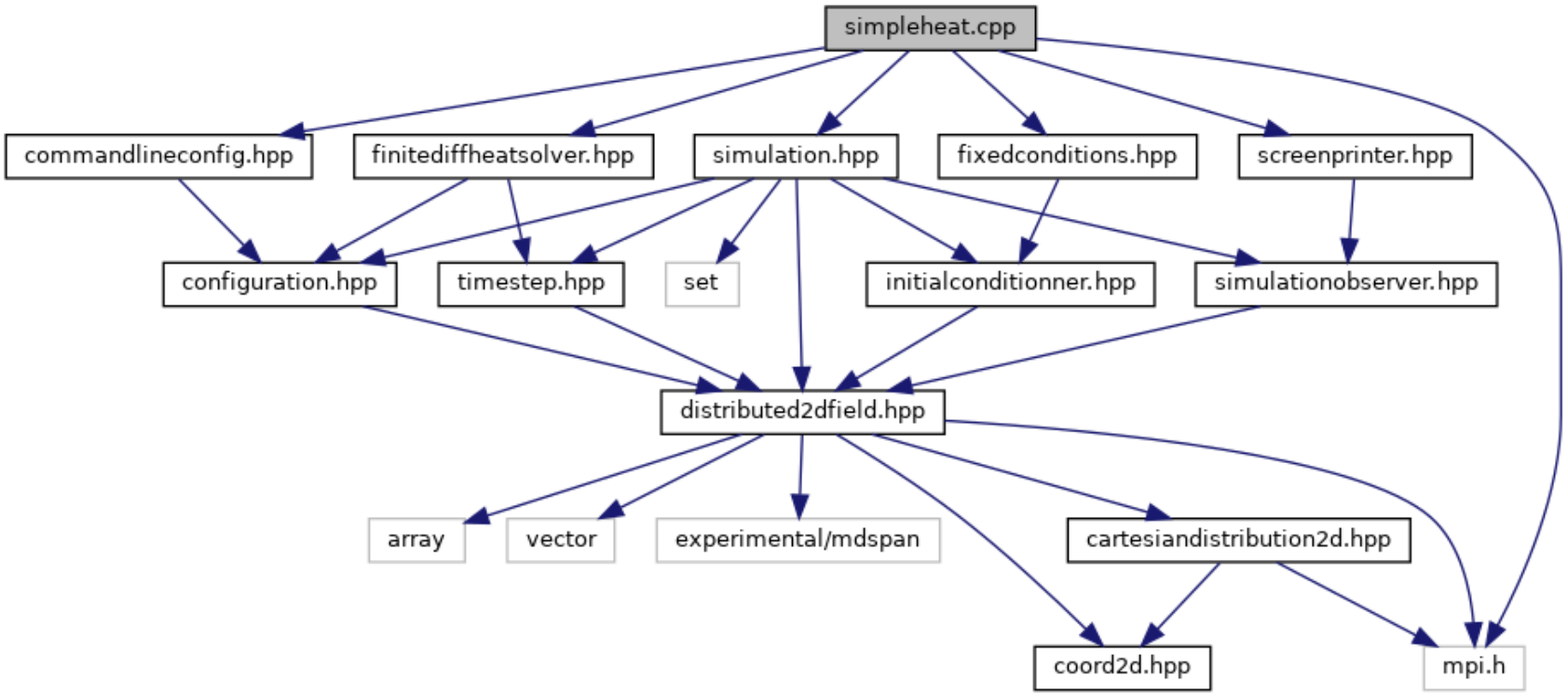
Présentation du projet GLCS

Géni Logiciel pour le Calcul Scientifique

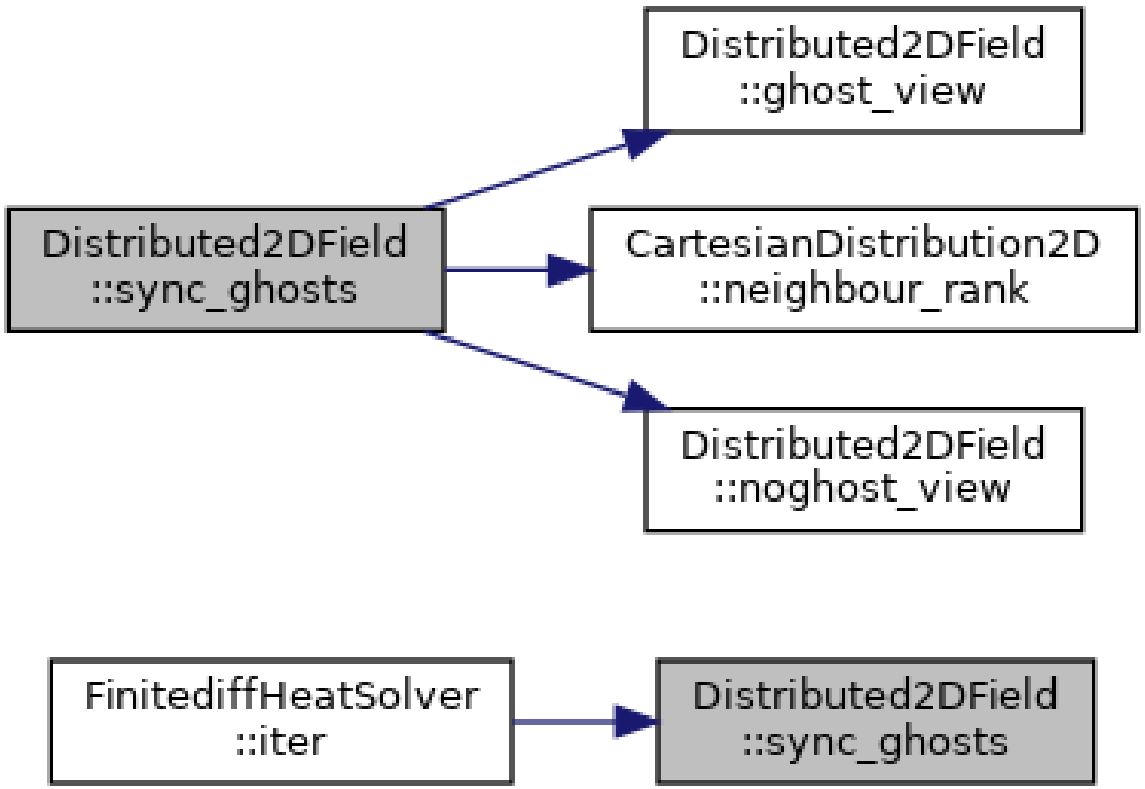
Réalisé par :
Khadimou DIOP
Nawal CHIBANE

28 janvier 2021

Vue globale du code



La fonction Distributed2DField::sync_ghosts



La classe CartesianDistribution2D

CartesianDistribution2D Class Reference

`#include <cartesiandistribution2d.hpp>`

Public Member Functions

	CartesianDistribution2D (MPI_Comm comm, **Shape2D** shape)
Shape2D	**extents** () const
int	**extent** (**Dimension2D** dim) const
Coord2D	**coord** () const
int	**coord** (**Dimension2D** dim) const
const MPI_Comm	**communicator** () const
MPI_Comm	**communicator** ()
int	**neighbour_rank** (**Direction2D** direction)
int	**size** () const
int	**rank** () const

Plan

- Introduction
- Système décriture des données
- Système de configuration
- Post-traitement des données
- Conclusion

Introduction

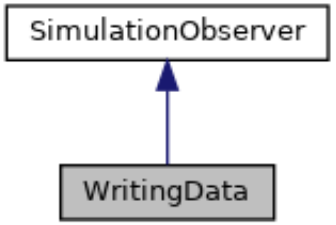
Le code fournis présente des failles :

- Le système d'écriture des données.
- Le système de configuration.

But : Améliorer ces deux systèmes.

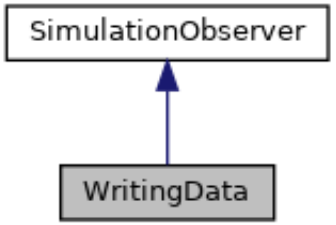
```
#include <writing_hdf5.hpp>
```

Inheritance diagram for WritingData:



[legend]

Collaboration diagram for WritingData:



[legend]

Public Member Functions

void **simulation_updated** (const **Distributed2DField** &data) override

Public Member Functions inherited from SimulationObserver

~SimulationObserver ()=default

The destructor. More...

