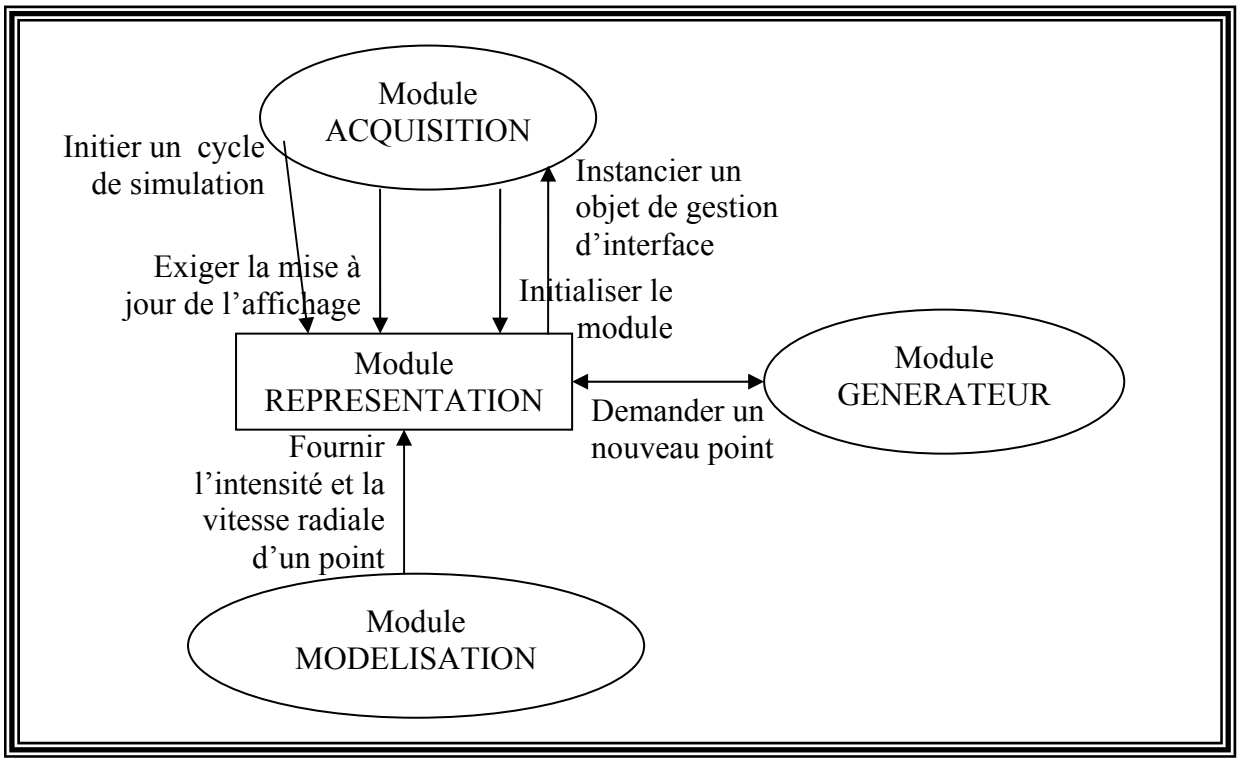
Description du module REPRESENTATION

1. Description du rôle du module

Le module REPRESENTATION doit assurer la mise à jour des différentes représentations graphiques de la simulation (exclusivement les graphes et autres représentations de la simulation en cours, et non pas la gestion de l'interface graphique proprement dite qui est effectuée par le module INTERFACE/ACQUISITION).
Il devra notamment implémenter les calculs et stocks de données nécessaires à cette mise à jour périodique (celle-ci ne s'effectuera pas en effet à chaque cycle de simulation, mais seulement sur l'impulsion du module INTERFACE/ACQUISITION).

2. Echanges avec les autres modules

A. Représentation visuelle



B. Echanges avec le module ACQUISITION

1. Phase d'initialisation

Le module REPRESENTATION devra être initialisé par le module ACQUISITION, c'est-à-dire que ce dernier module est chargé d'instancier les objets de REPRESENTATION, selon la méthode décrite dans la synthèse correspondant au module INTERFACE/ACQUISITION.
Durant cette phase d'initialisation, le module REPRESENTATION instanciera des objets de gestion d'interface à la demande du module ACQUISITION.

2. En fonctionnement normal

A la demande du module INTERFACE/ACQUISITION, un nouveau cycle de simulation est commencé dans le module, qui consiste à acquérir un nouveau point, puis à mettre à jour la base de données interne en accord avec les informations connues en ce point.
De même, sur demande du module INTERFACE/ACQUISITION, les différentes représentations graphiques seront mises à jour à l'aide de la base de données interne du module. (Cette mise à jour n'est pas faite dans le cycle de simulation pour des raisons de performance. Cf. note 1 §6)

