Description du module MODELISATION (rev. 2)

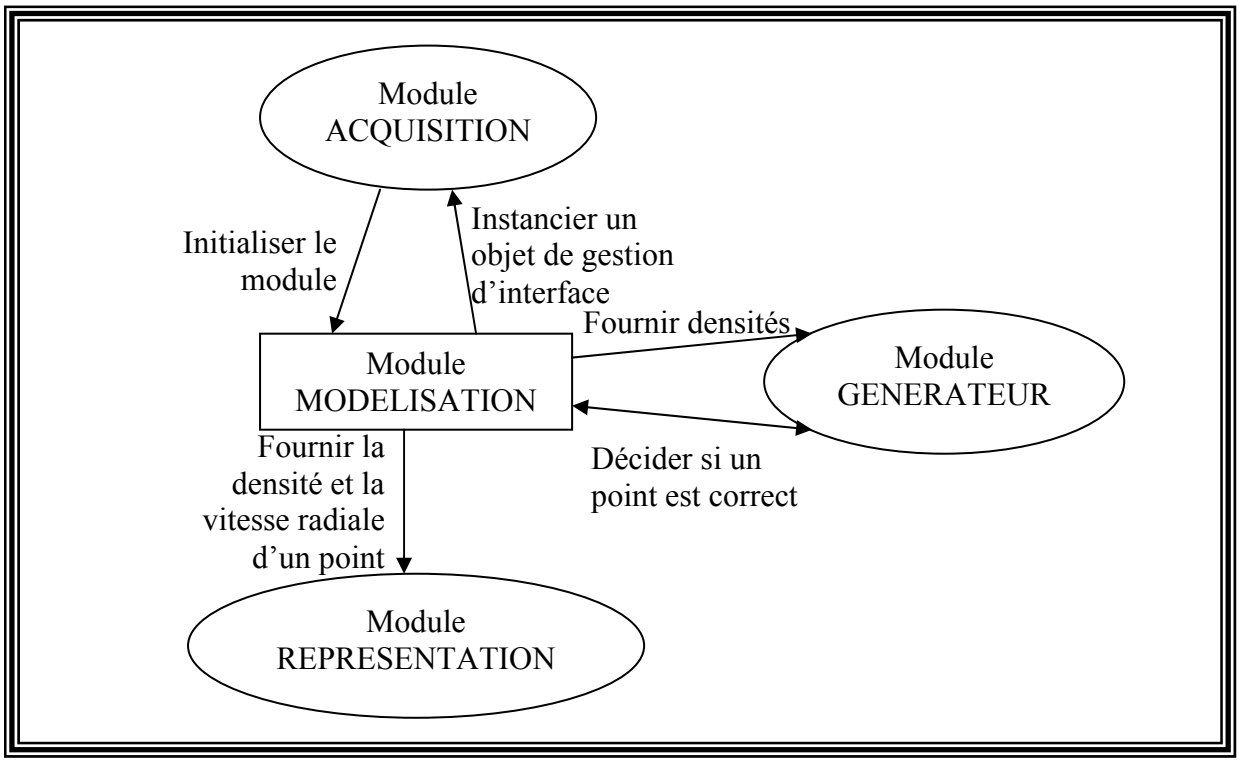
1. Description du rôle du module

Le module MODELISATION, étant donné un modèle éventuellement complexe préalablement défini, mais créé de façon hiérarchique à partir de modèles simples, doit implémenter tous les calculs nécessaires aux opérations suivantes :

- déterminer l'appartenance d'un point au modèle,
- déterminer la densité en un point,
- déterminer le vecteur vitesse en un point, et sa projection sur l'axe z.

2. Echanges avec les autres modules

A. Représentation visuelle



B. Echanges avec INTERFACE/ACQUISITION : initialisation et configuration

Le module sera initialisé pour un modèle particulier, les objets qui le composent étant créés hiérarchiquement via une interface graphique (qui passe par l'interface standardisée IParamétrable), définie indépendamment pour chaque sous-objet du modèle. Ces mini-interfaces se chargeront de remplir les paramètres adéquats de chaque sous objet à l'aide des entrées de l'utilisateur. Ces paramètres, outre les variables spécifiques aux différentes lois de densité et de champ de vitesse, comprendront une orientation, une origine et une échelle.

