

***Rapport de Projet de c++ :***

*Filière Génie Informatique*

***Réalisé par :***

* Ayoub bouaoud

***Encadré par :***

*Nadia Chouhad*

**Année universitaire : 2023/2024**

***REMERCIEMENT***

***A la fin de ce travail***

***Nous souhaitons exprimer nous profonde gratitude envers toutes les personnes qui ont contribué à la réalisation de ce projet.***

***Tout d'abord, nous tenons à remercier chaleureusement notre encadrant, [Mme.Chouhad Nadia], pour son accompagnement constant, ses précieux conseils et son soutien tout au long de ce projet. Ses remarques constructives et ses orientations qui nous été aider inestimable pour surmonter les défis rencontrés et atteindre les objectifs fixés.***

***PLAN :***

1. ***Introduction***
2. ***Objectifs du projet***
3. ***Étapes principales***
4. ***Outils et technologies utilisés***
5. ***Résultats attendus***
6. ***Explication du code de chaque***

***module***

***\*module shape***

***\*module lifeform***

***\*module simulation***

***\*fichier.cpp***

1. ***Conclusion***
2. ***Introduction***

*Le projet Microrécif s'inscrit dans le cadre des travaux pratiques des sections Électricité et Microtechnique. Son objectif principal est de développer un module de simulation pour modéliser des entités biologiques simples, en particulier des microrécifs. Un microrécif est une structure biologique à petite échelle, souvent constituée de colonies de micro-organismes. La modélisation de ces structures peut offrir des perspectives intéressantes dans des domaines tels que la biologie marine, la microbiologie et les systèmes écologiques simulés.*

*Ce projet est divisé en plusieurs rendus, chacun visant à implémenter et valider une partie spécifique du système global. Le premier rendu, qui fait l'objet de ce document, se concentre sur la mise au point d'un module shape pour gérer les formes géométriques simples et la lecture de fichiers de configuration pour initialiser le modèle de simulation.*

*Le développement du module shape inclut la définition et la manipulation de points, segments, carrés et cercles dans un plan bidimensionnel. Les fonctionnalités implémentées permettent de calculer l'écart angulaire entre segments, vérifier leur superposition et détecter les intersections.*

*Parallèlement, le module simulation se charge de lire les fichiers de configuration, d'initialiser l'état du modèle et de gérer les erreurs potentielles rencontrées lors de cette lecture.*

*Ce rapport détaille la conception et l'implémentation des modules shape et simulation, ainsi que les tests réalisés pour valider leur bon fonctionnement. Le projet utilise des concepts avancés de programmation en C++, en mettant un accent particulier sur la rigueur et la qualité du code à travers des tests unitaires et la gestion des erreurs.*

1. ***Objectifs du projet***

*L'objectif principal du projet Microrécif est de développer un modèle de simulation informatique pour la modélisation et la gestion de structures biologiques simples, telles que les microrécifs. Plus spécifiquement, ce projet vise à :*

*\*\*Développer un module Shape :*

*Concevoir des structures de données pour représenter des formes géométriques de base telles que des points, des segments, des carrés et des cercles.*

*Implémenter des fonctions pour calculer l'écart angulaire entre segments, vérifier la superposition et détecter les intersections.*

*\*\*Créer un module de simulation :*

*Lire et interpréter des fichiers de configuration pour initialiser l'état du modèle.*

*Gérer les erreurs potentielles lors de la lecture des fichiers, telles que des angles invalides ou des longueurs incorrectes.*

*\*\*Tester et valider les composants du modèle :*

*Effectuer des tests unitaires rigoureux pour chaque fonction du module Shape afin d'assurer leur bon fonctionnement.*

*Intégrer les tests dans le flux de développement pour détecter et corriger les erreurs rapidement.*

1. ***Étapes principales***

*Pour réaliser le projet informatique de la série Electricité et Microtechnique, voici les étapes à suivre d'après le document fourni :*

*1. Préparation et Compréhension du Projet*

*Lire attentivement le document de projet pour comprendre les objectifs et les contraintes.*

*Étudier les méthodes de développement de projet mentionnées dans la série théorique associée.*

*2. Objectifs du Premier Rendu (Rendu1)*

*Module Shape :*

*Développer le module shape en commençant par définir les structures de données simples pour les segments, carrés et cercles.*

*Créer le type S2d pour modéliser un point ou un vecteur dans le plan.*

*Valider chaque fonction du module shape avec des tests unitaires et du scaffolding.*

*Fonctions à inclure dans le module shape :*

*Calcul de l’écart angulaire entre deux segments.*

*Détection de la superposition de deux segments ayant un point commun.*

*Détection de l’intersection ou de la superposition de deux segments indépendants.*

*3. Développement du Modèle*

*Module Simulation :*

*Créer une fonction ou méthode de lecture pour initialiser une simulation à partir d’un fichier.*

*Le module simulation doit être le seul point d’entrée du Modèle vis-à-vis de l’extérieur.*

*Module Lifeform :*

*Pour ce rendu, une approche simplifiée est acceptée avec un seul attribut de type dans une classe unique Lifeform.*

*Module Message :*

*Utiliser ce module pour afficher des messages standardisés pour la tâche de lecture du Modèle sans modification du module.*

*4. Fonction Main du Module Projet*

*Analyser la ligne de commande pour vérifier si le nom de fichier de test est fourni.*

*Transmettre le nom du fichier à une fonction du module simulation.*

*5. Tests et Validation*

*Lecture de Fichier :*

*Arrêter le programme dès la première erreur trouvée en appelant la fonction d’affichage de message d’erreur et exit(EXIT\_FAILURE).*

*Appeler success() et quitter avec exit(0) si la lecture se termine sans erreurs.*

*Tests Unités pour Module Shape :*

*Écrire un programme de test (scaffolding) qui appelle toutes les fonctions du module et compare les résultats aux attentes calculées indépendamment.*

*Nommer et définir les prototypes des fonctions exportées dans shape.h pour les inclure dans le programme de test.*

