
136                        interfacegraphique.state = 0;
137                        break;
138                    case 'x':
139                    case 'X':
140                        interfacegraphique.state = 6;
141                        break;
142                    case 'c':
143                    case 'C':
144                        interfacegraphique.state = 7;
145                        break;
146                }
147            case 3:
148                switch (key) {
149                    case 'a':
150                    case 'A':
151                        interfacegraphique.state = 3;
152                        break;

```

```

153     }
154 case 4:
155     switch (key) {
156     case 's':
157     case 'S':
158         interfacegraphique.state = 0;
159         break;
160     }
161 case 5:
162     switch (key) {
163     case 's':
164     case 'S':
165         interfacegraphique.state = 0;
166         break;
167     case 'd':
168     case 'D':
169         interfacegraphique.state = 10;
170         break;
171     case 'T':
172     case 't':
173         interfacegraphique.state = 15;
174         break;
175     case 'Z':
176     case 'z':
177         interfacegraphique.state=16;
178         break;
179     }
180 case 6:
181     switch (key) {
182     case 's':
183     case 'S':
184         interfacegraphique.state = 0;
185         break;
186     }
187 case 7:
188     switch (key) {
189     case 's':
190     case 'S':
191         interfacegraphique.state = 0;
192         break;
193     case 'v':
194     case 'V':
195         interfacegraphique.state = 8;
196         break;
197     }
198 case 8:
199     switch (key) {
200     case 'p':
201     case 'P':
202         interfacegraphique.state = 9;
203         break;
204     case 's':
205     case 'S':
206         interfacegraphique.state = 0;
207         break;
208     }
209 case 9:
210     switch (key) {
211     case '&':
212     case '1':
213         interfacegraphique.state = 11;
214         break;
215     case 'é':
216     case '2':
217         interfacegraphique.state = 12;
218         break;
219     case '':
220     case '3':
221         interfacegraphique.state = 13;
222         break;
223     case 's':

```

```

224     case 'S':
225         interfacegraphique.state = 0;
226         break;
227     }
228     case 10:
229         switch (key) {
230             case 's':
231             case 'S':
232                 interfacegraphique.state = 0;
233                 break;
234             }
235     case 11:
236         switch (key) {
237             case 's':
238             case 'S':
239                 interfacegraphique.state = 0;
240                 break;
241             }
242     case 12:
243         switch (key) {
244             case 's':
245             case 'S':
246                 interfacegraphique.state = 0;
247                 break;
248             }
249     case 13:
250         switch (key) {
251             case 's':
252             case 'S':
253                 interfacegraphique.state = 0;
254                 break;
255             }
256     case 14:
257         switch (key) {
258             case 'e':
259             case 'E':
260                 PauseEvents.pause();
261                 break;
262             }
263     case 15:
264         switch (key) {
265             case 's':
266             case 'S':
267                 interfacegraphique.state = 0;
268                 break;
269             }
270     case 16:
271         switch (key) {
272             case 's':
273             case 'S':
274                 interfacegraphique.state = 0;
275                 break;
276             }
277         }
278     }
279 }
280 }
```

```

1 import java.sql.*;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4
5 //Chargement de la base de données
6 public class LectureDB {
7
8     public static Bateau recupererBateau(long mmsi) { // Recuperer un bateau de
9                     // mmsi donné
10    Bateau b = null;
11    try {
12        Connection connect = mySQL.ConnectionSQL();
13        Statement state = connect.createStatement();
```

```

14     ResultSet res = state
15         .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
16             Position.mmsi=Bateau.mmsi WHERE Bateau.mmsi = "
17             + mmsi + " ;");
18     res.first();
19     if (res.getInt("COUNT(DISTINCT Bateau.mmsi)") != 0)
20         b = recupererBateau(mmsi, connect);
21     else
22         System.err.println("MMSI inexistant : " + mmsi);
23     } catch (SQLException e) {
24         System.err.println(e.getMessage());
25     }
26
27     return b;
28 }
29
30 private static Bateau recupererBateau(long mmsi, Connection connect) { // Recuperer
31     // un
32     // bateau
33     // de
34     // mmsi
35     // donne
36     try {
37         // Creation d'un objet Statement
38         Statement state = connect.createStatement();
39         ResultSet res = state
40             .executeQuery("SELECT * FROM Bateau WHERE mmsi = " + mmsi
41             + " ;");
42         res.first();
43         res.getInt("mmsi");
44
45         String nom = res.getString("nom");
46         String nomCourant = res.getString("nom_courant");
47         double largeur = res.getDouble("largeur");
48         double longueur = res.getDouble("longueur");
49         int type = res.getInt("type");
50         String today = res.getString("temps_arrive");
51         Date ETA;
52         if (today != null) {
53             ETA = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
54                 .parse(today);
55         } else {
56             ETA = null;
57         }
58
59         res.close();
60
61         ResultSet res1 = state
62             .executeQuery("SELECT COUNT(*) FROM (SELECT Position.mmsi FROM Bateau INNER JOIN
63                 Position ON Bateau.mmsi = Position.mmsi WHERE Bateau.mmsi = "
64                 + mmsi + ") AS aux ;");
65         res1.first();
66         int r = res1.getInt("COUNT(*)");
67         res1.close();
68
69         ResultSet result = state
70             .executeQuery("SELECT Position.date_mesure, Position.longitude, Position.latitude,
71                 Position.x, Position.y, Position.sog, Position.cog, Position.vitesse_ang, Position.
72                 cap FROM Bateau INNER JOIN Position ON Bateau.mmsi = Position.mmsi WHERE Bateau.mmsi
73                 = "
74                 + mmsi + " ORDER BY date_mesure");
75
76         Position[] position = new Position[r];
77         int j = 0;
78         result.first();
79         do {
80             double d = result.getFloat(9);
81             int cap = (int) d;
82
83             today = result.getString("date_mesure");

```

```

80     Date date;
81     if (today != null) {
82         date = (new SimpleDateFormat ("yyyy-MM-dd HH:mm:ss"))
83             .parse (today);
84     } else {
85         date = null;
86     }
87
88     position [j] = new Position (date, result.getFloat (2),
89         result.getFloat (3), result.getFloat (4),
90         result.getFloat (5), result.getFloat (6),
91         result.getFloat (7), result.getFloat (8), cap);
92
93     j++;
94 } while (result.next ());
95 result.close ();
96 state.close ();
97 return new Bateau (mmsi, nom, nomCourant, type, position, longueur,
98 largeur, ETA, null);
99
100 } catch (Exception e) {
101     e.printStackTrace ();
102 }
103 return null;
104 }
105
106 public static Flotte recupererFlotteF () { // Idem recupererFlotte () mais
107     // renvoie une Flotte et non un
108     // tableau de bateau
109     return new Flotte (recupererFlotte ());
110 }
111
112 public static Flotte recupererFlotteF (long [] mmsi) { // Idem
113     // recupererFlotte (long
114     // [] mmsi) mais renvoie
115     // une Flotte et non un
116     // tableau de bateau
117     return new Flotte (recupererFlotte (mmsi));
118 }
119
120 public static Bateau [] recupererFlotte () { // Recuperer la flotte (tout les
121     // bateaux dans la DB) en
122     // itérant sur les mmsi dans la
123     // fonction recuperer bateau
124     try {
125
126         Connection connect = mySQL.ConnectionSQL ();
127         Statement state = connect.createStatement ();
128
129         ResultSet res = state.executeQuery ("SELECT COUNT(*) FROM Bateau;");
130         res.first ();
131         int nombre = res.getInt ("COUNT(*)");
132         System.out.println ("\n" + nombre
133             + " bateaux présents dans la base de données.");
134         res.close ();
135         if (nombre == 0) {
136             System.err
137                 .println ("Erreur : Pas de bateaux dans la Base de Données !");
138             return null;
139         }
140
141         Bateau [] flotte;
142         flotte = new Bateau [nombre];
143         int j = 0;
144
145         ResultSet res1 = state.executeQuery ("SELECT mmsi FROM Bateau;");
146
147         while (res1.next ()) {
148             long identifiant = res1.getInt ("mmsi");
149             System.out.printf ("%3d) ", j + 1);
150             flotte [j] = recupererBateau (identifiant, connect);

```

```

151         j++;
152     }
153     System.out.println(nombre + " bateaux chargés");
154     state.close();
155
156     return flotte;
157 }
158
159 } catch (Exception e) {
160     e.printStackTrace();
161 }
162 return null;
163 }
164
165 public static Bateau[] recupererFlotte(long[] mmsi) { // Recuperer
166     // uniquement les
167     // bateaux aux mmsi
168     // donnés en
169     // argument
170
171     try {
172
173         Connection connect = mySQL.ConnectionSQL();
174         Statement state = connect.createStatement();
175
176         // Tests pour vérifier que tous les mmsi donnés existe bien ...
177         String s = "";
178         for (int j = 0; j < mmsi.length; j++) {
179             s += mmsi[j] + "";
180             if (j != (mmsi.length - 1))
181                 s += ", ";
182         }
183         ResultSet res = state
184             .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
185                 Position.mmsi=Bateau.mmsi WHERE Bateau.mmsi IN (" +
186                 + s + ")");
187         res.first();
188         int nombre = res.getInt("COUNT(DISTINCT Bateau.mmsi)");
189
190         if (nombre == 0) {
191             System.out.println("Attention : Flotte vide !");
192             return null;
193         }
194
195         Bateau[] flotte;
196         flotte = new Bateau[nombre];
197         int index = 0;
198
199         for (int j = 0; j < mmsi.length; j++) {
200             res = state
201                 .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
202                     Position.mmsi=Bateau.mmsi WHERE Bateau.mmsi = "
203                     + mmsi[j] + " ");
204             res.first();
205             if (res.getInt("COUNT(DISTINCT Bateau.mmsi)") != 0) {
206                 System.out.printf("%3d ", index + 1);
207                 flotte[index] = recupererBateau(mmsi[j], connect);
208                 index++;
209             }
210         }
211         System.out.print(nombre + " bateaux chargés");
212         if (nombre != mmsi.length)
213             System.out.print(" (" + (mmsi.length - nombre)
214                         + " bateau(x) inexistant(s))");
215         System.out.println();
216         state.close();
217
218         return flotte;
219     }

```

```

220     } catch (Exception e) {
221         e.printStackTrace();
222     }
223     return null;
224 }
225
226 public static long renvoieMMSI(String nom) { // Renvoie un mmsi etant donné
227                                     // un nom de bateau
228     try {
229
230         Connection connect = mySQL.ConnectionSQL();
231         Statement state = connect.createStatement();
232
233         ResultSet res = state
234             .executeQuery("SELECT COUNT(*) FROM Bateau WHERE nom='"
235                         + nom + '\'' || nom_courant=''" + nom + "\';");
236         res.first();
237         int nombre = res.getInt("COUNT(*)");
238
239         if (nombre == 0) {
240             System.out.println("Nom entré incorrect !");
241             return 0;
242         } else if (nombre >= 2) {
243             System.out.println(nombre + " bateaux portent ce nom !");
244             return 0;
245         }
246
247         res.close();
248
249         ResultSet res1 = state
250             .executeQuery("SELECT mmsi FROM Bateau WHERE nom='"
251                         + nom + '\'' || nom_courant=''" + nom + "\';");
252         res1.first();
253         long mmsi = res1.getInt("mmsi");
254         res1.close();
255         state.close();
256
257         return mmsi;
258
259     } catch (Exception e) {
260         e.printStackTrace();
261     }
262     return 0;
263 }
264
265 /*
266 * Fonction pour le chargement de bateau entre deux dates
267 */
268
269 public static Bateau[] recupererFlotte(Date date1, Date date2) {
270     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
271     String dateString1 = sdf.format(date1);
272     String dateString2 = sdf.format(date2);
273     try {
274
275         Connection connect = mySQL.ConnectionSQL();
276         Statement state = connect.createStatement();
277
278         ResultSet res = state
279             .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
280                         Position.mmsi = Bateau.mmsi WHERE date_mesure >= \'
281                         + dateString1
282                         + '\' AND date_mesure <= \
283                         + dateString2 + "\';");
284         res.first();
285         int nombre = res.getInt("COUNT(DISTINCT Bateau.mmsi)");
286         System.out.println(nombre
287             + " bateaux présents dans la base de données.");
288         res.close();
289         if (nombre == 0) {

```

```

290     System.out.println("Attention : Flotte vide !");
291     return null;
292 }
293
294 Bateau[] flotte;
295 flotte = new Bateau[nombre];
296 int j = 0;
297
298 ResultSet res1 = state
299     .executeQuery("SELECT DISTINCT Bateau.mmsi FROM Bateau INNER JOIN Position ON Position.
300         mmsi=Bateau.mmsi WHERE date_mesure >= \'
301         + dateString1
302         + \"\ AND date_mesure <= \"
303         + dateString2 + "\';");
304
304 while (res1.next()) {
305     long identifiant = res1.getInt("mmsi");
306     System.out.printf("(%3d) ", j + 1);
307     flotte[j] = recupererBateau(identifiant, connect, date1, date2);
308     j++;
309 }
310 System.out.println(nombre + " bateaux chargés");
311 state.close();
312
313 return flotte;
314
315 } catch (Exception e) {
316     e.printStackTrace();
317 }
318 return null;
319 }
320
321 public static Bateau[] recupererFlotte(long[] mmsi, Date date1, Date date2) { // Recuperer
322     // uniquement
323     // les
324     // bateaux
325     // aux
326     // mmsi
327     // donnés
328     // en
329     // argument
330     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
331     String dateString1 = sdf.format(date1);
332     String dateString2 = sdf.format(date2);
333
334     try {
335
336         Connection connect = mySQL.ConnectionSQL();
337         Statement state = connect.createStatement();
338
339         // Tests pour vérifier que tous les mmsi donnés existe bien ...
340         String s = "";
341         for (int j = 0; j < mmsi.length; j++) {
342             s += mmsi[j] + "";
343             if (j != (mmsi.length - 1))
344                 s += ", ";
345         }
346         ResultSet res = state
347             .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
348                 Position.mmsi=Bateau.mmsi WHERE Bateau.mmsi IN (
349                     + s
350                     + ") AND date_mesure >= \
351                     + dateString1
352                     + \"\ AND date_mesure <= \" + dateString2 + "\';");
353         res.first();
354         int nombre = res.getInt("COUNT(DISTINCT Bateau.mmsi)");
355
356         if (nombre == 0) {
357             System.out.println("Attention : Flotte vide !");
358             return null;
359         }

```

