

PROJET C++ (Jules Roques - Zephirin Faure)

Généré par Doxygen 1.9.1

1 README	1
1.1 Auteurs	1
1.2 Compilation	1
1.3 Documentation	1
1.4 Utilisation	1
1.4.1 Définition de l'univers	2
1.4.2 Configuration	2
1.4.3 Sortie	3
1.5 Pour générer une vidéo à partir des images PNG	3
1.6 Pour générer un GIF à partir des images PNG	3
1.6.1 Tests	4
2 Conception	5
3 TO DO	7
3.0.1 Zeph	7
3.0.2 Jules	7
3.0.3 Reste	7
4 Index hiérarchique	9
4.1 Hiérarchie des classes	9
5 Index des classes	11
5.1 Liste des classes	11
6 Index des fichiers	13
6.1 Liste des fichiers	13
7 Documentation des classes	15
7.1 Référence de la classe Cell	15
7.1.1 Description détaillée	16
7.1.2 Documentation des constructeurs et destructeur	16
7.1.2.1 Cell()	16
7.1.3 Documentation des fonctions membres	17
7.1.3.1 addNeighbour()	17
7.1.3.2 applyForceOnNeighbours()	17
7.1.3.3 clearParticles()	17
7.1.3.4 getCoordinates()	17
7.1.3.5 getDimension()	18
7.1.3.6 getNbNeighbours()	18
7.1.3.7 getNeighbours()	18
7.1.3.8 getUniverse()	19
7.2 Référence de la classe ExternalForce	19
7.2.1 Description détaillée	20

7.2.2 Documentation des constructeurs et destructeur	20
7.2.2.1 ExternalForce() [1/2]	20
7.2.2.2 ExternalForce() [2/2]	20
7.2.3 Documentation des fonctions membres	20
7.2.3.1 applyOn()	20
7.2.3.2 setForceFunction()	20
7.3 Référence de la classe ExternBorderCell	21
7.3.1 Description détaillée	22
7.3.2 Documentation des constructeurs et destructeur	22
7.3.2.1 ExternBorderCell()	23
7.3.3 Documentation des fonctions membres	23
7.3.3.1 applyForceOnNeighbours()	23
7.3.3.2 clearParticles()	23
7.3.3.3 copyParticles()	24
7.4 Référence de la classe FiniteUniverse	24
7.4.1 Description détaillée	27
7.4.2 Documentation des constructeurs et destructeur	27
7.4.2.1 FiniteUniverse()	28
7.4.3 Documentation des fonctions membres	28
7.4.3.1 activateReflexionWithForces()	28
7.4.3.2 addParticle() [1/5]	29
7.4.3.3 addParticle() [2/5]	29
7.4.3.4 addParticle() [3/5]	29
7.4.3.5 addParticle() [4/5]	30
7.4.3.6 addParticle() [5/5]	30
7.4.3.7 applyExternalForces()	31
7.4.3.8 applyInteractionForces()	31
7.4.3.9 applyInternInterractionsForces()	32
7.4.3.10 applyLimitinterractionForces()	32
7.4.3.11 getBounds()	33
7.4.3.12 getLowerBound()	33
7.4.3.13 getoobbehavior()	33
7.4.3.14 getUpperBound()	33
7.4.3.15 handleOutOfBoundsParticles()	33
7.4.3.16 isInBounds()	34
7.4.3.17 reflectOutOfBoundsParticles()	35
7.4.3.18 removeOutOfBoundsParticles()	35
7.4.3.19 setApplyWallsForces()	36
7.4.3.20 setOOBBehavior()	36
7.4.3.21 teleportOutOfBoundsParticles()	36
7.4.3.22 updatePositions()	37
7.4.4 Documentation des fonctions amies et associées	37

7.4.4.1 operator<<	37
7.5 Référence de la classe GriddedUniverse	38
7.5.1 Description détaillée	41
7.5.2 Documentation des constructeurs et destructeur	41
7.5.2.1 GriddedUniverse()	41
7.5.3 Documentation des fonctions membres	41
7.5.3.1 getDimensions()	41
7.5.3.2 simulateStormerVerlet()	42
7.5.4 Documentation des fonctions amies et associées	42
7.5.4.1 operator<<	42
7.6 Référence de la classe Interaction	43
7.6.1 Description détaillée	43
7.6.2 Documentation des constructeurs et destructeur	43
7.6.2.1 Interaction()	43
7.6.3 Documentation des fonctions membres	43
7.6.3.1 operator()()	43
7.7 Référence de la classe InternCell	44
7.7.1 Description détaillée	45
7.7.2 Documentation des constructeurs et destructeur	46
7.7.2.1 InternCell()	46
7.7.3 Documentation des fonctions membres	46
7.7.3.1 addParticle()	46
7.7.3.2 applyForceOnNeighbours()	47
7.7.3.3 clearParticles()	47
7.7.3.4 computeInternInterractions()	47
7.7.3.5 getParticles()	48
7.8 Référence de la classe Particle	48
7.8.1 Documentation des constructeurs et destructeur	49
7.8.1.1 Particle()	49
7.8.2 Documentation des fonctions membres	50
7.8.2.1 addToForce()	50
7.8.2.2 addToForceCoord()	51
7.8.2.3 addToPosition()	51
7.8.2.4 addToSpeed()	52
7.8.2.5 applyExternalForces()	53
7.8.2.6 applyInteractionForcesOn()	53
7.8.2.7 distanceTo()	53
7.8.2.8 getDimension()	54
7.8.2.9 getForce()	55
7.8.2.10 getMass()	55
7.8.2.11 getName()	55
7.8.2.12 getOldForce()	56

7.8.2.13	getParticleCount()	56
7.8.2.14	getPosition()	56
7.8.2.15	getSpeed()	56
7.8.2.16	invertSpeed()	57
7.8.2.17	multiplySpeed()	58
7.8.2.18	setForce()	58
7.8.2.19	setForceToZero()	58
7.8.2.20	setOldForce()	59
7.8.2.21	setPosCoord()	59
7.8.2.22	setPosition()	59
7.8.2.23	setSpeed()	59
7.8.3	Documentation des fonctions amies et associées	60
7.8.3.1	operator<<	60
7.9	Référence de la classe Progressbar	60
7.9.1	Documentation des constructeurs et destructeur	61
7.9.1.1	~Progressbar()	61
7.9.1.2	Progressbar() [1/4]	61
7.9.1.3	Progressbar() [2/4]	61
7.9.1.4	Progressbar() [3/4]	61
7.9.1.5	Progressbar() [4/4]	62
7.9.2	Documentation des fonctions membres	62
7.9.2.1	operator=() [1/2]	62
7.9.2.2	operator=() [2/2]	62
7.9.2.3	reset()	62
7.9.2.4	set_closing_bracket_char()	62
7.9.2.5	set_done_char()	62
7.9.2.6	set_niter()	63
7.9.2.7	set_opening_bracket_char()	63
7.9.2.8	set_output_stream()	63
7.9.2.9	set_todo_char()	63
7.9.2.10	show_bar()	63
7.9.2.11	update()	63
7.10	Référence de la classe Universe	64
7.10.1	Description détaillée	66
7.10.2	Documentation des constructeurs et destructeur	66
7.10.2.1	Universe()	66
7.10.3	Documentation des fonctions membres	67
7.10.3.1	addExternalForce()	67
7.10.3.2	addInteraction()	67
7.10.3.3	addParticle() [1/4]	67
7.10.3.4	addParticle() [2/4]	68
7.10.3.5	addParticle() [3/4]	69

7.10.3.6 addParticle() [4 / 4]	69
7.10.3.7 applyExternalForces()	70
7.10.3.8 applyInteractionForces()	71
7.10.3.9 getBounds()	71
7.10.3.10 getDimension()	71
7.10.3.11 getInteractions()	72
7.10.3.12 getLastAddedParticlePointer()	72
7.10.3.13 getMaxForce()	72
7.10.3.14 getNbParticles()	72
7.10.3.15 getNbPastStates()	73
7.10.3.16 getParticles()	73
7.10.3.17 setCineticEnergyLimit()	73
7.10.3.18 simulateStormerVerlet()	74
7.10.3.19 updateForces()	75
7.10.3.20 updatePositions()	75
7.10.4 Documentation des fonctions amies et associées	76
7.10.4.1 operator<<	76
7.10.4.2 VisualGenerator	76
7.11 Référence de la classe Vector	76
7.11.1 Description détaillée	78
7.11.2 Documentation des constructeurs et destructeur	78
7.11.2.1 Vector() [1 / 3]	78
7.11.2.2 Vector() [2 / 3]	78
7.11.2.3 Vector() [3 / 3]	78
7.11.3 Documentation des fonctions membres	78
7.11.3.1 areAllCoordsGreater()	78
7.11.3.2 getData()	79
7.11.3.3 getDimension()	79
7.11.3.4 isInBounds()	80
7.11.3.5 norm()	81
7.11.3.6 operator!=()	82
7.11.3.7 operator*=()	82
7.11.3.8 operator+=()	82
7.11.3.9 operator-=()	82
7.11.3.10 operator==()	82
7.11.3.11 operator[]() [1 / 2]	83
7.11.3.12 operator[]() [2 / 2]	83
7.11.4 Documentation des fonctions amies et associées	83
7.11.4.1 max	83
7.11.4.2 min	83
7.11.4.3 operator<<	84
7.12 Référence de la classe VisualGenerator	84

7.12.1 Description détaillée	85
7.12.2 Documentation des constructeurs et destructeur	85
7.12.2.1 VisualGenerator()	85
7.12.3 Documentation des fonctions membres	85
7.12.3.1 generatePhoto()	85
7.12.3.2 generateVideo()	85
7.12.3.3 setImageSizes()	86
7.12.3.4 setPointSize()	86
7.12.3.5 setPointType()	87
8 Documentation des fichiers	89
8.1 Référence du fichier docs/conception.md	89
8.2 Référence du fichier docs/TO DO.md	89
8.3 Référence du fichier include/cell.hpp	89
8.3.1 Description détaillée	90
8.4 Référence du fichier include/config.hpp	90
8.4.1 Description détaillée	91
8.4.2 Documentation des macros	91
8.4.2.1 NDEBUG	91
8.4.2.2 PNG_OUTPUT	91
8.4.2.3 SHOW_PROGRESS_INFOS	92
8.4.2.4 XML_OUTPUT	92
8.5 Référence du fichier include/extern_border_cell.hpp	92
8.5.1 Description détaillée	93
8.6 Référence du fichier include/external_force.hpp	93
8.7 Référence du fichier include/finite_universe.hpp	94
8.7.1 Description détaillée	95
8.7.2 Documentation du type de l'énumération	95
8.7.2.1 OOBBehavior	95
8.8 Référence du fichier include/forces.hpp	96
8.8.1 Description détaillée	96
8.8.2 Documentation des fonctions	97
8.8.2.1 gravitationalForce()	97
8.8.2.2 gravitationalInteraction()	98
8.8.2.3 lennardJonesInteraction()	99
8.8.2.4 wallsForce()	100
8.9 Référence du fichier include/gridded_universe.hpp	101
8.10 Référence du fichier include/intern_cell.hpp	102
8.11 Référence du fichier include/interraction.hpp	103
8.11.1 Description détaillée	103
8.12 Référence du fichier include/particle.hpp	104
8.13 Référence du fichier include/progressbar.hpp	104

8.14 Référence du fichier include/universe.hpp	105
8.15 Référence du fichier include/vector.hpp	106
8.15.1 Description détaillée	107
8.16 Référence du fichier include/visual_generator.hpp	108
8.16.1 Documentation du type de l'énumération	109
8.16.1.1 PointType	109
8.17 Référence du fichier include/xassert.hpp	109
8.17.1 Description détaillée	110
8.17.2 Documentation des macros	110
8.17.2.1 xassert	110
8.17.3 Documentation des fonctions	110
8.17.3.1 _xassert() [1/2]	110
8.17.3.2 _xassert() [2/2]	111
8.18 Référence du fichier README.md	111
8.19 Référence du fichier src/cell.cpp	111
8.20 Référence du fichier src/extern_border_cell.cpp	111
8.21 Référence du fichier src/finite_universe.cpp	112
8.21.1 Documentation des fonctions	113
8.21.1.1 operator<<()	113
8.22 Référence du fichier src/forces.cpp	113
8.22.1 Documentation des fonctions	114
8.22.1.1 gravitationalForce()	114
8.22.1.2 gravitationalInteraction()	115
8.22.1.3 lennardJonesInteraction()	115
8.22.1.4 wallsForce()	116
8.23 Référence du fichier src/gridded_universe.cpp	117
8.23.1 Documentation des fonctions	118
8.23.1.1 isExternCoord()	118
8.23.1.2 isInternCoord()	118
8.23.1.3 operator<<()	118
8.24 Référence du fichier src/intern_cell.cpp	119
8.25 Référence du fichier src/main.cpp	119
8.25.1 Documentation des fonctions	120
8.25.1.1 main()	120
8.26 Référence du fichier test/main.cpp	121
8.26.1 Documentation des fonctions	121
8.26.1.1 main()	121
8.27 Référence du fichier src/particle.cpp	121
8.27.1 Documentation des fonctions	122
8.27.1.1 operator<<()	122
8.28 Référence du fichier src/universe.cpp	122
8.28.1 Documentation des fonctions	123

8.28.1.1 operator<<()	123
8.28.1.2 writeData()	124
8.28.1.3 writeDataVTK()	124
8.29 Référence du fichier src/vector.cpp	125
8.29.1 Documentation des fonctions	126
8.29.1.1 max()	126
8.29.1.2 min()	126
8.29.1.3 operator<<()	127
8.30 Référence du fichier src/visual_generator.cpp	127
8.30.1 Documentation des fonctions	128
8.30.1.1 setAxRange()	128
8.31 Référence du fichier test/particle_test.cpp	129
8.31.1 Documentation des fonctions	130
8.31.1.1 TEST() [1/9]	130
8.31.1.2 TEST() [2/9]	131
8.31.1.3 TEST() [3/9]	131
8.31.1.4 TEST() [4/9]	132
8.31.1.5 TEST() [5/9]	132
8.31.1.6 TEST() [6/9]	132
8.31.1.7 TEST() [7/9]	133
8.31.1.8 TEST() [8/9]	133
8.31.1.9 TEST() [9/9]	134
8.31.2 Documentation des variables	134
8.31.2.1 mass	134
8.31.2.2 name	134
8.31.2.3 p	134
8.31.2.4 pos	135
8.31.2.5 speed	135
8.32 Référence du fichier test/solar_system_tp2.cpp	135
8.32.1 Documentation des fonctions	136
8.32.1.1 main()	136
8.33 Référence du fichier test/vector_test.cpp	136
8.33.1 Documentation des fonctions	137
8.33.1.1 TEST() [1/8]	137
8.33.1.2 TEST() [2/8]	138
8.33.1.3 TEST() [3/8]	138
8.33.1.4 TEST() [4/8]	138
8.33.1.5 TEST() [5/8]	139
8.33.1.6 TEST() [6/8]	139
8.33.1.7 TEST() [7/8]	139
8.33.1.8 TEST() [8/8]	139

Chapitre 1

README

Ce projet vise à développer un programme en C++ pour simuler des systèmes de particules.

Pour lire la documentation Doxygen complète, ouvrir le PDF dans le répertoire `docs` ou bien exécuter la commande suivant depuis la racine du projet :

```
open docs/html/index.html
```

1.1 Auteurs

Jules Roques (roquesj)

Zephirin Faure (faurez)

1.2 Compilation

Lancer les commandes suivantes depuis la racine du projet pour le compiler :

```
mkdir build
cd build/
cmake ..
make
```

1.3 Documentation

L'ensemble de la documentation du projet peut être parcourue à l'aide des fichiers générés par Doxygen (cf. intro).

Pour plus d'information concernant la conception du code : `/user/4/.base/faurez/home/Cours/2A/C++/tps-cpp/docs/conception.md` "Conception".

Pour chaque classe, méthode et attribut, la documentation propose les schémas expliquant les héritages, les appels, etc.

1.4 Utilisation

L'exécutable du code principal sera généré dans le répertoire `build/src` et peut être exécuter en effectuant la commande suivante depuis le répertoire `build` du projet :

```
./src/main
```

1.4.1 Définition de l'univers

Pour définir le type d'univers et les particules de la simulation, modifier le corps du main dans `src/main.cpp`. Pourront y être définis :

1. **Les interactions** : Ce sont les forces qui s'exercent entre les particules. Ajouter des interactions entre particules en utilisant la méthode `addInteraction` de l'objet `universeGrid`. Par exemple :

```
```cpp universeGrid.addInteraction( [epsilon, sigma](const Particle& source, Particle& target) { lennardJonesInteraction(source, target, epsilon, sigma); } ); ```
```

 Interactions possibles :
  - `gravitationalInteraction`
  - `lennardJonesInteraction`
2. **Les forces externes** : Ce sont les forces qui s'exercent sur les particules individuellement en fonction de leur position dans l'univers. Ajouter une force externe en utilisant la méthode `addExternalForce`. Par exemple :  

```
```cpp universeGrid.addExternalForce( [G](Particle& target) { gravitationalForce(target, G); } ); ```
```