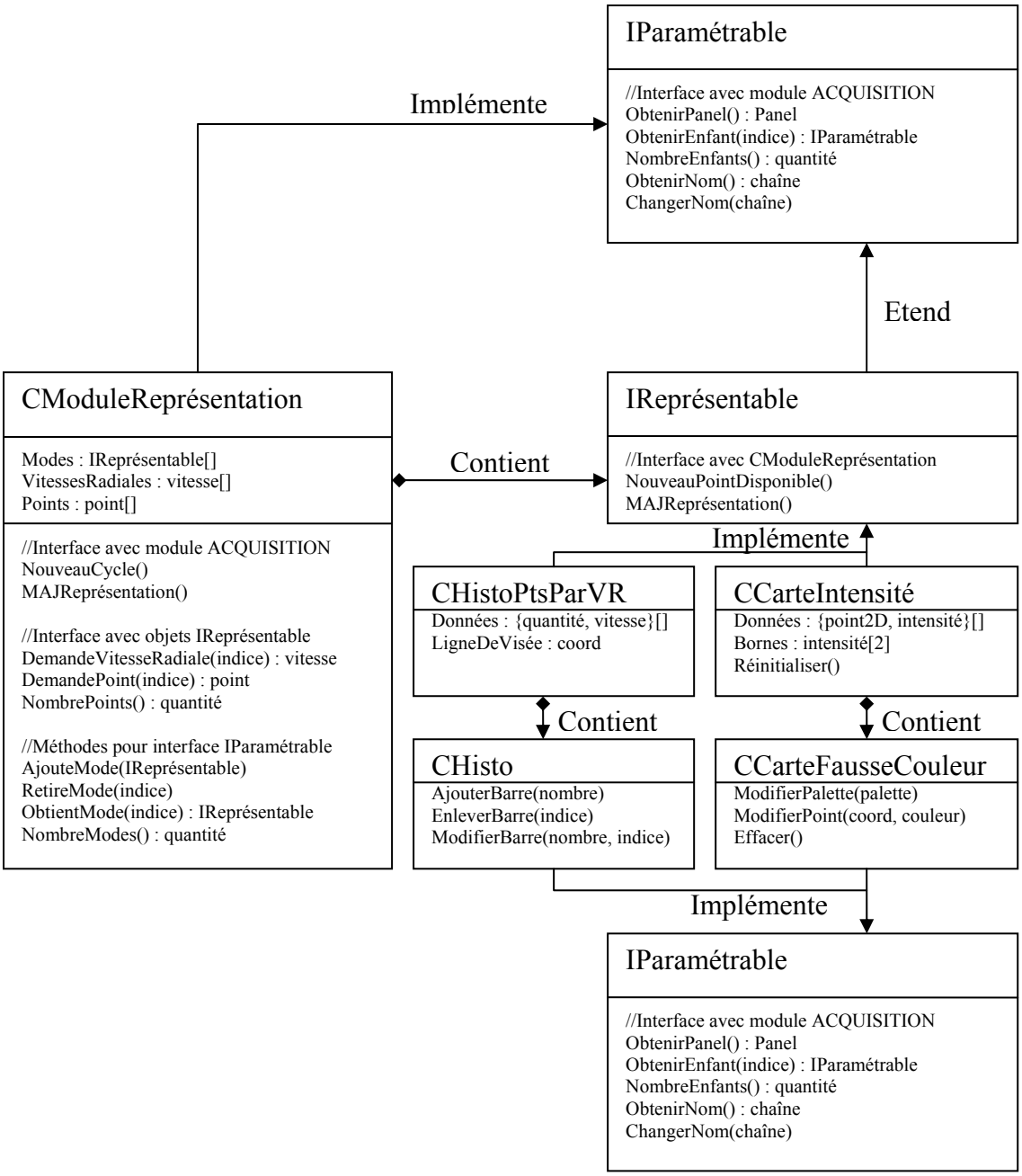
C. Echanges avec le module GENERATEUR

A chaque cycle de simulation, le module REPRESENTATION exigera du module GENERATEUR de fournir un nouveau point à l'intérieur du modèle simulé.

D. Echanges avec le module MODELISATION

A chaque cycle de simulation, il sera demandé au module MODELISATION la vitesse radiale au point fourni par le module GENERATEUR.

3. Diagramme de classes du module



4. Description des classes et interfaces

A. La classe CModuleRepresentation

Elle sera instanciée par un unique objet, qui sera le principal objet du module, créé à l'initialisation de l'applet.

1. Méthodes utilisées par le module ACQUISITION

- NouveauCycle() est la méthode qui sera utilisée par le module ACQUISITION pour signifier à l'objet qu'il doit initier un nouveau cycle de simulation.
- MAJRepresentation() provoque la mise à jour de tous les objets implémentant l'interface IReprésentable, contenus dans la table Modes

2. Méthodes utilisées par les objets contenus dans Modes

- DemandePoint(indice) permet d'obtenir le point du cycle de simulation courant pour un indice nul, sinon du indice^{ème} précédent cycle.
- DemandeVitesseRadiale(indice) permet d'obtenir la vitesse du point à l'indice indiqué. Cette vitesse est effectivement calculée une seule fois pour chaque point, au premier appel d'un des objets IReprésentable.
- NombrePoints() fournit le nombre de points déjà calculés au cours de la simulation.

