Ce fichier présente les différentes classes et les liens entre elles qui composent notre projet de simulation de nuages dans une région montagneuse.

Les classes :

Systeme : C'est la classe s'occupant de la simulation. Pour cela, il y à des pointeurs dans **Systeme** sur : **terrain**, **ChampPotentiels**, **nuages**, **chaine_mont**, **mult_riv**. Cela permet au **Systeme** de gérer tout ce qui se passe dans la simulation (les liens entre les classes sont gérés par le **Systeme**) Il y a en plus un vector<**dessinable***> dans lequel sont copiées les adresses des éléments qui seront effectivement affichés lors de l'affichage du **Systeme** qui est lui même **dessinable**.

Systeme3D : Cette classe hérite de **Systeme** et permet de séparer les parties graphique et simulation. Elle permet de pointer sur équivalents 3D des classes pointées dans **Systeme**. Afin que la méthode dessine utilise celles-ci par polymorphisme lors de la simulation 3D.

dessinable : La classe virtuelle pure qui permet d'afficher ses héritiers.

ChampPotentiels : Cette classe permet de calculer le vent dans la simulation. Elle possède un vector<vector<vector<**Potentiel**>>> en attribut qui stocke les information nécessaires au calcul du vent. Elle nécessite une **chaine_mont** pour s'initialiser. Elle hérite de **dessinable**.

Potentiel : Cette classe ne possède que deux **Vecteur2D**. L'un représente le potentiel et l'autre le laplacien utilisés pour le calcul du vent dans le complément mathématique.

Vecteur2D : Cette classe code une représentation de vecteurs de dimension 2 avec les opérations vectorielles de base.

gros_chaine_mont : La classe viruelle pure qui hérite de **dessinable** et est la base de chaque montagne. La création des rivière stocke dans **gros_chaine_mont** la position des rivières et lac afin de pouvoir renvoyer l'altitude

chaine_mont : C'est la classe qui gère les chaînes de montagnes dans le programme. Elle hérite de **gros_chaine_mont** et possède un collection hétérogène de **gros_chaine_mont** qui lui permet de gérer un ensemble de montagnes grâce au polymorphisme.

montagne : C'est la montagne gaussienne qui peut être un des plus petits éléments de **chaine_mont**.

montagne_lue : Cette classe permet de lire une montagne dans un fichier. Ou de générer aléatoirement une montagne un utilisant l'algorithme de diamant-carré si aucun fichier n'est spécifié cette montagne est enregistrée dans random.mont afin de pouvoir être réutilisée. Elle peut aussi être un des plus petits éléments de **chaine_mont**.

nuages : Cette classe s'occupe de l'évolution et de la création des **CubedAir**. Elle se construit à partir d'un **ChampPotentiels** initialisé pour calculer la répartition des nuages à partir du vent, de la température globale, de la pression etc. On peut aussi lui passer une **mult_riv** pour initialiser l'augmentation d'humidité locale due aux rivières. Elle hérite de **dessinable**.

nuage3D : La classe qui gère le dessin des nuages, elle hérite de **nuages** et affiche : les nuages, la température, la pression en OpenGL.

CubedAir : C'est la classe qui conserve les informations physiques locales nécessaires à la simulation des nuages. Elle permet aussi de faire une grande partie des calculs locaux. Elle hérite de **dessinable**.

terrain : La classe qui gère la végétation et le type de sol. Elle hérite de **dessinable** et possède une référence sur une **chaine_mont** et sur des **nuages** qui lui permettent de s'initialiser et d'évoluer correctement.

terrain3D : La classe qui gère le dessin du sol et de la montagne, elle hérite de **terrain** et affiche la montagne correctement colorée en fonction du sol. Son dessin est codé pour OpenGL.

mult_riv : La classe viruelle pure qui hérite de **dessinable** et est la base de chaque rivière.

rivers : C'est la classe qui gère les rivières dans le programme. Elle hérite de **mult_riv** et possède un collection hétérogène de **mult_riv** qui lui permet de gérer un ensemble de rivières grâce au polymorphisme.

rivers3D : La classe qui gère le dessin de la collection **riviere**, elle hérite de **rivers** et affiche les **riviere3D**.

riviere : C'est une rivière qui est le plus petit élément de **rivers**. Elle stocke aussi les lacs.

riviere : C'est une rivière qui est le plus petit élément de **rivers**. Elle stocke aussi les lacs. Elle pointe sur une **chaine_mont** pour en connaître l'altitude.

riviere3D : La classe qui gère le dessin des rivières et des lacs, elle hérite de **riviere** et affiche la rivière et le lac en OpenGL en fonction de l'altitude de la **chaine_mont**.

Application : La classe qui permet de gérer la simulation 3D en temps réel. Elle permet la programmation événementielle car c'est l'application pour wxWidgets. Elle possède un pointeur sur un **Systeme3D**, un pointeur sur un timer, un pointeur sur une **FenetreGL** et un pointeur sur une **FeneteControle**.

FenetreGL : La classe qui gère la vue OpenGL. Elle possède un pointeur sur une **VueOpenGL** et gère la fermeture des fenêtres à la fermeture du programme.

VueOpenGL : La classe qui gère le dessin OpenGL. Elle possède une **Camera** et la position de la souris. Elle fait appel dans sa méthode dessine au dessine de **Application** qui appelle le dessine de **Systeme3D**. Cela permet d'afficher en 3D tout ce qui doit l'être et la **Camera** gère le point de vue. Cette classe gère aussi les input clavier et souris.

Camera : La classe qui gère la position du point de vue pour **VueOpenGL**.

FeneteControle : La classe qui gère une fenêtre de contrôle modifiant divers choses (orage, dessin température et pression) en faisant appel à la méthode parametre de l'**Application** .

Parseur : Cette classe permet d'initialiser la simulation 3D à partir d'un fichier. Grâce à elle, il ne faut pas recompiler à chaque changement de paramètres de l'**Application**.

Les namespace :

Physique : Possède les constantes physiques nécessaires à la simulation.

Proprietes : Le namespace qui gère les variables utiles à la simulation.