 Forces externes possibles :
 - `gravitationalForce`
 - `wallsForce`
3. **Le type l'univers** : Définir les dimensions de l'univers en spécifiant les bornes inférieure et supérieure dans les vecteurs `lowerBound` et `upperBound` qui représentent chacun un sommet de l'univers. De plus l'univers peut être de deux types :
 - `finite_universe` : Un univers de taille finie dans lequel toutes les particules interagissent entre elles;
 - `gridded_universe` : Un univers de taille finie découpé en une grille de cellules telles que les particules n'interagissent qu'avec celles de la même cellule ou des cellules voisines.
4. **Les particules** : Ajouter des particules à l'univers en utilisant la méthode `addParticle`. Par exemple, pour ajouter des particules dans une région rectangulaire, utiliser une boucle imbriquée comme dans l'exemple suivant pour ajouter des particules rouges :

```
```cpp Vector bottomLeftCorner({100,60}); for (size_t i = 0; i < red_width; i++) { for (size_t j = 0; j < red_height; j++) { Vector position = bottomLeftCorner; position += Vector({i * spaceStep, j * spaceStep}); universeGrid.addParticle(position, red_speed, m); } } ```
```
5. **Les comportements aux limites** : Ce sont les conditions qui définissent le comportement des particules au bord de l'univers. Définir le comportement des particules lorsqu'elles atteignent les limites de l'univers avec la méthode `setLimitBehavior`. Il y a 3 comportements possibles :
  - `REFLEXION` : Les particules rebondissent sur les bords,
  - `ABSORPTION` : Les particules disparaissent,
  - `PERIODIC` : Les particules reviennent de l'autre côté de l'univers (uniquement pour `gridded_universe`).
6. **La simulation** : Configurer et lancer la simulation en utilisant la méthode `simulateStormerVerlet`, en spécifiant le pas de temps et le temps final de la simulation.

En suivant ces étapes, l'univers de simulation sera configuré et personnalisé les interactions et les forces appliquées aux particules pourront être personnalisées.

### 1.4.2 Configuration

Le projet peut être configuré via le fichier `include/config.hpp`. Ce fichier contient les définitions de constantes permettant de personnaliser le comportement du programme en préprocesseur.

Pour activer ou désactiver une fonctionnalité spécifique, commenter ou décommenter les lignes appropriées dans ce fichier.

Voici une explication de chaque configuration :

- `SHOW_PROGRESS_INFOS`: Lorsque activé, permet d'afficher des informations sur l'évolution du programme pendant son exécution.
- `NDEBUG`: Lorsque activé, toutes les assertions du programme sont désactivées, ce qui peut améliorer les performances en éliminant les vérifications supplémentaires pendant l'exécution. Cependant les erreurs potentielles ne seront pas détectées et signalées.
- `PNG_OUTPUT`: Lorsque activé, le programme génère la sortie dans un format d'images PNG, utile pour visualiser les résultats de manière graphique et générer ensuite une vidéo ou un gif.
- `XML_OUTPUT`: Lorsque activé, le programme génère la sortie au format XML, utile pour enregistrer les résultats dans un format structuré et lisible par machine, facilitant ainsi le traitement et l'analyse des données.

### 1.4.3 Sortie

Le programme offre deux possibilités pour la sortie des données :

#### 1. Sortie PNG

Lorsque l'option `PNG_OUTPUT` est activée dans le fichier de configuration, le programme génère un fichier ASCII `build/pastParticles.txt`. Ce fichier contient les positions et les forces des particules à chaque instant de la simulation. Ensuite, le programme utilise ces données pour générer un dossier d'images au format PNG `build/video`. Ce dossier contient 200 images, chacune représentant un instant de l'évolution de l'univers. Ces images peuvent être utilisées pour créer une vidéo ou un GIF animé de l'évolution de l'univers à l'aide de logiciels de montage vidéo ou de création de GIF.

Voici les commandes pour générer une vidéo ou un GIF à partir des images :

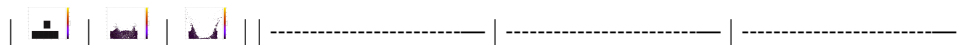
```
```bash
```

1.5 Pour générer une vidéo à partir des images PNG

```
ffmpeg -framerate 30 -pattern_type glob -i 'video/*.png' -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4
```

1.6 Pour générer un GIF à partir des images PNG

```
ffmpeg -i 'video/img%03d.png' -vf "fps=10,scale=320:-1:flags=lanczos" -c:v pam -f image2pipe - | convert -delay 5 -loop 0 -layers Optimize output.gif ```
```



1. Sortie XML (VTK)

Lorsque l'option `XML_OUTPUT` est activée dans le fichier de configuration, le programme génère un dossier `build/VTKFiles`. Ce dossier contient un fichier au format VTK pour chaque pas de temps de la simulation. Chaque fichier VTK représente l'état de l'univers à un instant donné et peut être visualisé à l'aide de logiciels de visualisation de données tels que Paraview. Ces fichiers VTK permettent une analyse détaillée de l'évolution de l'univers et offrent une visualisation 3D interactive de la simulation.

Pour visualiser les données avec Paraview, ouvrez le logiciel et ouvrez un fichier `.xml` à partir du dossier `VTKFiles`. Vous pouvez ensuite explorer les données, les filtrer et les visualiser de différentes manières pour mieux comprendre l'évolution du système de particules au fil du temps.

1.6.1 Tests


Pour tester les fonctionnalités de base de notre code, un exécutable de test utilisant Google Test (gtest) est généré dans le répertoire `build/test/test`. Cet exécutable permet de vérifier le bon fonctionnement des classes [Vector](#) et [Particle](#) implémentées.


Pour exécuter les tests, utiliser la commande suivante depuis le répertoire `build` du projet :

```
./test/test
```


Chapitre 2

Conception

 {width=1200px}

 {width=1200px}

Chapitre 3

TO DO

3.0.1 Zeph

- tests + ecriture xml
- revoir écriture de data dans fichier xml (écriture en binaire) car trop volumineux en ascii

3.0.2 Jules

- les conditions aux bord PERIODIC (periodic seulement pour un griddedUniverse) et BOUNCE_↔ REFLEXION (reflexion simple) ne fonctionnent pas
- gestion des erreurs (try / catch), par ex création / écriture fichier, ou si énergie cinétique négative

3.0.3 Reste

- optimisations (parcours des cellules voisines)

Chapitre 4

Index hiérarchique

4.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Cell	15
ExternBorderCell	21
InternCell	44
ExternalForce	19
Interaction	43
Particle	48
Progressbar	60
Universe	64
FiniteUniverse	24
GriddedUniverse	38
Vector	76
VisualGenerator	84

Chapitre 5

Index des classes

5.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Cell	Cells are elementary bricks forming a grid of the universe	15
ExternalForce	Stores a function that rules a force applied on particles	19
ExternBorderCell	Limits Cells contains copies of particles of a foreign cell, neighbour for a PERIDIC point of view. An offset is applied on the particles positions	21
FiniteUniverse	Universe with bounds	24
GriddedUniverse	A GriddedUniverse is a finite universe, separated into cells. Extends Universe	38
Interaction	Store a function that rules an interaction between particles	43
InternCell	Limits Cells are contains copies of the cellule of same coordinates, with a position offset	44
Particle	48
Progressbar	60
Universe	Represents an universe containing particles. Can be in dimension 1, 2, or 3	64
Vector	Vector class	76
VisualGenerator	Provides methods to generate visualisation of an universe	84

Chapitre 6

Index des fichiers

6.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

include/cell.hpp	89
Les cellules sont les cases élémentaires formant un maillage de l'univers	
include/config.hpp	90
Contains constants definitions for preprocessing of the code	
include/extern_border_cell.hpp	92
Cells for limit behavior of a PERIODIC universe	
include/external_force.hpp	93
include/finite_universe.hpp	94
A universe with limits	
include/forces.hpp	96
File containing different forces functions	
include/gridded_universe.hpp	101
include/intern_cell.hpp	102
include/interaction.hpp	103
To define an interaction between two particles	
include/particle.hpp	104
include/progressbar.hpp	104
include/universe.hpp	105
include/vector.hpp	106
File for vector calculations	
include/visual_generator.hpp	108
include/xassert.hpp	109
To define an assert macro	
src/cell.cpp	111
src/extern_border_cell.cpp	111
src/finite_universe.cpp	112
src/forces.cpp	113
src/gridded_universe.cpp	117
src/intern_cell.cpp	119
src/main.cpp	119
src/particle.cpp	121
src/universe.cpp	122
src/vector.cpp	125
src/visual_generator.cpp	127
test/main.cpp	121
test/particle_test.cpp	129
test/solar_system_tp2.cpp	135
test/vector_test.cpp	136

Chapitre 7

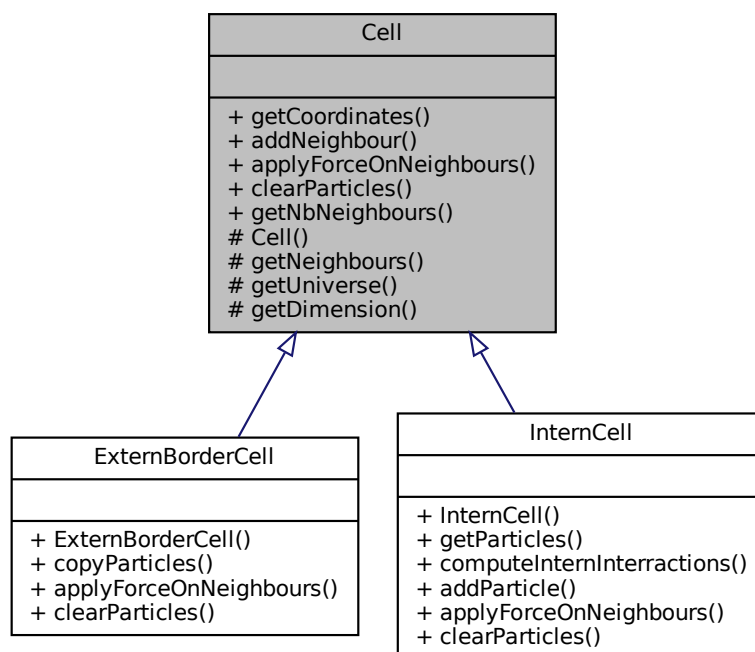
Documentation des classes

7.1 Référence de la classe Cell

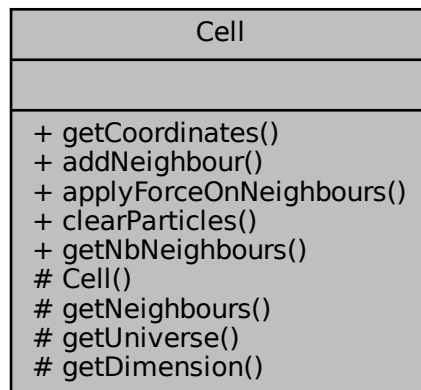
Cells are elementary bricks forming a grid of the universe.

```
#include <cell.hpp>
```

Graphe d'héritage de Cell:



Graphe de collaboration de Cell:



Fonctions membres publiques

- const std::vector< int > & [getCoordinates](#) () const
- void [addNeighbour](#) ([InternCell](#) *neighbour)
- *Adds a cell to the neighbours.*
- virtual void [applyForceOnNeighbours](#) () const =0
- *applies forces on neighbours particles (adds to existing forces)*
- virtual void [clearParticles](#) ()=0
- *Delete particles.*
- size_t [getNbNeighbours](#) () const

Fonctions membres protégées

- [Cell](#) (size_t dimension, [GriddedUniverse](#) *universe, std::vector< int > coordinates)
- std::list< [InternCell](#) * > [getNeighbours](#) () const
- [GriddedUniverse](#) * [getUniverse](#) () const
- size_t [getDimension](#) ()

7.1.1 Description détaillée

Cells are elementary bricks forming a grid of the universe.

7.1.2 Documentation des constructeurs et destructeur

7.1.2.1 Cell()

```

Cell::Cell (
    size_t dimension,
    GriddedUniverse * universe,
    std::vector< int > coordinates ) [protected]

```

7.1.3 Documentation des fonctions membres

7.1.3.1 addNeighbour()

```
void Cell::addNeighbour (
    InternCell * neighbour )
```

Adds a cell to the neighbours.

Paramètres

<i>neighbour</i>	pointer to cell
------------------	-----------------

7.1.3.2 applyForceOnNeighbours()

```
virtual void Cell::applyForceOnNeighbours ( ) const [pure virtual]
```

applies forces on neighbours particles (adds to existing forces)

Implémenté dans [InternCell](#), et [ExternBorderCell](#).

7.1.3.3 clearParticles()

```
virtual void Cell::clearParticles ( ) [pure virtual]
```

Delete particles.

Implémenté dans [InternCell](#), et [ExternBorderCell](#).

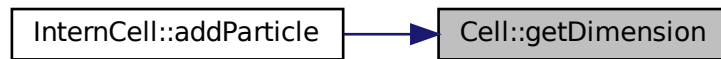
7.1.3.4 getCoordinates()

```
const std::vector<int>& Cell::getCoordinates ( ) const [inline]
```

7.1.3.5 getDimension()

```
size_t Cell::getDimension ( ) [inline], [protected]
```

Voici le graphe des appelants de cette fonction :



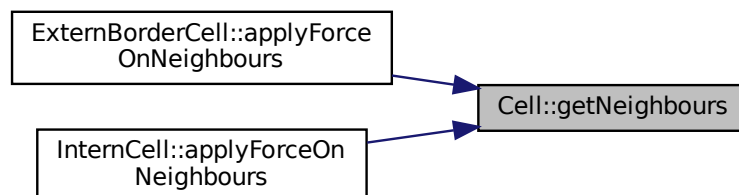
7.1.3.6 getNbNeighbours()

```
size_t Cell::getNbNeighbours ( ) const [inline]
```

7.1.3.7 getNeighbours()

```
std::list<InternCell*> Cell::getNeighbours ( ) const [inline], [protected]
```

Voici le graphe des appelants de cette fonction :



7.1.3.8 getUniverse()

```
GriddedUniverse* Cell::getUniverse ( ) const [inline], [protected]
```

Voici le graphe des appelants de cette fonction :



La documentation de cette classe a été générée à partir du fichier suivant :

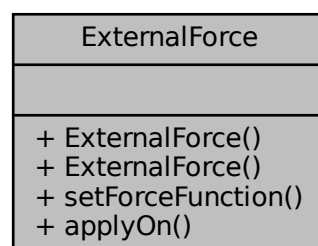
- [include/cell.hpp](#)
- [src/cell.cpp](#)

7.2 Référence de la classe ExternalForce

Stores a function that rules a force applied on particles.

```
#include <external_force.hpp>
```

Graphe de collaboration de ExternalForce:



Fonctions membres publiques

- [ExternalForce](#) ()
- [ExternalForce](#) (std::function< void([Particle](#) &)> forceFunction)
- void [setForceFunction](#) (std::function< void([Particle](#) &)> forceFunction)
- void [applyOn](#) ([Particle](#) &target) const

Computes and add the force applied on the given particle.

7.2.1 Description détaillée

Stores a function that rules a force applied on particles.

7.2.2 Documentation des constructeurs et destructeur

7.2.2.1 ExternalForce() [1/2]

```
ExternalForce::ExternalForce ( ) [inline]
```

7.2.2.2 ExternalForce() [2/2]

```
ExternalForce::ExternalForce (
    std::function< void(Particle &)> forceFunction ) [inline]
```

7.2.3 Documentation des fonctions membres

7.2.3.1 applyOn()

```
void ExternalForce::applyOn (
    Particle & target ) const [inline]
```

Computes and add the force applied on the given particle.

Paramètres

<i>target</i>	
---------------	--

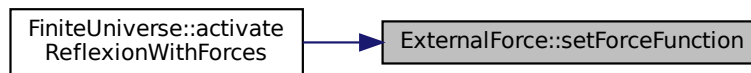
Renvoie

[Vector](#)

7.2.3.2 setForceFunction()

```
void ExternalForce::setForceFunction (
    std::function< void(Particle &)> forceFunction ) [inline]
```


Voici le graphe des appelants de cette fonction :



La documentation de cette classe a été générée à partir du fichier suivant :

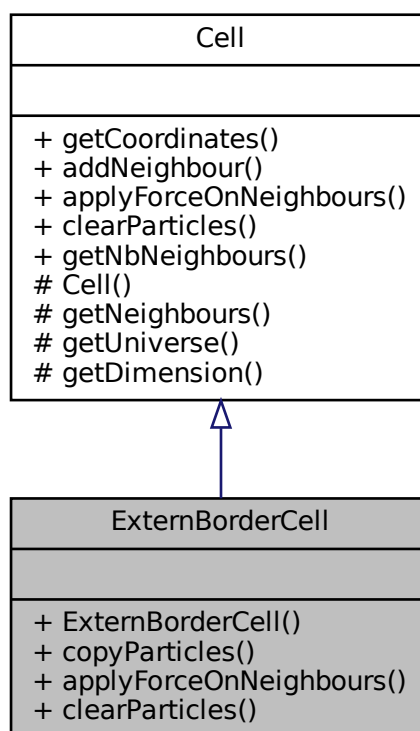
— [include/external_force.hpp](#)

7.3 Référence de la classe ExternBorderCell

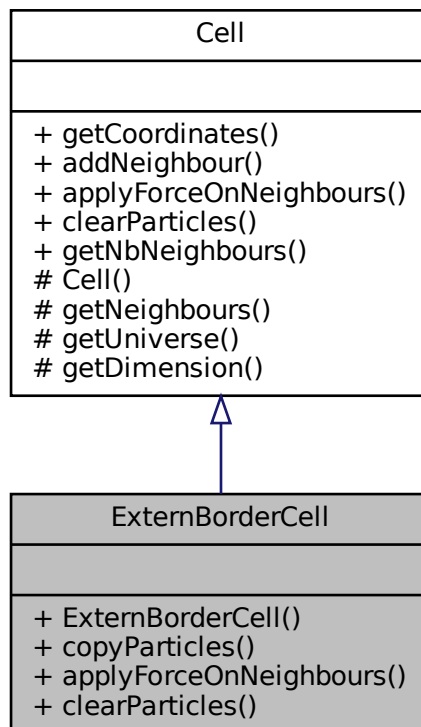
Limits Cells contains copies of particles of a foreign cell, neighbour for a PERIDIC point of view. An offset is applied on the particles positions.

```
#include <extern_border_cell.hpp>
```

Graphe d'héritage de ExternBorderCell:



Graphe de collaboration de ExternBorderCell:



Fonctions membres publiques

- `ExternBorderCell` (`size_t` dimension, `GriddedUniverse *universe`, `std::vector< int > coordinates`, `Vector` offset, `InternCell *copyCell`)
- `void copyParticles ()`
Copies particles from the _copyCell.
- `void applyForceOnNeighbours ()` const override
applies forces on neighbours particles (adds to existing forces)
- `void clearParticles ()` override
Delete particles.

Membres hérités additionnels

7.3.1 Description détaillée

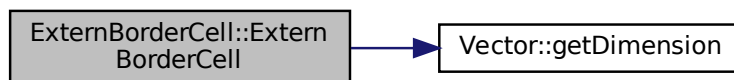
Limits Cells contains copies of particles of a foreign cell, neighbour for a PERIDIC point of view. An offset is applied on the particles positions.

7.3.2 Documentation des constructeurs et destructeur

7.3.2.1 ExternBorderCell()

```
ExternBorderCell::ExternBorderCell (
    size_t dimension,
    GriddedUniverse * universe,
    std::vector< int > coordinates,
    Vector offset,
    InternCell * copyCell ) [inline]
```

Voici le graphe d'appel pour cette fonction :



7.3.3 Documentation des fonctions membres

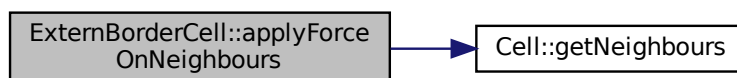
7.3.3.1 applyForceOnNeighbours()

```
void ExternBorderCell::applyForceOnNeighbours ( ) const [override], [virtual]
```

applies forces on neighbours particles (adds to existing forces)

Implémente [Cell](#).

Voici le graphe d'appel pour cette fonction :



7.3.3.2 clearParticles()

```
void ExternBorderCell::clearParticles ( ) [override], [virtual]
```

Delete particles.

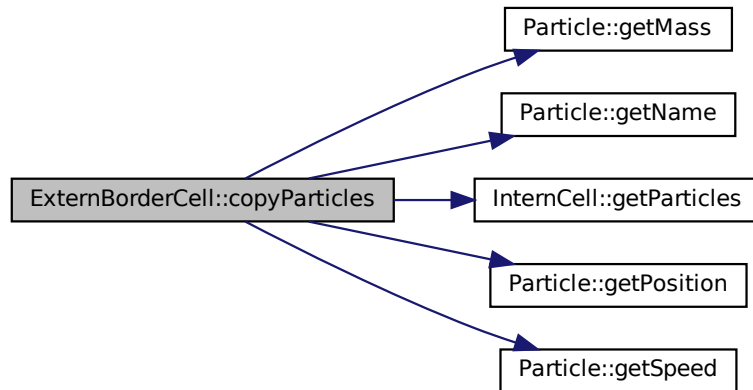
Implémente [Cell](#).