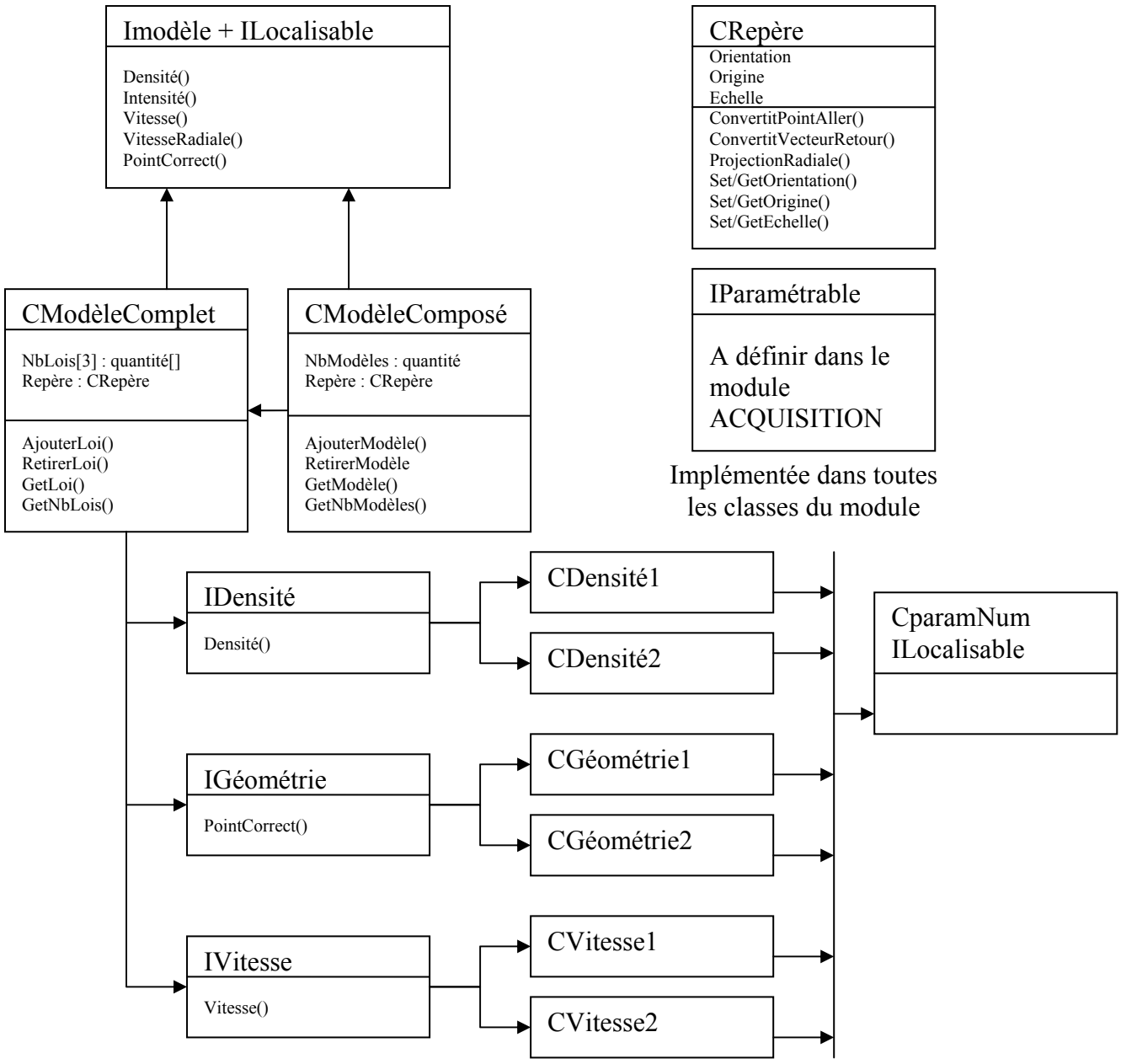
C. Echanges avec le Générateur

L'initialisation du générateur passe par la lecture des densités en un certain nombre de points. Celui-ci fera donc appel à la méthode densité en un grand nombre de points dans sa phase d'initialisation.
En cours de fonctionnement, le générateur fera appel à la méthode PointCorrect pour vérifier qu'un point généré est bien compris dans le modèle.