```

359
360     Bateau[] flotte;
361     flotte = new Bateau[nombre];
362     int index = 0;
363
364     for (int j = 0; j < mmsi.length; j++) {
365         res = state
366             .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
367                 Position.mmsi=Bateau.mmsi WHERE Bateau.mmsi = "
368                 + mmsi[j]
369                 + " AND date_mesure >= \\'"
370                 + dateString1
371                 + "\\' AND date_mesure <= \\'"
372                 + dateString2 + "\' ;");
373         res.first();
374         if (res.getInt("COUNT(DISTINCT Bateau.mmsi)") != 0) {
375             System.out.printf("(%3d) ", index + 1);
376             flotte[index] = recupererBateau(mmsi[j], connect, date1,
377                 date2);
378             index++;
379         }
380     }
381     System.out.print(nombre + " bateaux chargés");
382     if (nombre != mmsi.length)
383         System.out.print(" (" + (mmsi.length - nombre)
384                         + " bateau(x) inexistant(s))");
385     System.out.println();
386
387     state.close();
388
389     return flotte;
390 }
391 } catch (Exception e) {
392     e.printStackTrace();
393 }
394 return null;
395 }
396
397 public static Flotte recupererFlotteF(Date date1, Date date2) { // Idem
398     // recupererFlotte()
399     // mais
400     // renvoie
401     // une
402     // Flotte et
403     // non un
404     // tableau
405     // de bateau
406     return new Flotte(recupererFlotte(date1, date2));
407 }
408
409 public static Flotte recupererFlotteF(long[] mmsi, Date date1, Date date2) { // Idem
410     // recupererFlotte(long
411     // []
412     // mmsi)
413     // mais
414     // renvoie
415     // une
416     // Flotte
417     // et
418     // non
419     // un
420     // tableau
421     // de
422     // bateau
423     return new Flotte(recupererFlotte(mmsi, date1, date2));
424 }
425
426 public static Bateau recupererBateau(long mmsi, Date date1, Date date2) { // Recuperer
427     // un
428     // bateau

```

```

429 // de
430 // mmsi
431 // donné
432 Bateau b = null;
433 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
434 String dateString1 = sdf.format(date1);
435 String dateString2 = sdf.format(date2);
436 try {
437     Connection connect = mySQL.ConnectionSQL();
438     Statement state = connect.createStatement();
439     ResultSet res = state
440         .executeQuery("SELECT COUNT(DISTINCT Bateau.mmsi) FROM Bateau INNER JOIN Position ON
441             Position.mmsi=Bateau.mmsi WHERE Bateau.mmsi = "
442             + mmsi
443             + " AND date_mesure >= \'"
444             + dateString1
445             + "\' AND date_mesure <= \'" + dateString2 + "\' ;");
446     res.first();
447     if (res.getInt("COUNT(DISTINCT Bateau.mmsi)") != 0)
448         b = recupererBateau(mmsi, connect, date1, date2);
449     else
450         System.out.println("MMSI inexistant : " + mmsi);
451 } catch (SQLException e) {
452     System.out.println(e.getMessage());
453 }
454
455 return b;
456 }
457
private static Bateau recupererBateau(long mmsi, Connection connect,
458     Date date1, Date date2) { // Recuperer un bateau de mmsi donné
459     try {
460         // Cr ation d'un objet Statement
461         Statement state = connect.createStatement();
462         ResultSet res = state
463             .executeQuery("SELECT * FROM Bateau WHERE mmsi = " + mmsi
464             + ";");
465         res.first();
466         res.getInt("mmsi");
467
468         String nom = res.getString("nom");
469         String nomCourant = res.getString("nom_courant");
470         double largeur = res.getDouble("largeur");
471         double longueur = res.getDouble("longueur");
472         int type = res.getInt("type");
473         String today = res.getString("temps_arrive");
474         Date ETA;
475         if (today != null) {
476             ETA = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
477                 .parse(today);
478         } else {
479             ETA = null;
480         }
481
482         res.close();
483
484         String dateString1 = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
485             .format(date1); // Conversion du temps en ms depuis 1970 en
486             // une date ...
487         String dateString2 = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
488             .format(date2); // Conversion du temps en ms depuis 1970 en
489             // une date ...
490
491         ResultSet res1 = state
492             .executeQuery("SELECT COUNT(*) FROM (SELECT Position.mmsi FROM Bateau INNER JOIN
493             Position ON Bateau.mmsi = Position.mmsi WHERE Bateau.mmsi = "
494             + mmsi
495             + " AND date_mesure >= \'"
496             + dateString1
497             + "\' AND date_mesure <= \'"
498             + dateString2

```

```

498     + " \') AS aux ;");
499 res1.first();
500 int r = res1.getInt("COUNT(*)");
501 res1.close();
502 System.out.println("Chargement du bateau " + nom + " (" + mmsi
503     + ") : " + r + " positions."); // A effacer
504
505 ResultSet result = state
506     .executeQuery("SELECT Position.date_mesure, Position.longitude, Position.latitude,
507         Position.x, Position.y, Position.sog, Position.cog, Position.vitesse_ang, Position.
508         cap FROM Bateau INNER JOIN Position ON Bateau.mmsi = Position.mmsi WHERE Bateau.mmsi
509     = "
510     + mmsi
511     + " AND date_mesure >= \'"
512     + dateString1
513     + "\' AND date_mesure <= \'"
514     + dateString2
515     + "\' ORDER BY date_mesure");
516
517 Position[] position = new Position[r];
518 int j = 0;
519 result.first();
520 do {
521     double d = result.getFloat(9);
522     int cap = (int) d;
523
524     today = result.getString("date_mesure");
525     Date date;
526     if (today != null) {
527         date = (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"))
528             .parse(today);
529     } else {
530         date = null;
531     }
532
533     position[j] = new Position(date, result.getFloat(2),
534         result.getFloat(3), result.getFloat(4),
535         result.getFloat(5), result.getFloat(6),
536         result.getFloat(7), result.getFloat(8), cap);
537
538     j++;
539 } while (result.next());
540 result.close();
541 state.close();
542 return new Bateau(mmsi, nom, nomCourant, type, position, longueur,
543     largeur, ETA, null);
544
545 } catch (Exception e) {
546     e.printStackTrace();
547 }
548 return null;
}
}

```

```

1 import java.io.File;
2
3 //Chargement de l'arbre de probabilités pour le groupe "statistiques".
4
5 /**
6 * Lister le contenu d'un répertoire
7 */
8 public class ListerFichiers {
9
10    private String cheminInitial = "";
11    private String[] listeDesFichiers;
12    public int filecount = 0;
13
14    /**
15     * Constructeur
16     *

```

```

17 * @param path
18 *           chemin du répertoire
19 * @param subFolder
20 *           analyse des sous dossiers
21 */
22 public ListerFichiers(String path) {
23     super();
24     cheminInitial = path;
25     listeDesFichiers = null;
26 }
27
28 public String[] lister() {
29     return this.listDirectory(this.cheminInitial);
30 }
31
32 public boolean existe(String nomFichier) {
33     File file = new File(cheminInitial);
34     File[] files = file.listFiles();
35     if (files != null) {
36         for (int i = 0; i < files.length; i++) {
37             if (!files[i].isDirectory()
38                 && files[i].getName().equals(nomFichier)) {
39                 return true;
40             }
41         }
42     }
43     return false;
44 }
45
46 public int existeSauvegarde() {
47     File file = new File(cheminInitial);
48     File[] files = file.listFiles();
49     int max = 0;
50     if (files != null) {
51         for (int i = 0; i < files.length; i++) {
52             if (!files[i].isDirectory()
53                 && files[i].getName().startsWith("sauvegarde")
54                 && files[i].getName().endsWith(".sql")) {
55                 try {
56                     int nb = Integer.parseInt(files[i].getName().substring(
57                         10, files[i].getName().length() - 4));
58                     if (nb > max)
59                         max = nb;
60                 } catch (NumberFormatException e) {
61                     // Do nothing
62                 }
63             }
64         }
65     }
66     return max;
67 }
68
69 private String[] listDirectory(String dir) {
70     File file = new File(dir);
71     File[] files = file.listFiles();
72
73     System.out.println("Repérage Fichier : ");
74
75     if (files != null) {
76         for (int i = 0; i < files.length; i++) {
77             if (!files[i].isDirectory()
78                 && files[i].getName().endsWith(".obj")) {
79                 System.out.println("Fichier vu : " + files[i].getName());
80                 filecount++;
81             }
82         }
83     }
84
85     listeDesFichiers = new String[filecount];
86     filecount = 0;
87 }
```

```

88     for (int i = 0; i < files.length; i++) {
89         if (!files[i].isDirectory())
90             && files[i].getName().endsWith(".obj")) {
91             listeDesFichiers[filecount] = cheminInitial + "\\"
92             + files[i].getName();
93             filecount++;
94         }
95     }
96 } else {
97     System.out.println("Pas de fichier dans : " + dir);
98 }
99 return listeDesFichiers;
100}
101}
102}

```

```

1 //Opérations sur un matrice carré d'ordre 2.
2
3 public class Matrice {
4     double a11;
5     double a22;
6     double a12;
7     double a21;
8
9     public Matrice (double a, double b, double c , double d) {
10        a11 = a;
11        a12 = b;
12        a21 = c ;
13        a22 = d;
14    }
15
16    public double det () {
17        return (a11*a22 - a12*a21);
18    }
19
20    public Matrice inverse () {
21        double d = this.det();
22        return (new Matrice (a22/d,-a12/d, -a21/d, a11/d));
23    }
24
25    public void afficher () {
26        System.out.println("|" + a11 + " " +a12+"|");
27        System.out.println("|" + a21 + " " +a22+"|");
28    }
29}
30}

```

```

1 import processing.core.*;
2
3 //Création du menu graphique
4
5 public class Menu {
6     public static PApplet parent;
7
8     public static void Menu() {
9
10        if (parent == null) {
11            parent = interfacegraphique.truc;
12        }
13
14        parent.textSize(24);
15        parent.fill(parent.color(0, 0, 0));
16        parent.textAlign(PConstants.CENTER);
17        parent.text("Port de Marseille", (float) interfacegraphique.SIZE_X / 9,
18                   (float) 3 / 50 * (float) interfacegraphique.SIZE_Y);
19
20        // coordonnées du pointeur.
21
22        // Menu principal et sous-menus:
23        if (interfacegraphique.state == 0) {

```

```

24     parent.stroke(0, 0, 0, 200);
25     parent.fill(parent.color(152, 152, 152));
26     parent.rectMode(PConstants.CENTER);
27     parent.rect((float) (float) 10 / 11
28         * (float) interfacegraphique.SIZE_X,
29         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
30         (float) (float) 140 / 889
31             * (float) interfacegraphique.SIZE_X,
32             (float) (float) 120 / 500
33                 * (float) interfacegraphique.SIZE_Y, 18, 18, 18, 18);
34     parent.textSize(15);
35     parent.fill(0, 0, 0);
36     parent.text("Menu", (float) (float) interfacegraphique.SIZE_X
37         * (float) 10 / 11,
38         (float) (float) interfacegraphique.SIZE_Y * 21 / 50);
39     parent.text("Options (Q)", (float) interfacegraphique.SIZE_X
40         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y
41             * (float) 23 / 50);
42     parent.text("Remise", (float) interfacegraphique.SIZE_X
43         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
44     parent.text("Aázéro (O)", (float) interfacegraphique.SIZE_X
45         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y
46             * (float) 27 / 50);
47
48 } else if (interfacegraphique.state == 1) {
49     parent.stroke(0, 0, 0, 200);
50     parent.fill(parent.color(152, 152, 152));
51     parent.rectMode(PConstants.CENTER);
52     parent.rect((float) (float) 10 / 11
53         * (float) interfacegraphique.SIZE_X,
54         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
55         (float) (float) 140 / 889
56             * (float) interfacegraphique.SIZE_X,
57             (float) (float) 120 / 500
58                 * (float) interfacegraphique.SIZE_Y, 18, 18, 18, 18);
59     parent.textSize(13);
60     parent.fill(0, 0, 0);
61     parent.text("Modifier un senseur (W)",
62         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
63         (float) interfacegraphique.SIZE_Y * 21 / 50);
64     parent.text("Modes d'affichage(M)",
65         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
66         (float) interfacegraphique.SIZE_Y * (float) 23 / 50);
67     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
68         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
69
70 } else if (interfacegraphique.state == 2) {
71     parent.stroke(0, 0, 0, 200);
72     parent.fill(parent.color(152, 152, 152));
73     parent.rectMode(PConstants.CENTER);
74     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
75         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
76         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
77         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
78             18, 18, 18);
79     parent.textSize(13);
80     parent.fill(0, 0, 0);
81     parent.text("créer un senseur (Q)",
82         (float) interfacegraphique.SIZE_X * 809 / 889,
83         (float) interfacegraphique.SIZE_Y * 21 / 50);
84     parent.text(" modifier un senseur(C)",
85         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
86         (float) interfacegraphique.SIZE_Y * (float) 23 / 50);
87     parent.text("détruire un senseur(X)",
88         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
89         (float) interfacegraphique.SIZE_Y / 2);
90     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
91         * (float) 10 / 11,
92         (float) interfacegraphique.SIZE_Y * 27 / 50);
93
94 } else if (interfacegraphique.state == 3) {

```