3. Méthodes pour l'interface IReprésentable

- AjouteMode(IReprésentable) ajoute une représentation à la table Modes
- RetireMode(indice) retire la représentation identifiée par sa clé unique.
- ObtientMode(indice) permet d'obtenir la indice^{ème} représentation contenu dans Modes
- NombreModes() renvoie le nombre dereprésentations en cours

B. L'interface IReprésentable

C'est l'interface commune à toutes les représentations possibles. Pour l'implémenter, il faut définir les méthodes suivantes :

- NouveauPoint() permet de signaler à l'objet l'arrivée d'un nouveau point à ajouter à la base de données de la représentation.
- MAJRepresentation() signale que le module ACQUISITION requiert la mise à jour des représentations graphiques présentées à l'utilisateur.

C. Des exemples de représentations

1. La classe CHistoPtsParVR

Cette classe doit permettre d'afficher un histogramme du nombre de points par vitesse radiale pour une ligne de visée donnée (par ses coordonnées x, y dans la projection de l'objet simulé). Cette classe implémente l'interface IReprésentable ; c'est-à-dire que lorsqu'un nouveau point est disponible, elle incrémente la case correspondant à la vitesse de ce point, s'il fait partie de la ligne de visée sélectionnée. Lorsque la méthode MAJRepresentation() est appelée, ces changements sont répercutés sur l'objet de la classe CHisto, contenu dans CHistoPtsParVR.

2. La classe CCarteIntensité

Elle doit permettre d'afficher une carte en fausses couleurs de la projection de l'objet simulé, où la couleur représente l'intensité totale, ou pour une vitesse radiale donnée. Elle implémente l'interface IReprésentable. L'ajout d'un nouveau point consiste à ajouter la vitesse radiale aux coordonnées du point dans la projection, pour l'intensité totale ; dans le cas où l'on veut

l'intensité pour une vitesse radiale donnée, on ajoute la vitesse radiale uniquement si elle est à l'intérieur de bornes fixées par l'utilisateur.
A la demande, cette classe répercutera ces mises à jour dans l'objet qui gère le dessin effectif de la carte, de classe CCarteFausseCouleur.

D. Des exemples de classes graphiques

1. La classe CHisto

Elle doit gérer le dessin d'un histogramme.

2. La classe CCarteFausseCouleur

Elle gère le dessin d'une carte en fausses couleurs.

E. L'interface IParamétrable

Cette interface est rapidement présentée ici, mais est définie dans le module ACQUISITION. Elle doit être implémentée par tous les objets du module, car elle définit toutes les méthodes nécessaires au module ACQUISITION pour gérer leur configuration et leur présentation dans l'interface.

5. Détails ou problèmes éventuels à ne pas oublier

A. Séparation calcul/affichage

L'affichage systématique d'un nouveau point à l'écran risque de provoquer des ralentissements importants dans les calculs étant donné la latence élevée les opérations d'E/S. Une mise à jour périodique de l'affichage, de période quelques dizaines ou quelques centaines de cycles de simulations pourrait régler ce problème dans un premier temps. Si cela ne s'avérait pas suffisant, nous pourrons également utiliser les capacités multithread de Java, qui permettront d'exécuter le mise à jour graphique, et les calculs dans deux tâches différentes (ce qui signifie qu'elles n'auront plus à être synchronisées.)

B. La gestion de l'interface

Le module est effectivement chargé d'instancier des objets pour l'interface, des panels, qui permettront au module ACQUISITION de fournir à l'utilisateur une interface spécifique à chaque objet du module REPRESENTATION. Ces objets, ainsi que les méthodes implémentant l'interface IParamétrable seront cependant implémentés dans le cadre du module ACQUISITION.

C. Le stockage des points

Il est prévu de stocker tous les points calculés, ainsi que les vitesses radiales associées qui auront été nécessaires, ceci afin de permettre l'ajout de représentations en cours de simulation. Si cela s'avérait irréalisable, on pourrait réduire le stockage aux quelques dizaines de points

précédents, afin d'optimiser les calculs pour certaines représentations, ou même supprimer ce stockage.

D. Les cycles de simulation

Ils sont à l'initiative du module ACQUISITION. Celui-ci donne en quelque sorte son accord à la poursuite de la simulation en appelant la méthode NouveauCycle(). Ceci lui permet d'interrompre de fait la simulation en ne l'appelant pas. Au cours de chaque cycle, l'objet CModuleReprésentation appelle la méthode NouveauPoint() de chacun des objets de la table Modes.

6. Découpage des tâches

Suivent les axes de travail de ce module :

- programmation de la classe CModuleReprésentation
  - évaluer l'utilité d'un stockage des points (oui, non, combien), et éventuellement le programmer
  - programmer les 3 groupes de méthodes (cf diag. de classes)
- programmation de la représentation en histogramme discutée plus haut
  - CHisto
  - CHistoPtsParVR
- programmation de la représentation en fausses couleurs
  - CCarteIntensité
  - CCarteFausseCouleur
- Fonctions de test