D. Echanges avec le module REPRESENTATION

En règle générale, le module REPRESENTATION fera appel à la méthode VitesseRadiale uniquement, car les informations sur la densité sont déjà « utilisées » dans la génération des points ; cependant il sera toujours possible d'utiliser les méthodes densité ou intensité pour des représentations particulières.

3. Diagramme de classes du module



4. Description des classes et interfaces

A. Interface IModèle

Elle définit les méthodes qui doivent être implémentées pour définir un modèle exhaustif : il sera implémenté par CModèleComplet, et CModèleComposé, mais pourra également être utilisé par la suite pour définir des modèles supplémentaires. Ces derniers pourront ainsi être ajoutés facilement : il suffira qu'ils implémentent cette interface pour pouvoir être utilisés.

La méthode VitesseRadiale devra renvoyer la projection selon l'axe x de l'objet. Ainsi, le repère de l'objet qui se trouve au sommet de la hiérarchie des modèles définira la direction d'observation.

B. Interface IParamétrable

Cf. explications module INTERFACE/ACQUISITION

C. Classe CRepère

Cette classe définit des objets qui implémentent un changement de repère. De tels objets pourront être ajoutés en tant que propriété à chaque objet de la hiérarchie des modèles où on veut définir un repère local (par rapport au repère de l'objet parent).

Les différents champs de la classe CRepere :

- Orientation contient les 3 angles, entre les axes de la base parente et de la base locale.
- Origine coordonnées de l'origine du repère local exprimées dans le repère parent.
- Echelle contient les normes des 3 vecteurs définissant la base locale en prenant comme unité les normes des vecteurs de la base parente.

Les différentes méthodes sont les suivantes :

- ConvertitPointAller() : convertit les coordonnées d'un point exprimées dans le repère parent, vers des coordonnées exprimées dans le repère local.
- ConvertitVecteurRetour() : convertit les coordonnées d'un vecteur exprimées dans la base locale, vers des coordonnées exprimées dans la base parente.
- ProjectionRadiale() : fournit la projection sur l'axe x (ligne de visée) d'un vecteur exprimé dans la base locale.
- Les autres méthodes sont des accesseurs.
- La propriété Orientation représente deux angles, entre les axes de la base parente et de la base locale.
- La propriété Origine contient les coordonnées de l'origine du repère local exprimées dans le repère parent.
- La propriété Echelle contient les normes des 3 vecteurs définissant la base locale, en prenant comme unité les normes des vecteurs de la base parente. (ie ce sont les facteurs d'échelle appliqués à chaque axe).

D. Classe CModèleComplet

Cette classe définit l'objet qui crée un modèle à part entière, à partir de différentes lois de densité, de géométrie, et de champs de vitesses.

Les champs de la classe CmodeleComplet :
- TYPELOIS : Tableau qui contient les types associés à chacune des lois.
- Des entiers stockant les différentes lois : ex : LD_GAUSSIENNE pour une loi de densité gaussienne.
- VectDensite : tableau contenant les lois de densité
- VectGeometrie : tableau contenant les lois de géometrie
- VectVitesse : tableau contenant les lois de vitesse
- VectLoi : vecteur des lois (indistinctement)
- CRepere Repere repère du ModeleComplet.

- Point en cours de coordonnées Px, Py, Pz
- Densite du point en cours : DensitePoint
- Coordonnées du vecteur vitesse du point en cours : Vx,Vy, Vz
- Vitesse radiale du point en cours :Vr.
- Pcorrect : variable stockant si oui ou non le point en cours est correct.
- Nbdimension variable qui stocke le nombre de dimensions des lois de densité
la dimension sera mise à 3 si au moins une des lois est à 3D

Les différentes méthodes sont :