*6. Gestion des Erreurs*

*Erreurs à détecter lors de la lecture du fichier :*

*L’âge doit être strictement positif.*

*La longueur des segments doit être comprise dans les limites spécifiées.*

*L’angle des segments doit être compris entre [-π, π].*

*Le rayon des scavengers doit être dans les limites spécifiées.*

*Vérifier l’unicité des identificateurs de coraux et l’existence des coraux référencés par les scavengers.*

*Utilisation de epsil\_zero :*

*Considérer epsil\_zero comme nul pendant les tests de lecture de fichier.*

*Ajouter un paramètre supplémentaire dans les fonctions de shape pour activer ou désactiver l’utilisation de epsil\_zero.*

1. ***Outils et technologies utilisés***

***Langages de Programmation***

***C/C++ :*** *Langages principaux pour développer les modules et fonctions nécessaires.*

*Environnements de Développement*

***IDE (Integrated Development Environment) :***

*CodeBlocks : Un IDE open-source pour C/C++, utile pour l'édition, la compilation et le débogage.*

*Visual Studio Code : Un éditeur de code léger et extensible, souvent utilisé avec des extensions pour C/C++.*

1. ***Résultats attendus***

*Le résultat attendu pour ce projet est structuré en plusieurs étapes, chacune ayant des objectifs spécifiques. Voici un résumé des résultats attendus pour le rendu1 :*

***1. Mise au Point du Module shape***

***Validation des Fonctions :*** *Chaque fonction du module shape doit être validée par des tests extensifs, y compris des tests unitaires et de scaffolding.*

*\*Définition des Structures de Données :*

*Type S2d pour modéliser un point ou un vecteur dans le plan.*

*Structures de données pour les segments, carrés et cercles.*

*\*Fonctions à Implémenter :*

*Calcul de l'écart angulaire entre deux segments.*

*Détection de la superposition de deux segments ayant un point commun.*

*Détection de l'intersection ou de la superposition de deux segments indépendants.*

***2. Ébauche du Modèle Module simulation :*** *Doit agir comme le seul point d’entrée du modèle et gérer les tâches de mise à jour de la simulation, dessin, lecture et écriture de fichiers.*

*\*Lecture de Fichier : Implémentation de la fonction de lecture pour initialiser une simulation et détecter des erreurs prédéfinies.*

*\*Gestion des Erreurs : Le programme doit détecter et gérer les erreurs lors de la lecture du fichier de configuration, arrêtant le programme en cas d'erreur.*

***3. Module projet Fonction main :***

*Récupérer le nom du fichier de test depuis la ligne de commande.*

*Transmettre ce nom de fichier à une fonction du module simulation.*

***4. Tests et Validation Tests Unitaires :*** *Mise en place de tests unitaires pour le module shape.*

*\*Utilisation du Module message :*

*Affichage de messages standardisés pour les erreurs et succès lors de la lecture de fichiers.*

***5. Documentation et Méthode de Travail***

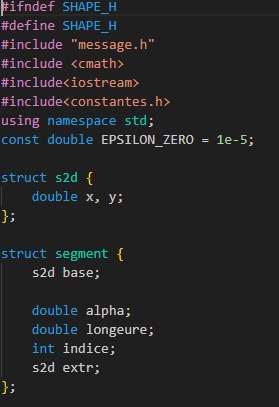
***Documentation du Code*** *:*

*Bien commenter le code et fournir une documentation adéquate.*

*Organisation des Tests : Développer une batterie de tests indépendamment des tests fournis pour s'assurer de la robustesse du code.*

***6- Explication du code de chaque module***

***\*module shape :***

******

***Dans le cadre de notre module de programmation avancée en C++, nous avons entrepris de concevoir et de développer un projet intitulé "Simulation de l’évolution de micro-organismes dans une goutte d’eau en 2D". Ce projet vise à modéliser les interactions entre les algues et les coraux dans un environnement simulé, en mettant en œuvre des concepts avancés de la programmation en C++.***

***Les entités Corail, Algue et Scavenger jouent un rôle crucial dans la nature en assurant sa stabilité. Notre projet ambitionne de créer un outil logiciel capable de simuler la croissance des algues et des coraux, de détecter les collisions et les superpositions, et de fournir des analyses précises de leur évolution.***

***En utilisant une approche orientée objet, nous avons développé des classes C++ représentant les différentes entités et leurs interactions. Des algorithmes de détection de collisions et de vérification de la validité des coordonnées ont également été implémentés.***

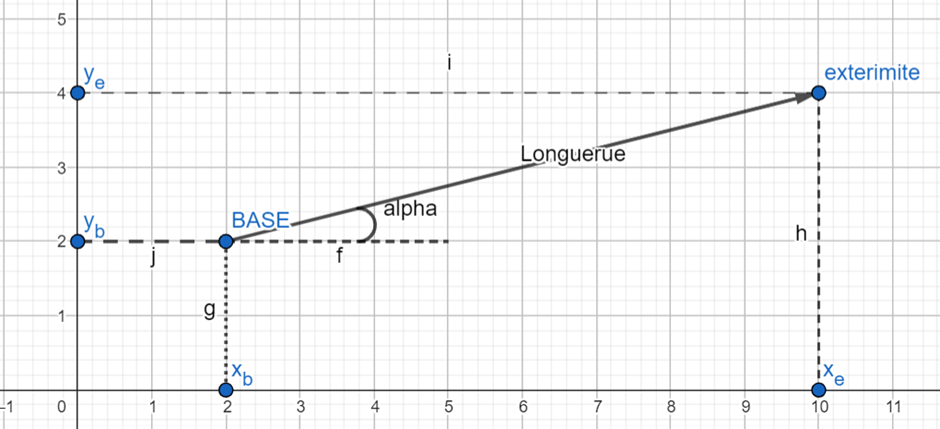
***La partie Shape[.cc/.h] :***

***Structure s2d :***

***Présente les coordonnées d'un point/vecteur dans un plan orthonormé (o,i,j). Pour l'initialiser, on utilise initialise\_s2d qui prend comme arguments double x et double y et retourne un point ou vecteur avec les mêmes coordonnées.***