```

95     parent.stroke(0, 0, 0, 200);
96     parent.fill(parent.color(152, 152, 152));
97     parent.rectMode(PConstants.CENTER);
98     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
99                 (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
100                (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
101                (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
102                18, 18, 18);
103    parent.textSize(13);
104    parent.fill(200, 200, 255, 200);
105    parent.text("Annuler(A)", (float) interfacegraphique.SIZE_X
106      * (float) 10 / 11,
107      (float) interfacegraphique.SIZE_Y * 27 / 50);
108
109 } else if (interfacegraphique.state == 4) {
110     interfacegraphique.START = parent.millis();
111     parent.stroke(0, 0, 0, 200);
112     parent.fill(parent.color(152, 152, 152));
113     parent.rectMode(PConstants.CENTER);
114     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
115                 (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
116                 (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
117                 (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
118                 18, 18, 18);
119    parent.textSize(13);
120    parent.fill(0, 0, 0);
121    parent.text("en travaux",
122               (float) interfacegraphique.SIZE_X * 809 / 889,
123               (float) interfacegraphique.SIZE_Y * 24 / 50);
124    parent.text("Sortir(S)",
125               (float) interfacegraphique.SIZE_X * 809 / 889,
126               (float) interfacegraphique.SIZE_Y * 27 / 50);
127 } else if (interfacegraphique.state == 5) {
128     parent.stroke(0, 0, 0, 200);
129     parent.fill(parent.color(152, 152, 152));
130     parent.rectMode(PConstants.CENTER);
131     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
132                 (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
133                 (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
134                 (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
135                 18, 18, 18);
136    parent.textSize(13);
137    parent.fill(0, 0, 0);
138    parent.text("affichage du quadrillage(D)",
139               (float) interfacegraphique.SIZE_X * (float) 10 / 11,
140               (float) interfacegraphique.SIZE_Y * 19 / 50);
141    parent.text("Prévision statistique(T)",
142               (float) interfacegraphique.SIZE_X * (float) 10 / 11,
143               (float) interfacegraphique.SIZE_Y * (float) 21 / 50);
144    parent.text("Zones Sécurisées(Z)",
145               (float) interfacegraphique.SIZE_X * (float) 10 / 11,
146               (float) interfacegraphique.SIZE_Y * (float) 23 / 50);
147    parent.text("Affichages des Zones(!)",
148               (float) interfacegraphique.SIZE_X * 809 / 889,
149               (float) interfacegraphique.SIZE_Y / 2);
150    parent.text("Sortir(S)",
151               (float) interfacegraphique.SIZE_X * 809 / 889,
152               (float) interfacegraphique.SIZE_Y * 27 / 50);
153
154 } else if (interfacegraphique.state == 6) {
155     parent.stroke(0, 0, 0, 200);
156     parent.fill(parent.color(152, 152, 152));
157     parent.rectMode(PConstants.CENTER);
158     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
159                 (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
160                 (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
161                 (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
162                 18, 18, 18);
163     parent.textSize(13);
164     parent.fill(0, 0, 0);
165     parent.text("en travaux", (float) interfacegraphique.SIZE_X

```

```

166     * (float) 10 / 11,
167     (float) interfacegraphique.SIZE_Y * 24 / 50);
168 parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
169     * (float) 10 / 11,
170     (float) interfacegraphique.SIZE_Y * 27 / 50);
171
172 } else if (interfacegraphique.state == 7) {
173     parent.stroke(0, 0, 0, 200);
174     parent.fill(parent.color(152, 152, 152));
175     parent.rectMode(PConstants.CENTER);
176     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
177         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
178         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
179         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
180         18, 18, 18);
181     parent textSize(13);
182     parent.fill(0, 0, 0);
183     parent.text("Modifiez la position.", 
184         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
185         (float) interfacegraphique.SIZE_Y * (float) 23 / 50);
186     parent.text("Passer au sens.(V)", (float) interfacegraphique.SIZE_X
187         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
188     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
189         * (float) 10 / 11,
190         (float) interfacegraphique.SIZE_Y * 27 / 50);
191
192 } else if (interfacegraphique.state == 8) {
193     parent.stroke(0, 0, 0, 200);
194     parent.fill(parent.color(152, 152, 152));
195     parent.rectMode(PConstants.CENTER);
196     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
197         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
198         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
199         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
200         18, 18, 18);
201     parent textSize(13);
202     parent.fill(0, 0, 0);
203     parent.text("passer à la portée(P)", 
204         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
205         (float) interfacegraphique.SIZE_Y * 24 / 50);
206     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
207         * (float) 10 / 11,
208         (float) interfacegraphique.SIZE_Y * 26 / 50);
209
210 } else if (interfacegraphique.state == 9) {
211     parent.stroke(0, 0, 0, 200);
212     parent.fill(parent.color(152, 152, 152));
213     parent.rectMode(PConstants.CENTER);
214     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
215         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
216         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
217         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
218         18, 18, 18);
219     parent textSize(13);
220     parent.fill(0, 0, 0);
221     parent.text("Choisissez le type de senseur",
222         (float) interfacegraphique.SIZE_X * (float) 10 / 11,
223         (float) interfacegraphique.SIZE_Y * 21 / 50);
224     parent.text("(1) ou (2) ou(3)", (float) interfacegraphique.SIZE_X
225         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y
226         * (float) 23 / 50);
227     parent.text("Terminer(S)", (float) interfacegraphique.SIZE_X
228         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
229
230 } else if (interfacegraphique.state == 10) {
231     // quadrillage
232     parent.stroke(200);
233     for (int i = 0; i < interfacegraphique.FACTEUR_DE_DIVISION; i++) {
234         parent.line(0, i * (float) interfacegraphique.SIZE_Y
235             / interfacegraphique.FACTEUR_DE_DIVISION,
236             (float) interfacegraphique.SIZE_X, i

```

```

237         * (float) interfacegraphique.SIZE_Y
238         / interfacegraphique.FACTEUR_DE_DIVISION);
239     }
240     for (int i = 0; i < 2 * interfacegraphique.FACTEUR_DE_DIVISION; i++) {
241         parent.line(i * (float) interfacegraphique.SIZE_Y
242         / interfacegraphique.FACTEUR_DE_DIVISION, 0, i
243         * (float) interfacegraphique.SIZE_Y
244         / interfacegraphique.FACTEUR_DE_DIVISION,
245         (float) interfacegraphique.SIZE_Y);
246     }
247     parent.noStroke();
248
249     parent.stroke(0, 0, 200);
250     parent.fill(parent.color(152, 152, 152));
251     parent.rectMode(PConstants.CENTER);
252     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
253         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
254         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
255         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
256         18, 18, 18);
257     parent.textSize(13);
258     parent.fill(0, 0, 0);
259     parent.text("Terminer(S)", (float) interfacegraphique.SIZE_X
260         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
261 } else if (interfacegraphique.state == 11) {
262     parent.stroke(0, 0, 200);
263     parent.fill(parent.color(152, 152, 152));
264     parent.rectMode(PConstants.CENTER);
265     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
266         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
267         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
268         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
269         18, 18, 18);
270     parent.textSize(13);
271     parent.fill(0, 0, 0);
272     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
273         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
274 }
275 } else if (interfacegraphique.state == 12) {
276     parent.stroke(0, 0, 200);
277     parent.fill(parent.color(152, 152, 152));
278     parent.rectMode(PConstants.CENTER);
279     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
280         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
281         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
282         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
283         18, 18, 18);
284     parent.textSize(13);
285     parent.fill(0, 0, 0);
286     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
287         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
288 }
289 } else if (interfacegraphique.state == 13) {
290     parent.stroke(0, 0, 200);
291     parent.fill(parent.color(152, 152, 152));
292     parent.rectMode(PConstants.CENTER);
293     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
294         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
295         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
296         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
297         18, 18, 18);
298     parent.textSize(13);
299     parent.fill(0, 0, 0);
300     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
301         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
302 }
303 } else if (interfacegraphique.state == 14) {
304     parent.stroke(0, 0, 200);
305     parent.fill(parent.color(152, 152, 152));
306     parent.rectMode(PConstants.CENTER);
307     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,

```

```

308     (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
309     (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
310     (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
311     18, 18, 18);
312 parent.setTextSize(13);
313 parent.fill(200, 200, 255, 200);
314 parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
315     * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
316
317 } else if (interfacegraphique.state == 15) {
318     parent.stroke(0, 0, 0, 200);
319     parent.fill(parent.color(152, 152, 152));
320     parent.rectMode(PConstants.CENTER);
321     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
322         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
323         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
324         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
325         18, 18, 18);
326     parent.setTextSize(13);
327     parent.fill(0, 0, 0);
328     parent.text("Pointez un bateau", (float) interfacegraphique.SIZE_X
329         * (float) 10 / 11,
330         (float) interfacegraphique.SIZE_Y * 22 / 50);
331     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
332         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
333
334 } else if (interfacegraphique.state == 16) {
335     parent.stroke(0, 0, 0, 255);
336     parent.fill(parent.color(152, 152, 152));
337     parent.rectMode(PConstants.CENTER);
338     parent.rect((float) 10 / 11 * (float) interfacegraphique.SIZE_X,
339         (float) interfacegraphique.SIZE_Y * (float) 23 / 50,
340         (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
341         (float) 120 / 500 * (float) interfacegraphique.SIZE_Y, 18,
342         18, 18, 18);
343     parent.setTextSize(13);
344     parent.fill(0, 0, 0);
345     parent.text("Sortir(S)", (float) interfacegraphique.SIZE_X
346         * (float) 10 / 11, (float) interfacegraphique.SIZE_Y / 2);
347
348     parent.stroke(0, 0, 0, 200);
349     parent.fill(parent.color(152, 152, 152));
350     parent.rectMode(parent.CENTER);
351     parent.rect((float) 175 / 889 * interfacegraphique.SIZE_X,
352         (float) 230 / 500 * interfacegraphique.SIZE_Y, (float) 300
353             / 889 * interfacegraphique.SIZE_X, (float) 290
354                 / 500 * interfacegraphique.SIZE_Y, (float) 18,
355                     (float) 18, (float) 18, (float) 18);
356
357     parent.setTextSize(13);
358     parent.fill(0, 0, 0, 255);
359     parent.textAlign(parent.CENTER);
360     int k1 = 1;
361     int k2 = 1;
362     parent.text("Bateaux en ZONE SECURISEE 1:", (float) 100 / 889
363         * (float) interfacegraphique.SIZE_X,
364         (float) interfacegraphique.SIZE_Y / 5);
365     parent.text("Bateaux en ZONE SECURISEE 2:", (float) 250 / 889
366         * (float) interfacegraphique.SIZE_X,
367         (float) interfacegraphique.SIZE_Y / 5);
368     for (int i = 0; i < GestionPort.flotte.getbateaux().length; i++) {
369         Position position = GestionPort.flotte.getbateau(i).getPosNorm(
370             interfacegraphique.TIME);
371         if (position == null) {
372         } else {
373             double[] tab = {
374                 GestionPort.flotte.getbateau(i)
375                     .getPosNorm(interfacegraphique.TIME).getX(),
376                 GestionPort.flotte.getbateau(i)
377                     .getPosNorm(interfacegraphique.TIME).getY() };

```

```

379         if (Zone.ZONE_SECURISE1(tab)) {
380
381             parent.text((int) GestionPort.flotte.getBateau(i)
382                         .getMMSI(), (float) 100 / 889
383                         * (float) interfacegraphique.SIZE_X,
384                         (float) (100 + 10 * k1) / 500
385                         * (float) interfacegraphique.SIZE_Y);
386             k1++;
387         }
388         if (Zone.ZONE_SECURISE2(tab)) {
389
390             parent.text((int) GestionPort.flotte.getBateau(i)
391                         .getMMSI(), (float) 250 / 889
392                         * (float) interfacegraphique.SIZE_X,
393                         (float) (100 + 10 * k2) / 500
394                         * (float) interfacegraphique.SIZE_Y);
395             k2++;
396         }
397     }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
```

```

1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5 import java.sql.Connection;
6 import java.sql.DriverManager;
7 import java.sql.SQLException;
8 import java.sql.Statement;
9
10 // Connexion avec la base de donnée en mySQL
11
12 public class mySQL {
13     private static Connection connection = null;
14
15     public static Connection ConnectionSQL() {
16         if (connection != null)
17             return connection;
18
19         try {
20             System.out.print("\nConnexion à la base de donnée mySQL... ");
21             Class.forName("com.mysql.jdbc.Driver");
22             // System.out.println("Driver O.K.");
23
24             String url = "jdbc:mysql://localhost/mig";
25             String user = "user";
26             String passwd = "password";
27             connection = DriverManager.getConnection(url, user, passwd);
28             System.out.println("ok");
29             return connection;
30         } catch (ClassNotFoundException e) {
31             System.out.println("\nErreur de Connexion à la base de donnée : "
32                             + e.getMessage());
33             e.printStackTrace();
34         } catch (SQLException e) {
35             System.out.println("\nErreur de Connexion à la base de donnée : "
36                             + e.getMessage());
37             e.printStackTrace();
38         }
39         return null;
40     }
41
42     public static void ViderTable() {
43         System.out.print("\nVidage de la base de donnée...");
```

```

44     Connection connect = ConnectionSQL();
45     try {
46         Statement state = connect.createStatement();
47         state.executeUpdate("DROP TABLE IF EXISTS Bateau");
48         state.executeUpdate("DROP TABLE IF EXISTS Position");
49         System.out.println("Table Vidée !");
50         state.close();
51     } catch (SQLException e) {
52         System.out.println("Erreur relative à la base de donnée : "
53             + e.getMessage());
54         e.printStackTrace();
55     }
56 }
57
58 public static void ChargerTable() {
59     ChargerTable(null);
60 }
61
62 public static void ChargerTable(String s) {
63
64     try {
65         System.out.print("\nChargement de la base de donnée ");
66
67         if (s == null) { // Si on ne précise pas de chemin de chargement, on
68             // se place par défaut dans : \mesures
69         ListerFichiers listeurFichier = new ListerFichiers(System
70             .getProperties().get("user.dir") + "\\mesures");
71         int aCharger = listeurFichier.existeSauvegarde();
72         if (aCharger == 0) {
73             System.out.println("\nPas de base de donnée trouvée ...");
74             return;
75         }
76         System.out.print("sauvegarde" + aCharger + ".sql ... ");
77         s = System.getProperties().get("user.dir")
78             + "\\mesures\\sauvegarde" + aCharger + ".sql";
79     }
80     String con = new String(
81         "mysql -u user -ppassword --default-character-set=utf8 -e \"source "
82         + s + "\\ mig\"");
83
84     File tmpFile = new File("tmp.bat");
85     FileOutputStream out = new FileOutputStream(tmpFile);
86     out.write(con.getBytes());
87     out.close();
88
89     String[] commande = { "tmp.bat" };
90     Runtime r = Runtime.getRuntime();
91     Process p = r.exec(commande);
92     p.waitFor();
93     System.out.println("ok\nBase de donnée chargée depuis " + s);
94     tmpFile.delete();
95 } catch (FileNotFoundException e) {
96     // TODO Auto-generated catch block
97     e.printStackTrace();
98 } catch (IOException e) {
99     // TODO Auto-generated catch block
100    e.printStackTrace();
101 } catch (InterruptedException e) {
102     // TODO Auto-generated catch block
103     e.printStackTrace();
104 }
105
106 }
107
108 public static void EnregistrerTable() {
109     EnregistrerTable(null);
110 }
111
112 public static void EnregistrerTable(String s) {
113     try {
114         System.out.print("\nEnregistrement de la base de donnée ...");

```