7.3.3.3 copyParticles()

```
void ExternBorderCell::copyParticles ( )
```

Copies particles from the `_copyCell`.

Voici le graphe d'appel pour cette fonction :



La documentation de cette classe a été générée à partir du fichier suivant :

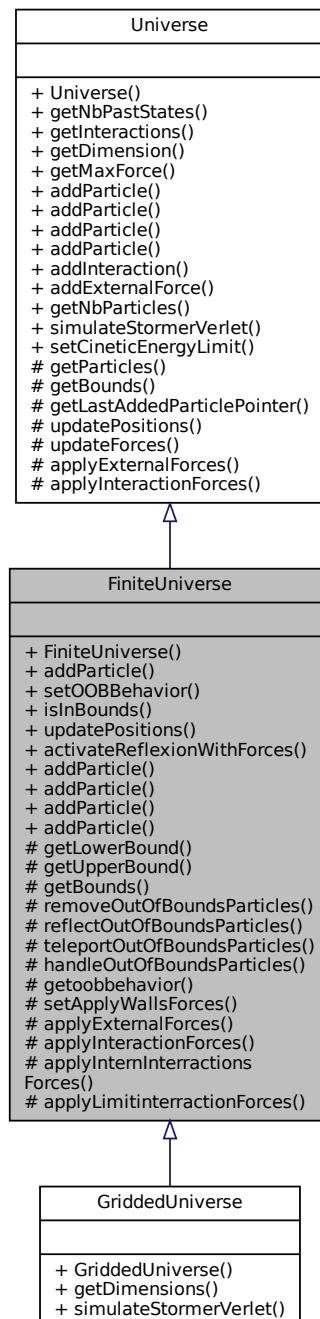
- [include/extern_border_cell.hpp](#)
- [src/extern_border_cell.cpp](#)

7.4 Référence de la classe FiniteUniverse

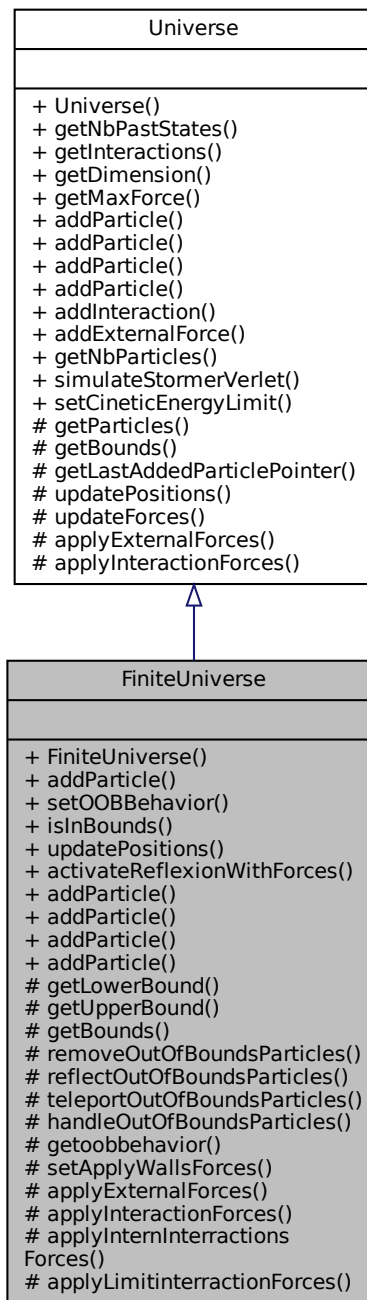
[Universe](#) with bounds.

```
#include <finite_universe.hpp>
```

Graphe d'héritage de FiniteUniverse:



Graphe de collaboration de FiniteUniverse:



Fonctions membres publiques

- **FiniteUniverse** (**Vector** lowerBound, **Vector** upperBound)
- virtual void **addParticle** (**Vector** posCoords, **Vector** speedCoords, double **mass**, std::string **name**) override
Adds a particle to the gridded universe.
- void **setOOBBehavior** (**OOBBehavior** lb)
Set the Limit Behavior of particles. REFLEXION makes particle stay in, ABSORPTION delete particles, PERIODIC teleports particles to the other side.

- bool `isInBounds` (const `Particle` &p)
Says if particle is in the bounds of the universe.
- virtual void `updatePositions` (double timeStep) override
Updates particles positions in Stormer Verlet algorithm. Handles cases about being outside of the finite universe.
- void `activateReflexionWithForces` (double epsilon, double sigma)
Activate reflexion mode with a repulsing force from the walls.
- virtual void `addParticle` (`Vector` pos, `Vector` speed, double mass, std::string name)
Adds a particle into the universe.
- void `addParticle` (std::initializer_list< double > posCoords, std::initializer_list< double > speedCoords, double mass, std::string name)
Adds a particle into the universe.
- void `addParticle` (`Vector` pos, `Vector` speed, double mass)
Adds a particle into the universe. Give it automaticly a name.
- void `addParticle` (std::initializer_list< double > posCoords, std::initializer_list< double > speedCoords, double mass)
Adds a particle into the universe. Give it automaticly a name.

Fonctions membres protégées

- const `Vector` & `getLowerBound` () const
- const `Vector` & `getUpperBound` () const
- std::pair< `Vector`, `Vector` > `getBounds` () const override
Get the bounds of the universe, i.e. the min and max vectors of all past particles (not present particles)
- void `removeOutOfBoundsParticles` ()
Removes particles outside of the bounds of the finite universe.
- void `reflectOutOfBoundsParticles` ()
Reflect particles outside of the bounds of the finite universe.
- void `teleportOutOfBoundsParticles` ()
Teleport particles outside of the bounds of the finite universe to the other side.
- void `handleOutOfBoundsParticles` ()
Handle out of bounds particles.
- `OOBBehavior` `getoobbehavior` () const
- void `setApplyWallsForces` (bool apply)
- void `applyExternalForces` () override
Applies external forces on particles in the universe. Linear complexity. Deal with limits forces.
- void `applyInteractionForces` () override
Applies forces between particles in the universe. Quadratic complexity.
- virtual void `applyInternInteractionsForces` ()
Applies forces between particles in the universe, but not limit interactions (like for PERIODIC)
- void `applyLimitinteractionForces` ()
Deal with limits forces, for exple when the universe is PERIODIC.

Amis

- std::ostream & `operator<<` (std::ostream &strm, `FiniteUniverse` universe)
Overrides the << operator.

7.4.1 Description détaillée

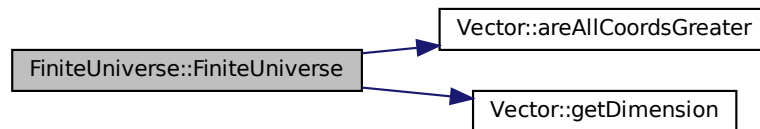
`Universe` with bounds.

7.4.2 Documentation des constructeurs et destructeur

7.4.2.1 FiniteUniverse()

```
FiniteUniverse::FiniteUniverse (
    Vector lowerBound,
    Vector upperBound )
```

Voici le graphe d'appel pour cette fonction :



7.4.3 Documentation des fonctions membres

7.4.3.1 activateReflexionWithForces()

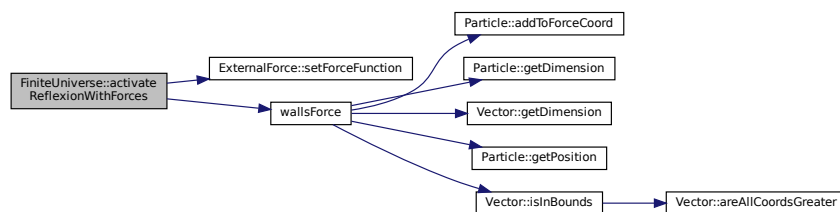
```
void FiniteUniverse::activateReflexionWithForces (
    double epsilon,
    double sigma )
```

Activate reflexion mode with a repulsing force from the walls.

Paramètres

<i>epsilon</i>	
<i>sigma</i>	

Voici le graphe d'appel pour cette fonction :



7.4.3.2 addParticle() [1/5]

```
void Universe::addParticle
```

Adds a particle into the universe. Give it automaticly a name.

Paramètres

<i>posCoords</i>	
<i>speedCoords</i>	
<i>mass</i>	

7.4.3.3 addParticle() [2/5]

```
void Universe::addParticle
```

Adds a particle into the universe.

Paramètres

<i>dimension</i>	
<i>posCoords</i>	
<i>speedCoords</i>	
<i>mass</i>	
<i>name</i>	

7.4.3.4 addParticle() [3/5]

```
void Universe::addParticle
```

Adds a particle into the universe. Give it automaticly a name.

Paramètres

<i>pos</i>	
<i>speed</i>	
<i>mass</i>	

Voici le graphe d'appel pour cette fonction :



7.4.3.5 addParticle() [4/5]

```
void Universe::addParticle
```

Adds a particle into the universe.

Paramètres

<i>dimension</i>	
<i>pos</i>	
<i>speed</i>	
<i>mass</i>	
<i>name</i>	

7.4.3.6 addParticle() [5/5]

```
void FiniteUniverse::addParticle (
    Vector posCoords,
    Vector speedCoords,
    double mass,
    std::string name ) [override], [virtual]
```

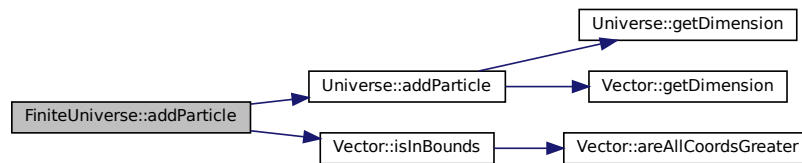
Adds a particle to the gridded universe.

Paramètres

<i>p</i>	the particle to add
----------	---------------------

Réimplémentée à partir de [Universe](#).

Voici le graphe d'appel pour cette fonction :



7.4.3.7 applyExternalForces()

```
void FiniteUniverse::applyExternalForces ( ) [override], [protected], [virtual]
```

Applies external forces on particles in the universe. Linear complexity. Deal with limits forces.

Réimplémentée à partir de [Universe](#).

Voici le graphe d'appel pour cette fonction :



7.4.3.8 applyInteractionForces()

```
void FiniteUniverse::applyInteractionForces ( ) [override], [protected], [virtual]
```

Applies forces between particles in the universe. Quadratic complexity.

Réimplémentée à partir de [Universe](#).

Voici le graphe d'appel pour cette fonction :

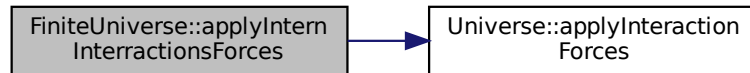


7.4.3.9 applyInternInteractionsForces()

```
void FiniteUniverse::applyInternInteractionsForces ( ) [protected], [virtual]
```

Applies forces between particles in the universe, but not limit interactions (like for PERIODIC)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.4.3.10 applyLimitinterractionForces()

```
void FiniteUniverse::applyLimitinterractionForces ( ) [protected]
```

Deal with limits forces, for exple when the universe is PERIODIC.

Voici le graphe des appelants de cette fonction :



7.4.3.11 getBounds()

```
std::pair< Vector, Vector > FiniteUniverse::getBounds ( ) const [override], [protected], [virtual]
```

Get the bounds of the universe, i.e. the min and max vectors of all past particles (not present particles)

Renvoie

```
const std::pair<Vector, Vector>&
```

Réimplémentée à partir de [Universe](#).

7.4.3.12 getLowerBound()

```
const Vector& FiniteUniverse::getLowerBound ( ) const [inline], [protected]
```

7.4.3.13 getoobbehavior()

```
OOBBehavior FiniteUniverse::getoobbehavior ( ) const [inline], [protected]
```

7.4.3.14 getUpperBound()

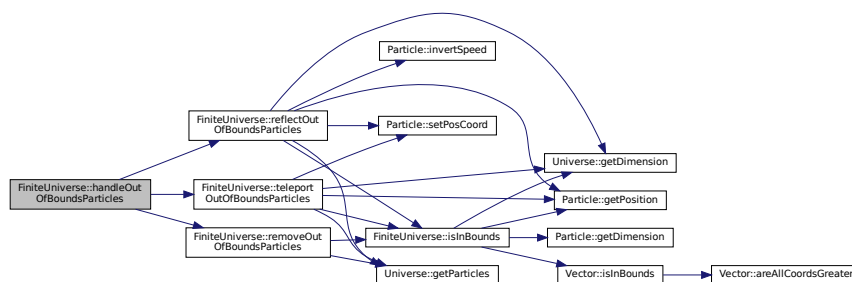
```
const Vector& FiniteUniverse::getUpperBound ( ) const [inline], [protected]
```

7.4.3.15 handleOutOfBoundsParticles()

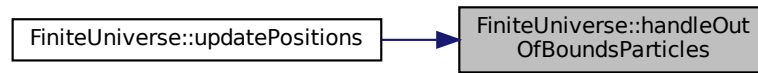
```
void FiniteUniverse::handleOutOfBoundsParticles ( ) [protected]
```

Handle out of bounds particles.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.4.3.16 isInBounds()

```
bool FiniteUniverse::isInBounds (
    const Particle & p )
```

Says if particle is in the bounds of the universe.

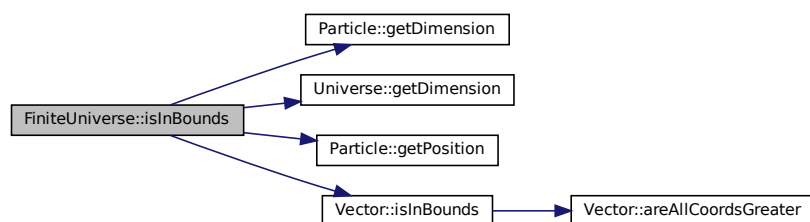
Paramètres

<i>p</i>	
----------	--

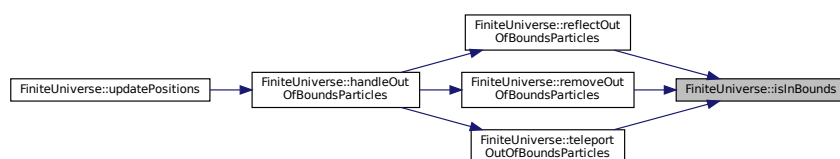
Renvoie

true if particle in the bounds of the universe
false if not

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

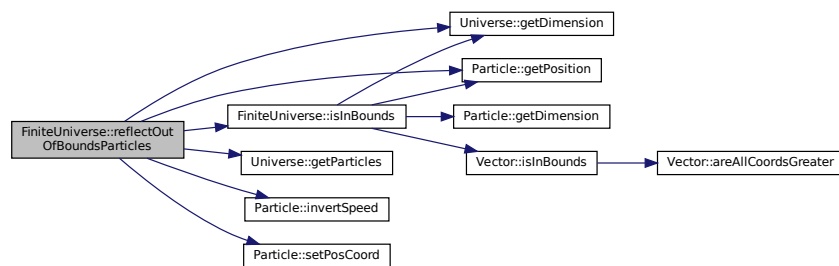


7.4.3.17 reflectOutOfBoundsParticles()

```
void FiniteUniverse::reflectOutOfBoundsParticles ( ) [protected]
```

Reflect particles outside of the bounds of the finite universe.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

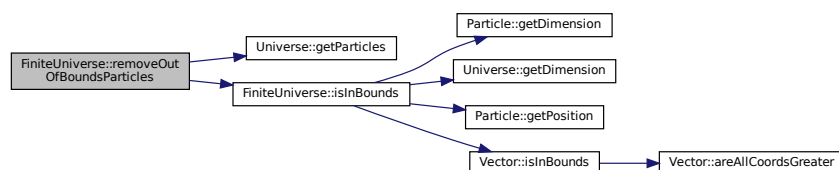


7.4.3.18 removeOutOfBoundsParticles()

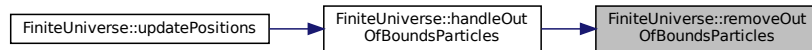
```
void FiniteUniverse::removeOutOfBoundsParticles ( ) [protected]
```

Removes particles outside of the bounds of the finite universe.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.4.3.19 setApplyWallsForces()

```
void FiniteUniverse::setApplyWallsForces (
    bool apply ) [inline], [protected]
```

7.4.3.20 setOOBBehavior()

```
void FiniteUniverse::setOOBBehavior (
    OOBBehavior lb )
```

Set the Limit Behavior of particles. REFLEXION makes particle stay in, ABSORPTION delete particles, PERIODIC teleports particles to the other side.

Paramètres

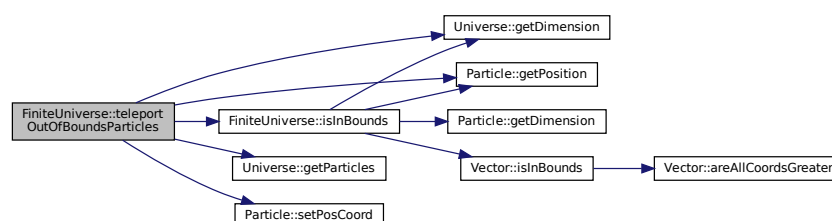
<i>lb</i>	
-----------	--

7.4.3.21 teleportOutOfBoundsParticles()

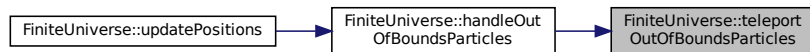
```
void FiniteUniverse::teleportOutOfBoundsParticles ( ) [protected]
```

Teleport particles outside of the bounds of the finite universe to the other side.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



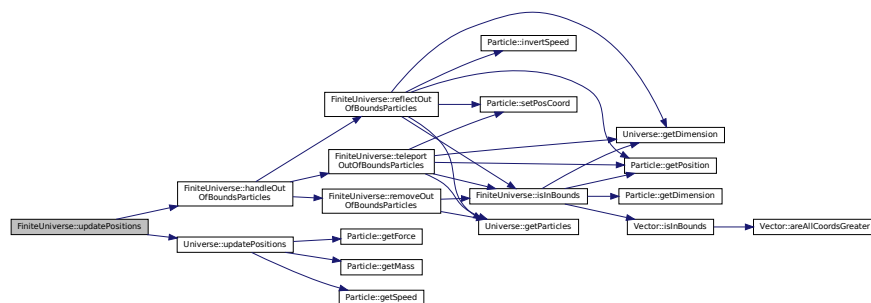
7.4.3.22 updatePositions()

```
void FiniteUniverse::updatePositions (
    double timeStep ) [override], [virtual]
```

Updates particles positions in Stormer Verlet algorithm. Handles cases about being outside of the finite universe.

Réimplémentée à partir de [Universe](#).

