I-Outils Mathématiques

Vecteur

Notre classe Vecteur a comme seul attribut un vector de double, ce qui permet d'avoir une flexibilité sur sa dimension (qui n'est pas possible avec un array). Cet attribut est privé ; on offre à l'utilisateur tout ce qu'il lui faudra comme opérations mathématiques nécessaires dans une simulation physique. On a donc implémenté l'addition, la soustraction, la multiplication par un scalaire, le produit (scalaire et vectoriel), la direction et la norme euclidienne des vecteurs tel qu'elles existent dans l'espace vectoriel R^n. Ces méthodes nous permettent de raccourcir des calculs fastidieux qui adviennent dans les méthodes de la classe Toupie.

Matrice

Notre classe Matrice a comme seul attribut un array d'arrays de doubles (de taille 3 par 3) ; on utilise ici des arrays car seul les matrices de dimension 3 par 3 sont utile dans une simulation physique. Cet attribut est prive ; on offre à l'utilisateur tout ce qu'il lui faudra comme opérations mathématiques nécessaires dans une simulation physique. On a donc implémenté l'addition, la soustraction, la multiplication (par un scalaire, entre 2 matrices et entre une matrice et un vecteur), la transposée et l'inverse tel qu'elles existent dans l'espace vectoriel des matrices 3 par 3. On utilise les Matrices pour changer de repère et pour représenter des tenseurs d'inertie d'une toupie.

Integrateur

Notre classe Integrateur représente un processus mathématique permettant de résoudre une équation différentielle numériquement. On l'utilise pour pouvoir approximer les solutions des équations du mouvement de nos objets physiques, ce qui nous permet de dessiner l'évolution de ces objets dans le temps. La classe Integrateur est abstraite, car il n'existe pas d'algorithme général pour résoudre une équation différentielle quelconque. En effet, dans le cadre de notre projet, on utilise seulement des intégrateurs numériques permettant de résoudre des équations différentielles ordinaires du deuxième ordre données explicitement (et qui ne dépendent pas du temps). Spécifiquement, on a développé les méthodes d'Euler-Cromer, de Newmark et de Runge-Kutta, qui correspondent chacune à une sous-classe différente d'Integrateur. On offre donc à l'utilisateur la possibilité, en donnant un pas de temps d'intégration et le Systeme qu'il veut simuler, de calculer l'évolution du Système physique en temps réel. L'utilisateur pourrait aussi comparer la performance de différents Integrateurs en changeant la méthode d'intégration (et donc l'Integrateur) de son Système.

II-Objets :

Dessinable

Notre classe Dessinable représente tout objet qu'on puisse crée qui peut être dessiné d'une manière ou d'une autre (à travers un SupportADessin). Cette classe est abstraite car il n'existe pas une forme générale concrète d'un Dessinable. Être un Dessinable signifie simplement avoir la propriété d'être dessine, et donc tout objet qu'on croit possible de dessiner est une sous-classe de Dessinable. On offre donc à l'utilisateur la possibilité de pouvoir dessiner tous nos Dessinables (ayant préalablement choisi un SupportADessin). Les

sous-classes de la classe Dessinable sont les divers Toupies qu'on a implémenté dans notre projet.

Toupie

Notre classe Toupie représente un modèle pour pouvoir simuler une toupie quelconque, avec la possibilité de pouvoir observer l'évolution de la toupie de divers façon (une simulation textuelle (TextViewer), graphique (VueOpenGL), ou un graphe des grandeur physiques de la Toupie (QCustomPlot)). L'objet Toupie a (d'une manière générale) comme attribut une masse, des coefficients d'inertie, un Vecteur de paramètres et un Vecteur contenant les dérivées de ces paramètres. Dans le cadre le plus général, un solide indéformable peut être modélise par 6 données : 3 coordonnées pour la position d'un point quelconque du solide (normalement choisi comme le centre de masse) décrivant la translation du solide et 3 angles correspondant aux angles d'Euler décrivant la rotation du solide. Une toupie générale comporte aussi un point de contact et une distance entre son point de contact et son centre de masse. L'ensemble de ces données permet de nous donner toute les libertés pour simuler une toupie (qui ne glisse pas soumit a seul son poids). Une Toupie comporte aussi un (pointeur vers un) SupportADessin (en tant qu'objet Dessinable) qui permet à l'utilisateur de choisir la manière dont il veut dessiner la Toupie. On offre aussi la possibilité à l'utilisateur de choisir une GrandeurPhysique (attribut de type enum d'une toupie) qui fera varier la couleur de la toupie (en fonction de cette grandeur) en simulation graphique. L'utilisateur peut aussi choisir la possibilité de tracer le centre de masse (à travers un attribut booléen de la Toupie) dans une simulation graphique ou textuelle. Pour l'affichage graphique de la trace du centre de masse, la Toupie comporte aussi une Mémoire (classe décrite plus tard). Nous offrons à l'utilisateur une multitude de méthodes permettant de calculer certaine grandeur physiques intéressantes de la Toupie (moment cinétique, énergie, etc.). L'utilisateur pourrait donc s'intéresser a l'évolution de ces grandeurs dans le temps (et par exemple noter l'invariance de certaines d'entre eux).