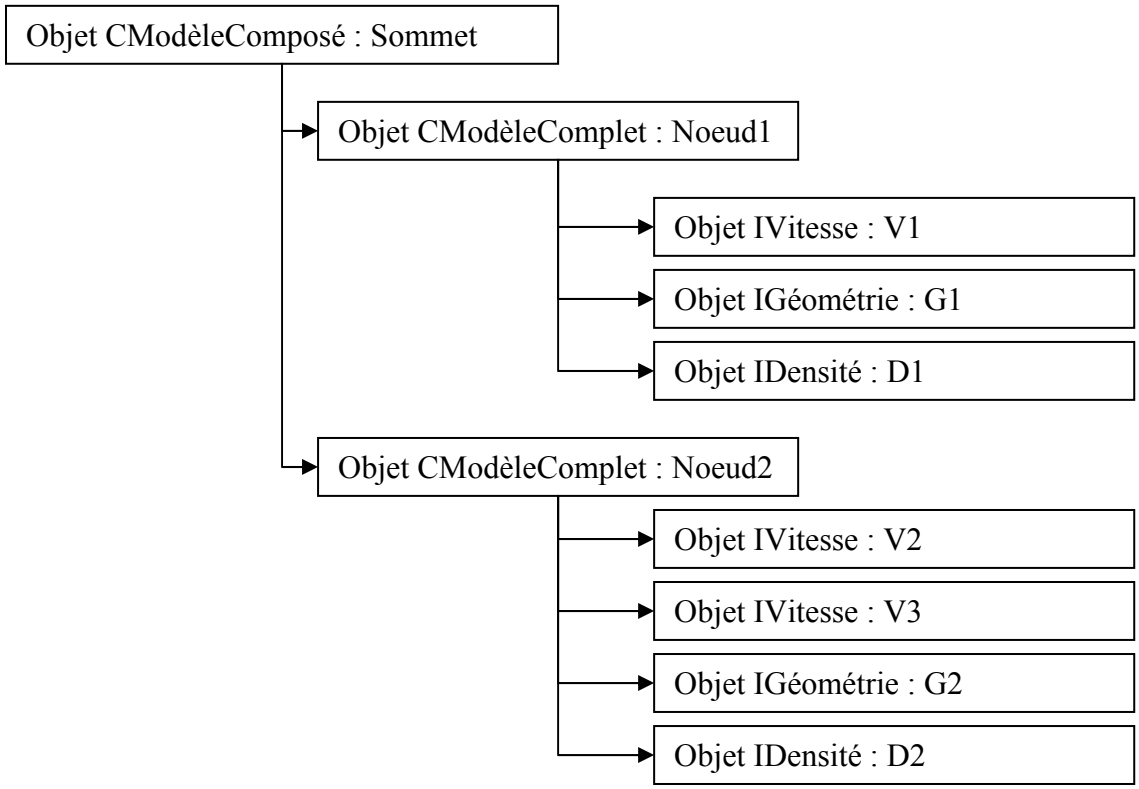
- AjouterLoi / RetirerLoi / GetLoi / GetNbLois permettent de gérer la liste des lois contenues dans l'objet, et dont l'aggrégation forme le modèle. Pour être complet, un modèle devra bien sûr contenir au moins une loi de chaque type.
- La méthode Vitesse va faire appel à la méthode Vitesse (pour le point en cours) de chaque objet Vitesse en executant la méthode Vitesse de chaque interface IVitesse du tableau TabVitesse.
- La méthode VitesseRadiale se contente de projeter le résultat de la méthode vitesse sur l'axe x : ligne de visée.
- La méthode Densite va faire appel à la méthode Densite(pour le point en cours) de chaque objet Densite en executant la méthode densite de chaque interface IDensite du tableau TabDensite.
- La méthode PointCorrect fait un appel à chaque loi de géométrie (pour le point en cours si au moins une renvoie une valeur négative Pointcorrect met le champs Pcorrect à 0, sinon si au moins une renvoit une valeur positive Pcorrect est mis à une valeur >0, sinon Pcorrect est mis à 0.
- MAJConfig met à jour le repère avec les valeurs entrées par l'utilisateur.
- On redéfinit les méthodes de l'interface IParametrable implémentée par la classe.