Voici le graphe d'appel pour cette fonction :



7.4.4 Documentation des fonctions amies et associées

7.4.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & strm,
    FiniteUniverse universe ) [friend]
```

Overrides the << operator.

Paramètres

<i>strm</i>	
<i>universe</i>	

Renvoie

`std::ostream&`

La documentation de cette classe a été générée à partir du fichier suivant :

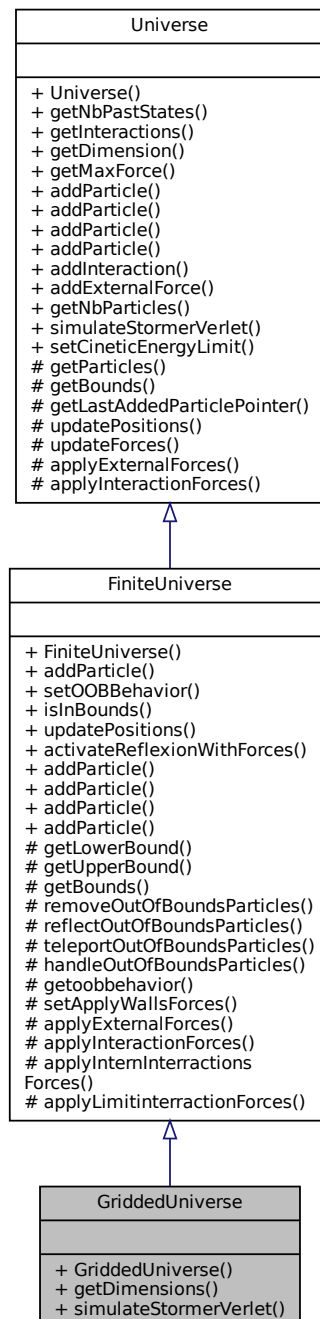
- [include/finite_universe.hpp](#)
- [src/finite_universe.cpp](#)

7.5 Référence de la classe GriddedUniverse

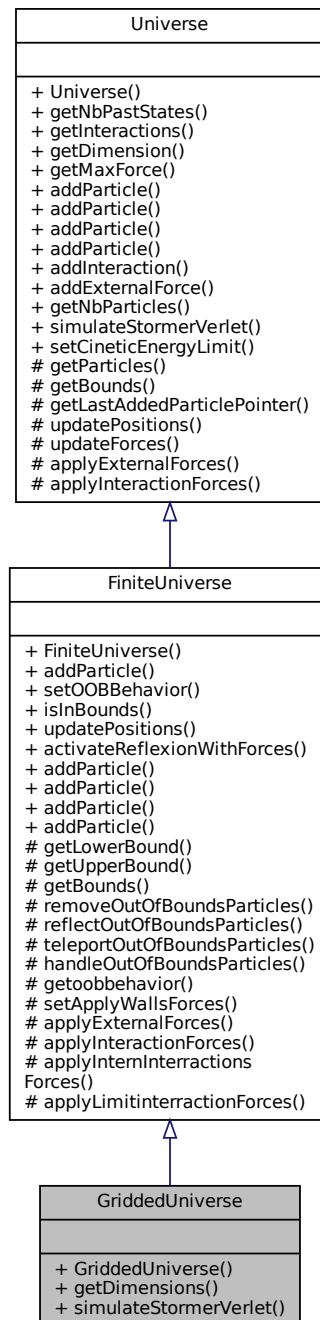
A [GriddedUniverse](#) is a finite universe, separated into cells. Extends [Universe](#).

```
#include <gridded_universe.hpp>
```

Graphe d'héritage de GriddedUniverse:



Graphe de collaboration de GriddedUniverse:



Fonctions membres publiques

- [GriddedUniverse](#) ([Vector](#) lowerBound, [Vector](#) upperBound, double cellSide)
Create a [GriddedUniverse](#). Improve efficiency while computing interactions between particles. If a particle quits the domain during the simulation, it is deleted from the gridded universe.
- const std::vector< int > & [getDimensions](#) () const
- void [simulateStormerVerlet](#) (double timeStep, double finalTime) override
Simulates the movement of particles in the universe using the Störmer-Verlet method Before simulation, fill cells with particles.

Amis

— `std::ostream & operator<< (std::ostream &strm, GriddedUniverse universe)`

Membres hérités additionnels

7.5.1 Description détaillée

A [GriddedUniverse](#) is a finite universe, separated into cells. Extends [Universe](#).

7.5.2 Documentation des constructeurs et destructeur

7.5.2.1 GriddedUniverse()

```
GriddedUniverse::GriddedUniverse (
    Vector lowerBound,
    Vector upperBound,
    double cellSide )
```

Create a [GriddedUniverse](#). Improve efficiency while computing interactions between particles. If a particle quits the domain during the simulation, it is deleted from the gridded universe.

Paramètres

<i>cellSide</i>	distance with which we can neglect interactions between two particles
<i>lowerBound</i>	one extreme corner of the area of the universe
<i>upperBound</i>	the other extreme corner, must have greater coordinates

7.5.3 Documentation des fonctions membres

7.5.3.1 getDimensions()

```
const std::vector<int>& GriddedUniverse::getDimensions ( ) const [inline]
```

Voici le graphe des appelants de cette fonction :



7.5.3.2 simulateStormerVerlet()

```
void GriddedUniverse::simulateStormerVerlet (
    double timeStep,
    double finalTime ) [override], [virtual]
```

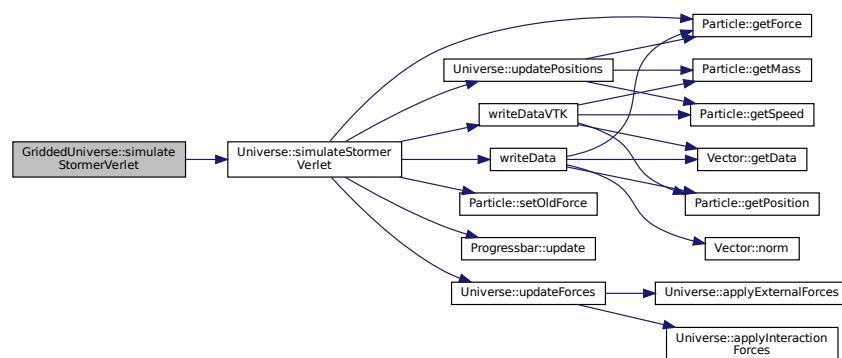
Simulates the movement of particles in the universe using the Störmer-Verlet method Before simulation, fill cells with particles.

Paramètres

<i>timeStep</i>	
<i>finalTime</i>	End time of the simulation

Réimplémentée à partir de [Universe](#).

Voici le graphe d'appel pour cette fonction :



7.5.4 Documentation des fonctions amies et associées

7.5.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & strm,
    GriddedUniverse universe ) [friend]
```

La documentation de cette classe a été générée à partir du fichier suivant :

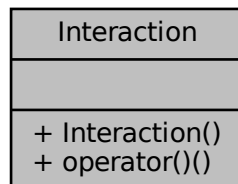
- [include/gridded_universe.hpp](#)
- [src/gridded_universe.cpp](#)

7.6 Référence de la classe Interaction

Store a function that rules an interaction between particles.

```
#include <interaction.hpp>
```

Graphe de collaboration de Interaction:



Fonctions membres publiques

- [Interaction](#) (std::function< void(const [Particle](#) &, [Particle](#) &)> interactionFunction)
- void [operator\(\)](#) (const [Particle](#) &source, [Particle](#) &target) const
Computes and applies the force implied by source on target.

7.6.1 Description détaillée

Store a function that rules an interaction between particles.

7.6.2 Documentation des constructeurs et destructeur

7.6.2.1 Interaction()

```
Interaction::Interaction (
    std::function< void(const Particle &, Particle &)> interactionFunction ) [inline]
```

7.6.3 Documentation des fonctions membres

7.6.3.1 operator()()

```
void Interaction::operator() (
    const Particle & source,
    Particle & target ) const [inline]
```

Computes and applies the force implied by source on target.

Paramètres

<i>source</i>	the particle that applies force
<i>target</i>	the particle that receives

Renvoie

[Vector](#)

La documentation de cette classe a été générée à partir du fichier suivant :

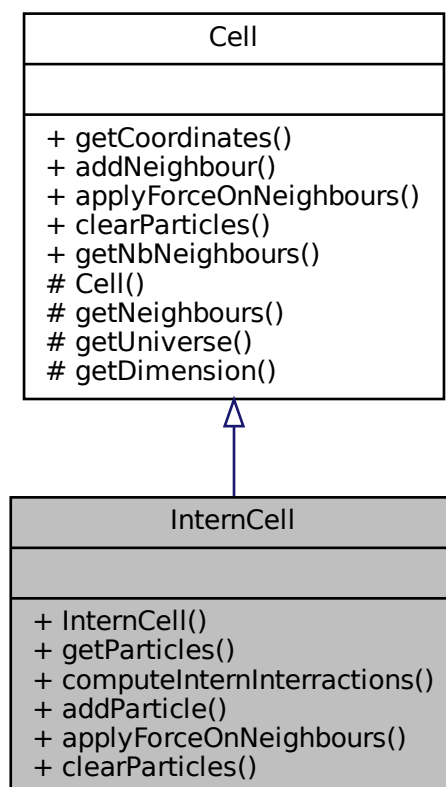
— [include/interraction.hpp](#)

7.7 Référence de la classe InternCell

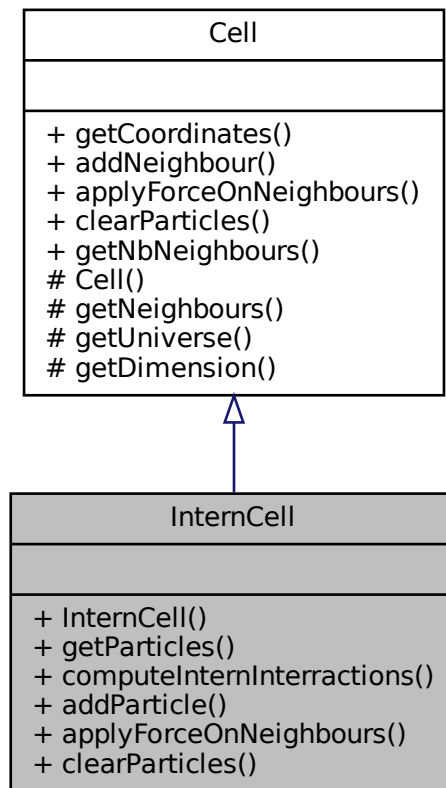
Limits Cells are contains copies of the cellule of same coordinates, with a position offset.

```
#include <intern_cell.hpp>
```

Graphe d'héritage de InternCell:



Graphe de collaboration de InternCell:



Fonctions membres publiques

- `InternCell` (`size_t dimension`, `GriddedUniverse *universe`, `std::vector< int > coordinates`)
- `const std::list< Particle * > getParticles ()` const
- `void computeInternInteractions ()`
apply forces between particles in the cell (adds to existing forces)
- `void addParticle (Particle *p)`
Adds a cell to the neighbours Dimensions must match.
- `void applyForceOnNeighbours ()` const override
applies forces on neighbours particles (adds to existing forces)
- `void clearParticles ()` override
Delete particles.

Membres hérités additionnels

7.7.1 Description détaillée

Limits Cells are contains copies of the cellule of same coordinates, with a position offset.

7.7.2 Documentation des constructeurs et destructeur

7.7.2.1 InternCell()

```
InternCell::InternCell (
    size_t dimension,
    GriddedUniverse * universe,
    std::vector< int > coordinates )
```

Voici le graphe d'appel pour cette fonction :



7.7.3 Documentation des fonctions membres

7.7.3.1 addParticle()

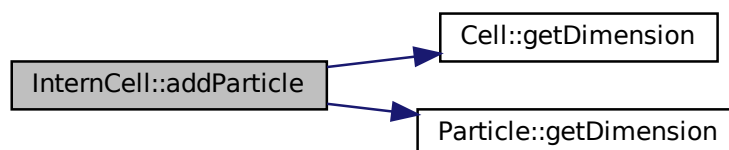
```
void InternCell::addParticle (
    Particle * p )
```

Adds a cell to the neighbours Dimensions must match.

Paramètres

<i>p</i>	pointer to particle
----------	---------------------

Voici le graphe d'appel pour cette fonction :



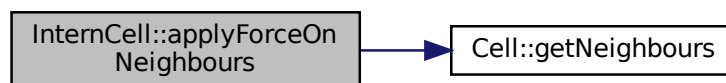
7.7.3.2 applyForceOnNeighbours()

```
void InternCell::applyForceOnNeighbours ( ) const [override], [virtual]
```

applies forces on neighbours particles (adds to existing forces)

Implémente [Cell](#).

Voici le graphe d'appel pour cette fonction :



7.7.3.3 clearParticles()

```
void InternCell::clearParticles ( ) [override], [virtual]
```

Delete particles.

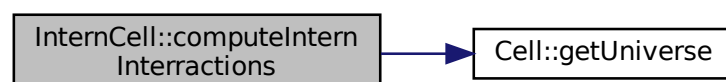
Implémente [Cell](#).

7.7.3.4 computeInternInteractions()

```
void InternCell::computeInternInteractions ( )
```

apply forces between particles in the cell (adds to existing forces)

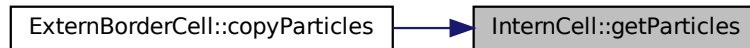
Voici le graphe d'appel pour cette fonction :



7.7.3.5 getParticles()

```
const std::list<Particle*> InternCell::getParticles ( ) const [inline]
```

Voici le graphe des appelants de cette fonction :



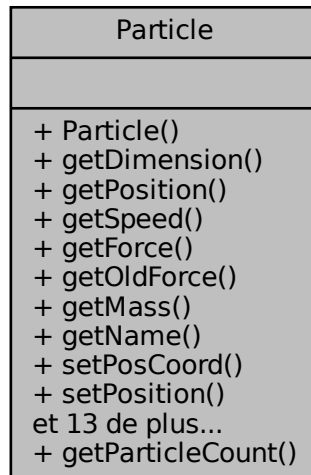
La documentation de cette classe a été générée à partir du fichier suivant :

- include/[intern_cell.hpp](#)
- src/[intern_cell.cpp](#)

7.8 Référence de la classe Particle

```
#include <particle.hpp>
```

Graphe de collaboration de Particle:



Fonctions membres publiques

- `Particle (Vector pos, Vector speed, double mass, std::string name)`
Construct a new *Particle* object.
- `size_t getDimension () const`
- `const Vector & getPosition () const`
- `const Vector & getSpeed () const`
- `const Vector & getForce () const`
- `const Vector & getOldForce () const`
- `double getMass () const`
- `const std::string & getName () const`
- `void setPosCoord (size_t coord, double value)`
- `void setPosition (const Vector &pos)`
- `void setSpeed (const Vector &speed)`
- `void setForce (const Vector &force)`
- `void setOldForce (const Vector &oldForce)`
- `void multiplySpeed (double scalar)`
Multiply the speed by a scalar.
- `void addToForceCoord (size_t i, double value)`
Adds to the *i* th force coordinate the value given.
- `void addToForce (const Vector &vect)`
Adds to existing force the vector given.
- `void addToPosition (const Vector &vect)`
Adds to existing position the vector given.
- `void addToSpeed (const Vector &vect)`
Adds to existing speed the vector given.
- `void setForceToZero ()`
Set the force of particle to zero.
- `double distanceTo (const Particle &other) const`
Calculates distance from this particle to the other.
- `void invertSpeed (size_t i)`
invert the speed coordinate *i*
- `void applyExternalForces (const std::list< ExternalForce > &extForces)`
Apply to a particle the external forces (adds to existing force)
- `void applyInteractionForcesOn (Particle &other, const std::list< Interaction > &interactions) const`
Apply to a particle the force implied by calling particle (adds to existing force)

Fonctions membres publiques statiques

- `static int getParticleCount ()`

Amis

- `std::ostream & operator<< (std::ostream &strm, const Particle &p)`
Overrides the << operator.

7.8.1 Documentation des constructeurs et destructeur

7.8.1.1 Particle()

```
Particle::Particle (
    Vector pos,
    Vector speed,
    double mass,
    std::string name )
```

Construct a new *Particle* object.

Paramètres

<i>dimension</i>	dimension of the particle
<i>pos</i>	
<i>speed</i>	
<i>mass</i>	
<i>name</i>	

Voici le graphe d'appel pour cette fonction :



7.8.2 Documentation des fonctions membres

7.8.2.1 addToForce()

```
void Particle::addToForce (
    const Vector & vect )
```

Adds to existing force the vector given.

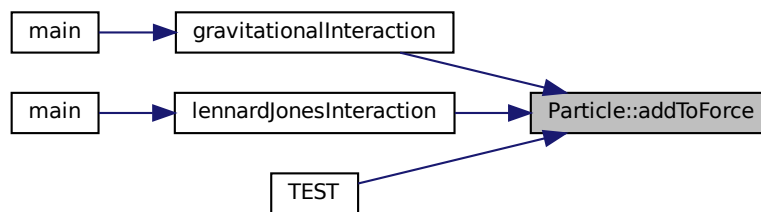
Paramètres

<i>vect</i>	
-------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.8.2.2 addToForceCoord()

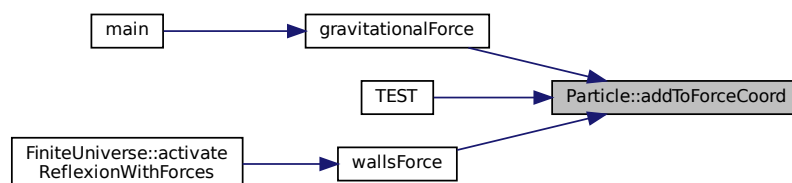
```
void Particle::addToForceCoord (
    size_t i,
    double value )
```

Adds to the *i* th force coordinate the value given.

Paramètres

<i>i</i>	
<i>value</i>	

Voici le graphe des appelants de cette fonction :



7.8.2.3 addToPosition()

```
void Particle::addToPosition (
    const Vector & vect )
```

Adds to existing position the vector given.

Paramètres

<i>vect</i>	
-------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.8.2.4 addToSpeed()

```
void Particle::addToSpeed (
    const Vector & vect )
```

Adds to existing speed the vector given.

Paramètres

<i>vect</i>	
-------------	--

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.8.2.5 applyExternalForces()

```
void Particle::applyExternalForces (
    const std::list< ExternalForce > & extForces )
```

Apply to a particle the extarnel forces (adds to existing force)

Paramètres

<i>extForces</i>	
------------------	--

7.8.2.6 applyInteractionForcesOn()

```
void Particle::applyInteractionForcesOn (
    Particle & other,
    const std::list< Interaction > & interactions ) const
```

Apply to a particle the force implied by calling particle (adds to existing force)

Paramètres

<i>other</i>	the particle to apply force on
--------------	--------------------------------

7.8.2.7 distanceTo()

```
double Particle::distanceTo (
    const Particle & other ) const
```

Calculates distance from this particle to the other.