```

115     if (s == null) { // Si on ne précise pas de chemin d'enregistrement,
116         // on se place par défaut dans : \mesures
117         ListerFichiers listeurFichier = new ListerFichiers(System
118             .getProperties().get("user.dir") + "\\mesures");
119         int aSauver = listeurFichier.existeSauvegarde();
120         s = System.getProperties().get("user.dir")
121             + "\\mesures\\sauvegarde" + (aSauver + 1) + ".sql";
122         System.out.println(" sauvegarde" + (aSauver + 1) + ".sql");
123     }
124
125     String con = new String("mysqldump -u user -ppassword mig > "
126         + "\"" + s + "\"");
127     File tmpFile = new File("tmp.bat");
128     FileOutputStream out = new FileOutputStream(tmpFile);
129     out.write(con.getBytes());
130     out.close();
131
132     String [] commande = { "tmp.bat" };
133     Runtime r = Runtime.getRuntime();
134     Process p = r.exec(commande);
135     p.waitFor();
136     System.out.println("Base de donnée enregistrée sous " + s + ".");
137     tmpFile.delete();
138
139 } catch (FileNotFoundException e) {
140     System.out.println("Erreur");
141     e.printStackTrace();
142 } catch (IOException e) {
143     System.out.println("Erreur");
144     e.printStackTrace();
145 } catch (InterruptedException e) {
146     System.out.println("Erreur");
147     e.printStackTrace();
148 }
149
150 }
151
152 public static void FermerConnection() {
153     try {
154         if (connection != null) {
155             connection.close();
156             System.out.println("Connection SQL fermée");
157         }
158     } catch (SQLException e) {
159         e.printStackTrace();
160     }
161 }
162 }
163 }
```

```

1 import java.util.Scanner;
2
3 import processing.core.PApplet;
4
5 //Fonction pause dans le menu de l'interface graphique.
6
7 public class PauseEvents {
8     static PApplet parent;
9     public static boolean trajaafficher = false;
10
11    public static void pause() {
12        if (parent == null) {
13            parent = interfacegraphique.truc;
14        }
15
16        if (interfacegraphique.j == 0) {
17            interfacegraphique.j = 1;
18            parent.loop();
19            interfacegraphique.PAUSE = (int) (parent.millis() - interfacegraphique.START)
20                / 100 - interfacegraphique.PAUSE;
21    }
22}
```

```

22 } else {
23     interfacegraphique.j = 0;
24     Bateau bateauchoisi = getBateau(GestionPort.flotte);
25     if (bateauchoisi != null) {
26         trajaafficher = true;
27
28         System.out.println(bateauchoisi);
29         interfacegraphique.bateauchoisi = 1;
30     } else {
31         trajaafficher = false;
32         System.out.println("bateau null");
33     }
34     parent.noLoop();
35     interfacegraphique.PAUSE = interfacegraphique.TIME;
36
37 }
38
39 }
40
41 public static double[][] creerTableauCoord(Bateau b) {
42     int longueur = 0;
43
44     for (int j = 0; j < interfacegraphique.TIME - b.getMinPosNorm(); j++) {
45         if (b.getPosNorm(j) != null) {
46             longueur++;
47         }
48     }
49     double[][] trajectoirePassee = new double[longueur][2];
50     int x = 3;
51     double[][] trajectoirePasseeTronquee = new double[x][2];
52
53     if (interfacegraphique.TIME < b.getMinPosNorm()) {
54         System.err.println("Ce bateau n'est pas encore parti");
55     } else if (b.getMaxPosNorm() < interfacegraphique.TIME) {
56         System.err.println("ce bateau a fini son parcours");
57     } else {
58         for (int pos = 0; pos < longueur; pos++) {
59             if (b.getPosNorm(pos) != null) {
60                 trajectoirePassee[pos][0] = (int) (b.getPosNorm(pos).getX());
61                 trajectoirePassee[pos][1] = (int) (b.getPosNorm(pos).getY());
62             }
63         }
64     }
65
66     if (longueur == 0) {
67     } else {
68         // for(int k=longueur-x+1;k<=longueur;k++){
69         // trajectoirePasseeTronquee[k-longueur+x-1]=trajectoirePassee[longueur-x-(k-longueur-x+1)];
70         if (longueur >= 3) {
71
72             trajectoirePasseeTronquee[2][0] = trajectoirePassee[longueur - 1][0];
73             trajectoirePasseeTronquee[2][1] = trajectoirePassee[longueur - 1][1];
74             trajectoirePasseeTronquee[1][0] = trajectoirePassee[longueur - 2][0];
75             trajectoirePasseeTronquee[1][1] = trajectoirePassee[longueur - 2][1];
76             trajectoirePasseeTronquee[0][0] = trajectoirePassee[longueur - 3][0];
77             trajectoirePasseeTronquee[0][1] = trajectoirePassee[longueur - 3][1];
78         } else if (longueur == 1) {
79             trajectoirePasseeTronquee[0][0] = trajectoirePassee[longueur - 1][0];
80             trajectoirePasseeTronquee[0][1] = trajectoirePassee[longueur - 1][1];
81         }
82
83     // }
84 }
85
86 return trajectoirePasseeTronquee;
87 }
88
89 public static Bateau choisirBateau(int mmsi, Flotte flotte) {
90     for (int i = 0; i < flotte.getNombreBateaux(); i++) {
91         if (flotte.getBateau(i).getMMSI() == mmsi) {
92             return flotte.getBateau(i);

```

```

93     }
94
95     }
96     return null;
97 }
98
99 public static int stringToInt( String s) {
100    int res = 0;
101    boolean b = true;
102
103    for (int j = 0; j < s.length(); j++) {
104        if (Character.isDigit(s.charAt(j)) == false) {
105            b = false;
106        }
107    }
108
109    if (s.equals("") == false && b == true) {
110        res = Integer.parseInt(s);
111    }
112    return res;
113 }
114
115 public static int typeMmsi() {
116    Scanner sc = new Scanner(System.in);
117    String s = "2";
118
119    s = sc.nextLine();
120
121    System.out.println(s);
122    return stringToInt(s);
123 }
124
125
126 public static Bateau getBateau(Flotte flotte) {
127    return choisirBateau(typeMmsi(), flotte);
128 }
129
130
131 // affichage des ellipses
132
133 public static double[] t(double x, double y) {
134    double[] tab = new double[2];
135    tab[0] = x;
136    tab[1] = y;
137    return tab;
138 }
139
140
141 public static double[] in1 = new double[7];
142 public static double[][] ce1 = new double[7][7];
143 public static Matrice[] cov1 = new Matrice[7];
144 public static double[][] moy1 = new double[7][2];
145 public static double[] in2 = new double[7];
146 public static double[][] ce2 = new double[7][7];
147 public static double[][] moy3 = new double[6][2];
148 public static double[] in3 = new double[6];
149 public static Matrice[] cov2 = new Matrice[7];
150 public static double[][] moy2 = new double[7][2];
151 public static double[][] ce3 = new double[6][6];
152 public static Matrice[] cov3 = new Matrice[6];
153
154 public static void bidouille() {
155    in1[0] = 0.3;
156    in1[1] = 0.2;
157    in1[2] = 0.2;
158    in1[3] = 0.2;
159    in1[4] = 0.05;
160    in1[5] = 0.05;
161    in1[6] = 0.;
162
163    ce1[0][0] = 0.7677765;
164    ce1[0][1] = 0.23222348;

```

```

164 ce1 [ 1 ][ 1 ] = 0.87665;
165 ce1 [ 1 ][ 2 ] = 0.12334;
166 ce1 [ 2 ][ 2 ] = 0.968425;
167 ce1 [ 2 ][ 3 ] = 0.031574;
168 ce1 [ 3 ][ 3 ] = 0.99695221;
169 ce1 [ 3 ][ 4 ] = 0.00304778;
170 ce1 [ 4 ][ 4 ] = 0.99998;
171 ce1 [ 4 ][ 5 ] = 0.00002;
172 ce1 [ 5 ][ 5 ] = 0.9999995;
173 ce1 [ 5 ][ 6 ] = 0.0000005;
174 ce1 [ 6 ][ 6 ] = 1;
175
176 cov1 [ 0 ] = new Matrice(3256.16, -123.78, -123.78, 3866.62);
177 cov1 [ 1 ] = new Matrice(1524.6, 1379.38, 1379.38, 8117.7);
178 cov1 [ 2 ] = new Matrice(2185.61, 2867.311, 2867.311, 16147.3869);
179 cov1 [ 3 ] = new Matrice(11018.73, 7078.666577865967, 7078.666577866,
180 26729.609395456);
181 cov1 [ 4 ] = new Matrice(74545.8, 24700.665, 24700.665, 15531.919);
182 cov1 [ 5 ] = new Matrice(97583.655, 10463.55127, 10463.55127, 3501.8226);
183 cov1 [ 6 ] = new Matrice(62336, 61791, 61791, 99744);
184
185 moy1 [ 0 ] = t(632.9390663542, 456.88818);
186 moy1 [ 1 ] = t(624.79364, 383.957);
187 moy1 [ 2 ] = t(604.4087698, 278.0835);
188 moy1 [ 3 ] = t(550.3348, 200.08);
189 moy1 [ 4 ] = t(434.582946, 165.8854098);
190 moy1 [ 5 ] = t(353.08639, 172.826034);
191 moy1 [ 6 ] = t(345, 200);
192
193 in2 [ 0 ] = 0.3;
194 in2 [ 1 ] = 0.2;
195 in2 [ 2 ] = 0.2;
196 in2 [ 3 ] = 0.2;
197 in2 [ 4 ] = 0.05;
198 in2 [ 5 ] = 0.05;
199 in2 [ 6 ] = 0;
200
201 ce2 [ 0 ][ 0 ] = 0.76754;
202 ce2 [ 0 ][ 1 ] = 0.23246;
203 ce2 [ 1 ][ 1 ] = 0.8593;
204 ce2 [ 1 ][ 2 ] = 0.1407;
205 ce2 [ 2 ][ 2 ] = 0.96534;
206 ce2 [ 2 ][ 3 ] = 0.03466;
207 ce2 [ 3 ][ 3 ] = 0.99654;
208 ce2 [ 3 ][ 4 ] = 0.00346;
209 ce2 [ 4 ][ 4 ] = 0.99998;
210 ce2 [ 4 ][ 5 ] = 0.00002;
211 ce2 [ 5 ][ 5 ] = 0.99999999457;
212 ce2 [ 5 ][ 6 ] = 0.00000000543;
213 ce2 [ 6 ][ 6 ] = 1;
214
215 cov2 [ 0 ] = new Matrice(891.39668, 1144.571, 1144.571, 3872.909);
216 cov2 [ 1 ] = new Matrice(1160.916779, 2548.697665, 2548.697665, 7866.67385);
217 cov2 [ 2 ] = new Matrice(1498.2066777, 3816.947979, 3816.947979,
218 14802.92447);
219 cov2 [ 3 ] = new Matrice(10811.385, 14872.387884, 14872.387884, 22583.6735);
220 cov2 [ 4 ] = new Matrice(44141.129, 30035.99, 30035.99, 20804.664877);
221 cov2 [ 5 ] = new Matrice(30293.628, 17263.89, 17263.89, 10276.8557);
222 cov2 [ 6 ] = new Matrice(926, 2411, 2411, 6500);
223
224 moy2 [ 0 ] = t(656.9485849242876, 455.1996254511965);
225 moy2 [ 1 ] = t(635.4330927238937, 379.6467843012057);
226 moy2 [ 2 ] = t(606.3886384727871, 279.6707844070029);
227 moy2 [ 3 ] = t(541.640779978425, 203.11535328469807);
228 moy2 [ 4 ] = t(450.79861128143796, 152.995945264709);
229 moy2 [ 5 ] = t(410.57378256274495, 131.65138132843478);
230 moy2 [ 6 ] = t(419.80380181178873, 130.00791926297612);
231
232 in3 [ 0 ] = 0.3;
233 in3 [ 1 ] = 0.2;
234 in3 [ 2 ] = 0.2;

```

```

235     in3[3] = 0.2;
236     in3[4] = 0.05;
237     in3[5] = 0.05;
238
239     ce3[0][0] = 0.77;
240     ce3[0][1] = 0.23;
241     ce3[1][1] = 0.98;
242     ce3[1][2] = 0.02;
243     ce3[2][2] = 0.92;
244     ce3[2][3] = 0.08;
245     ce3[3][3] = 0.99;
246     ce3[3][4] = 0.01;
247     ce3[4][4] = 0.99;
248     ce3[4][5] = 0.01;
249     ce3[5][5] = 1;
250
251     cov3[0] = new Matrice(1790, -475, -475, 2280);
252     cov3[1] = new Matrice(333, 443, 443, 4370);
253     cov3[2] = new Matrice(907, -1282, -2282, 13202);
254     cov3[3] = new Matrice(2164, -6800, -6800, 26508);
255     cov3[4] = new Matrice(6209, -9523, -9523, 15089);
256     cov3[5] = new Matrice(36, -41, -41, 47);
257
258     moy3[0] = t(658, 460);
259     moy3[1] = t(642, 362);
260     moy3[2] = t(633, 284);
261     moy3[3] = t(636, 199);
262     moy3[4] = t(562, 170);
263     moy3[5] = t(543, 166);
264
265 }
266
267 public static hmmcontinus2 hmm1 = new hmmcontinus2(7, in1, ce1, moy1, cov1);
268 public static hmmcontinus2 hmm2 = new hmmcontinus2(7, in2, ce2, moy2, cov2);
269 public static hmmcontinus2 hmm3 = new hmmcontinus2(6, in3, ce3, moy3, cov3);
270
271 public static void afficherEllipses(double[][] pos) {
272     bidouille();
273     hmm1.cons(6, in1, ce1, moy1, cov1);
274     hmm2.cons(7, in2, ce2, moy2, cov2);
275     hmm3.cons(7, in3, ce3, moy3, cov3);
276
277     int l = pos.length;
278     double[][] position = new double[l][2];
279     for (int i = 0; i < l; i++) {
280         double[] t = new double[2];
281         t[0] = ConversionPixel.conversionPixelX((float) pos[i][0]);
282         t[1] = ConversionPixel.conversionPixelY((float) pos[i][1]);
283         position[i] = t.clone();
284     }
285     double[] proba = new double[3];
286     proba[0] = hmm1.proba(hmm2.traitemenDonnées(position));
287     proba[1] = hmm2.proba(hmm2.traitemenDonnées(position));
288     proba[2] = hmm3.proba(hmm2.traitemenDonnées(position));
289     for (int i = 0; i < 3; i++) {
290         if (!(proba[i] > 0))
291             proba[i] = 0;
292     }
293     double max = proba[0];
294     int m = 0;
295     if (proba[1] > proba[0]) {
296         max = proba[1];
297         m = 1;
298     }
299     if (proba[2] > max) {
300         max = proba[2];
301         m = 2;
302     }
303
304     AffichageHmm.Affiche(m + 1);
305

```