***Structure segment :***

***Présentation d'un segment défini par sa base (origine), un écart angulaire et une norme. Il suffit juste d'initialiser ces trois paramètres pour obtenir l'extrémité.***

******

***pour avoir les cordonnées de l’extimité :***

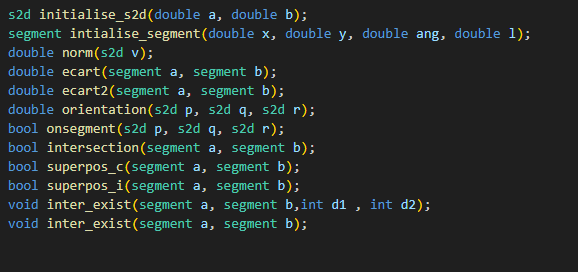
***sin⁡α=((y\_e-y\_b))/longueure***

***cos⁡α=((x\_e-x\_b))/longueure***

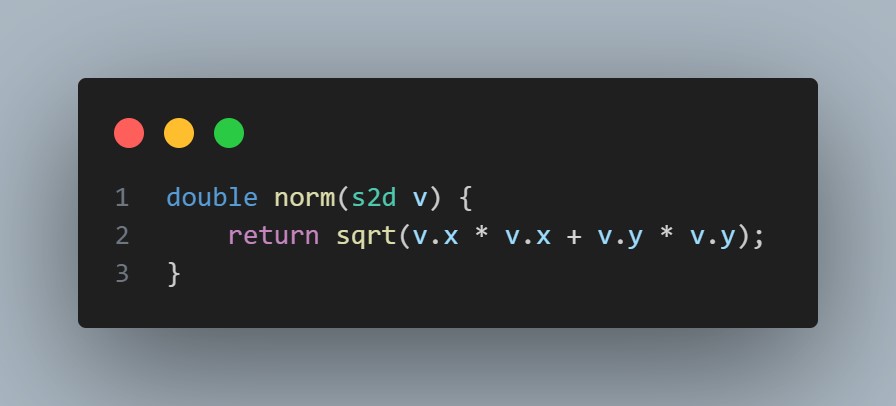
***Alors :***

***ye = longueure . sin⁡α + yb***

***xe = longueure . cos⁡α + xb***

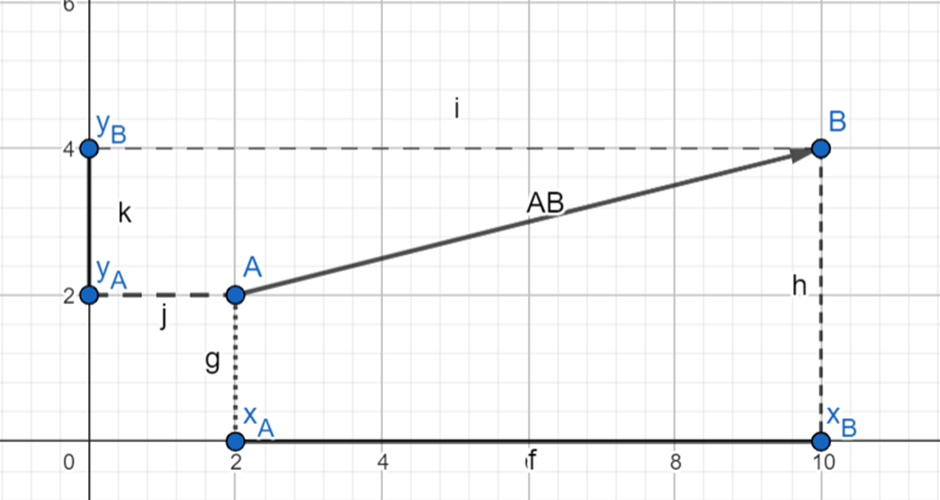
******

***La fonction norme:***

******

***Elle prend comme argument un vecteur et retourne son longueur selon le théorème de Pythagore d'après les coordonnées***