On a implémenté 5 objets plus spécifiques qui permettent de voir certaines applications intéressantes de notre cadre général d'une Toupie. Un Solide de Révolution, modélise avec une liste de rayons, une hauteur, et une masse volumique, permettant à l'utilisateur de créer une forme quelconque d'une Toupie et d'observer son évolution d'une manière graphique ou textuelle. Un Cône Simple, modélise par une hauteur et un rayon, le cadre le plus simple d'une Toupie non-glissante soumise à seul son poids. Une Toupie Chinoise, modélise par une sphère d'un rayon donne tronquée d'une hauteur donne ; le point de contact de cette Toupie ce déplace. Un Pendule, modéliser par une longueur et une masse, cet objet constitue un pendule simple. Une Masse qui Tombe, représentant simplement un point matériel en chute libre.

Systeme

Notre classe Systeme est constitué d'une collection hétérogène de Toupies et d'un Integrateur permettant de simuler leur évolution. Un Systeme représente l'entièreté du système physique que l'utilisateur pourrait considérer pendant une simulation. L'utilisateur peut ajouter autant de Toupie qu'il veut au Systeme, et comparer leurs comportements différents. Notre Systeme n'est pas un Dessinable (ce n'est pas un objet physique, mais plutôt un repère dans lequel on choisit de dessiner nos Toupies divers). On offre à

l'utilisateur la possibilité d'afficher le contenu du système, de le dessiner, et de tracer les centres de masse des Toupies qu'il contient.

SupportADessin

SupportADessin permet à toutes toupies de se dessiner mais aussi de tracer le déplacement de leur centre de masse. Cependant cette classe est abstraite car il n'existe pas de forme concrète de SupportADessin. En effet, il est impossible de dessiner une toupie de manière universelle.

TextViewer

Cette classe hérite de SupportADessin et contient un flot dans lequel il est possible d'afficher nos données. TextViewer nous permet de dessiner une toupie de manière textuelle c'est-à-dire, de pouvoir afficher les paramètres et les dérivées de paramètres (d'un Dessinable) un à un. De plus, il est aussi possible d'afficher les coordonnées du point G à chaque instant grâce aux méthodes trace.

III-Graphisme :

VueOpenGL

Dans un premier lieu, cette classe hérite de SupportADessin car elle permet de représenter de manière graphique un Dessinable. En effet, VueOpenGl nous sert à créer le dessin de toutes les Toupies et de la trace de leur centre de masse mais aussi, d'un « sol » et d'un système d'axe cartésien.

Cependant, considérer VueOpenGL comme permettant seulement le dessin des Toupies seraient un peu réducteur car il est aussi possible de construire tout type de formes (dont des sphères approximées) et appliquer des shaders qui vont permettre de donner des textures aux formes dessinées.

Enfin, VueOpenGL représente le point de vue de l'utilisateur qui lui permet d' observer les objets dessinés. Il sera possible de contrôler ce point de vue grâce à la classe suivante (GLWidget). VueOpenGl offre aussi à l'utilisateur la possibilité d'avoir une vue tangentielle à une Toupie.

GLWidget

Nous utilisons la classe GLWidget comme étant la classe gérant toutes manipulation propres à la fenêtre d'affichage. Cette classe pourra donc permettre à l'utilisateur de redimensionner la fenêtre, de se déplacer dans l'espace (à partir d'un point de vue) et de contrôler aussi la dimension de temps.
Enfin, il est possible d'afficher des objets (préalablement dessinés dans VueOpenGL) dans cette fenêtre d'affichage.

QCustomPlot

Nous avons trouvé un programme permettant de remplacer gnuplot sur Qt. En effet, ce programme permet plus de libertés sur l'affichage. On l'a utilisé pour comparer un ConeSimple à un Solide de Revolution (de forme conique) mais aussi pour afficher nos invariants du mouvement d'une toupie conique avec différents intégrateurs.

Pour utiliser ce programme nous avons dû créé une fenêtre d'affichage (un QWidget) avec des boutons permettant de choisir les paramètres désirés.

Autres

Les classes GLSphere et GLSphereCoupe nous permette d'approximer une sphère et une sphère tronque utilisé dans VueOpenGl dans l'affichage graphique de toupie.

IV-Utilitaires

Constant

Ce fichier comporte des constantes et des méthodes (« modulo_2pi » et « affiche_erreurs ») qui sont utilisé dans la plupart de notre programme. Cela nous permet donc d'éviter toutes redondance sur la déclaration de nos constantes.

Mémoire

La classe mémoire comporte un tableau (deque) de valeurs et une taille fixe. Une fois cette taille depasser, le tableau se vide de ses éléments les plus anciens. On utilise cette classe comme attribut d'une Toupie, ce qui nous permet de tracer la position du centre de masse dans un affichage graphique.