```
306     }
307 }
308 }
```

```
1 import java.util.Arrays;
2
3
4 public class perturbation {
5
6     public final static int X = 82;
7     public final static int Y = 50;
8     public final static int T = 24 * 60 / 2;
9     public final static int deltaT = 30;
10
11    public static int[][] insererLongueur(int[][] t) {
12        int[][] res = new int[t.length + 1][2];
13
14        res[0][0] = t.length;
15        for (int i = 1; i < t.length + 1; i++) {
16            res[i][0] = t[i - 1][0];
17            res[i][1] = t[i - 1][1];
18        }
19
20        return res;
21    }
22
23    public static Bateau[] recopierTableau(Bateau[] flotte) {
24        Bateau[] res = new Bateau[flotte.length];
25        for (int i = 0; i < flotte.length; i++) {
26            res[i] = flotte[i];
27        }
28        return res;
29    }
30
31 /**
32 * Lisse la trajectoire en fin de traitement (minimise les changements de cap)
33 *
34 * @param traj : trajectoire actuelle
35 *
36 * @return traj : trajectoire lissée
37 */
38    public static int[][] lissageTrajDiag (int[][] traj) {
39        int lg = traj[0][0]; // longueur du trajet
40        int[][] compteur = new int[X][Y];
41        contour.initialiserMatrice(compteur);
42        int k = 0;
43
44        // Remplissage de la matrice compteur
45        for (int i = 1; i < lg+1; i++) {
46            compteur[traj[i][0]][traj[i][1]] = k;
47            k++;
48        }
49
50        // Nouvelle trajectoire
51        return contour.compteurToChemin(compteur, traj[1][0], traj[1][1], k, 0, true);
52    }
53
54 /**
55 * Vérifie si le point est dans la zone (existe dans la matrice carte)
56 */
57    public static boolean existe (int x, int y) {
58        return (x >= 0 && x < X) && (y >= 0 && y < Y);
59    }
60
61 /**
62 * Renvoie la carte des zones interdites selon la position des bateaux davantage prioritaires
63 *
64 * @param carte : carte des obstacles
65 * @param flotte : liste de Bateau déployés dans la zone
66 * @param indC : indice du bateau courant dans flotte
67 *
```

```

68 * @return bool : true si la fonction a modifié la carte ; false sinon
69 */
70 public static boolean interdireZones (boolean[][] carte, Bateau[] flotte, int indC) {
71     Bateau b0 = flotte[indC];
72     boolean modif = false;
73
74     for (int i = 0; i < indC; i++) {
75         Bateau b1 = flotte[i];
76         int zoneEvitement = 1;
77
78         int temp = b0.getMinPosNorm() - b1.getMinPosNorm();
79         // b0 est le dernier bateau arrivant dans la zone
80         if (temp > 0) { }
81     else {
82         Bateau transit = b1;
83         b1 = b0;
84         b0 = transit;
85     }
86     temp = Math.abs(temp);
87
88     for (int t = 1; t < b1.traj.length && t + temp < b0.traj.length; t++) {
89         int t0 = t + temp;
90         int t1 = t;
91         if (Math.pow(b1.traj[t1][0] - b0.traj[t0][0], 2) + Math.pow(b1.traj[t1][1] - b0.traj[t0][1], 2) <= Math.pow(zoneEvitement, 2)) {
92             int x1 = b1.traj[t1][0];
93             int y1 = b1.traj[t1][1];
94
95             for (int k0 = -zoneEvitement; k0 <= zoneEvitement; k0++) {
96                 for (int k1 = -zoneEvitement; k1 <= zoneEvitement; k1++) {
97                     if (existe(x1+k0, y1+k1)) { carte[x1+k0][y1+k1] = false; modif = true; } //&& (k0 ==
98                         0 || k1 == 0)
99                 }
100            }
101        }
102    }
103 }
104 return modif;
105 }
106
107 /**
108 * Calcule les nouvelles trajectoires des bateaux déployés dans la zone pour éviter toute
109 * collision
110 *
111 * @param flotte : flotte dans l'état prévu par les modèles statistiques
112 *
113 * @return flotte : flotte dont les trajectoires ont été modifiées pour éviter les collisions
114 */
115 public static Bateau[] gerer(Bateau[] flotte0) {
116     // Initialisation de la carte
117     Bateau[] flotte = recopierTableau(flotte0); // On recopie pour éviter de modifier l'ordre des
118     // bateaux à l'extérieur
119     boolean[][] carte = new boolean[X][Y];
120     reinitialiser(carte);
121
122     // On trie les bateaux du plus prioritaire au moins prioritaire
123     Arrays.sort(flotte);
124
125     // Pour chaque bateau
126     for (int i = 0; i < flotte.length; i++) {
127         // On regarde s'il entre en collision avec les autres
128         while (interdireZones (carte, flotte, i)) {
129             // On lance la procédure de contour
130             // On remplace par la nv traj
131             flotte[i].traj = contour.parcours(carte, flotte[i].traj); //contour.parcours(carte, flotte[
132                         i].traj);
133             reinitialiser(carte);
134         }
135     }

```

```

134     return flotte;
135 }
136
137
138 /**
139 * Affiche la carte (int) en repère cartésien
140 */
141 public static void afficher(int [][] a) {
142     int n = a[0].length;
143     int m = a.length;
144
145     for (int i=n-1; i >= 0; i--) {
146         for (int j=0; j < m; j++) {
147             System.out.printf("%3d | ", a[j][i]);
148         }
149         System.out.println();
150     }
151 }
152 /**
153 * Affiche la carte (boolean) en repère cartésien
154 */
155 public static void afficher(boolean [][] a) {
156     int n = a[0].length;
157     int m = a.length;
158
159     for (int i=n-1; i >= 0; i--) {
160         for (int j=0; j < m; j++) {
161             if (a[j][i]) { System.out.print(" 1 |"); }
162             else { System.out.print(" 0 |"); }
163         }
164         System.out.println();
165     }
166 }
167 /**
168 * Affiche le tableau des positions dans l'ordre
169 */
170 public static void afficher_traj(int [][] a) {
171     int n = a.length;
172     int m = a[0].length;
173
174     for (int i=0; i < n; i++) {
175         for (int j=0; j < m; j++) {
176             System.out.printf("%3d | ", a[i][j]);
177         }
178         System.out.println();
179     }
180 }
181 /**
182 * Affiche le tableau
183 */
184 public static void afficher(int [] a) {
185     int n = a.length;
186     for (int i=0; i < n; i++) {
187         System.out.printf("%2d | ", a[i]);
188     }
189 }
190
191 public static void reinitialiser(boolean [][] carte) {
192     for(int i=0; i < carte.length; i++) {
193         for(int j=0; j < carte[0].length; j++) {
194             carte[i][j] = true;
195         }
196     }
197 }
198
199 /**
200 * Vérifie que le bateau ne sort pas des limites de la carte
201 * (évite de croire qu'il y a des bugs d'indice qui n'existent pas)
202 *
203 * @param b : Bateau dont il faut tester la trajectoire
204 */

```

```

205 * @return void : mais affiche un message d'erreur et quitte en cas de problème
206 */
207 public static void neSortPas(Bateau b) {
208     neSortPas(b.raj);
209 }
210 public static void neSortPas(int [][] traj) {
211     for (int i = 1; i < traj.length; i++) {
212         if (!existe(traj[i][0], traj[i][1])) {
213             System.err.printf("ERREUR : le bateau quitte la carte à l'instant %d\n", i);
214             System.exit(1);
215         }
216     }
217 }
218
// Fonction test
219 public static void main(String [] args) {
220     boolean [][] carte = new boolean[X][Y];
221
222     for(int i=0; i < carte.length; i++) {
223         for(int j=0; j < carte[0].length; j++) {
224             carte[i][j] = true;
225         }
226     }
227
228     int [][] p0 = new int [28][2];
229     int [][] p1 = new int [28][2];
230
231     p0[0][0] = p0.length-1;
232     p1[0][0] = p1.length-1;
233
234     for (int i = 1; i < p0.length; i++) {
235         p0[i][0] = 30-i;
236         p0[i][1] = 15;
237     }
238     for (int i = 1; i < p1.length; i++) {
239         p1[i][0] = 15;
240         p1[i][1] = i;
241     }
242 }
243
244 Bateau b0 = new Bateau("Un", p0, 10, 10, 1, 12);
245 Bateau b1 = new Bateau("Deux", p1, 8, 3, 1, 33);
246
247 neSortPas(b0);
248 neSortPas(b1);
249
250 Bateau [] flotte = {b0, b1};
251
252 gerer(flotte);
253 afficher_raj(flotte[0].raj);
254 System.out.println();
255 afficher_raj(flotte[1].raj);
256 }
257
258 }

```

```

1 import java.util.Date;
2
3 /*Un objet "position" est contenu dans chaque bateau
4 et contient lui-même des informations relatives aux positions du bateau.*/
5
6 public class Position {
7
8     private double longitude;
9     private double latitude;
10    private double x;
11    private double y;
12    private double sog; // Vitesse par rapport au sol
13    private double cog; // Cap par rapport au sol
14    private int cap;
15    private double vitesseAng;
16    private Date temps;

```

```

17
18 public Position(Date temps, double longitude, double latitude, double x,
19     double y, double sog, double cog, double tauxRotation, int cap) {
20     super();
21     this.longitude = longitude;
22     this.latitude = latitude;
23     this.sog = sog;
24     this.cog = cog;
25     this.cap = cap;
26     this.vitesseAng = tauxRotation;
27     this.temps = temps;
28     this.x = x;
29     this.y = y;
30 }
31
32 public double getLongitude() {
33     return longitude;
34 }
35
36 public double getLatitude() {
37     return latitude;
38 }
39
40 public double getX() {
41     return x;
42 }
43
44 public double getY() {
45     return y;
46 }
47
48 public double getSog() {
49     return sog;
50 }
51
52 public double getCog() {
53     return cog;
54 }
55
56 public double getCap() {
57     return cap;
58 }
59
60 public double getVitesseAng() {
61     return vitesseAng;
62 }
63
64 public Date getTemps() {
65     return temps;
66 }
67
68 public String toString() {
69     String chaine;
70     chaine = "Date : " + temps + "\nLongitude : " + longitude
71         + ", Latitude : " + latitude + ", X : " + x + ", Y : " + y;
72     return chaine;
73 }
74 }
```

```

1 import java.io.FileNotFoundException;
2 import java.io.IOException;
3
4 /* Classe utilis? par le groupe "statistiques", qui cr?
5 le tableau des positions les plus probables que va emprunter le bateau*/
6
7 /* Ne pas oublier de rajouter la commande "-Xss20m" dans VM arguments (dans Run configurations,
8 arguments),
9 * sinon vous aurez une StackOverflowException.
10 *
11 * Mode d'emploi :
12 * - Redefinissez avant tout les constantes CHEMIN_DONNES et CHEMIN_ARBRES.
```

```

12  /*
13   * - La fonction generer_arbre doit être appelée au lancement du programme pour générer l'arbre.
14   *
15   * - La fonction trajectoireLaPlusProbable calcule la trajectoire la plus probable sous la forme de
16   * int[][] à partir de la
17   * trajectoire passée du bateau donnée sous la forme de int[][] ( coordonnées des cases) ou
18   * double [][] ( coordonnées métriques).
19   * La longueur maximale de la trajectoire calculée peut éventuellement être spécifiée , sachant
20   * que quoi qu'il en soit ,
21   * la somme des longueurs de la trajectoire passée et de la trajectoire calculée ne pourra
22   * excéder HAUTEUR_MAX.
23   *
24   * - La fonction probabilitePresence calcule la probabilité de présence du bateau dans chaque case
25   * à chaque instant ultérieur .
26   * Le résultat est donné sous la forme de double[][] où le double de coordonnées (t , i , j ) ,
27   * compris entre 0 et 1 ,
28   * est la probabilité que le bateau se trouve dans la case de coordonnées (i , j ) à l'instant t
29   * ultérieur .
30   * La longueur maximale de la trajectoire calculée (i.e. le t maximal) peut également être
31   * spécifié , mais on aura la
32   * même limite qu'avec la fonction trajectoireLaPlusProbable à propos de la somme des longueurs
33   * des trajectoires .
34   * Le boolean normaliser permet de normaliser ou non la table renvoyée , c'est-à-dire que s'il
35   * vaut true , toutes les
36   * probabilités de présence à un instant donné seront multipliées par la même constante de
37   * sorte que la probabilité
38   * maximale à un instant donné soit ramenée à 1 .
39 */
40 public class PredicTraj {
41
42     /** Incrémentation de temps entre deux pointages de position */
43     public static final int INCREMENT_TEMPS = 30;
44     /** Hauteur maximale de l'arbre */
45     public static final int HAUTEUR_MAX = 20;
46
47     /** Dossier dans lequel sont stockées les données */
48     public static final String CHEMIN_DONNEES = "D:\\\\Documents\\\\MIG\\\\Base de données\\\\Sys\\\\Données";
49     /** Dossier dans lequel sont stockés les arbres calculés */
50     public static final String CHEMIN_ARBRES = "C:\\\\Documents and Settings\\\\Administrateur\\\\Bureau\\\\
51 paulmain5\\\\src\\\\data\\\\Arbres";
52
53     /** Variable contenant l'arbre statistique utilisé pour la prédiction de trajectoire */
54     public static ArbreProba arbre;
55
56     /** Fonction générant l'arbre statistique . Elle doit être appelée au lancement du programme . */
57     public static void generer_arbre() {
58         try {
59             System.out.println("Lecture de l'arbre depuis le fichier... ");
60             arbre = ArbreProba.lireDepuisFichier(CHEMIN_ARBRES + "\\" + INCREMENT_TEMPS + "_" +
61                 HAUTEUR_MAX + "_"
62                 + Etat.N1 + "_" + Etat.N2 + ".obj");
63         } catch (FileNotFoundException e) {
64             System.out.println("L'arbre n'a encore jamais été calculé. Calcul de l'arbre en cours... ");
65
66             ArbreProba.nbNoeuds = 0;
67             arbre = new ArbreProba();
68             arbre.ajouterDossierSys(CHEMIN_DONNEES, 1000 * INCREMENT_TEMPS, HAUTEUR_MAX);
69             System.out.println("Calcul de l'arbre terminée. Enregistrement de l'arbre... ");
70             arbre.enregistrerDansFichier(CHEMIN_ARBRES + "\\" + INCREMENT_TEMPS + "_" + HAUTEUR_MAX + "_"
71                 + Etat.N1
72                 + "_" + Etat.N2 + ".obj");
73             System.out.println("Arbre enregistré.");
74
75         } catch (IOException e) {
76             System.out.println("Erreur de lecture. L'arbre est recalculé... ");
77
78             ArbreProba.nbNoeuds = 0;
79             arbre = new ArbreProba();
80             arbre.ajouterDossierSys(CHEMIN_DONNEES, 1000 * INCREMENT_TEMPS, HAUTEUR_MAX);
81             System.out.println("Calcul de l'arbre terminée. Enregistrement de l'arbre... ");
82
83         }
84     }

```