***dans cette figure :***

******

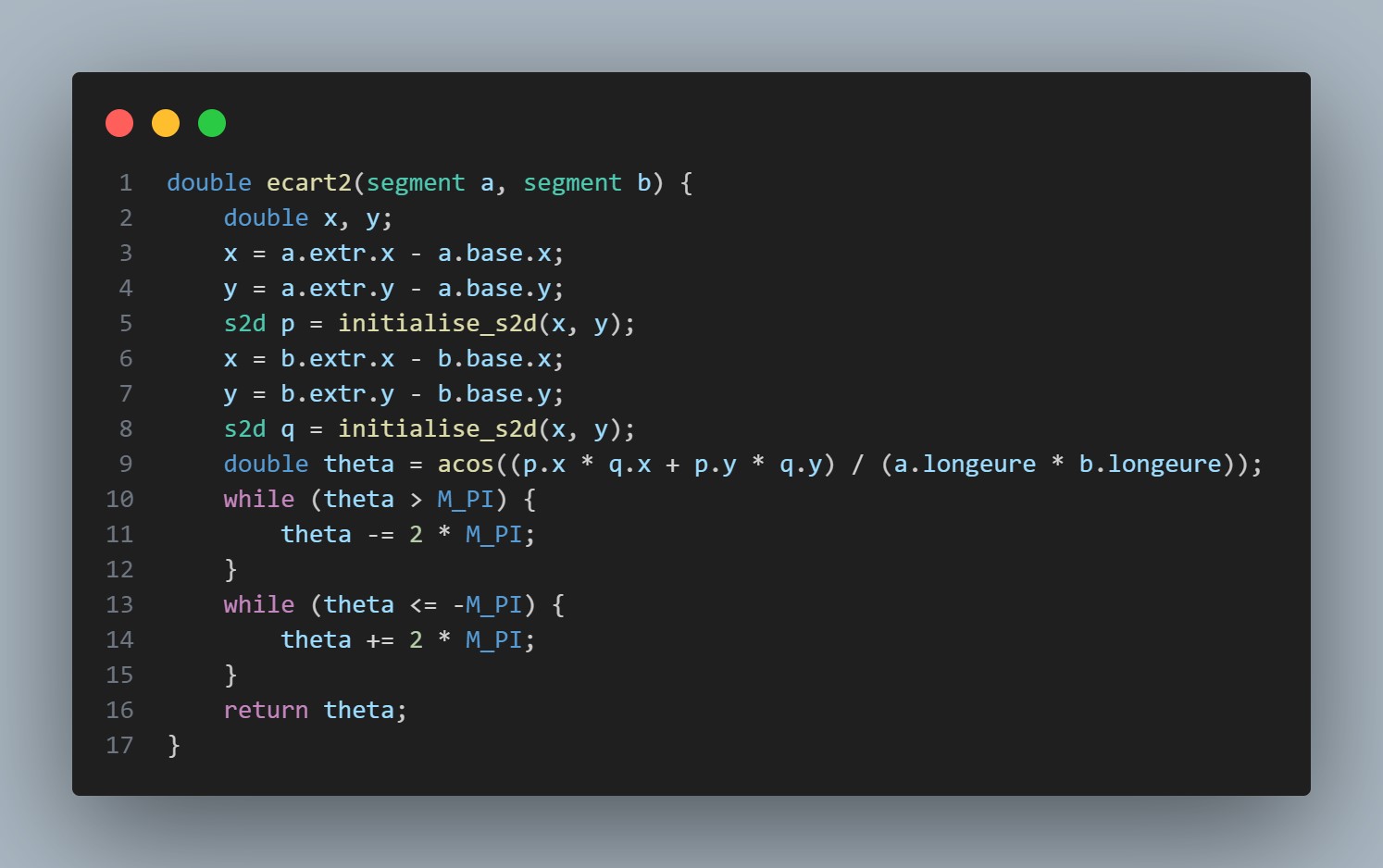
***par translation on obtient un triangle rectangle cela nous amène à dire :***

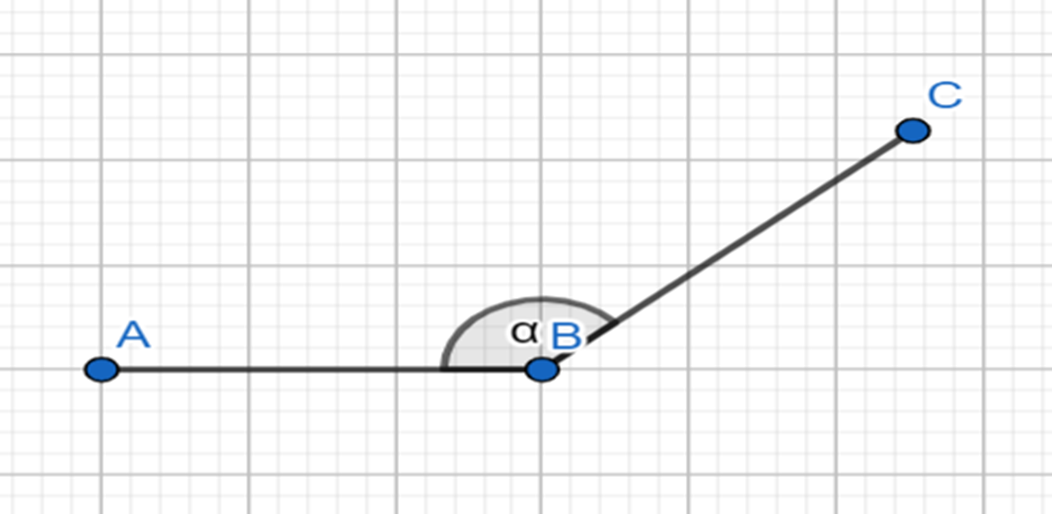
***〖(AB)〗^2=〖(yb-ya)〗^2+〖(xb-xa)〗^2***

***Alors :***

***|AB|=√(〖(yb-ya)〗^2+〖(xb-xa)〗^2 )***

***L’écart angulaire() :***

******

******

***cette fonction retourne l' écart angulaire entre 2 segment dans l ‘intervalle ]-π ,π]***

***on se base sur les propriétés analytique et géométrique de produit scalaire :***

***Soient : u ⃗ (■(x1@y1)) et v ⃗ (■(x2@y2))***

***1.u ⃗.v ⃗= y1.y2+ x1.x2 et 2. u ⃗.v ⃗= |v ⃗ |.|u ⃗ |.cos⁡〖((u ⃗.v ⃗ ) ̂)〗***

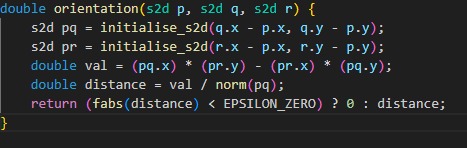
***De 1 et 2 cherchant alpha entre eux :***

***cos⁡〖((u ⃗.v ⃗ ) ̂)〗=( y1.y2+ x1.x2 )/( |v ⃗ |.|u ⃗ | ) □(⇔┴1 ) (u ⃗.v ⃗ ) ̂=cos^(-1)⁡〖(( y1.y2+ x1.x2 )/( |v ⃗ |.|u ⃗ | ))〗***

