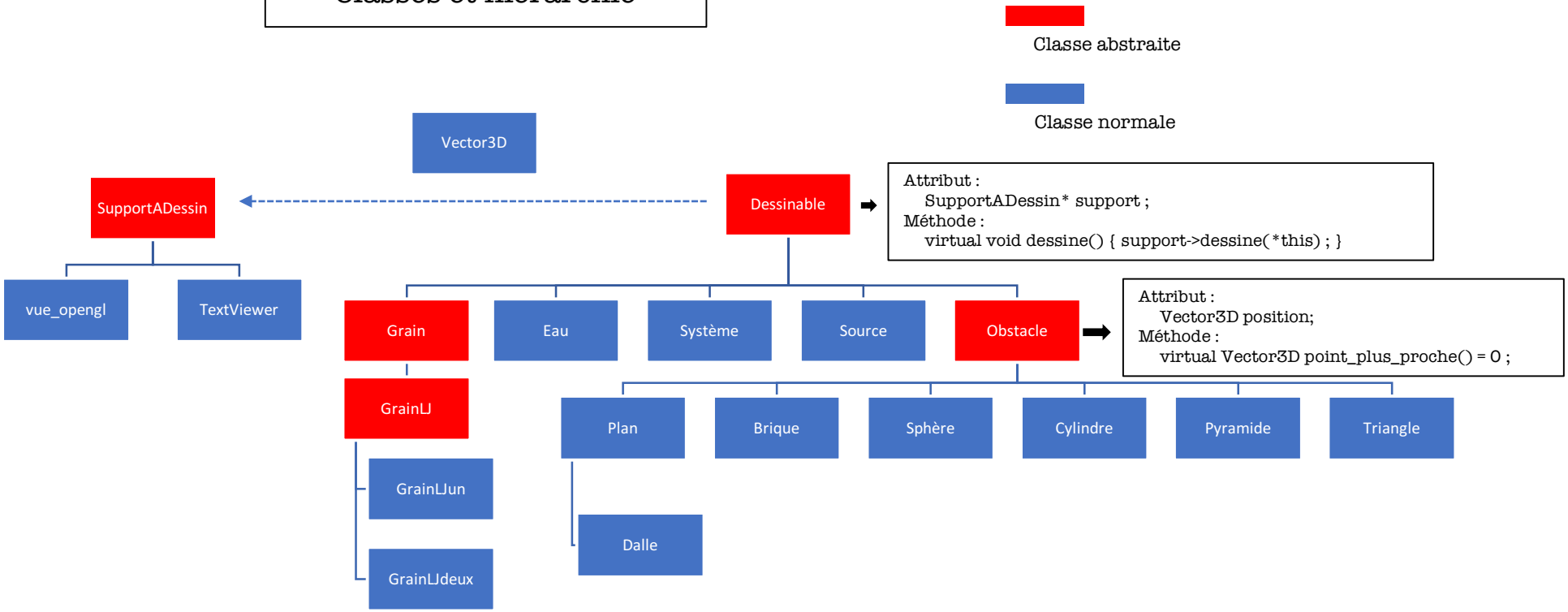
Classes et hiérarchie



Conception :

Vector3D:
La première classe que nous avons créé c'est la classe Vector3D. Elle permet de faciliter la manipulation des vecteurs pendant la suite du projet. Elle a 3 attributs : les composantes du vecteur en x, y et z.
Elle n'a pas de sous-classes.

SupportADessin (abstraite):

Cette classe permet de gérer le dessin des différents éléments de notre programme à travers la classe *Dessinable* (expliqué plus bas). Nous avons deux *SupportADessin,* le *Textviewer* (affichage en mode texte) et *vue_opengl* (affichage graphique avec Open_GL).

Dessinable (abstraite):

La super-classe qui désigne tous les éléments du programmer qui peuvent être dessinés. Ses sous-classes principales sont *Obstacle, Grain* et *Système*. Nous avons en plus décidé de dessiner la *Source*, et des volumes d'*Eau*. Tous les objets de la classe *Dessinable* ont comme attribut un pointeur sur un *SupportADessin*. Ils ont aussi une fonction virtuelle *dessine()*, qui va appeler la fonction une autre fonction *dessine* de cette façon : `Dessinable ::dessine() { support->dessine(*this) ; }`

Grain : public Dessinable :

Une des classes les plus importantes du programme. Un grain est défini par sa position, sa vitesse, son rayon, la somme des forces qui sont appliquées dessus (hors poids) et le milieu dans lequel il se trouve (eau ou air). Tous les grains ont aussi une densité, qui est donc un attribut mis en *static*. On a défini un *namespace* dans le fichier grain.h, dans lequel on a stocké des valeurs de constantes physique nécessaires pour les calculs de force. La méthode principale de cette classe est *move()*, qui permet de faire bouger le grain pendant un temps dt selon les lois de Newton.