```

68     arbre.enregistrerDansFichier(CHEMIN_ARBRES + "\\" + INCREMENT_TEMPS + "_" + HAUTEUR_MAX + " - "
69         " + Etat.N1
70         + " - " + Etat.N2 + ".obj");
71     System.out.println("Arbre enregistré.");
72 }
73 } finally {
74     System.out.println("L'arbre statistique est prêt à être utilisé.");
75 }
76 }

77 /* Fonctions renvoyant la trajectoire la plus probable */
78 public static int[][] trajectoireLaPlusProbable(int[][] trajectoirePassee, int
79     longueurTrajectoireCalculee) {
80     return arbre.sousArbre(new Chemin(trajectoirePassee)).cheminLePlusProbable(
81         longueurTrajectoireCalculee)
82         .toArray();
83 }
84

85 public static int[][] trajectoireLaPlusProbable(int[][] trajectoirePassee) {
86     return arbre.sousArbre(new Chemin(trajectoirePassee)).cheminLePlusProbable().toArray();
87 }

88 public static int[][] trajectoireLaPlusProbable(double[][] trajectoirePassee, int
89     longueurTrajectoireCalculee) {
90     return arbre.sousArbre(new Chemin(trajectoirePassee)).cheminLePlusProbable(
91         longueurTrajet
92         .to
93         .probabilités de présence */
94 public static double[][][] probabilitePresence(int[][] trajectoirePassee, boolean normaliser,
95     int longueurTrajet
96         .probabilités de présence */
97 public static double[][][] probabilitePresence(int[][] trajectoirePassee, boolean normaliser,
98     int longueurTrajet
99         .probabilités de présence */
100    .probabilités de présence */
101    .probabilités de présence */
102    .probabilités de présence */
103    .probabilités de présence */
104    .probabilités de présence */
105    .probabilités de présence */
106    .probabilités de présence */
107    .probabilités de présence */
108    .probabilités de présence */
109    .probabilités de présence */
110    .probabilités de présence */
111    .probabilités de présence */
112    .probabilités de présence */
113    .probabilités de présence */
114    .probabilités de présence */
115    .probabilités de présence */
116    .probabilités de présence */
117    .probabilités de présence */
118    .probabilités de présence */
119    .probabilités de présence */
120    .probabilités de présence */
121    .probabilités de présence */
122    .probabilités de présence */
123    .probabilités de présence */
124 }
```

```

1 import processing.core.*;
2
3 // Echelle
4
5 public class Scale {
```

```

6 public static PApplet parent ;
7
8 // Echelle
9 public static void Scale() {
10    if (parent == null) {
11       parent = interfacegraphique.truc ;
12    }
13    parent.stroke(0);
14    parent.fill(0);
15    // en abscisse
16    parent.line((float) 50, (float) interfacegraphique.SIZE_Y - 50, 100,
17                (float) interfacegraphique.SIZE_Y - 50);
18    parent.triangle((float) 100, (float) interfacegraphique.SIZE_Y - 45,
19                    (float) 100, (float) interfacegraphique.SIZE_Y - 55,
20                    (float) 105, (float) interfacegraphique.SIZE_Y - 50);
21    parent.textAlign(PConstants.CENTER);
22    parent textSize(10);
23    parent.text("Est", (float) 115, (float) interfacegraphique.SIZE_Y - 50);
24    parent.text("X mètres", (float) 75,
25                (float) interfacegraphique.SIZE_Y - 35);
26    // en ordonnée
27
28    parent.line((float) 50, (float) interfacegraphique.SIZE_Y - 50,
29                (float) 50, (float) interfacegraphique.SIZE_Y - 100);
30    parent.triangle((float) 45, (float) interfacegraphique.SIZE_Y - 100,
31                    (float) 55, (float) interfacegraphique.SIZE_Y - 100,
32                    (float) 50, (float) interfacegraphique.SIZE_Y - 105);
33    parent.text("Nord", (float) 50, (float) interfacegraphique.SIZE_Y - 110);
34
35    parent.noFill();
36    parent.noStroke();
37
38 }
39
40 }
```

```

1 import processing.core.*;
2
3 // Cr ation et modification des senseurs
4
5 public class Senseur {
6
7    // attributs
8    double xpos;
9    double ypos;
10   double portee;
11   double ouverture_angulaire;
12   double sens;
13   int sensor_type;
14   PApplet parent;
15
16   // constructeur
17   public Senseur(PApplet p, double a, double b, double c, double d, double e,
18                  int f) {
19      xpos = a;
20      ypos = b;
21      portee = c;
22      ouverture_angulaire = d;
23      sens = e;
24      sensor_type = f;
25      parent = p;
26
27   }
28
29   // m ethodes
30   public void dessinerSenseur() {
31
32      switch (sensor_type) {
33
34         case 1:
```

```

36 // balayage de la caméra.
37 for (int k = 0; k < 20; k++) {
38     parent.fill(200, 200, 200, 100);
39     if (parent.millis() % 1000 < (k + 1) * 50
40         && parent.millis() % 1000 >= k * 50 && k < 10) {
41         parent.arc(
42             (float) (xpos),
43             (float) (ypos),
44             (float) (portee * 2),
45             (float) (portee * 2),
46             (float) (sens - (ouverture_angulaire) / 2 + ouverture_angulaire
47                     * k / 10),
48             (float) (sens - (ouverture_angulaire) / 2 + ouverture_angulaire
49                     * (k + 1) / 10));
50     } else if (parent.millis() % 1000 < (k + 1) * 50
51         && parent.millis() % 1000 >= k * 50 && k >= 10) {
52         parent.arc(
53             (float) (xpos),
54             (float) (ypos),
55             (float) (portee * 2),
56             (float) (portee * 2),
57             (float) (sens + (ouverture_angulaire) / 2 - ouverture_angulaire
58                     * (k - 9) / 10),
59             (float) (sens + (ouverture_angulaire) / 2 - ouverture_angulaire
60                     * (k - 10) / 10));
61     }
62 }
63
64 parent.fill(40, 70, 120, 200);
65 break;
66
67 case 2:
68
69 // balayage de la caméra.
70 for (int k = 0; k < 20; k++) {
71     parent.fill(200, 200, 200, 100);
72     if (parent.millis() % 1000 < (k + 1) * 50
73         && parent.millis() % 1000 >= k * 50 && k < 10) {
74         parent.arc(
75             (float) (xpos),
76             (float) (ypos),
77             (float) (portee * 2),
78             (float) (portee * 2),
79             (float) (sens - (ouverture_angulaire) / 2 + ouverture_angulaire
80                     * k / 10),
81             (float) (sens - (ouverture_angulaire) / 2 + ouverture_angulaire
82                     * (k + 1) / 10));
83     } else if (parent.millis() % 1000 < (k + 1) * 50
84         && parent.millis() % 1000 >= k * 50 && k >= 10) {
85         parent.arc(
86             (float) (xpos),
87             (float) (ypos),
88             (float) (portee * 2),
89             (float) (portee * 2),
90             (float) (sens + (ouverture_angulaire) / 2 - ouverture_angulaire
91                     * (k - 9) / 10),
92             (float) (sens + (ouverture_angulaire) / 2 - ouverture_angulaire
93                     * (k - 10) / 10));
94     }
95 }
96
97 parent.fill(200, 55, 12, 200);
98 break;
99
100 case 3:
101
102 for (int k = 0; k < 20; k++) {
103     if (parent.millis() % 1000 < (k + 1) * 50
104         && parent.millis() % 1000 >= k * 50) {
105         parent.fill(200, 200, 200, 100);
106     }

```

```

107     parent.arc(
108         (float) (xpos),
109         (float) (ypos),
110         (float) (portee * 2),
111         (float) (portee * 2),
112         (float) (sens - (ouverture_angulaire) / 2 + ouverture_angulaire
113             * k / 10),
114         (float) (sens - (ouverture_angulaire) / 2 + ouverture_angulaire
115             * (k + 1) / 10));
116     }
117 }
118 parent.fill(124, 124, 124, 200);
119 break;
120
121 case 4:
122     parent.fill(0, 120, 120);
123     break;
124
125 case 5:
126     parent.fill(120, 0, 200);
127     break;
128 }
129
130 parent.arc((float) (xpos), (float) (ypos), (float) (5), (float) (5),
131     (float) (0), 2 * PConstants.PI);
132 parent.noStroke();
133 parent.fill(0, 0, 0, 40);
134 parent.arc((float) (xpos), (float) (ypos), (float) (portee * 2),
135     (float) (portee * 2),
136     (float) (sens - (ouverture_angulaire) / 2),
137     (float) (sens + (ouverture_angulaire) / 2));
138 parent.stroke(200);
139
140 // effacer les senseurs.
141 if (interfacegraphique.state == 6
142     && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
143         (float) interfacegraphique.xpos1,
144         (float) interfacegraphique.ypos1) < 25) {
145     interfacegraphique.xpos1 = -1000;
146     interfacegraphique.ypos1 = -1000;
147 } else if (interfacegraphique.state == 6
148     && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
149         (float) interfacegraphique.xpos2,
150         (float) interfacegraphique.ypos2) < 25) {
151     interfacegraphique.xpos2 = -1000;
152     interfacegraphique.ypos2 = -1000;
153 } else if (interfacegraphique.state == 6
154     && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
155         (float) interfacegraphique.xpos3,
156         (float) interfacegraphique.ypos3) < 25) {
157     interfacegraphique.xpos3 = -1000;
158     interfacegraphique.ypos3 = -1000;
159 }
160
161 // modifier les senseurs.
162
163 if (interfacegraphique.state == 7
164     && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
165         (float) interfacegraphique.xpos1,
166         (float) interfacegraphique.ypos1) < 25) {
167
168     interfacegraphique.xpos1 = parent.mouseX;
169     interfacegraphique.ypos1 = parent.mouseY;
170 }
171 if (interfacegraphique.state == 8
172     && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
173         (float) interfacegraphique.xpos1,
174         (float) interfacegraphique.ypos1) < 25) {
175     interfacegraphique.sens1 = PApplet
176         .acos((float) (parent.mouseX - interfacegraphique.xpos1)
177             / PApplet.dist((float) interfacegraphique.xpos1,

```

```

178         (float) interfacegraphique.ypos1,
179         (float) parent.mouseX,
180         (float) parent.mouseY));
181     }
182     if (interfacegraphique.state == 11) {
183         interfacegraphique.portee1 = PApplet.dist((float) parent.mouseX,
184             (float) parent.mouseY, (float) interfacegraphique.xpos1,
185             (float) interfacegraphique.ypos1) * 2;
186     }
187
188     if (interfacegraphique.state == 7
189         && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
190             (float) interfacegraphique.xpos2,
191             (float) interfacegraphique.ypos2) < 25) {
192
193         interfacegraphique.xpos2 = parent.mouseX;
194         interfacegraphique.ypos2 = parent.mouseY;
195     }
196     if (interfacegraphique.state == 8
197         && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
198             (float) interfacegraphique.xpos2,
199             (float) interfacegraphique.ypos2) < 25) {
200
201         interfacegraphique.sens2 = PApplet
202             .acos((float) (parent.mouseX - interfacegraphique.xpos2)
203                 / PApplet.dist((float) interfacegraphique.xpos2,
204                     (float) interfacegraphique.ypos2,
205                     (float) parent.mouseX,
206                     (float) parent.mouseY));
207     }
208     if (interfacegraphique.state == 12) {
209         interfacegraphique.portee2 = 2 * PApplet.dist(
210             (float) parent.mouseX, (float) parent.mouseY,
211             (float) interfacegraphique.xpos2,
212             (float) interfacegraphique.ypos2);
213     }
214     if (interfacegraphique.state == 7
215         && PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
216             (float) interfacegraphique.xpos3,
217             (float) interfacegraphique.ypos3) < 25) {
218
219         interfacegraphique.xpos3 = parent.mouseX;
220         interfacegraphique.ypos3 = parent.mouseY;
221     }
222     if (interfacegraphique.state == 13) {
223         interfacegraphique.portee3 = 2 * PApplet.dist(
224             (float) parent.mouseX, (float) parent.mouseY,
225             (float) interfacegraphique.xpos1,
226             (float) interfacegraphique.ypos1);
227
228 // affichage des informations avec la souris
229     if (PApplet.dist((float) parent.mouseX, (float) parent.mouseY,
230         (float) xpos, (float) ypos) <= 10) {
231         parent.textSize(13);
232         parent.rectMode(PConstants.CENTER);
233         parent.stroke(50, 50, 50);
234         parent.fill(152, 152, 152);
235         parent.rect((float) interfacegraphique.SIZE_X * (float) 10 / 11,
236             (float) interfacegraphique.SIZE_Y * (float) 43 / 50,
237             (float) 140 / 889 * (float) interfacegraphique.SIZE_X,
238             (float) 4 / 25 * (float) interfacegraphique.SIZE_Y, 18, 18,
239             18, 18);
240         parent.fill(0, 0, 0, 255);
241         parent.TextAlign(PConstants.CENTER);
242         parent.text("Type de Senseur:" + " " + sensor_type,
243             (float) interfacegraphique.SIZE_X - (float) 80 / 889
244             * (float) interfacegraphique.SIZE_X,
245             (float) interfacegraphique.SIZE_Y - (float) 90 / 500
246             * (float) interfacegraphique.SIZE_Y);
247         parent.text("Position : (" + xpos + "," + ypos + ")",
248             (float) interfacegraphique.SIZE_X - (float) 80 / 889

```