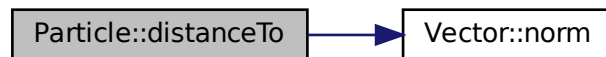
Paramètres

<i>other</i>	
--------------	--

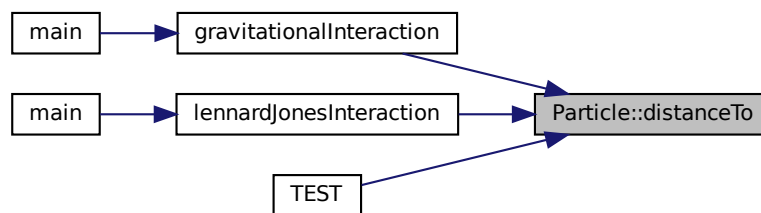
Renvoie

double

Voici le graphe d'appel pour cette fonction :

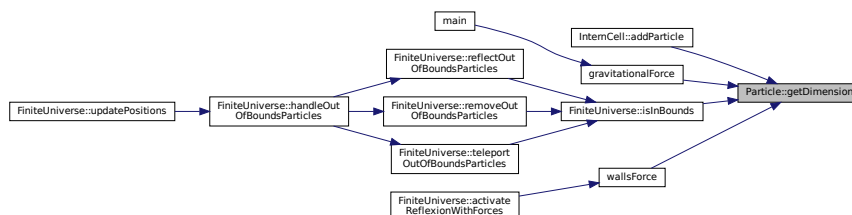


Voici le graphe des appelants de cette fonction :

7.8.2.8 `getDimension()`

```
size_t Particle::getDimension ( ) const [inline]
```

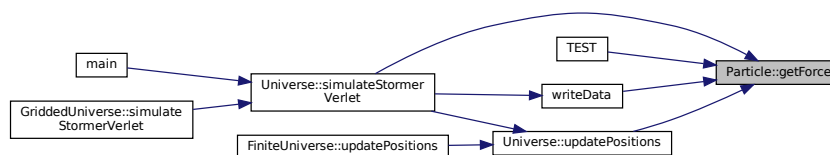
Voici le graphe des appelants de cette fonction :



7.8.2.9 getForce()

```
const Vector& Particle::getForce ( ) const [inline]
```

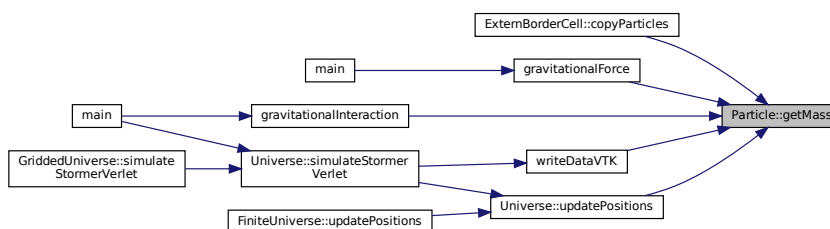
Voici le graphe des appelants de cette fonction :



7.8.2.10 getMass()

```
double Particle::getMass ( ) const [inline]
```

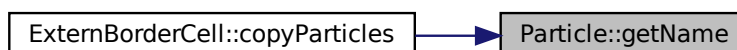
Voici le graphe des appelants de cette fonction :



7.8.2.11 getName()

```
const std::string& Particle::getName ( ) const [inline]
```

Voici le graphe des appelants de cette fonction :



7.8.2.12 getOldForce()

```
const Vector& Particle::getOldForce ( ) const [inline]
```

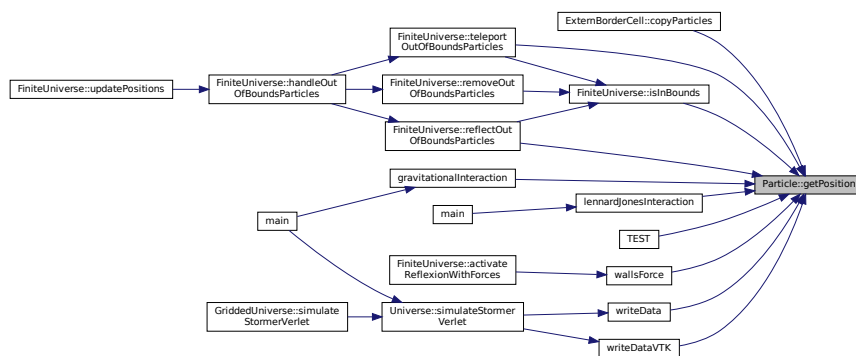
7.8.2.13 getParticleCount()

```
static int Particle::getParticleCount ( ) [inline], [static]
```

7.8.2.14 getPosition()

```
const Vector& Particle::getPosition ( ) const [inline]
```

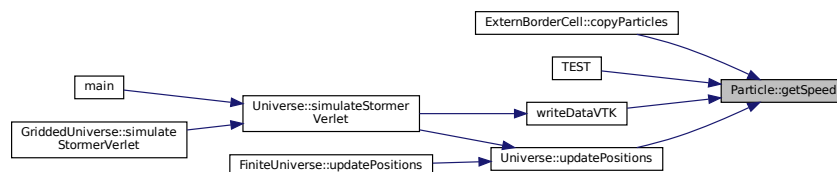
Voici le graphe des appelants de cette fonction :



7.8.2.15 getSpeed()

```
const Vector& Particle::getSpeed ( ) const [inline]
```

Voici le graphe des appelants de cette fonction :



7.8.2.16 invertSpeed()

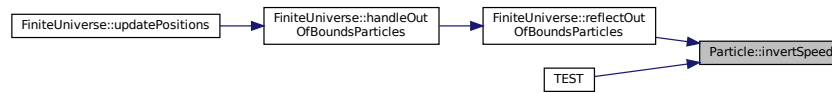
```
void Particle::invertSpeed (
    size_t i )
```

invert the speed coordinate i

Paramètres

<i>i</i>	
----------	--

Voici le graphe des appelants de cette fonction :



7.8.2.17 multiplySpeed()

```
void Particle::multiplySpeed (
    double scalar ) [inline]
```

Multiply the speed by a scalar.

Paramètres

<i>scalar</i>	
---------------	--

7.8.2.18 setForce()

```
void Particle::setForce (
    const Vector & force ) [inline]
```

7.8.2.19 setForceToZero()

```
void Particle::setForceToZero ( )
```

Set the force of particle to zero.

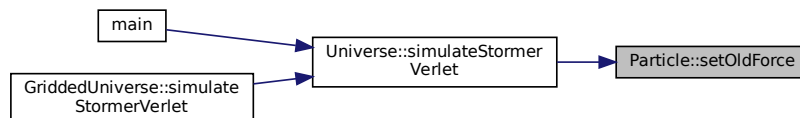
Voici le graphe des appelants de cette fonction :



7.8.2.20 setOldForce()

```
void Particle::setOldForce (
    const Vector & oldForce ) [inline]
```

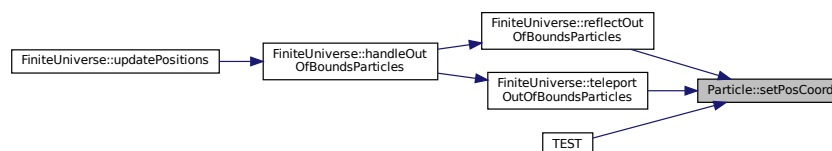
Voici le graphe des appelants de cette fonction :



7.8.2.21 setPosCoord()

```
void Particle::setPosCoord (
    size_t coord,
    double value )
```

Voici le graphe des appelants de cette fonction :



7.8.2.22 setPosition()

```
void Particle::setPosition (
    const Vector & pos ) [inline]
```

7.8.2.23 setSpeed()

```
void Particle::setSpeed (
    const Vector & speed ) [inline]
```

7.8.3 Documentation des fonctions amies et associées

7.8.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & strm,
    const Particle & p ) [friend]
```

Overrides the << operator.

Paramètres

<i>strm</i>	
<i>p</i>	

Renvoie

std::ostream&

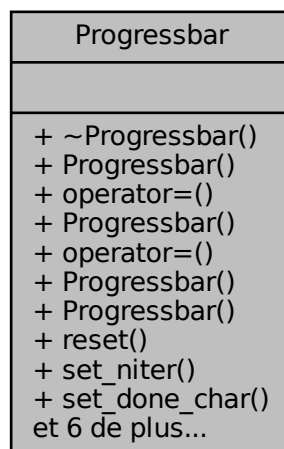
La documentation de cette classe a été générée à partir du fichier suivant :

- include/[particle.hpp](#)
- src/[particle.cpp](#)

7.9 Référence de la classe Progressbar

```
#include <progressbar.hpp>
```

Graphe de collaboration de Progressbar:



Fonctions membres publiques

- `~Progressbar()`=default
- `Progressbar(Progressbar const &)=delete`
- `Progressbar & operator= (Progressbar const &)=delete`
- `Progressbar(Progressbar &&)=delete`
- `Progressbar & operator= (Progressbar &&)=delete`
- `Progressbar()`
- `Progressbar(int n, bool showbar=true, std::ostream &out=std::cerr)`
- void `reset()`
- void `set_niter(int iter)`
- void `set_done_char(const std::string &sym)`
- void `set_todo_char(const std::string &sym)`
- void `set_opening_bracket_char(const std::string &sym)`
- void `set_closing_bracket_char(const std::string &sym)`
- void `show_bar(bool flag=true)`
- void `set_output_stream(const std::ostream &stream)`
- void `update()`

7.9.1 Documentation des constructeurs et destructeur

7.9.1.1 `~Progressbar()`

```
Progressbar::~~Progressbar ( ) [default]
```

7.9.1.2 `Progressbar()` [1/4]

```
Progressbar::Progressbar (
    Progressbar const & ) [delete]
```

7.9.1.3 `Progressbar()` [2/4]

```
Progressbar::Progressbar (
    Progressbar && ) [delete]
```

7.9.1.4 `Progressbar()` [3/4]

```
Progressbar::Progressbar ( ) [inline]
```

7.9.1.5 Progressbar() [4/4]

```
Progressbar::Progressbar (
    int n,
    bool showbar = true,
    std::ostream & out = std::cerr ) [inline]
```

7.9.2 Documentation des fonctions membres

7.9.2.1 operator=() [1/2]

```
Progressbar& Progressbar::operator= (
    Progressbar && ) [delete]
```

7.9.2.2 operator=() [2/2]

```
Progressbar& Progressbar::operator= (
    Progressbar const & ) [delete]
```

7.9.2.3 reset()

```
void Progressbar::reset ( ) [inline]
```

7.9.2.4 set_closing_bracket_char()

```
void Progressbar::set_closing_bracket_char (
    const std::string & sym ) [inline]
```

7.9.2.5 set_done_char()

```
void Progressbar::set_done_char (
    const std::string & sym ) [inline]
```

7.9.2.6 set_niter()

```
void Progressbar::set_niter (
    int iter ) [inline]
```

7.9.2.7 set_opening_bracket_char()

```
void Progressbar::set_opening_bracket_char (
    const std::string & sym ) [inline]
```

7.9.2.8 set_output_stream()

```
void Progressbar::set_output_stream (
    const std::ostream & stream ) [inline]
```

7.9.2.9 set_todo_char()

```
void Progressbar::set_todo_char (
    const std::string & sym ) [inline]
```

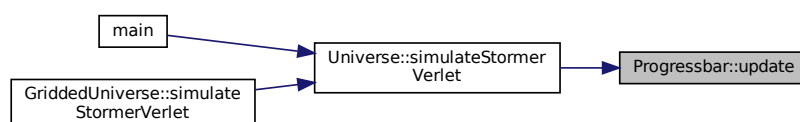
7.9.2.10 show_bar()

```
void Progressbar::show_bar (
    bool flag = true ) [inline]
```

7.9.2.11 update()

```
void Progressbar::update ( ) [inline]
```

Voici le graphe des appelants de cette fonction :



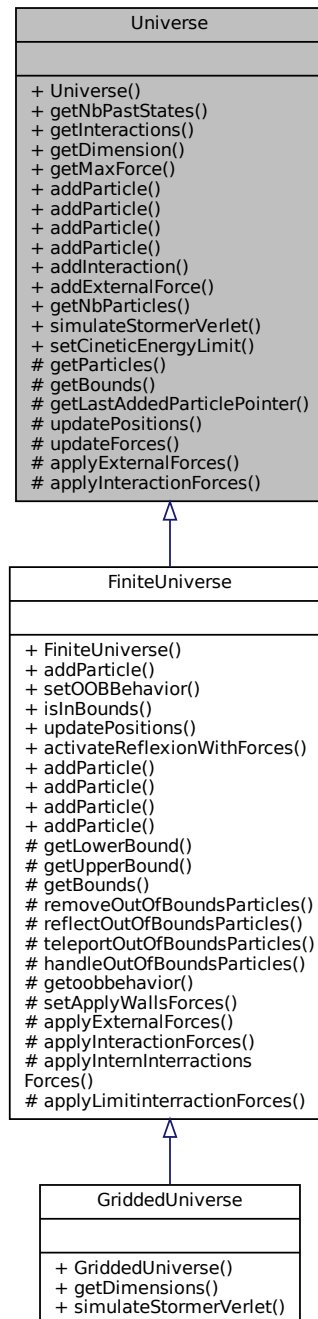
La documentation de cette classe a été générée à partir du fichier suivant :
— [include/progressbar.hpp](#)

7.10 Référence de la classe Universe

Represents an universe containing particles. Can be in dimension 1, 2, or 3.

```
#include <universe.hpp>
```

Graphe d'héritage de Universe:



Graphe de collaboration de Universe:



Fonctions membres publiques

- [Universe](#) (size_t dimension)
Constructor of class [Universe](#).
- size_t [getNbPastStates](#) () const
Get the number of states the universe has been in in the past (not including present).
- const std::list< [Interaction](#) > & [getInteractions](#) () const
Get the universe interactions.
- size_t [getDimension](#) () const
Get the dimension of the universe.
- double [getMaxForce](#) () const
Get the maximum force any particle has never felt.
- virtual void [addParticle](#) ([Vector](#) pos, [Vector](#) speed, double mass, std::string name)
Adds a particle into the universe.
- void [addParticle](#) (std::initializer_list< double > posCoords, std::initializer_list< double > speedCoords, double mass, std::string name)
Adds a particle into the universe.
- void [addParticle](#) ([Vector](#) pos, [Vector](#) speed, double mass)
Adds a particle into the universe. Give it automaticly a name.
- void [addParticle](#) (std::initializer_list< double > posCoords, std::initializer_list< double > speedCoords, double mass)
Adds a particle into the universe. Give it automaticly a name.
- void [addInteraction](#) (std::function< void(const [Particle](#) &, [Particle](#) &)> interactionFunction)
Adds interaction between two particles in the universe. The force function must compute the force applied by the 1st argument particle on the 2nd.
- void [addExternalForce](#) (std::function< void([Particle](#) &)> forceFunction)
Adds force on particles in the universe.

- int [getNbParticles](#) () const
Renvoie le nombre de particules dans l'univers.
- virtual void [simulateStormerVerlet](#) (double timeStep, double finalTime)
Simulates the movement of particles in the universe using the Störmer-Verlet method.
- void [setCineticEnergyLimit](#) (double cineticEnergyLimit)
Set the total Cinetic Energy Limit for the universe. Avoids speed divergence.

Fonctions membres protégées

- std::list< [Particle](#) > & [getParticles](#) ()
Get list of particles reference.
- virtual std::pair< [Vector](#), [Vector](#) > [getBounds](#) () const
Get the bounds of the universe, i.e. the min and max vectors of all past particles (not present particles)
- [Particle](#) * [getLastAddedParticlePointer](#) ()
Get the Last [Particle](#) Pointer Useful in the method addParticle of [GriddedUniverse](#).
- virtual void [updatePositions](#) (double timeStep)
Updates particles positions in Stormer Verlet algorithm.
- virtual void [updateForces](#) ()
Updates forces applied on particles in the universe. (Forces from interactions and external forces)
- virtual void [applyExternalForces](#) ()
Applies external forces on particles in the universe. Linear complexity.
- virtual void [applyInteractionForces](#) ()
Applies forces between particles in the universe. Quadratic complexity.

Amis

- class [VisualGenerator](#)
- std::ostream & [operator<<](#) (std::ostream &strm, const [Universe](#) &univers)
Surcharge de l'opérateur de flux de sortie pour afficher les informations sur l'univers.

7.10.1 Description détaillée

Represents an universe containing particles. Can be in dimension 1, 2, or 3.

7.10.2 Documentation des constructeurs et destructeur

7.10.2.1 Universe()

```
Universe::Universe (
    size_t dimension )
```

Constructor of class [Universe](#).

Paramètres

<i>dimension</i>	Universe dimension (1D, 2D or 3D)
------------------	---

7.10.3 Documentation des fonctions membres

7.10.3.1 addExternalForce()

```
void Universe::addExternalForce (
    std::function< void(Particle &)> forceFunction )
```

Adds force on particles in the universe.

Paramètres

<i>forceFunction</i>	
----------------------	--

7.10.3.2 addInteraction()

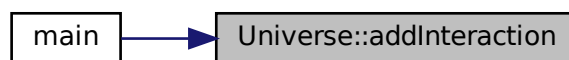
```
void Universe::addInteraction (
    std::function< void(const Particle &, Particle &)> interactionFunction )
```

Adds interaction between two particles in the universe. The force function must compute the force applied by the 1st argument particle on the 2nd.

Paramètres

<i>interactionFunction</i>	
----------------------------	--

Voici le graphe des appelants de cette fonction :



7.10.3.3 addParticle() [1/4]

```
void Universe::addParticle (
    std::initializer_list< double > posCoords,
    std::initializer_list< double > speedCoords,
    double mass )
```

Adds a particle into the universe. Give it automatically a name.

Paramètres

<i>posCoords</i>	
<i>speedCoords</i>	
<i>mass</i>	

Voici le graphe d'appel pour cette fonction :

7.10.3.4 `addParticle()` [2/4]

```

void Universe::addParticle (
    std::initializer_list< double > posCoords,
    std::initializer_list< double > speedCoords,
    double mass,
    std::string name )
  
```

Adds a particle into the universe.

Paramètres

<i>dimension</i>	
<i>posCoords</i>	
<i>speedCoords</i>	
<i>mass</i>	
<i>name</i>	

Voici le graphe d'appel pour cette fonction :



7.10.3.5 addParticle() [3/4]

```
void Universe::addParticle (
    Vector pos,
    Vector speed,
    double mass )
```

Adds a particle into the universe. Give it automaticly a name.

Paramètres

<i>pos</i>	
<i>speed</i>	
<i>mass</i>	

Voici le graphe d'appel pour cette fonction :

**7.10.3.6 addParticle()** [4/4]