E. Classe CModèleComposé

Cette classe définit l'objet destiné à se trouver au sommet de la hiérarchie du modèle : il permet de combiner différents modèles ensemble. Les méthodes AjouterModèle,

- RetirerModèle, GetModèle, GetNbModèles permettent de gérer la liste des modèles ainsi combinés.
- La méthode Densite fait appel à la méthode Densite de tous les CModeleComplet contenus dans le tableau TabModeleComplet.
- La méthode vitesse fait appel à la méthode Vitesse de tous les CModeleComplet contenus dans le tableau TabModeleComplet.
- La méthode VitesseRadiale se contente de projeter le résultat de la méthode vitesse sur l'axe x : ligne de visée.
- Méthode PointCorrect : Pour chaque sous modèle, on appelle PointCorrect, si au moins une réponse est positive on renvoit un nombre >0, sinon on renvoit 0.
- SetupPanel met en place les contrôles graphiques.
- On redéfinie les méthodes de l'interface IParametrable.

La hiérarchie prévue pour une modélisation est la suivante (par exemple):



F. Interfaces IVitesse, IGéométrie, IDensité

Ces trois interfaces définissent les méthodes à définir pour implémenter chacun des trois types de lois possibles. Un objet qui implémente une de ces trois interface, en implémentant IModele peut être utilisée dans un CModèleComplet. Il doit aussi étendre CparamNum et implémenter l'interface ILocalisable

5. Séquences de fonctionnement

A. Initialisation d'un modèle simple

- Création d'un CModèleComplet par le module ACQUISITION
- Utilisation par ACQUISITION de son interface IParamétrable
- Suite aux interactions de l'utilisateur : appel de AjouterLoi pour ajouter une loi de densité, de vitesse ou de géométrie.
- Création d'un CDensité, CVitesse, ou CGéométrie
- Utilisation par ACQUISITION de l'interface IParamétrable de l'objet correspondant

Répétition à volonté des étapes iv et v.

B. Initialisation d'un modèle composé

- Création d'un CModèleComposé par le module ACQUISITION
- Utilisation par ACQUISITION de son interface IParamétrable

- Suite aux interactions de l'utilisateur : appel de AjouterModèle pour ajouter un sous-modèle.
- Création d'un CModèleComposé ou d'un CModèleComplet
- Utilisation par ACQUISITION de l'interface IParamétrable de l'objet correspondant

Répétition à volonté des étapes iv et v.

C. Appel de Densité, Vitesse

Supposons que c'est un appel de densité(), et que la racine est un CModèleComposé :
Pour chaque sous-modèle contenu, on appelle PointCorrect() ; si la réponse est positive, on appelle Densité() et le résultat est ajouté au sous-total, sinon on passe au sous-modèle suivant.
Lorsqu'on a parcouru tous les sous-modèles, le sous-total est renvoyé.
(idem pour vitesse)

Supposons que c'est un appel de PointCorrect() et que la racine est un CModèleComposé :
Pour chaque sous-modèle contenu, on appelle PointCorrect() ;si au moins une réponse est positive, on renvoie >0, sinon 0.

Supposons que c'est un appel de densité(), et que la racine est un CModèleComplet :
On appelle PointCorrect(), si la réponse est positive, on calcule : on appelle chaque loi de densité, et on fait la somme des résultats pour la renvoyer ensuite.
(idem pour vitesse)

Supposons que c'est un appel de PointCorrect(), et que la racine est un CModèleComplet :
On appelle chaque loi de géométrie,
si au moins une renvoie une valeur négative, on renvoie 0,
sinon si au moins une renvoie une valeur positive, on renvoie >0,
sinon on renvoie 0.

D. Appels de PointCorrect

Nous souhaitons pouvoir combiner des modèles de la façon suivante :

- dans un CModèleComposé, les différents modèles doivent être ajoutés (ie les densités de points sont ajoutées, les vitesses également mais dans leur cas addition vectorielle)
- dans un CModèleComplet, les lois de densité et de vitesse doivent être ajoutées également ; cependant on doit pouvoir définir des zones d'exclusion ou d'inclusion à l'aide des IGéométrie

Dans ce but, l'appel à PointCorrect d'un IGéométrie doit pouvoir renvoyer trois valeurs différentes : positif pour un point à inclure, nul pour indifférent, négatif pour un point à exclure.
Par contre la méthode PointCorrect d'un IModèle n'a que deux valeurs possibles (point dans ou en dehors du modèle) vrai, ou faux

E. Utilisation de CRepere

- Dans tous les appels de fonctions faisant intervenir un point comme argument, la méthode appelée fera en premier lieu convertir les coordonnées du point dans son

repère personnel par la méthode ConvertitPointAller() de Crepere. Cette méthode peut aussi convertir les coordonnées d'un vecteur si son dernier argument est égal à zéro.