```

249     * (float) interfacegraphique.SIZE_X,
250     (float) interfacegraphique.SIZE_Y - (float) 70 / 500
251     * (float) interfacegraphique.SIZE_Y);
252 parent.text(
253     "Portée:" + " "
254     + Truncate.TruncateDouble(portee / 27.51, 1) + " "
255     + "km", (float) interfacegraphique.SIZE_X
256     - (float) 80 / 889
257     * (float) interfacegraphique.SIZE_X,
258     (float) interfacegraphique.SIZE_Y
259     - (float) interfacegraphique.SIZE_Y / 10);
260 }
261 }
262
263 public static boolean sensorRange1(float x, float y) {
264     boolean presence;
265     presence = false;
266
267     if (Math.sqrt(Math.pow(interfacegraphique.xpos1 - x, 2)
268         + Math.pow(interfacegraphique.ypos1 - y, 2)) <= interfacegraphique.portee1
269     && y >= interfacegraphique.ypos1
270         + (x - interfacegraphique.xpos1)
271         * Math.tan(-(interfacegraphique.sens1
272             - interfacegraphique.angle1 / 2 + 2 * Math.PI / 3))
273     && y <= interfacegraphique.ypos1
274         + (x - interfacegraphique.xpos1)
275         * Math.tan(-(interfacegraphique.sens1
276             + interfacegraphique.angle1 / 2 + 2 * Math.PI / 3))) {
277
278         presence = true;
279     }
280     return presence;
281 }
282
283 public static boolean sensorRange2(float x, float y) {
284     boolean presence;
285     presence = false;
286
287     if (Math.sqrt(Math.pow(interfacegraphique.xpos2 - x, 2)
288         + Math.pow(interfacegraphique.ypos2 - y, 2)) <= interfacegraphique.portee2
289     && y >= interfacegraphique.ypos2
290         - (x - interfacegraphique.xpos2)
291         * Math.tan(-(interfacegraphique.sens2 - interfacegraphique.angle2 / 2))
292     && y <= interfacegraphique.ypos2
293         - (x - interfacegraphique.xpos2)
294         * Math.tan(-(interfacegraphique.sens2 + interfacegraphique.angle2 / 2))) {
295
296         presence = true;
297     }
298     return presence;
299 }
300
301 public static boolean sensorRange3(float x, float y) {
302     boolean presence;
303     presence = false;
304
305     if (Math.sqrt(Math.pow(interfacegraphique.xpos3 - x, 2)
306         + Math.pow(interfacegraphique.ypos3 - y, 2)) <= interfacegraphique.portee3) {
307
308         presence = true;
309     }
310     return presence;
311 }
312 }
313 }
```

```

1 import processing.core.*;
2 import java.text.SimpleDateFormat;
3
4 //Création de la variable "temps".
5
```

```

6 public class Timer {
7
8     public static PApplet parent;
9
10    public static void afficherDate(Flotte flotte) {
11        if (parent == null) {
12            parent = interfacegraphique.truc;
13        }
14        SimpleDateFormat sdf = new SimpleDateFormat ("yyyy-MM-dd HH:mm:ss");
15        int k = 0;
16
17        String dateformatee = sdf.format (GestionPort.flotte
18            .getDate (interfacegraphique.TIME));
19        parent.fill (0, 0, 0);
20        parent.textSize (12);
21        parent.text (dateformatee, interfacegraphique.SIZE_X / 9,
22            interfacegraphique.SIZE_Y / 10);
23        parent.noFill ();
24    }
25
26    public static void timer () {
27        if (parent == null) {
28            parent = interfacegraphique.truc;
29        }
30        // Pour modifier la rapidité de l'animation, on le fait ici, ET dans
31        // Pauseevents
32        interfacegraphique.TIME = (int) ((parent.millis () - interfacegraphique.START) / 100 -
33            interfacegraphique.PAUSE);
34        if (interfacegraphique.j == 1) {
35        }
36    }
37}

```

```

1 import processing.core.PApplet;
2 //Tracé de trajectoire à partir d'un tableau de positions.
3 public class TrajPrev {
4     public static PApplet parent;
5
6     // à utiliser dans la classe dessiner dans la boucle qui parcourt tous les
7     // bateaux.
8     // le tab est donné par une des fonctions de charles/romain
9
10    public static void dessiner (int [][] tab) {
11
12        if (interfacegraphique.state == 15) {
13
14            parent.stroke (255, 100, 0);
15            int k = 1;
16            while (k - 1 < tab.length) {
17
18                float x1 = ConversionPixel.conversionPixelX ((float) Etat
19                    .toMetrique (tab [k]) [0]);
20                float y1 = ConversionPixel.conversionPixelY ((float) Etat
21                    .toMetrique (tab [k]) [1]);
22
23                float x2 = ConversionPixel.conversionPixelX ((float) Etat
24                    .toMetrique (tab [k + 1]) [0]);
25                float y2 = ConversionPixel.conversionPixelY ((float) Etat
26                    .toMetrique (tab [k + 1]) [1]);
27
28                parent.line (x1, y1, x2, y2);
29
30                k++;
31            }
32
33            parent.noStroke ();
34
35        }
36    }

```

```

1 import processing.core.*;
2
3 // Affichage de la trajectoire passée d'un bateau.
4 public class TrajRec {
5     public static PApplet parent;
6
7     public static void trajRec(int i, int t) {
8         if (parent == null) {
9             parent = interfacegraphique.truc;
10        }
11
12        if (GestionPort.flotte.getBateau(i).getPosNorm(t) == null
13            || t < interfacegraphique.TIME - 30) {
14        } else {
15            if (GestionPort.flotte.getBateau(i).getPosNorm(t - 2) == null) {
16            } else {
17
18                float xj = (float) ConversionPixel
19                    .conversionPixelX((float) GestionPort.flotte
20                        .getBateau(i).getPosNorm(t).getX());
21                float yj = (float) ConversionPixel
22                    .conversionPixelY((float) GestionPort.flotte
23                        .getBateau(i).getPosNorm(t).getY());
24                float x2j = (float) ConversionPixel
25                    .conversionPixelX((float) GestionPort.flotte
26                        .getBateau(i).getPosNorm(t - 2).getX());
27                float y2j = (float) ConversionPixel
28                    .conversionPixelY((float) GestionPort.flotte
29                        .getBateau(i).getPosNorm(t - 2).getY());
30
31                if (parent.dist(xj, yj, x2j, y2j) > 100) {
32
33                } else {
34                    // détection sensorielle
35                    if (Senseur.sensorRange1(xj, yj) == true
36                        || Senseur.sensorRange2(xj, yj) == true
37                        || Senseur.sensorRange3(xj, yj) == true) {
38                        parent.fill(parent.color(0, 255, 0, 200));
39                        parent.stroke(parent.color(0, 255, 0, 200));
40                    } else {
41
42                        parent.stroke(parent.color(255, 0, 0, 200));
43                    }
44                    parent.rectMode(PConstants.CENTER);
45                    parent.line((float) xj, (float) yj, (float) x2j,
46                                (float) y2j);
47                    parent.noStroke();
48
49                    trajRec(i, t - 2);
50
51                }
52            }
53        }
54    }
55}

```

```

1 import processing.core.*;
2 // Tracé des carrés colorés pour le groupe "statistiques"
3 public class TrajStat {
4     public static PApplet parent;
5
6     // à utiliser dans la classe dessiner dans la boucle qui parcourt tous les
7     // bateaux.
8     // le tab est donné par une des fonctions de charles
9
10    public static void dessiner(int[][] tab) {
11        if (parent == null) {
12            parent = interfacegraphique.truc;

```

```

13 }
14 if (interfacegraphique.state == 15) {
15
16     parent.stroke(255, 100, 0);
17     int k = 0;
18     while (k < tab.length - 2) {
19
20         float x1 = ConversionPixel.conversionPixelX((float) Etat
21             .toMetrique(tab[k])[0]);
22         float y1 = ConversionPixel.conversionPixelY((float) Etat
23             .toMetrique(tab[k])[1]);
24
25         float x2 = ConversionPixel.conversionPixelX((float) Etat
26             .toMetrique(tab[k + 1])[0]);
27         float y2 = ConversionPixel.conversionPixelY((float) Etat
28             .toMetrique(tab[k + 1])[1]);
29
30         parent.line(x1, y1, x2, y2);
31
32         k++;
33     }
34
35     parent.noStroke();
36
37 }
38
39 }
40

```

```

1 import java.math.RoundingMode;
2 import java.text.DecimalFormat;
3
4 // Troncature d'un type float
5 public class Truncate {
6
7     public static String TruncateFloat(float f, int i) {
8         String t = "";
9         for (int j = 0; j < i; j++) {
10            t = t + "#";
11        }
12        DecimalFormat df = new DecimalFormat("##." + t);
13        df.setRoundingMode(RoundingMode.DOWN);
14        return df.format(f);
15    }
16
17     public static String TruncateDouble(double d, int j) {
18         String t = "";
19         for (int k = 0; k < j; k++) {
20            t = t + "#";
21        }
22        DecimalFormat df = new DecimalFormat("##." + t);
23        df.setRoundingMode(RoundingMode.DOWN);
24        return df.format(d);
25    }
26

```

```

1 import java.util.*;
2
3 // Délimitation des zones du port.
4 public class Zone {
5     public HashSet<Etat> ensembleCase = new HashSet<Etat>();
6
7     // Méthode add: Ajouter un élément à une zone
8     public void add(Etat e) {
9         ensembleCase.add(e);
10    }
11
12    // Constructeur
13    public Zone(HashSet<Etat> e) {
14        ensembleCase = e;

```

```

15 }
16
17 // Fonction : renvoie un booléen testant la présence d'un point dans un
18 // triangle .
19 // Test la présence du point "point" (en coordUTM) dans le triangle m1 n1 p1
20 // (en coordMercator);
21 public static boolean testInclusion(double[] point, double[] m1,
22         double[] n1, double[] p1) {
23     double[] m2 = ConversionMercator.conversion(m1);
24     double[] n2 = ConversionMercator.conversion(n1);
25     double[] p2 = ConversionMercator.conversion(p1);
26
27     // Coordonnées y = ax + b des droites MP NP et NP;
28     double aMN = (m2[1] - n2[1]) / ((m2[0] - n2[0]));
29     double aMP = (m2[1] - p2[1]) / ((m2[0] - p2[0]));
30     double aNP = (n2[1] - p2[1]) / ((n2[0] - p2[0]));
31     double bMN = (m2[0] * n2[1] - m2[1] * n2[0]) / (m2[0] - n2[0]);
32     double bMP = (m2[0] * p2[1] - m2[1] * p2[0]) / (m2[0] - p2[0]);
33     double bNP = (p2[0] * n2[1] - p2[1] * n2[0]) / (p2[0] - n2[0]);
34
35     double x = point[0];
36     double y = point[1];
37
38     boolean b1, b2, b3;
39
40     if (p2[1] > (aMN * p2[0] + bMN)) {
41         b1 = true;
42     } else {
43         b1 = false;
44     }
45
46     if (m2[1] > (aNp * m2[0] + bNP)) {
47         b2 = true;
48     } else {
49         b2 = false;
50     }
51
52     if (n2[1] > (aMP * n2[0] + bMP)) {
53         b3 = true;
54     } else {
55         b3 = false;
56     }
57
58     if (((b1) && (y >= aMN * x + bMN)) || ((!b1) && (y <= aMN * x + bMN)))
59         && (((b2) && (y >= aNP * x + bNP)) || ((!b2) && (y <= aNP * x
60             + bNP)))
61         && (((b3) && (y >= aMP * x + bMP)) || ((!b3) && (y <= aMP * x
62             + bMP))) {
63         return true;
64     } else
65         return false;
66 }
67
68 // Constructeur d'une zone à partir de 3 points quelconques différents
69 // (Triangle non plat);
70 public Zone(double[] m1, double[] n1, double[] p1) {
71     Etat m2 = new Etat(m1[0], m1[1]);
72     Etat n2 = new Etat(n1[0], n1[1]);
73     Etat p2 = new Etat(p1[0], p1[1]);
74
75     int i1 = Math.min(Math.min(m2.x, n2.x), p2.x);
76     int i2 = Math.max(Math.max(m2.x, n2.x), p2.x);
77
78     int j1 = Math.min(Math.min(m2.y, n2.y), p2.y);
79     int j2 = Math.max(Math.max(m2.y, n2.y), p2.y);
80
81     for (int i = i1; i <= i2; i++) {
82         for (int j = j1; j <= j2; j++) {
83             Etat e = new Etat(0, 0);
84             e.x = i;
85             e.y = j;

```

```

86     double[] f = e.toMetrique();
87     if (testInclusion(f, m1, n1, p1)) {
88         ensembleCase.add(e);
89     }
90 }
91 }
92 }
93
94 // Afficher le contenu d'une zone sous la forme de doublets [i,j];
95 public void afficherLeContenu() {
96     Iterator<Etat> its = ensembleCase.iterator();
97     System.out.println("Le contenu de la zone est :");
98     while (its.hasNext()) {
99         System.out.println(its.next());
100    }
101 }
102
103 // Test d'inclusion d'un état dans une zone
104 public boolean contain(Etat e) {
105     boolean b = false;
106     Iterator<Etat> its = ensembleCase.iterator();
107     while (its.hasNext()) {
108         if (e.equals(its.next())) {
109             b = true;
110        }
111    }
112    return b;
113 }
114
115 // Union de deux zones : Attention: modifie l'EnsembleCase de la zone
116 // courante
117 public void union(Zone z) {
118     ensembleCase.addAll(z.ensembleCase);
119 }
120
121 // ..... Définition des points Maritimes : BOUEES .....
122
123 // MOUILLAGE_NORD:
124 static double[] MFN0 = { 43.429609, 4.921473 };
125 static double[] MFN1 = { 43.417800, 4.908033 };
126 static double[] MFN2 = { 43.402317, 4.906217 };
127 static double[] MFN3 = { 43.397317, 4.914233 };
128 static double[] MFN4 = { 43.391483, 4.940400 };
129 static double[] MFN5 = { 43.383233, 4.958050 };
130 static double[] MFN6 = { 43.396233, 4.981950 };
131 static double[] MFN7 = { 43.425495, 4.968594 };
132 static double[] MFN8 = { 43.427677, 4.941815 };
133
134 // MOUILLAGE_EST:
135 static double[] MFE1 = { 43.375650, 5.001050 };
136 static double[] MFE2 = { 43.375667, 4.978167 };
137 static double[] MFE3 = { 43.363150, 4.972900 };
138 static double[] MFE4 = { 43.325767, 4.991483 };
139 static double[] MFE5 = { 43.325683, 5.038117 };
140
141 // MOUILLAGE_OUEST:
142 static double[] MFO1 = { 43.390267, 4.933900 };
143 static double[] MFO2 = { 43.325683, 4.922850 };
144 static double[] MFO3 = { 43.325767, 4.970033 };
145 static double[] MFO4 = { 43.377950, 4.955033 };
146 static double[] MFO5 = { 43.386650, 4.942233 };
147
148 // MOUILLAGE_INTERDIT
149 static double[] MIO1 = { 43.388333, 4.991667 };
150 static double[] MIO2 = { 43.388333, 4.973333 };
151 static double[] MIO3 = { 43.375717, 4.973333 };
152 static double[] MIO4 = { 43.375800, 5.000183 };
153
154 // CHENAL_SUD
155 // Bordure-Est (Rail Montant)
156 static double[] PCO1 = { 43.376167, 4.966333 };

```