***on utilisant deux boucle while***

***qui vont soustraire ou ajouter 2π jusqu' au alpha soit compris entre ]-π,π]***

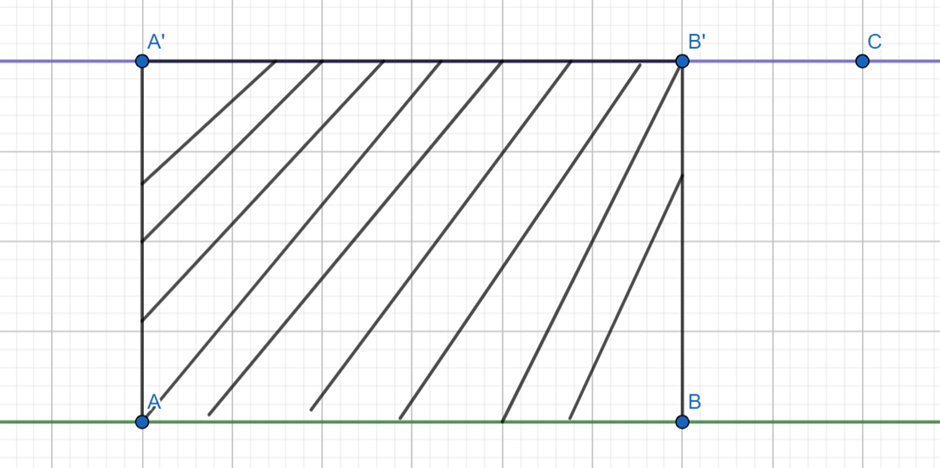
***La fonction orientation :***

******

***il retourne la distance vertical signé entre un point et une droit dirigé par un vecteur directeur (AB) ⃗***

***on calcule tout d ‘abord la surface signé en utilisant les propriétés analytique de produit vectoriel entre AB et BC :***

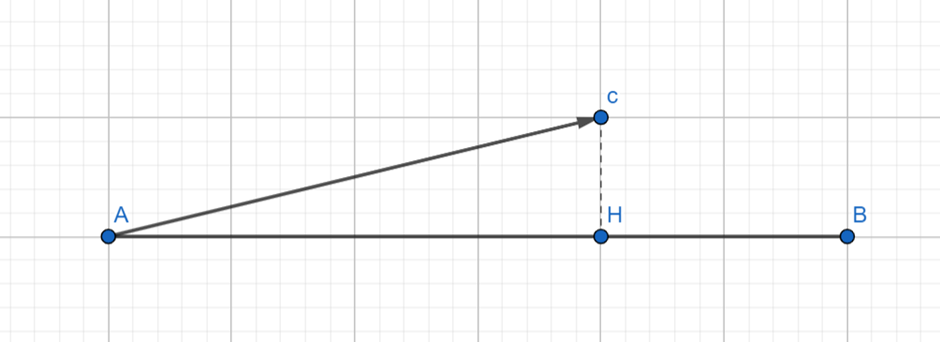
***(AB) ⃗^(BC) ⃗= y1.x2- x1.y2 avec (AB) ⃗(x1¦y1)et (BC) ⃗(x2¦y2)***

******

notre objet c’est trouver la distance B’B : BB'=((AB) ⃗^(BC) ⃗)/|(BC) ⃗ |

si la distance et presque nul on retourne 0 sinon on retourne d(C,AB) signé .

onsemengent() : sachant tous que le produit scalaire est une valeur algébrique représente la projection de C sur AB :