```
void Universe::addParticle (
    Vector pos,
    Vector speed,
    double mass,
    std::string name ) [virtual]
```

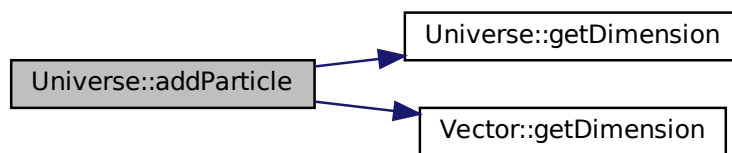
Adds a particle into the universe.

Paramètres

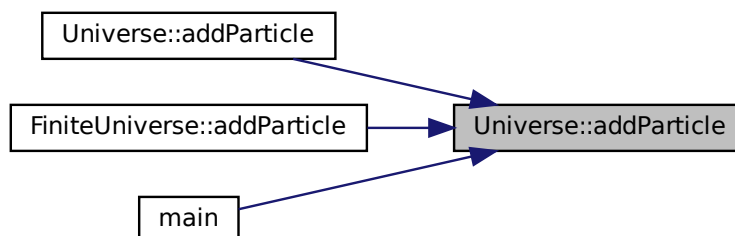
<i>dimension</i>	
<i>pos</i>	
<i>speed</i>	
<i>mass</i>	
<i>name</i>	

Réimplémentée dans [FiniteUniverse](#).

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



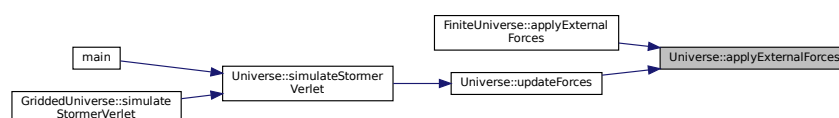
7.10.3.7 applyExternalForces()

```
void Universe::applyExternalForces ( ) [protected], [virtual]
```

Applies external forces on particles in the universe. Linear complexity.

Réimplémentée dans [FiniteUniverse](#).

Voici le graphe des appelants de cette fonction :



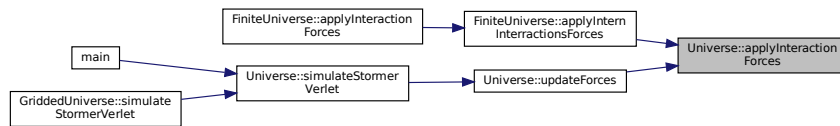
7.10.3.8 applyInteractionForces()

```
void Universe::applyInteractionForces ( ) [protected], [virtual]
```

Applies forces between particles in the universe. Quadratic complexity.

Réimplémentée dans [FiniteUniverse](#).

Voici le graphe des appelants de cette fonction :



7.10.3.9 getBounds()

```
std::pair< Vector, Vector > Universe::getBounds ( ) const [protected], [virtual]
```

Get the bounds of the universe, i.e. the min and max vectors of all past particles (not present particles)

Renvoie

```
const std::pair<Vector, Vector>&
```

Réimplémentée dans [FiniteUniverse](#).

7.10.3.10 getDimension()

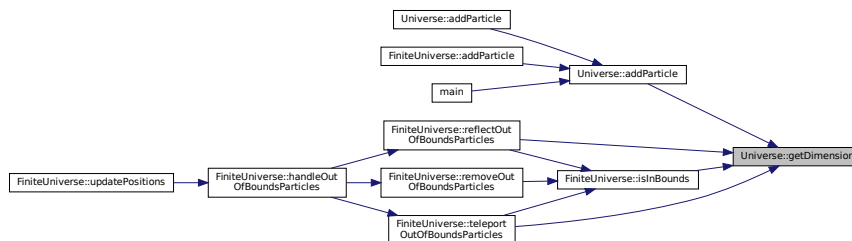
```
size_t Universe::getDimension ( ) const [inline]
```

Get the dimension of the universe.

Renvoie

```
size_t
```

Voici le graphe des appelants de cette fonction :



7.10.3.11 getInteractions()

```
const std::list<Interaction>& Universe::getInteractions ( ) const [inline]
```

Get the universe interactions.

Renvoie

```
const std::list<Interaction>&
```

7.10.3.12 getLastAddedParticlePointer()

```
Particle* Universe::getLastAddedParticlePointer ( ) [inline], [protected]
```

Get the Last [Particle](#) Pointer Useful in the method `addParticle` of [GriddedUniverse](#).

Renvoie

```
Particle*
```

7.10.3.13 getMaxForce()

```
double Universe::getMaxForce ( ) const [inline]
```

Get the maximum force any particle has never felt.

Renvoie

```
double
```

7.10.3.14 getNbParticles()

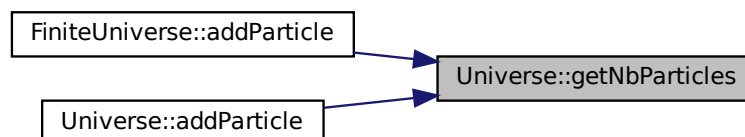
```
int Universe::getNbParticles ( ) const [inline]
```

Renvoie le nombre de particules dans l'univers.

Renvoie

Le nombre de particules dans l'univers

Voici le graphe des appelants de cette fonction :



7.10.3.15 getNbPastStates()

```
size_t Universe::getNbPastStates ( ) const [inline]
```

Get the number of states the universe has been in in the past (not including present).

Renvoie

size_t

7.10.3.16 getParticles()

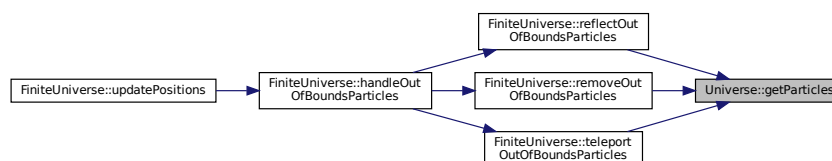
```
std::list<Particle>& Universe::getParticles ( ) [inline], [protected]
```

Get list of particles reference.

Renvoie

std::list<Particle>&

Voici le graphe des appelants de cette fonction :



7.10.3.17 setCineticEnergyLimit()

```
void Universe::setCineticEnergyLimit (
    double cineticEnergyLimit ) [inline]
```

Set the total Cinetic Energy Limit for the universe. Avoids speed divergence.

Paramètres

<i>cel</i>	
------------	--

7.10.3.18 simulateStormerVerlet()

```
void Universe::simulateStormerVerlet (
    double timeStep,
    double finalTime ) [virtual]
```

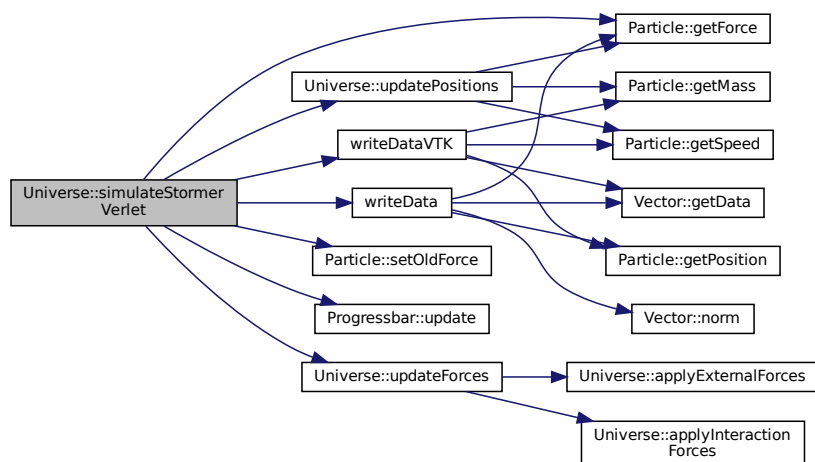
Simulates the movement of particles in the universe using the Störmer-Verlet method.

Paramètres

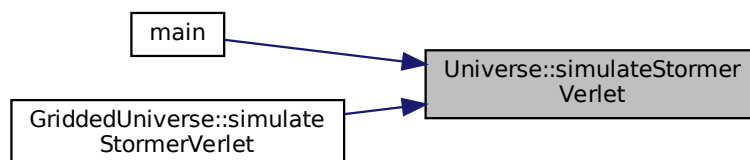
<i>timeStep</i>	
<i>finalTime</i>	End time of the simulation

Réimplémentée dans [GriddedUniverse](#).

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

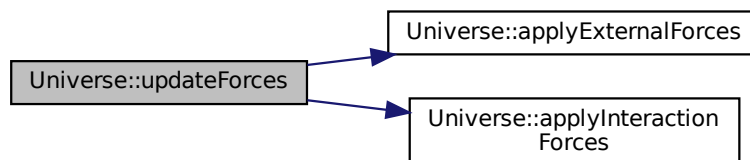


7.10.3.19 updateForces()

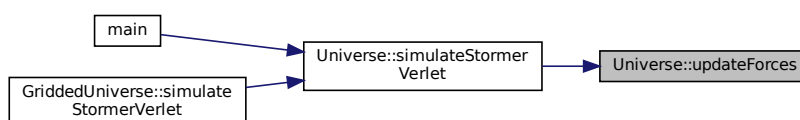
```
void Universe::updateForces ( ) [protected], [virtual]
```

Updates forces applied on particles in the universe. (Forces from interactions and external forces)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



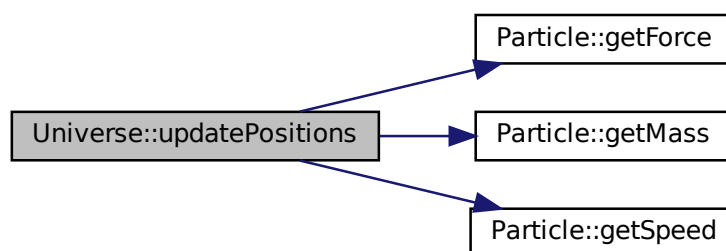
7.10.3.20 updatePositions()

```
void Universe::updatePositions ( double timeStep ) [protected], [virtual]
```

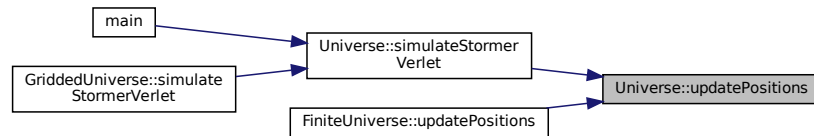
Updates particles positions in Stormer Verlet algorithm.

Réimplémentée dans [FiniteUniverse](#).

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



7.10.4 Documentation des fonctions amies et associées

7.10.4.1 `operator<<`

```
std::ostream& operator<< (
    std::ostream & strm,
    const Universe & univers ) [friend]
```

Surcharge de l'opérateur de flux de sortie pour afficher les informations sur l'univers.

Paramètres

<i>strm</i>	Le flux de sortie
<i>univers</i>	L'univers à afficher

Renvoie

`std::ostream&`, le flux de sortie

7.10.4.2 `VisualGenerator`

```
friend class VisualGenerator [friend]
```

La documentation de cette classe a été générée à partir du fichier suivant :

- [include/universe.hpp](#)
- [src/universe.cpp](#)

7.11 Référence de la classe `Vector`

`Vector` class.

```
#include <vector.hpp>
```


Graphe de collaboration de Vector:

Vector
<ul style="list-style-type: none"> + Vector() + Vector() + Vector() + operator[]() + operator[]() + getData() + getDimension() + operator==() + operator!=() + operator+=() + operator-=() + operator*=() + areAllCoordsGreater() + isInBounds() + norm()

Fonctions membres publiques

- `Vector` (size_t size)
- `Vector` ()
- `Vector` (std::initializer_list< double > values)
- double & `operator[]` (size_t index)
- double `operator[]` (size_t index) const
- const std::vector< double > & `getData` () const
- size_t `getDimension` () const
- bool `operator==` (const `Vector` &other) const
- bool `operator!=` (const `Vector` &other) const
- void `operator+=` (const `Vector` &other)
- void `operator-=` (const `Vector` &other)
- void `operator*=` (double scalar)
- bool `areAllCoordsGreater` (const `Vector` &other) const
- *Tell for each dimension if the coords of the calling vector is greater than the other vector given.*
- bool `isInBounds` (const `Vector` &lowerBound, const `Vector` &upperBound) const
- *Tell if a vector is in a certain cubic space.*
- double `norm` () const
- *give the norm of the vector i.e. square root of sum of squares*

Amis

- `Vector` min (`Vector` v1, `Vector` v2)
- *Returns vector with min on each coordinates.*
- `Vector` max (`Vector` v1, `Vector` v2)
- *Returns vector with max on each coordinates.*
- std::ostream & `operator<<` (std::ostream &strm, const `Vector` &v)
- *Overrides the << operator.*

7.11.1 Description détaillée

[Vector](#) class.

7.11.2 Documentation des constructeurs et destructeur

7.11.2.1 [Vector\(\)](#) [1/3]

```
Vector::Vector (
    size_t size ) [inline]
```

7.11.2.2 [Vector\(\)](#) [2/3]

```
Vector::Vector ( ) [inline]
```

7.11.2.3 [Vector\(\)](#) [3/3]

```
Vector::Vector (
    std::initializer_list< double > values ) [inline]
```

7.11.3 Documentation des fonctions membres

7.11.3.1 [areAllCoordsGreater\(\)](#)

```
bool Vector::areAllCoordsGreater (
    const Vector & other ) const
```

Tell for each dimension if the coords of the calling vector is greater than the other vector given.

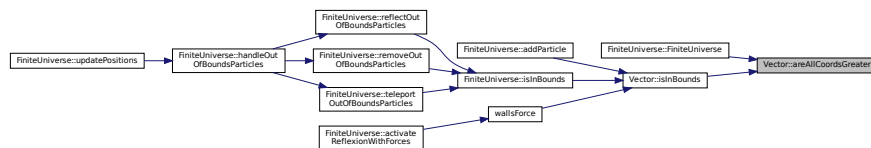
Paramètres

<i>other</i>	
--------------	--

Renvoie

true if all coords are greater
false if not

Voici le graphe des appelants de cette fonction :



7.11.3.2 getData()

```
const std::vector<double>& Vector::getData ( ) const [inline]
```

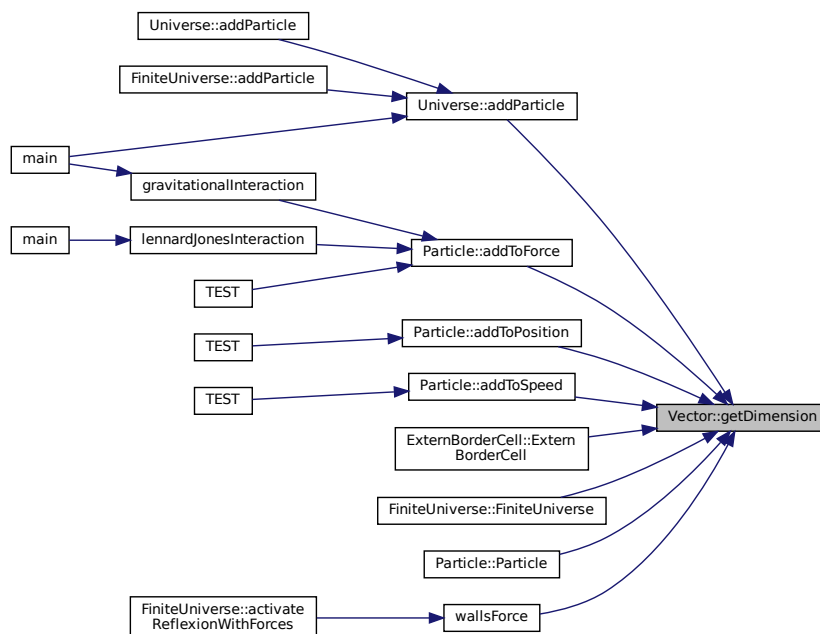
Voici le graphe des appelants de cette fonction :



7.11.3.3 getDimension()

```
size_t Vector::getDimension ( ) const [inline]
```

Voici le graphe des appelants de cette fonction :



7.11.3.4 isInBounds()

```

bool Vector::isInBounds (
    const Vector & lowerBound,
    const Vector & upperBound ) const

```

Tell if a vector is in a certain cubic space.

Paramètres

<i>lowerBound</i>	
<i>upperBound</i>	

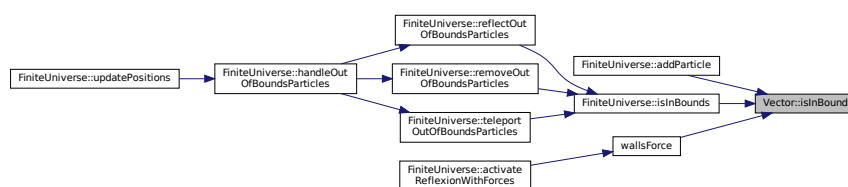
Renvoie

true if the vector is in bounds given
false if not

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**7.11.3.5 norm()**

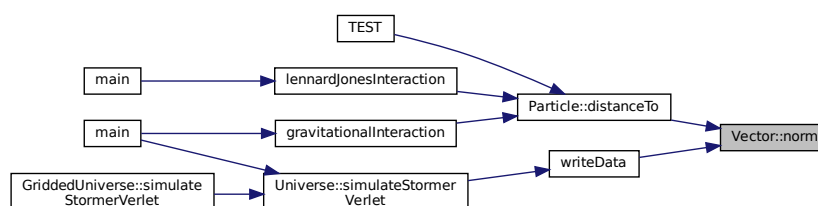
```
double Vector::norm ( ) const
```

give the norm of the vector i.e. square root of sum of squares

Renvoie

double

Voici le graphe des appelants de cette fonction :



7.11.3.6 operator!=()

```
bool Vector::operator!= (
    const Vector & other ) const
```

Voici le graphe d'appel pour cette fonction :



7.11.3.7 operator*=()

```
void Vector::operator*= (
    double scalar )
```

7.11.3.8 operator+=()

```
void Vector::operator+= (
    const Vector & other )
```

7.11.3.9 operator-=()

```
void Vector::operator-= (
    const Vector & other )
```

7.11.3.10 operator==()

```
bool Vector::operator== (
    const Vector & other ) const
```

Voici le graphe des appelants de cette fonction :



7.11.3.11 operator[]() [1/2]

```
double& Vector::operator[] (
    size_t index ) [inline]
```

7.11.3.12 operator[]() [2/2]

```
double Vector::operator[] (
    size_t index ) const [inline]
```

7.11.4 Documentation des fonctions amies et associées**7.11.4.1 max**

```
Vector max (
    Vector v1,
    Vector v2 ) [friend]
```

Returns vector with max on each coordinates.

Paramètres

<i>v1</i>	
<i>v2</i>	

Renvoie

Vector

7.11.4.2 min

```
Vector min (
    Vector v1,
    Vector v2 ) [friend]
```

Returns vector with min on each coordinates.

Paramètres

<i>v1</i>	
<i>v2</i>	

Renvoie

[Vector](#)

7.11.4.3 operator<<

```
std::ostream& operator<< (
    std::ostream & strm,
    const Vector & v ) [friend]
```

Overrides the << operator.

Paramètres

<i>strm</i>	
<i>v</i>	

Renvoie

std::ostream&

La documentation de cette classe a été générée à partir du fichier suivant :

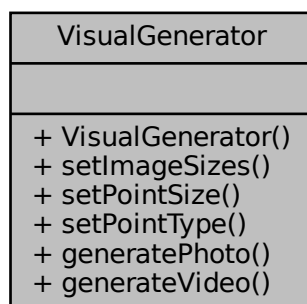
- include/[vector.hpp](#)
- src/[vector.cpp](#)

7.12 Référence de la classe VisualGenerator

Provides methods to generate visualisation of an universe.