```

157 static double[] PCO2 = { 43.199000, 5.054333 };
158 // Séparation
159 static double[] PCO3 = { 43.377000, 4.960833 };
160 static double[] PCO4 = { 43.199000, 5.030167 };
161 // Bordure-Ouest (Rail Descendant)
162 static double[] PCO5 = { 43.377950, 4.955033 };
163 static double[] PCO6 = { 43.199000, 5.006400 };
164
165 // ACCES_LAVERA
166 static double[] PCOL1 = { 43.363150, 4.972900 };
167 static double[] PCOL2 = { 43.393883, 4.985767 };
168 static double[] PCOL3 = { 43.396367, 4.984917 };
169 static double[] PCOL4 = { 43.390392, 4.979317 };
170 static double[] PCOL5 = { 43.384617, 4.962167 };
171
172 // ACCES_GOLFE_FOS
173 static double[] PCOF1 = { 43.384617, 4.962167 };
174 static double[] PCOF2 = { 43.383233, 4.958050 };
175 static double[] PCOF3 = { 43.391483, 4.940400 };
176 static double[] PCOF4 = { 43.397317, 4.914233 };
177 static double[] PCOF5 = { 43.402317, 4.906217 };
178 static double[] PCOF6 = { 43.397367, 4.903083 };
179 static double[] PCOF7 = { 43.390983, 4.932233 };
180 static double[] PCOF8 = { 43.386650, 4.942233 };
181 static double[] PCOF9 = { 43.377950, 4.955033 };
182 static double[] PCOF10 = { 43.376167, 4.966333 };
183
184 // ZONE_SECURISE1
185 static double[] PSEC1 = { 43.380700, 4.903083 };
186 static double[] PSEC2 = { 43.394132, 4.852636 };
187 static double[] PSEC3 = { 43.393258, 4.827832 };
188 static double[] PSEC4 = { 43.428798, 4.823025 };
189 static double[] PSEC5 = { 43.456592, 4.856155 };
190 static double[] PSEC6 = { 43.429421, 4.909714 };
191
192 // ZONE_SECURISE2
193 static double[] MI1 = { 43.398486, 5.001198 };
194 static double[] MI2 = { 43.402134, 4.996906 };
195 static double[] MI3 = { 43.394713, 4.986907 };
196 static double[] MI4 = { 43.391781, 4.990340 };
197
198 // ..... FIN de Définition des points Maritimes : BOUEES.....
199
200 // _____ Définition des zones particulières du
201 // port à l'aide des états
202
203 public static Zone MOUILLAGE_OUEST() {
204     Zone T001 = new Zone(MFO2, MFO3, MFO4);
205     Zone T002 = new Zone(MFO2, MFO4, MFO5);
206     Zone T003 = new Zone(MFO2, MFO5, MFO1);
207     T001.union(T002);
208     T001.union(T003);
209     return (T001);
210 }
211
212 public static Zone MOUILLAGE_EST() {
213     Zone T001 = new Zone(MFE5, MFE4, MFE3);
214     Zone T002 = new Zone(MFE5, MFE3, MFE2);
215     Zone T003 = new Zone(MFE5, MFE2, MFE1);
216     T001.union(T002);
217     T001.union(T003);
218     return (T001);
219 }
220
221 public static Zone MOUILLAGE_NORD() {
222     Zone T001 = new Zone(MFN8, MFN7, MFN6);
223     Zone T002 = new Zone(MFN8, MFN6, MFN5);
224     Zone T003 = new Zone(MFN8, MFN5, MFN4);
225     Zone T004 = new Zone(MFN8, MFN4, MFN3);
226     Zone T005 = new Zone(MFN8, MFN3, MFN2);
227     Zone T006 = new Zone(MFN8, MFN2, MFN1);

```

```

228 Zone T007 = new Zone(MFN8, MFN1, MFN0);
229 T001.union(T002);
230 T001.union(T003);
231 T001.union(T004);
232 T001.union(T005);
233 T001.union(T006);
234 T001.union(T007);
235 return (T001);
236 }
237
238 public static Zone MOUILLAGE_INTERDIT() {
239     Zone T001 = new Zone(MIO3, MIO2, MIO1);
240     Zone T002 = new Zone(MIO3, MIO1, MIO4);
241     T001.union(T002);
242     return (T001);
243 }
244
245 public static Zone CHENAL_SUD_MONTANT() {
246     Zone T001 = new Zone(PCO1, PCO2, PCO4);
247     Zone T002 = new Zone(PCO1, PCO3, PCO4);
248     T001.union(T002);
249     return (T001);
250 }
251
252 public static Zone CHENAL_SUD_DESCENDANT() {
253     Zone T001 = new Zone(PCO5, PCO6, PCO4);
254     Zone T002 = new Zone(PCO5, PCO3, PCO4);
255     T001.union(T002);
256     return (T001);
257 }
258
259 public static Zone ACCES_LAVERA() {
260     Zone T001 = new Zone(PCOL1, PCOL5, PCOL2);
261     Zone T002 = new Zone(PCOL2, PCOL4, PCOL3);
262     Zone T003 = new Zone(PCOL5, PCOL4, PCOL1);
263     T001.union(T002);
264     T001.union(T003);
265     return (T001);
266 }
267
268 public static Zone ZONE_SECURISE1() {
269     Zone T001 = new Zone(PSEC6, PSEC1, PSEC2);
270     Zone T002 = new Zone(PSEC6, PSEC2, PSEC3);
271     Zone T003 = new Zone(PSEC6, PSEC3, PSEC4);
272     Zone T004 = new Zone(PSEC6, PSEC4, PSEC5);
273     T001.union(T002);
274     T001.union(T003);
275     T001.union(T004);
276     return (T001);
277 }
278
279 public static Zone ZONE_SECURISE2() {
280     Zone T001 = new Zone(MI1, MI2, MI3);
281     Zone T002 = new Zone(MI1, MI3, MI4);
282     T001.union(T002);
283     return (T001);
284 }
285
286 public static Zone ACCES_GOLFE_FOS() {
287     Zone T001 = new Zone(PCOF6, PCOF5, PCOF4);
288     Zone T002 = new Zone(PCOF6, PCOF4, PCOF7);
289
290     Zone T003 = new Zone(PCOF4, PCOF7, PCOF3);
291     Zone T004 = new Zone(PCOF7, PCOF3, PCOF8);
292     Zone T005 = new Zone(PCOF2, PCOF8, PCOF3);
293     Zone T006 = new Zone(PCOF8, PCOF9, PCOF2);
294     Zone T007 = new Zone(PCOF2, PCOF9, PCOF10);
295     Zone T008 = new Zone(PCOF2, PCOF1, PCOF10);
296     T001.union(T002);
297     T001.union(T003);
298     T001.union(T004);

```

```

299     T001.union(T005);
300     T001.union(T006);
301     T001.union(T007);
302     T001.union(T008);
303     return (T001);
304 }
305
306 // ----- FIN DE Définition des zones particulières du
307 // port -----
308
309 // ----- Test de présence d'un point UTM dans une zone
310 // ( sans tenir compte des états ) -----
311
312 public static boolean MOUILLAGE_OUEST(double[] point) {
313     boolean b1 = testInclusion(point, MFO2, MFO3, MFO4);
314     boolean b2 = testInclusion(point, MFO2, MFO4, MFO5);
315     boolean b3 = testInclusion(point, MFO2, MFO5, MFO1);
316     boolean b = b1 || b2 || b3;
317     return (b);
318 }
319
320 public static boolean MOUILLAGE_EST(double[] point) {
321     boolean b1 = testInclusion(point, MFE5, MFE4, MFE3);
322     boolean b2 = testInclusion(point, MFE5, MFE3, MFE2);
323     boolean b3 = testInclusion(point, MFE5, MFE2, MFE1);
324     boolean b = b1 || b2 || b3;
325     return (b);
326 }
327
328 public static boolean MOUILLAGE_NORD(double[] point) {
329     boolean b1 = testInclusion(point, MFN8, MFN7, MFN6);
330     boolean b2 = testInclusion(point, MFN8, MFN6, MFN5);
331     boolean b3 = testInclusion(point, MFN8, MFN5, MFN4);
332     boolean b4 = testInclusion(point, MFN8, MFN4, MFN3);
333     boolean b5 = testInclusion(point, MFN8, MFN3, MFN2);
334     boolean b6 = testInclusion(point, MFN8, MFN2, MFN1);
335     boolean b7 = testInclusion(point, MFN8, MFN1, MFN0);
336     boolean b = b1 || b2 || b3 || b4 || b5 || b6 || b7;
337     return (b);
338 }
339
340 public static boolean MOUILLAGE_INTERDIT(double[] point) {
341     boolean b1 = testInclusion(point, MIO3, MIO2, MIO1);
342     boolean b2 = testInclusion(point, MIO3, MIO1, MIO4);
343     boolean b = b1 || b2;
344     return (b);
345 }
346
347 public static boolean CHENAL_SUD_MONTANT(double[] point) {
348     boolean b1 = testInclusion(point, PCO1, PCO2, PCO4);
349     boolean b2 = testInclusion(point, PCO1, PCO3, PCO4);
350     boolean b = b1 || b2;
351     return (b);
352 }
353
354 public static boolean CHENAL_SUD_DESCENDANT(double[] point) {
355     boolean b1 = testInclusion(point, PCO5, PCO6, PCO4);
356     boolean b2 = testInclusion(point, PCO5, PCO3, PCO4);
357     boolean b = b1 || b2;
358     return (b);
359 }
360
361 public static boolean ACCES_LAVERA(double[] point) {
362     boolean b1 = testInclusion(point, PCOL1, PCOL5, PCOL2);
363     boolean b2 = testInclusion(point, PCOL2, PCOL4, PCOL3);
364     boolean b3 = testInclusion(point, PCOL5, PCOL4, PCOL1);
365     boolean b = b1 || b2 || b3;
366     return (b);
367 }
368
369 public static boolean ZONE_SECURISE1(double[] point) {

```

```

370     boolean b1 = testInclusion(point, PSEC6, PSEC1, PSEC2);
371     boolean b2 = testInclusion(point, PSEC6, PSEC2, PSEC3);
372     boolean b3 = testInclusion(point, PSEC6, PSEC3, PSEC4);
373     boolean b4 = testInclusion(point, PSEC6, PSEC4, PSEC5);
374     boolean b = b1 || b2 || b3 || b4;
375     return (b);
376 }
377
378 public static boolean ZONE_SECURISE2(double[] point) {
379     boolean b1 = testInclusion(point, MI1, MI2, MI3);
380     boolean b2 = testInclusion(point, MI1, MI3, MI4);
381     boolean b = b1 || b2;
382     return (b);
383 }
384
385 public static boolean ACCES_GOLFE_FOS(double[] point) {
386     boolean b1 = testInclusion(point, PCOF6, PCOF5, PCOF4);
387     boolean b2 = testInclusion(point, PCOF6, PCOF4, PCOF7);
388     boolean b3 = testInclusion(point, PCOF4, PCOF7, PCOF3);
389     boolean b4 = testInclusion(point, PCOF7, PCOF3, PCOF8);
390     boolean b5 = testInclusion(point, PCOF2, PCOF8, PCOF3);
391     boolean b6 = testInclusion(point, PCOF8, PCOF9, PCOF2);
392     boolean b7 = testInclusion(point, PCOF2, PCOF9, PCOF10);
393     boolean b8 = testInclusion(point, PCOF2, PCOF1, PCOF10);
394     boolean b = b1 || b2 || b3 || b4 || b5 || b6 || b7 || b8;
395     return (b);
396 }
397
398 // Savoir dans quelle(s) Zone se trouve le point "point" (coordUTM)
399 public static void dansQuelleZone(double[] point) {
400
401     int e = 0;
402
403     if (MOUILLAGE_OUEST(point)) {
404         System.out.println("Bateau dans la zone: MOUILLAGE OUEST");
405         e = 1;
406     }
407     if (MOUILLAGE_EST(point)) {
408         System.out.println("Bateau dans la zone: MOUILLAGE EST");
409         e = 1;
410     }
411     if (MOUILLAGE_NORD(point)) {
412         System.out.println("Bateau dans la zone: MOUILLAGE NORD");
413         e = 1;
414     }
415     if (MOUILLAGE_INTERDIT(point)) {
416         System.out.println("Bateau dans la zone: MOUILLAGE INTERDIT");
417         e = 1;
418     }
419     if (CHENAL_SUD_MONTANT(point)) {
420         System.out.println("Bateau dans la zone: CHENAL SUD, SENS MONTANT");
421         e = 1;
422     }
423     if (CHENAL_SUD_DESCENDANT(point)) {
424         System.out.println("Bateau dans la zone: CHENAL SUD, SENS DESCENDANT");
425         e = 1;
426     }
427     if (ACCES_LAVERA(point)) {
428         System.out.println("Bateau dans la zone: ACCES LAVERA");
429         e = 1;
430     }
431     if (ZONE_SECURISE1(point)) {
432         System.out.println("Bateau dans la zone: ZONE SECURISE 1");
433         e = 1;
434     }
435     if (ACCES_GOLFE_FOS(point)) {
436         System.out.println("Bateau dans la zone: ACCES GOLF FOS");
437         e = 1;
438     }
439     if (ZONE_SECURISE2(point)) {

```

```
441     System.out.println("Bateau dans la zone: ZONE SECURISE 2");
442     e = 1;
443 }
444
445 if (e == 0) {
446     System.out.println("Bateau dans aucune des zones definies");
447 }
448 }
449
450 // :::::::::::::::::::: TEST
451 // ::::::::::::::::::::
452
453 public static void main(String args[]) {
454     double[] t = { 43.354725, 4.974134 };
455
456     double[] s = ConversionMercator.conversion(t);
457     dansQuelleZone(s);
458     Etat e = new Etat(43.354725, 4.974134);
459     System.out.println(e);
460
461 }
462 }
```