Exemple d'écriture

```
at t=1 : [
  [ 782.527 245.29 59.0213 10.3906 1.289 0.107 0.005375 0.000125 ]
  [ 972.799 332.173 82.1755 14.4183 1.73963 0.136625 0.00625 0.000125 ]
  [ 972.799 332.173 82.1755 14.4183 1.73963 0.136625 0.00625 0.000125 ]
  [ 782.527 245.29 59.0213 10.3906 1.289 0.107 0.005375 0.000125 ]
]
glcs@ed2a1186d8ce:/data/build$ h5dump my_file.h5
HDF5 "my_file.h5" {
GROUP "/" {
   DATASET "data" {
      DATATYPE  H5T_IEEE_F64LE
      DATASPACE  SIMPLE { ( 4, 4 ) / ( 4, 4 ) }
      DATA {
      (0,0): 782.527, 245.29, 59.0213, 10.3906,
      (1,0): 1.289, 2097.15, 972.799, 332.173,
      (2,0): 82.1755, 14.4183, 1.73963, 2097.15,
      (3,0): 972.799, 332.173, 82.1755, 14.4183
      }
   }
}
}
```

Système de configuration

```
$ mpirun -n 2 ./simpleheat 10 4 8 1 2 0.125 1 1
```

Cette manière de faire présente plusieurs inconvénients :

- Nous n'avons pas d'information sur l'identité des paramètres.
- Obligation de respecter l'ordre défini.
- Difficulté de gérer un grand nombre de paramètres.
- Mauvaise UX.

La solution

- Identification des paramètres
- Gestion des paramètres par défaut
- Choix d'un fichier de configuration

Une option par paramètre

- **nb_iter** : le nombre d'itérations
- **height** : longueur de l'espace de données global.
- **width** : largeur de l'espace de données global.
- **process_height** : la distribution des données selon la largeur.
- **process_width** : la distribution des données selon la longueur.
- **delta_t** : pas de discrétisation temporel.
- **delta_x** : pas de discrétisation spatial selon la longueur.
- **delta_y** : pas de discrétisation spatial selon la largeur.
- **file** : le fichier de configuration.
- **freq** : frequence d'écriture des données.

Choix de l'outil

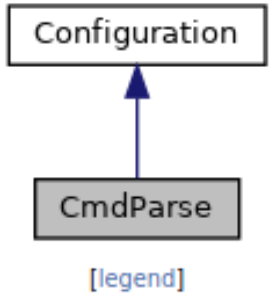
Choix très vague :
Boost, Clara, CLI11, The Lean Mean C++, getopt, Ketopt, args ...

Notre choix : Qt5

- Framework crossplateform développé en C++ pour C++ étendu pour les autres langages.
- Offre des composants GUI et plein d'autres librairies
- QCommandLineParser && QCommandLineOption
- QSettings et la gestion des fichiers dont .INI
- Facilité d'utilisation
- Parfait pour des perspectives d'ajout de GUI au simulateur et visualisation.

Réalisation

Collaboration diagram for CmdParse:



Public Member Functions

	CmdParse (int argc, char **const argv)
int	**nb_iter** () const override
Coord2D	**global_shape** () const override
Coord2D	**dist_extents** () const override
double	**delta_t** () const override
std::array< double, 2 >	**delta_space** () const override
int	**freq** () const override

▸ **Public Member Functions inherited from Configuration**

FIGURE – Classe CmdParse

Logique du code

- Lecture automatique d'un fichier **default.ini**
- Mise à jour des constantes (valeurs par défaut)
- Création d'une option pour chaque paramètre
- Récupération des valeurs (fichier de config, par défaut, options)
- Mise à jour des variables locales

Nouvel usage

- Utilisant les valeurs par défaut

```
$ mpirun -n 2 ./simpleheat
```

- Utilisant un fichier de config

```
$ mpirun -n 2 ./simpleheat --file=../config.ini
```

- Utilisant les option

```
$ mpirun -n 2 ./simpleheat --nb_iter=10 --height=4
--width=8   --process_height=1 --process_width=2
--delta_t=0.125 --delta_x=1 --delta_y=1
```

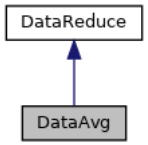
- Combinaison de ces méthodes : la priorité pour les valeurs introduites en option, puis celles du fichier de configuration si présent, puis celles par défaut.

Post traitement des données

Calcul de la moyenne et écriture dans un fichier hdf5

- Interface DataReduce
- Classe DataAvg
- Classe write_avg

Collaboration diagram for DataAvg:



Public Member Functions

	DataAvg (const **Distributed2DField** &data)
double	**average** () const override
Public Member Functions inherited from DataReduce	
virtual	**~DataReduce** ()=default
	The destructor. More...

Public Member Functions

void **write_avg** (const double &avg)

Member Function Documentation

write_avg()

void Write::write_avg (const double & avg)

Conclusion

- Vision sur les applications de simulation (conception & performances)
- Outils et bonnes pratiques du génie logiciel

  Inversion de contrôle, pattern Observer, gestion automatique des versions avec git, génération automatique de la documentation avec Doxygen, utilisation de la technologie de conteneurisation docker, structuration du code (commentaires, indentation, règles de nommage ...)