```
#include <visual_generator.hpp>
```

Graphe de collaboration de VisualGenerator:



Fonctions membres publiques

- `VisualGenerator` (const `Universe` *universe)
- void `setImageSizes` (size_t width, size_t height)
Set the sizes for images that will be created.
- void `setPointSize` (double pointSize)
Set the size of the points for the plots.
- void `setPointType` (`PointType` pointType)
Set the shape of the points plotted Accepts NO_SYMBOL, PLUS, CROSS, STAR, BOX, BOX_F, CIRCLE, CIRCLE←
- void `generatePhoto` () const
Generates a photo of the current state of the universe.
- void `generateVideo` (size_t numberFrames) const
Generates a video (multiple images) of the past states the universe has been into.

7.12.1 Description détaillée

Provides methods to generate visualisation of an universe.

7.12.2 Documentation des constructeurs et destructeur

7.12.2.1 VisualGenerator()

```
VisualGenerator::VisualGenerator (
    const Universe * universe )
```

7.12.3 Documentation des fonctions membres

7.12.3.1 generatePhoto()

```
void VisualGenerator::generatePhoto ( ) const
```

Generates a photo of the current state of the universe.

7.12.3.2 generateVideo()

```
void VisualGenerator::generateVideo (
    size_t numberFrames ) const
```

Generates a video (multiple images) of the past states the universe has been into.

Paramètres

<i>numberFrames</i>	number of images to generate (must be smaller than past states)
---------------------	---

Voici le graphe des appelants de cette fonction :

**7.12.3.3 setImageSizes()**

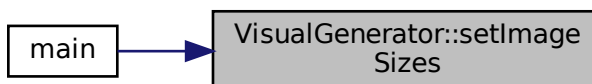
```
void VisualGenerator::setImageSizes (
    size_t width,
    size_t height )
```

Set the sizes for images that will be created.

Paramètres

<i>width</i>	
<i>height</i>	

Voici le graphe des appelants de cette fonction :

**7.12.3.4 setPointSize()**

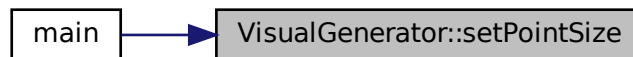
```
void VisualGenerator::setPointSize (
    double pointSize )
```

Set the size of the points for the plots.

Paramètres

<i>pointSize</i>	
------------------	--

Voici le graphe des appelants de cette fonction :



7.12.3.5 setPointType()

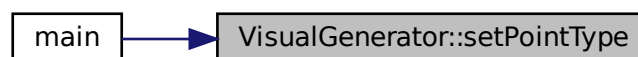
```
void VisualGenerator::setPointType (
    PointType pointType ) [inline]
```

Set the shape of the points plotted Accepts NO_SYMBOL, PLUS, CROSS, STAR, BOX, BOX_F, CIRCLE, CIRCLE_F.

Paramètres

<i>pointType</i>	
------------------	--

Voici le graphe des appelants de cette fonction :



La documentation de cette classe a été générée à partir du fichier suivant :

- [include/visual_generator.hpp](#)
- [src/visual_generator.cpp](#)

Chapitre 8

Documentation des fichiers

8.1 Référence du fichier docs/conception.md

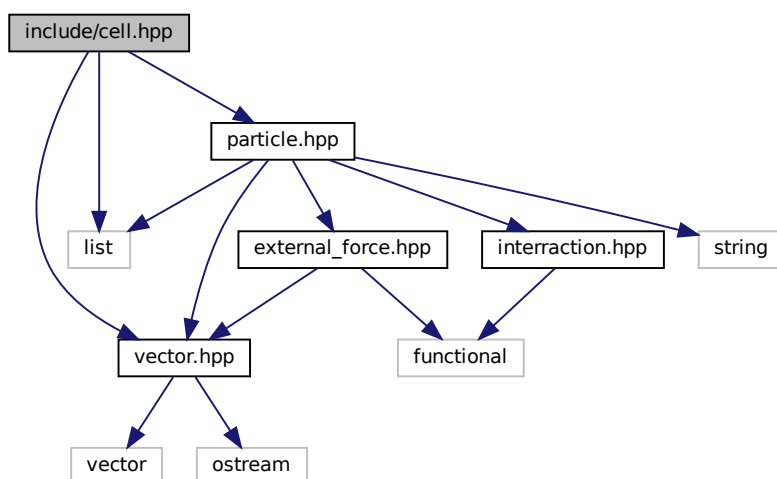
8.2 Référence du fichier docs/TO DO.md

8.3 Référence du fichier include/cell.hpp

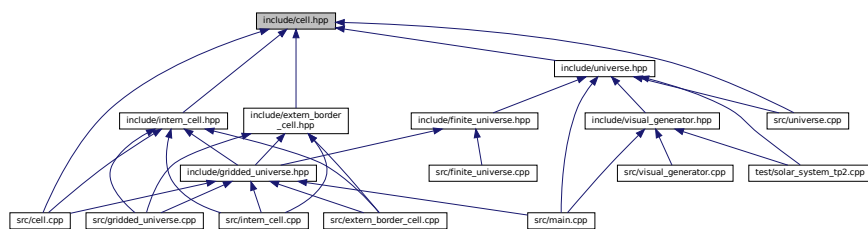
Les cellules sont les cases élémentaires formant un maillage de l'univers.

```
#include <list>
#include "vector.hpp"
#include "particle.hpp"
```

Graphe des dépendances par inclusion de cell.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [Cell](#)
Cells are elementary bricks forming a grid of the universe.

8.3.1 Description détaillée

Les cellules sont les cases élémentaires formant un maillage de l'univers.

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

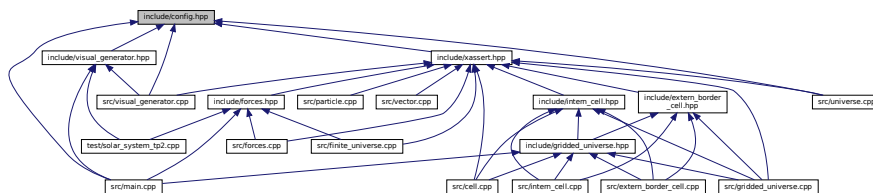
Date

2024-04-08

8.4 Référence du fichier include/config.hpp

Contains constants definitions for preprocessing of the code.

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Macros

- #define [SHOW_PROGRESS_INFOS](#)
Enables the display of progress information during the program's execution.
- #define [NDEBUG](#)
Disables all assertions (xassert) to potentially improve performance.
- #define [PNG_OUTPUT](#)
Enables the generation of output in PNG format.
- #define [XML_OUTPUT](#)
Enables the generation of output in XML format.

8.4.1 Description détaillée

Contains constants definitions for preprocessing of the code.

Auteur

jules roques (jules.roques@grenoble-inp.org)

This header file defines several preprocessor directives that control various aspects of the program's behavior, including debugging, output formats, and progress information.

Version

0.1

Date

2024-04-12

8.4.2 Documentation des macros

8.4.2.1 NDEBUG

```
#define NDEBUG
```

Disables all assertions (xassert) to potentially improve performance.

When defined, this directive deactivates all xassert statements in the code, which can help improve performance by avoiding the overhead of these checks. Uncomment this line to enable this behavior.

8.4.2.2 PNG_OUTPUT

```
#define PNG_OUTPUT
```

Enables the generation of output in PNG format.

When defined, this directive allows the program to write its output in PNG format. This is useful for generating visual representations of the simulation data.

8.4.2.3 SHOW_PROGRESS_INFOS

```
#define SHOW_PROGRESS_INFOS
```

Enables the display of progress information during the program's execution.

When defined, this directive allows the program to output information about its progress, which can be useful for monitoring and debugging long-running tasks.

8.4.2.4 XML_OUTPUT

```
#define XML_OUTPUT
```

Enables the generation of output in XML format.

When defined, this directive allows the program to write its output in XML format. This can be useful for exporting data in a structured and widely-used format for further processing or analysis. Uncomment this line to enable this behavior.

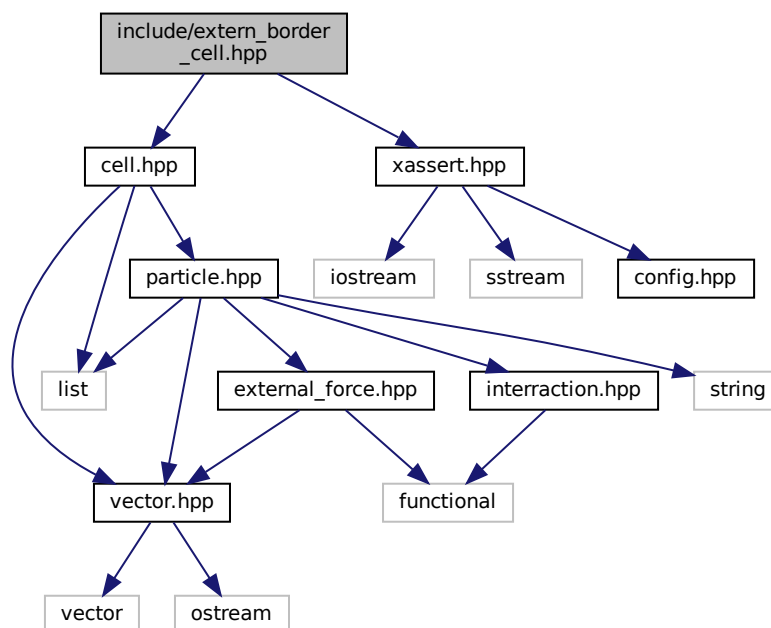
8.5 Référence du fichier include/extern_border_cell.hpp

Cells for limit behavior of a PERIODIC universe.

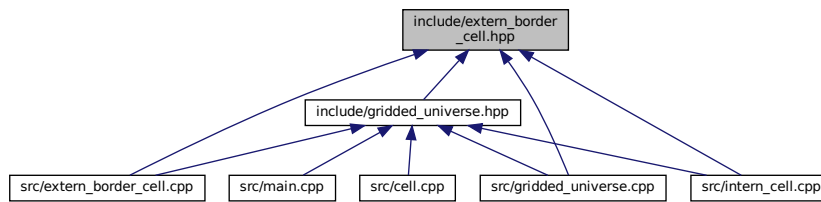
```
#include "cell.hpp"
```

```
#include "xassert.hpp"
```

Graphe des dépendances par inclusion de extern_border_cell.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

— class [ExternBorderCell](#)

Limits Cells contains copies of particles of a foreign cell, neighbour for a PERIODIC point of view. An offset is applied on the particles positions.

8.5.1 Description détaillée

Cells for limit behavior of a PERIODIC universe.

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

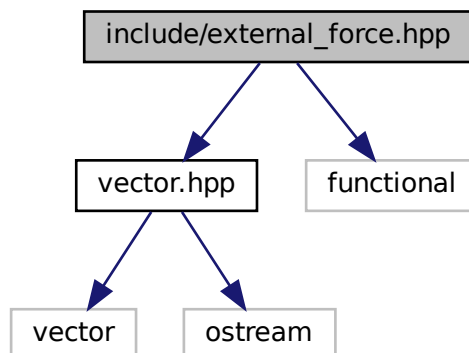
Date

2024-05-20

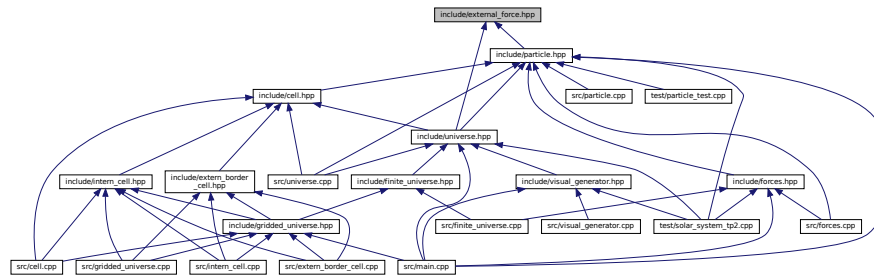
8.6 Référence du fichier include/external_force.hpp

```
#include "vector.hpp"
#include <functional>
```

Graphe des dépendances par inclusion de external_force.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

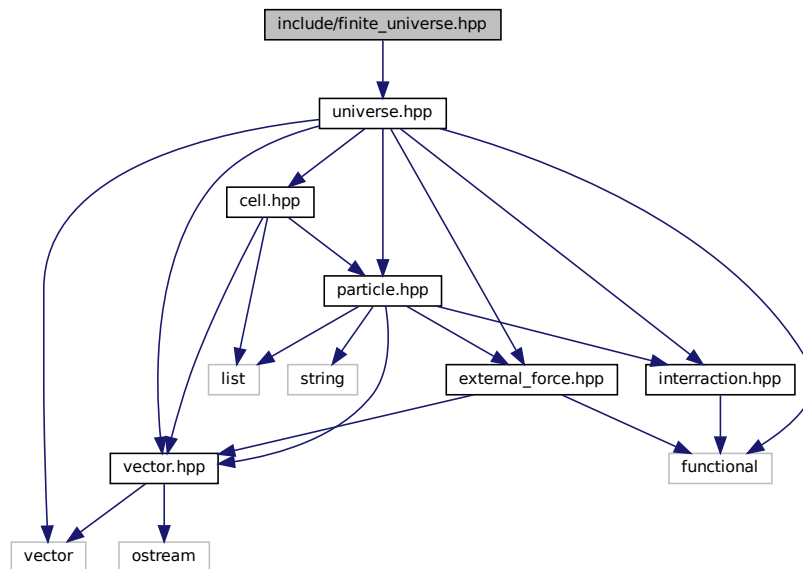
- class [ExternalForce](#)
Stores a function that rules a force applied on particles.

8.7 Référence du fichier include/finite_universe.hpp

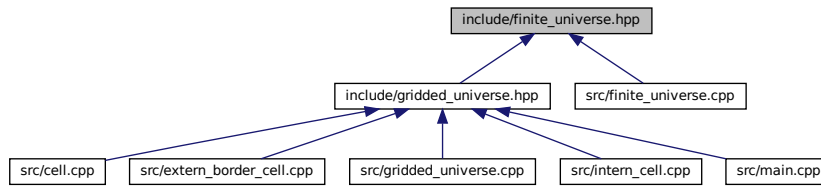
A universe with limits.

```
#include "universe.hpp"
```

Graphe des dépendances par inclusion de finite_universe.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

— class `FiniteUniverse`
Universe with bounds.

Énumérations

— enum `OOBBehavior` { `PERIODIC` , `REFLEXION` , `ABSORPTION` }

8.7.1 Description détaillée

A universe with limits.

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

Date

2024-04-13

8.7.2 Documentation du type de l'énumération

8.7.2.1 OOBBehavior

enum `OOBBehavior`

Valeurs énumérées

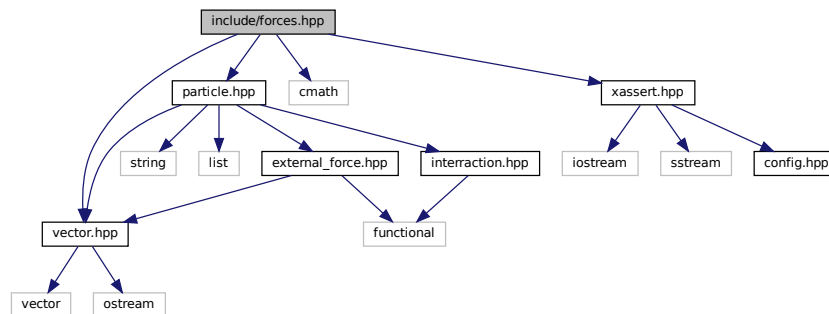
PERIODIC	
REFLEXION	
ABSORPTION	

8.8 Référence du fichier include/forces.hpp

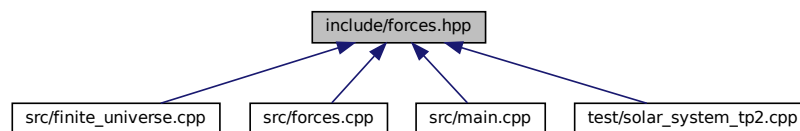
File containing different forces functions.

```
#include <vector.hpp>
#include <particle.hpp>
#include <cmath>
#include <xassert.hpp>
```

Graphe des dépendances par inclusion de forces.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Fonctions

- void [gravitationalInteraction](#) (const [Particle](#) &source, [Particle](#) &target)
Computes the gravitational force applied by source on target particle.
- void [lennardJonesInteraction](#) (const [Particle](#) &source, [Particle](#) &target, double epsilon, double sigma)
Computes the Lennard Jones force applied by source on target particle.
- void [gravitationalForce](#) ([Particle](#) &target, double G)
Adds the gravitational force applied on a particle to the existing force. The gravitational field is applied on the last dimension of the particle.
- void [wallsForce](#) ([Particle](#) &target, [Vector](#) lowerBound, [Vector](#) upperBound, double epsilon, double sigma)
Applies on the particle the strength of the walls (c.f. TP6)

8.8.1 Description détaillée

File containing different forces functions.

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

Date

2024-04-12

8.8.2 Documentation des fonctions

8.8.2.1 gravitationalForce()

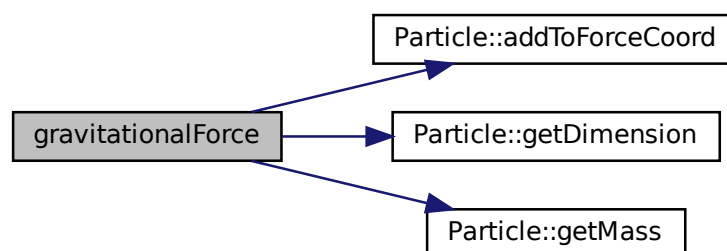
```
void gravitationalForce (
    Particle & target,
    double G )
```

Adds the gravitational force applied on a particle to the existing force. The gravitational field is applied on the last dimension of the particle.

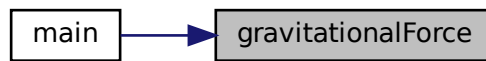
Paramètres

<i>source</i>	
<i>target</i>	

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



8.8.2.2 gravitationalInteraction()

```
void gravitationalInteraction (
    const Particle & source,
    Particle & target )
```

Computes the gravitational force applied by source on target particle.

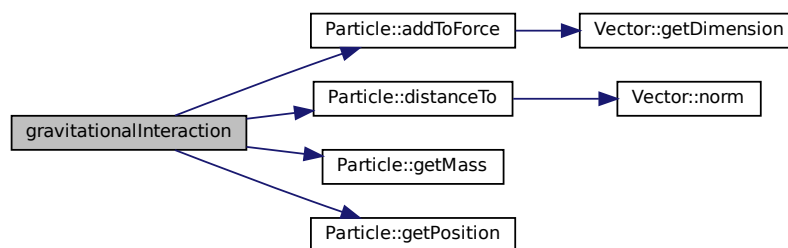
Paramètres

<i>source</i>	
<i>target</i>	

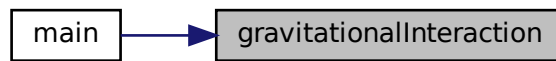
Renvoie

[Vector](#)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



8.8.2.3 lennardJonesInteraction()