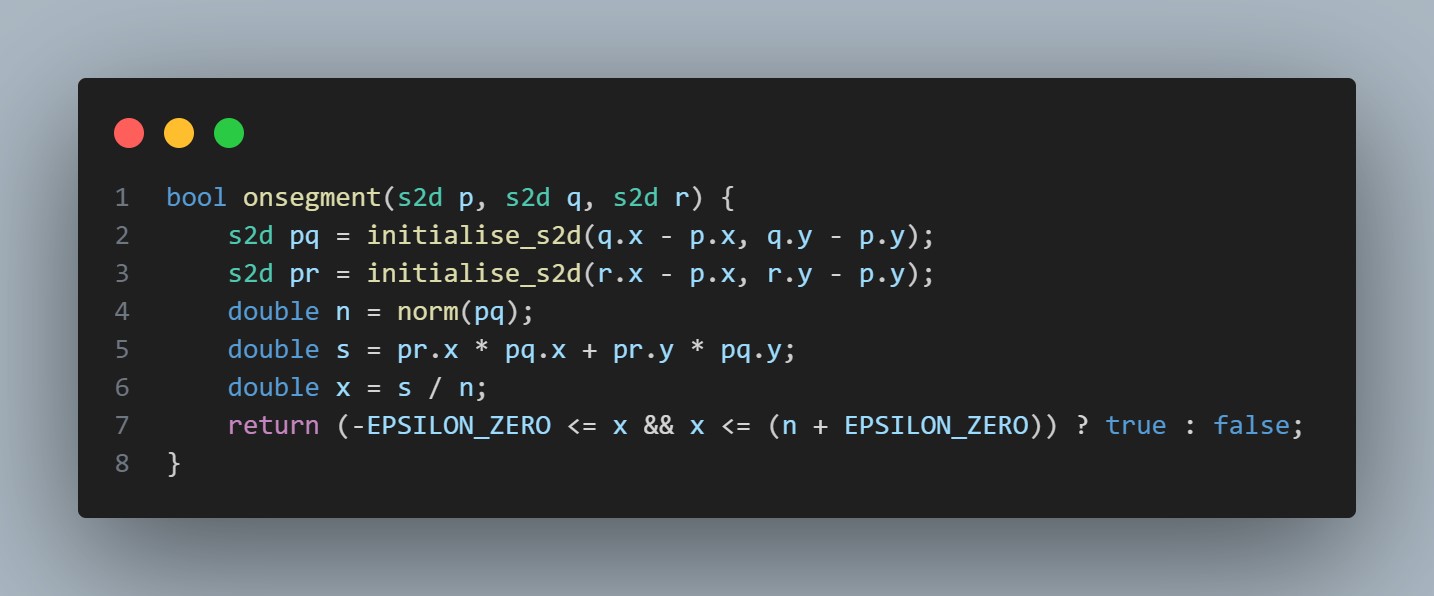


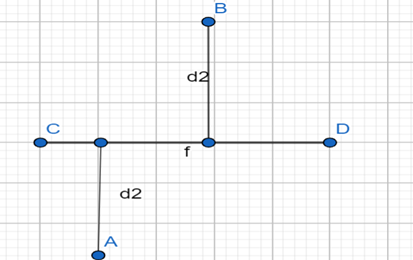
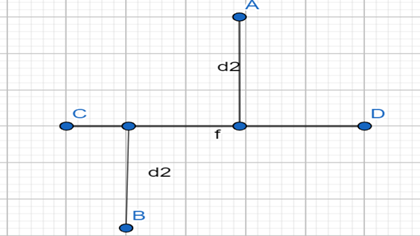
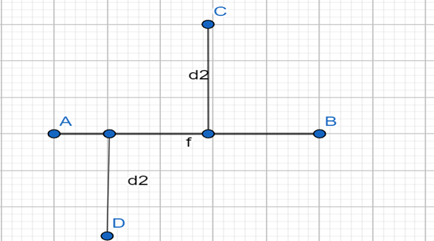
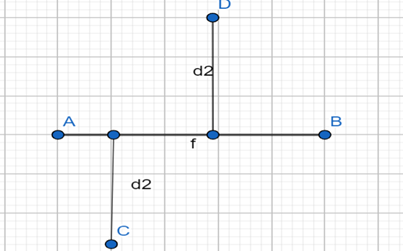
on divise cet valeur sur la norme AB et elle est compris entre 0 et AB donc A,C,B sont alignés on retourne true sinon on return false ;

détection de l’ intersection :

on a besoin des 2 fonction précédent ( Onsegment() et orientation())

les cas d intersection simple(orientation):





alors on conclure que :

on doit comparer la distance entre point et un segment donc 4 comparison ⇔ 4 distances :

d(A,[CD])=d1 => la vertical entre la droite passant par a || (CD)

d(B,[CD])=d2 => la vertical entre la droite passant par b || (CD)

d(C,[AB])=d3 => la vertical en la droite passant par c|| (AB)

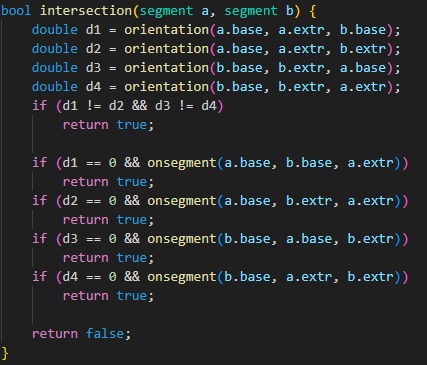
d(D,[AB])=d4 => la vertical en la droite passant par d || (AB)

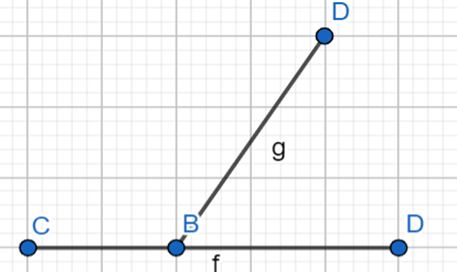
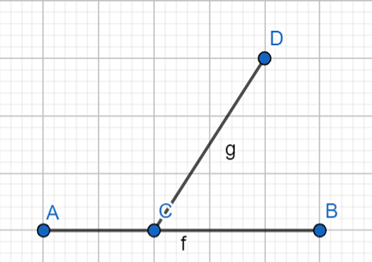
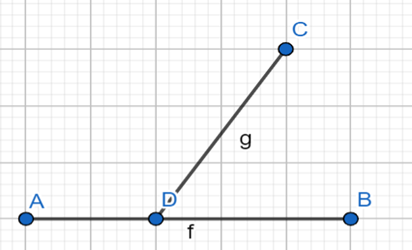
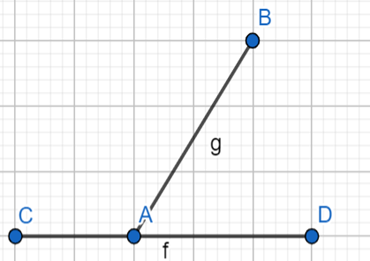
d1 et d2 doivent avoir un signe différent => d1 ≠d2

d3 et d4 doivent avoir un signe différent=> d3 ≠d4

alors la condition pour le première cas : ( d1 ≠d2 et d1 ≠d2 )

les cas d intersection composees (onsement() et oriontation ):





même chose ici on distingue 4 cas mais en s’ intéresse a une point l’autre peut être en dessus ou dessous :

C ∈ [AB] et d(C,(AB))= 0 ⇔ [AB] ∩ [CD]

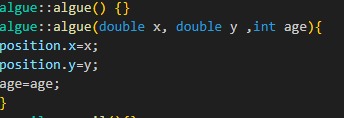
D ∈ [AB] et d(D,(AB))= 0 ⇔ [AB] ∩ [CD]

A ∈ [cd] et d(A,(CD))= 0 ⇔ [AB] ∩ [CD]

B ∈ [cd] et d(B,(CD))= 0 ⇔ [AB] ∩ [CD]

**\*module lifeform**

**Ce code C++ est une partie d'un programme qui gère différentes formes de vie (algues, coraux et charognards) dans un environnement. Chaque type de forme de vie a ses propres caractéristiques et méthodes pour vérifier la validité de ses propriétés. Voici une explication détaillée du code :**

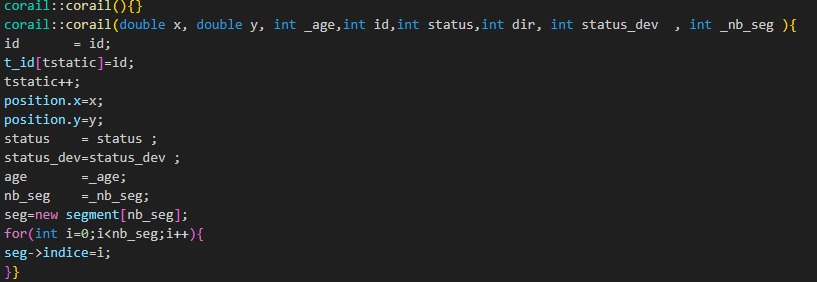
****

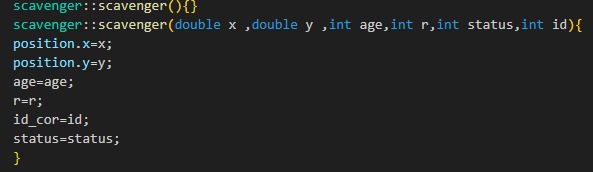
**Constructeur par défaut : Initialise un corail sans définir ses propriétés.**