- Dans la fonction vitesse, retournant un vecteur, on fera changer ses coordonnées vers le repère parent en appelant la méthode ConvertitVecteurRetour() au préalable. Cette fonction peut aussi s'utiliser en mode point (dernier argument non nul).

Les deux méthodes précédentes utilisent TournerRepere, TranslaterRepere, Dilater.

- TournerRepere convertit les coordonnées d'un point pour une rotation de repère d'angle a (en rad) entre l'axe X et x l'autre angle de rotation étant nul, les deux repère ayant même centre.
- La méthode TranslaterRepere convertit les coordonnées d'un point d'un repère vers un autre repère dont les axes sont parallèles au premier mais dont le centre est obtenu par translation.
- La méthode Dilater effectue le produit scalaire entre le vecteur V et le vecteur E.
  Le vecteur résultat est obtenue en dilatant ses composantes avec les 3 réels contenus dans le vecteur E.
- La fonction décalage provoque sur une vecteur de taille 3 un décalage à gauche ou à droite de 1 unité, cette méthode est utilisée uniquement en interne.
- La méthode ProjectionRadiale est destinée à être utilisée par la méthode VitesseRadiale d'un IModèle. (Cette dernière appelera d'abord la méthode Vitesse() pour obtenir le vecteur vitesse au point demandé, puis le projettera sur l'axe x à l'aide de ProjectionRadiale).

6. Précisions sur quelques méthodes

A. Densité, Vitesse ou PointCorrect dans CDensite, CVitesse et CGeometrie

Ces classes contiennent chacune une loi par exemple la classe CdensiteExponentielle implémente une loi de densité exponentielle.
Elles étendent CParamNum et implémente les interfaces IGeometrie, ILocalisable.
Les classes CDensite : contiennent les lois de densité.
Les classes CGeometrie : contiennent les lois de géométrie.
Les classes CVitesse : contiennent les lois de vitesse.
Si on veut ajouter une nouvelle loi il suffit de créer une de ces trois types de classe.
Nous détaillerons les opérations à effectuer pour faire ces ajouts dans le paragraphe suivant.

B. Ajout de nouvelles lois

1. Modifications à faire dans CmodeleComplet :

- Incrémenter le champs NBLOIS.

- Ajouter à la fin de TYPELOIS le numéro indiquant le type de la loi à ajouter, par exemple 2 si on veut ajouter une nouvelle géométrie : CgeometrieNouvelle.

- Ajouter une étiquette identifiant la loi avec un numéro suivant immédiatement celui de la dernière étiquette par exemple :

public final static int LG_NOUVELLEGEOMETRIE = 15;

- Modifier la méthode AjouterLoi :

Ex : on rajoute dans le switch case :

```
case LG_NOUVELLE LOI :
    loi = new CGeometrieNouvelle(this, 1);
    break;
```

2. Ecrire une nouvelle classe ex : CgeometrieNouvelle

- Définir les 2 static NomParam et Val.
- Mettre dans le constructeur le nom de la loi, le nombre de paramètre, passer NomParam et Val. Puis utiliser SetV pour récupérer les valeurs.

La syntaxe utilisé est détaillé dans les classes déjà écrites : ex CGeometrieExponentielle.

- Dans la fonction principale (ex PointCorrect pour une loi de géométrie) on récupère les variables à l'aide de GetV.

3. Modifier Traduit.java et Ilocalisable.java

Ces deux fichiers se trouvent dans le répertoire userinterface.
Ces modifications ont pour but de mettre à jour les noms utilisés dans l'applet pour la traduction anglais-francais.

C. Intensité

L'intensité correspond en fait à la densité pondérée par la vitesse radiale. Intensité() fera donc appel aux deux méthodes correspondantes pour renvoyer son résultat. Cette méthode n'est peut être pas utile ; nous verrons plus tard s'il convient de la conserver.