GrainLJ : public Grain :

Cette classe a les mêmes attributs que la classe Grain, mais contient des méthodes qui permettent de calculer les forces dues à l'interaction entre le grain et des obstacles ou d'autres grains.

GrainLJun : public GrainLJ, GrainLJdeux : public GrainLJ :

Ceux-ci sont deux sous-classes de GrainLJ, et diffèrent que par la façon dont elles sont dessinées (par exemple de couleurs différentes).

Obstacle : public Dessinable :

Ceci est une super-classe, car ils existent de nombreux différents obstacles qui héritent de cette classe. Comme attribut elle contient juste une position. Elle a une fonction virtuelle pure *point_plus_proche()* qui est redéfinie pour chaque obstacle du programme.

Plan : public Obstacle :

Un plan est défini par une position et un vecteur normal

Dalle : public Dalle :

Une dalle est un plan avec des dimensions fini, i.e. une portion de plan. Elle a donc une longueur et largeur.

Brique : public Obstacle :

Une brique est un parallélépipède rectangle, définie par un vecteur normal, une hauteur, longueur et largeur. Le calcul du point le plus proche ce fait un calculant le point le plus proche sur chacune des faces et en choisissant le minimum d'entre eux.

Sphère : public Obstacle :

Une sphère est un obstacle défini par sa position et un rayon.

Cylindre : public Obstacle :

Un cylindre est un obstacle défini par un rayon, un vecteur normal et une hauteur.

Triangle : public Obstacle :

Un triangle est un obstacle mais n'est jamais utilisé indépendamment, mais sert à construire la pyramide (voir Pyramide). Il a comme attributs ses trois sommets. Elle contient une méthode qui permet de vérifier si un point se trouve à l 'intérieur du triangle.

Pyramide : public Obstacle :

La classe Pyramide permet de construire un seul type spécifique de pyramide, à savoir une pyramide à 5 faces (4 triangles et un base). Elle a comme attribut les dimension et vecteurs qui décrivent la base, une hauteur et un sommet (pointe de la pyramide). Comme la Brique, on trouve le point le plus proche en calculant la distance par rapport à chaque face.

Source : public Dessinable :

Une source permet de créer des grains. Elle est caractérisée par une position, un débit, un grain de référence, une vitesse initiale et deux écarts types, un pour la vitesse et un pour le rayon des grains créés. Le grain de référence permet de créer un nouveau grain grâce à la copie polymorphique. La vitesse du nouveau grain est après adapté à la vitesse initiale passée en argument, avec une déviation aléatoire déterminée en fonction de l'écart type (grâce au générateur aléatoire). La création de grains ce fait pendant un temps donné dt, et le nombre de grains créés dépend du débit.

Eau : public Dessinable :

Un volume d'Eau est une partie du système où le milieu est défini comme étant de l'eau plutôt que de l 'air. Il existe trois types de volumes d'Eau différents : un niveau d'eau universel (tout le système en dessous d'une certaine hauteur se trouve sous l'eau), une « couche » d'eau (une surface d'eau infinie, mais avec une profondeur donnée), ou bien encore le volume d'eau délimité (décrit par une profondeur, longueur et largeur). Un grain qui rentre dans un volume d'eau subira d'autres forces de frottement liquide et une autre poussé d'Archimède.

Système : public Dessinable :

Le Système est la classe principale de notre programme. Elle contient tous les grains, obstacles, sources et volumes d'eaux (sous formes de tableaux dynamiques de pointeurs). Les grains du système sont stockés par un principe de quadrillage un utilisant une *map*, c'est –à-dire chaque grain se trouve dans une case d'une taille fixe tel que deux grains distants de plus que 2 cases n'interagissent pas. Pour faire évoluer le système, la classe a une méthode *evolue()*, qui calcule toutes les forces appliquées sur les grains du système et les fait bouger en fonction de ces forces.