**Constructeur paramétré : Initialise un corail avec une position (x, y), un âge \_age, un identifiant id, un statut status, une direction dir, un statut de développement status\_dev et un nombre de segments \_nb\_seg.**

**Met à jour un tableau statique t\_id avec l'identifiant id.**

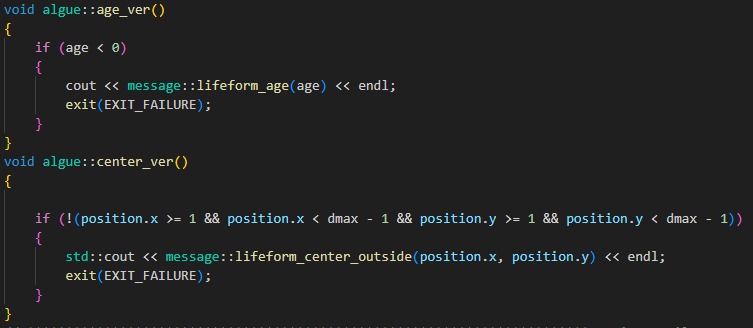
**Initialise chaque segment avec un indice.**

****

****

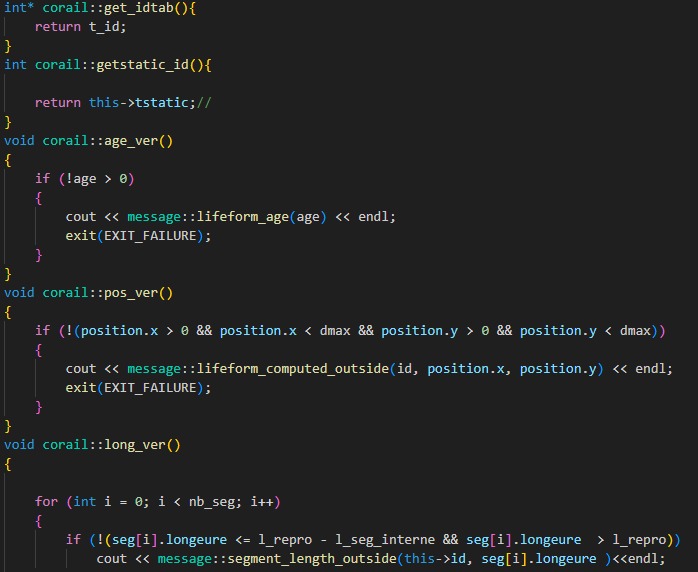
**Constructeur par défaut : Initialise un charognard sans définir ses propriétés.**

**Constructeur paramétré : Initialise un charognard avec une position (x, y), un âge age, un rayon r, un statut status et un identifiant id.**

****

**age\_ver : Vérifie si l'âge de l'algue est négatif et affiche un message d'erreur si c'est le cas.**

**center\_ver : Vérifie si l'algue est à l'intérieur des limites autorisées (dmax) et affiche un message d'erreur si ce n'est pas le cas.**

****

**get\_idtab : Retourne le tableau d'identifiants statiques des coraux.**

**getstatic\_id : Retourne le compteur statique tstatic.**

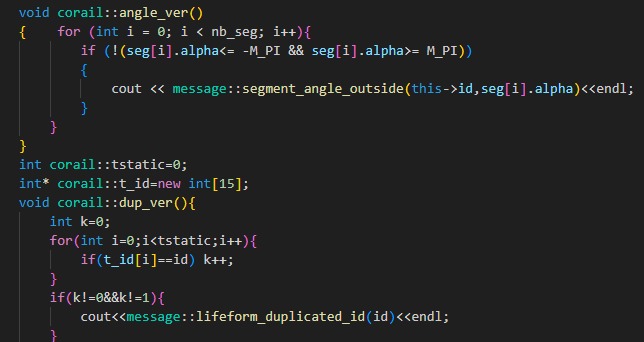
**age\_ver : Vérifie si l'âge du corail est positif, sinon affiche un message d'erreur.**

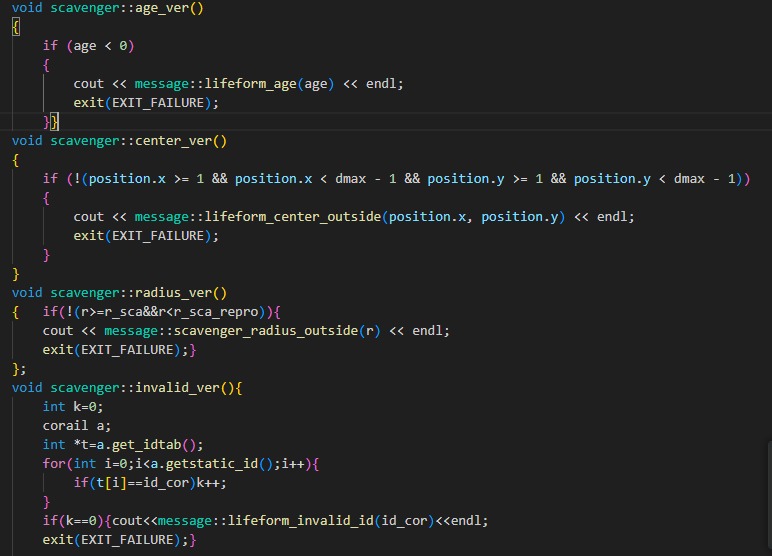
**pos\_ver : Vérifie si le corail est à l'intérieur des limites autorisées (dmax), sinon affiche un message d'erreur.**

**long\_ver : Vérifie la longueur de chaque segment et affiche un message d'erreur si elle n'est pas dans les limites autorisées.**

**angle\_ver : Vérifie l'angle de chaque segment et affiche un message d'erreur si l'angle n'est pas dans les limites autorisées.**

**dup\_ver : Vérifie les duplications d'identifiant et affiche un message d'erreur s'il y a des doublons.**

****

****

**age\_ver : Vérifie si l'âge du charognard est négatif et affiche un message d'erreur si c'est le cas.**

**center\_ver : Vérifie si le charognard est à l'intérieur des limites autorisées (dmax) et affiche un message d'erreur si ce n'est pas le cas.**

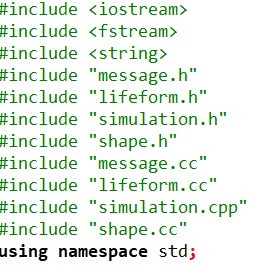
**radius\_ver : Vérifie si le rayon du charognard est dans les limites autorisées et affiche un message d'erreur si ce n'est pas le cas.**

**invalid\_ver : Vérifie si l'identifiant du corail associé au charognard est valide en parcourant les identifiants de coraux.**

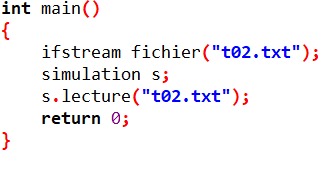
**\*\*resume Ce code définit des classes pour les algues, les coraux et les charognards, chaque classe ayant ses propres constructeurs et méthodes de vérification pour s'assurer que leurs propriétés sont valides. Les méthodes de vérification sont conçues pour identifier et signaler les erreurs concernant l'âge, la position, la longueur et l'angle des segments, ainsi que les identifiants des formes de vie, garantissant ainsi l'intégrité des données dans l'environnement simulé.**

***\*fichier.cpp***

Ce code semble être un programme en C++ qui effectue une simulation en lisant des données à partir d'un fichier texte (t02.txt). Il utilise plusieurs fichiers d'en-tête (message.h, lifeform.h, simulation.h, shape.h) et leurs implémentations correspondantes (message.cc, lifeform.cc, simulation.cpp, shape.cc).



Ces directives incluent différentes bibliothèques standard de C++ (iostream, fstream, string) ainsi que les fichiers d'en-tête (message.h, lifeform.h, simulation.h, shape.h) et leurs implémentations (message.cc, lifeform.cc, simulation.cpp, shape.cc). Cela donne accès aux fonctionnalités et aux définitions de ces fichiers dans le programme.



\*\*Ouvre un flux de fichier en lecture (ifstream) nommé fichier et tente de l'ouvrir avec le fichier texte t02.txt.

Crée une instance de la classe simulation appelée s.

Appelle la méthode lecture() de l'objet s pour lire et traiter les données du fichier t02.txt.

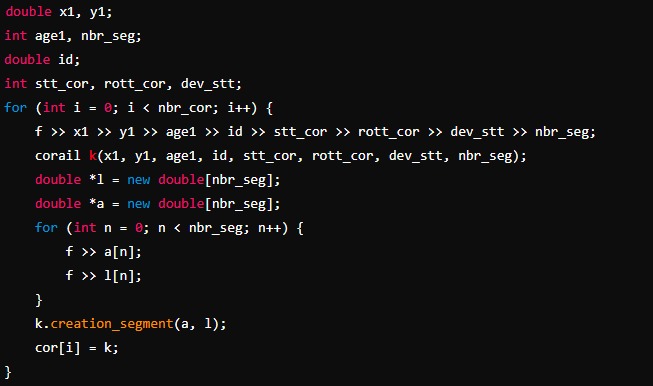
Retourne 0 pour indiquer une sortie sans erreur du programme

\*\*La fonction lecture() est appelée sur l'objet simulation, mais son fonctionnement détaillé dépend de l'implémentation de la classe simulation dans le fichier simulation.cpp.

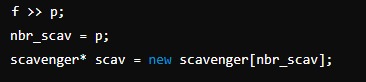
***\*module simulation***

*Le processus est similaire à celui des algues : la prochaine valeur lue est le nombre de coraux nbr\_cor.*

*Un tableau dynamique cor de type corail est créé pour stocker nbr\_cor coraux.*

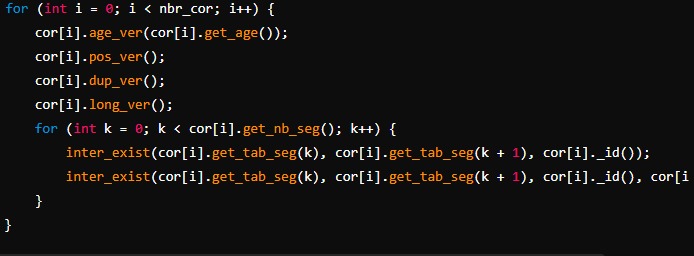
**

*Les informations pour chaque corail, y compris les segments associés, sont lues et stockées dans un tableau dynamique.*

**

*La prochaine valeur lue est le nombre de charognards nbr\_scav.*

*Un tableau dynamique scav de type scavenger est créé pour stocker nbr\_scav charognards.*

**

*\*\*Des méthodes de vérification pour les coraux sont appelées, incluant des vérifications d'intersections entre segments*

*\*\*Ce code lit les données des algues, des coraux et des charognards à partir d'un fichier, initialise les objets correspondants, puis applique diverses vérifications et traitements sur ces objets. Le résultat final est une simulation écologique basée sur les données fournies.*

***7-Conclusion***

***En conclusion, ce projet de simulation écologique fournit une base solide pour la modélisation d'écosystèmes complexes. Il permet de visualiser et de comprendre les dynamiques écologiques en utilisant une approche modulaire et extensible. Les résultats obtenus peuvent servir de support pour des recherches écologiques, des études environnementales, ou encore des applications éducatives, offrant ainsi une compréhension approfondie des interactions biologiques au sein des écosystèmes simulés***