```
void lennardJonesInteraction (
    const Particle & source,
    Particle & target,
    double epsilon,
    double sigma )
```

Computes the Lennard Jones force applied by source on target particle.

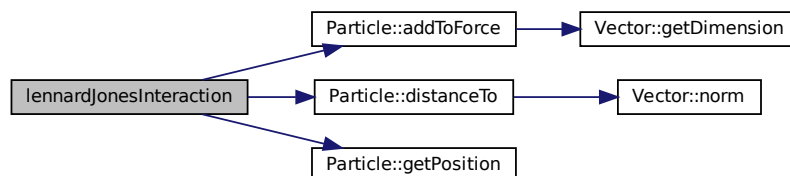
Paramètres

<i>source</i>	
<i>target</i>	
<i>epsilon</i>	
<i>sigma</i>	

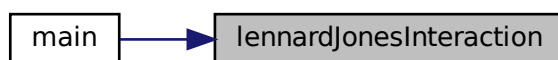
Renvoie

`Vector`

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



8.8.2.4 wallsForce()

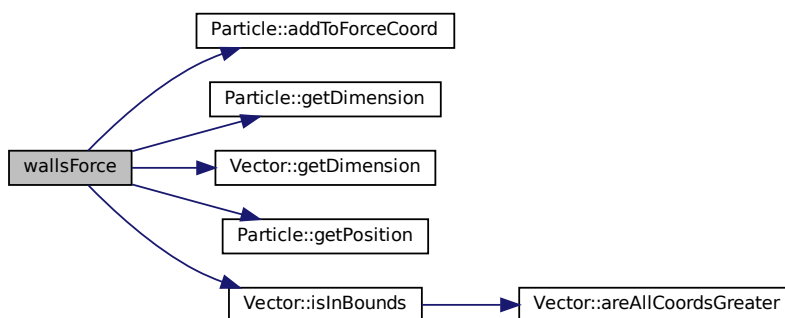
```
void wallsForce (
    Particle & target,
    Vector lowerBound,
    Vector upperBound,
    double epsilon,
    double sigma )
```

Applies on the particle the strength of the walls (c.f. TP6)

Paramètres

<i>target</i>	
<i>lowerBound</i>	
<i>upperBound</i>	

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



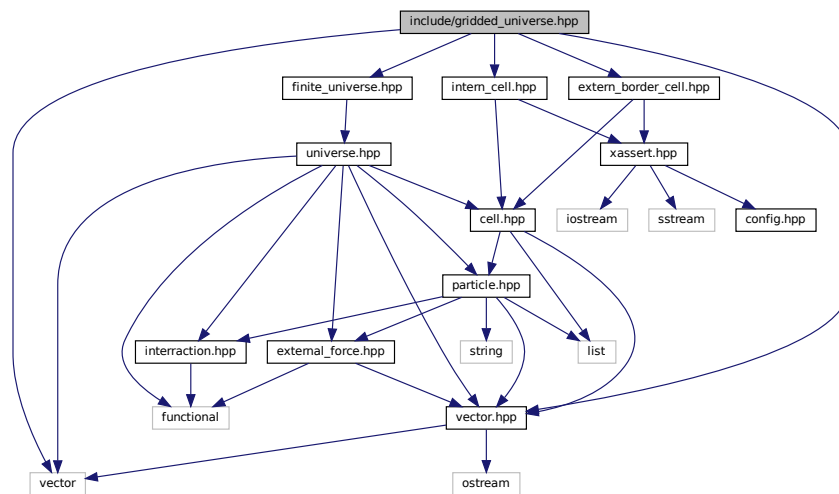
8.9 Référence du fichier include/gridded_universe.hpp

```

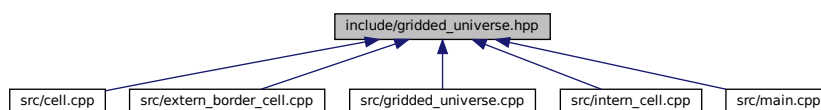
#include <vector>
#include "vector.hpp"
#include "finite_universe.hpp"
#include "intern_cell.hpp"
#include "extern_border_cell.hpp"

```

Graphe des dépendances par inclusion de gridded_universe.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



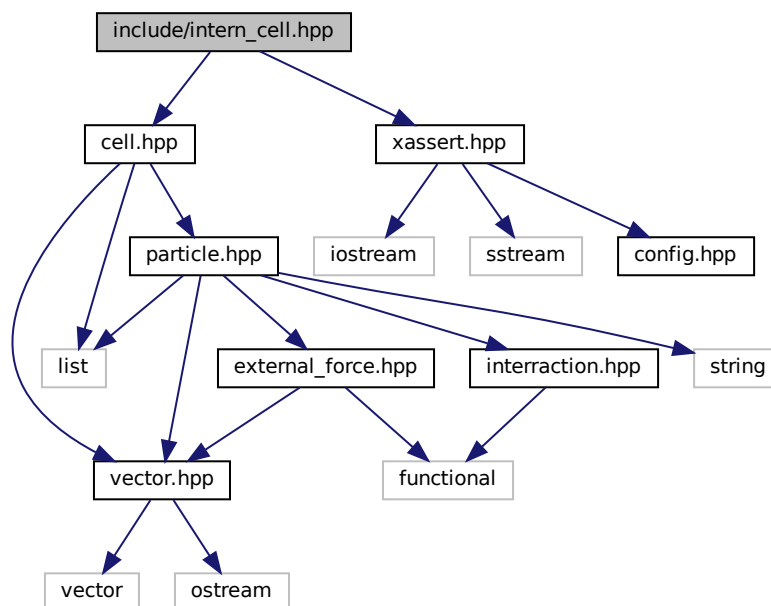
Classes

- class [GriddedUniverse](#)
A [GriddedUniverse](#) is a finite universe, separated into cells. Extends [Universe](#).

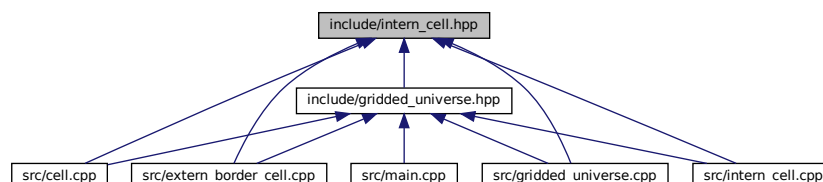
8.10 Référence du fichier include/intern_cell.hpp

```
#include "cell.hpp"
#include "xassert.hpp"
```

Graphe des dépendances par inclusion de intern_cell.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

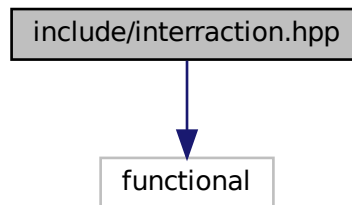
- class [InternCell](#)
Limits Cells are contains copies of the cellule of same coordinates, with a position offset.

8.11 Référence du fichier include/interraction.hpp

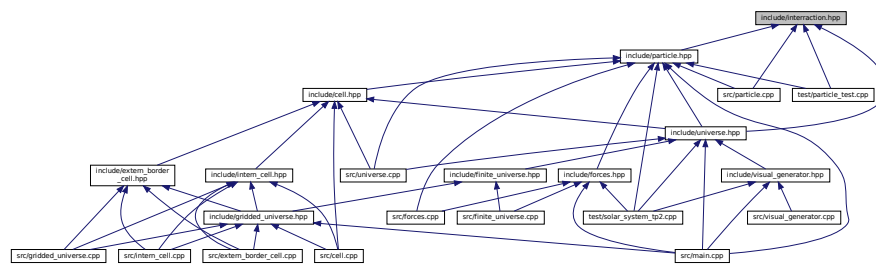
to define an interaction between two particles

```
#include <functional>
```

Graphe des dépendances par inclusion de interraction.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [Interaction](#)
Store a function that rules an interaction between particles.

8.11.1 Description détaillée

to define an interaction between two particles

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

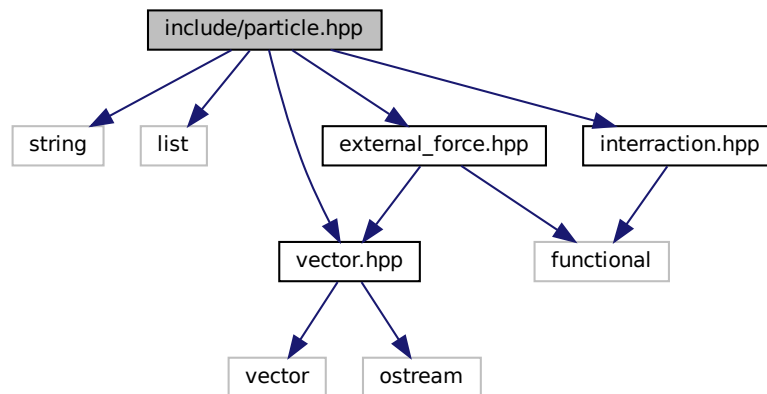
Date

2024-04-12

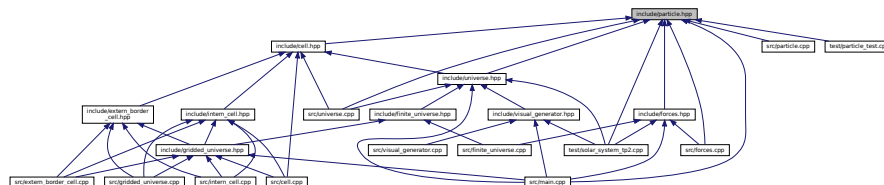
8.12 Référence du fichier include/particle.hpp

```
#include <string>
#include <list>
#include "vector.hpp"
#include "interaction.hpp"
#include "external_force.hpp"
```

Graphe des dépendances par inclusion de particle.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

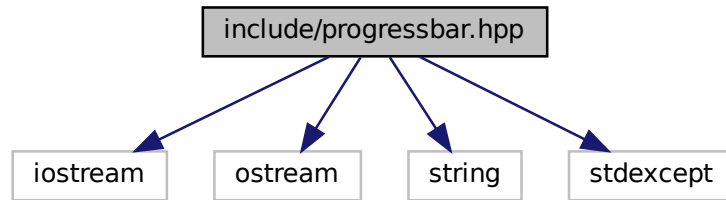
— class [Particle](#)

8.13 Référence du fichier include/progressbar.hpp

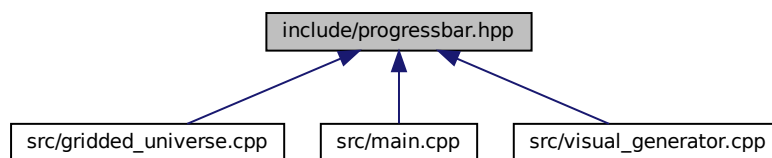
```
#include <iostream>
#include <ostream>
#include <string>
```

```
#include <stdexcept>
```

Graphe des dépendances par inclusion de progressbar.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



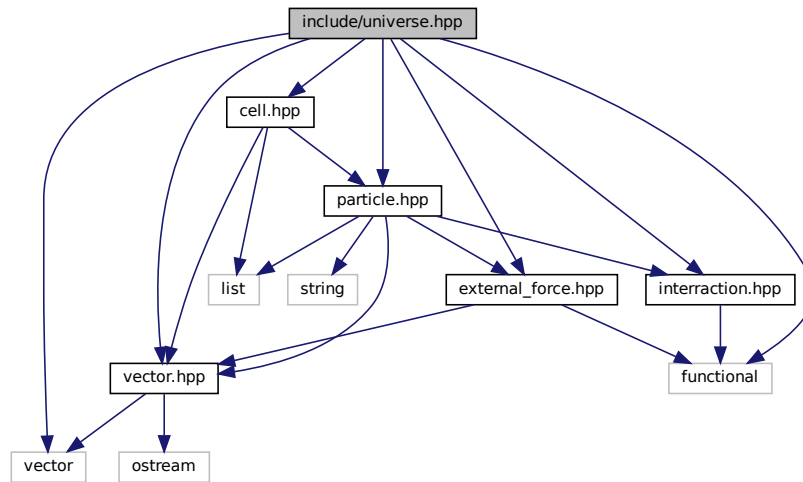
Classes

— class [Progressbar](#)

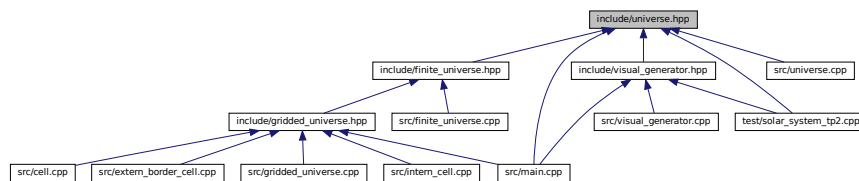
8.14 Référence du fichier include/universe.hpp

```
#include <vector>
#include <functional>
#include "particle.hpp"
#include "vector.hpp"
#include "cell.hpp"
#include "interaction.hpp"
#include "external_force.hpp"
```

Graphe des dépendances par inclusion de universe.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

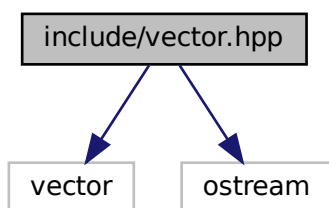
- class [Universe](#)
Represents an universe containing particles. Can be in dimension 1, 2, or 3.

8.15 Référence du fichier include/vector.hpp

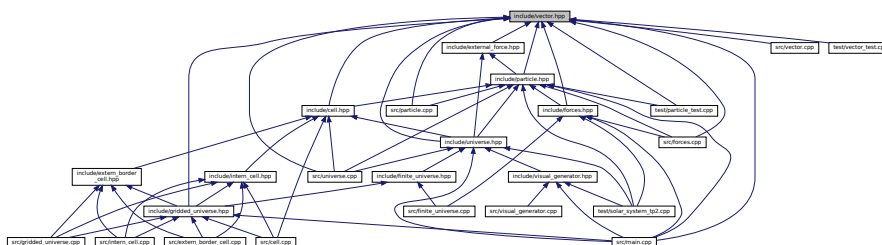
File for vector calculations.

```
#include <vector>
#include <ostream>
```

Graphe des dépendances par inclusion de vector.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

— class [Vector](#)
[Vector](#) class.

8.15.1 Description détaillée

File for vector calculations.

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

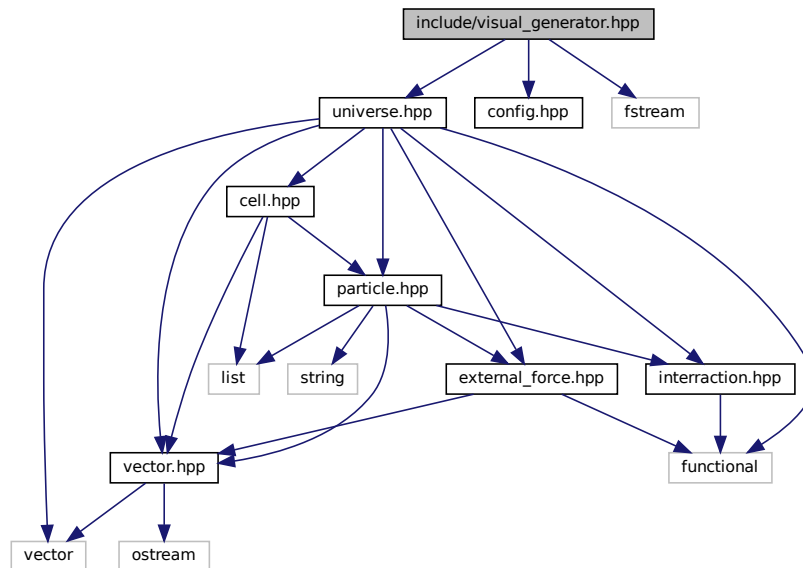
Date

2024-04-08

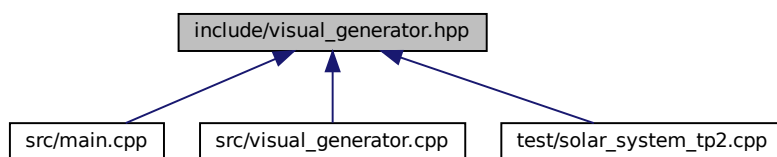
8.16 Référence du fichier include/visual_generator.hpp

```
#include <universe.hpp>
#include <config.hpp>
#include <fstream>
```

Graphe des dépendances par inclusion de visual_generator.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

- class [VisualGenerator](#)
Provides methods to generate visualisation of an universe.

Énumérations

- enum [PointType](#) {
 [NO_SYMBOL](#) , [PLUS](#) , [CROSS](#) , [STAR](#) ,
 [BOX](#) , [BOX_F](#) , [CIRCLE](#) , [CIRCLE_F](#) }

8.16.1 Documentation du type de l'énumération

8.16.1.1 PointType

enum `PointType`

Valeurs énumérées

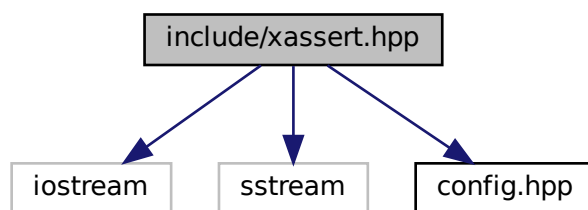
NO_SYMBOL	
PLUS	
CROSS	
STAR	
BOX	
BOX_F	
CIRCLE	
CIRCLE_F	

8.17 Référence du fichier include/xassert.hpp

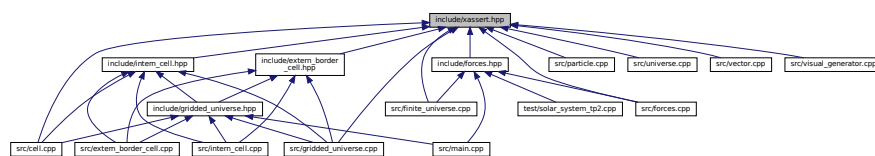
To define an assert macro.

```
#include <iostream>
#include <sstream>
#include <config.hpp>
```

Graphe des dépendances par inclusion de xassert.hpp:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Macros

— #define `xassert`(expr, msg) `_xassert`(#expr, expr, __FILE__, __LINE__, __func__, msg)

Fonctions

— void `_xassert` (const char *expr_str, bool expr, const char *file, int line, const char *function, std::stringstream &&msg)
— void `_xassert` (const char *expr_str, bool expr, const char *file, int line, const char *function, std::string &&msg)

8.17.1 Description détaillée

To define an assert macro.

Auteur

jules roques (jules.roques@grenoble-inp.org)

Version

0.1

Date

2024-04-09

8.17.2 Documentation des macros

8.17.2.1 xassert

```
#define xassert(  
    expr,  
    msg ) _xassert(#expr, expr, __FILE__, __LINE__, __func__, msg)
```

8.17.3 Documentation des fonctions

8.17.3.1 _xassert() [1/2]

```
void _xassert (  
    const char * expr_str,  
    bool expr,  
    const char * file,  
    int line,  
    const char * function,  
    std::string && msg ) [inline]
```

8.17.3.2 `_xassert()` [2/2]

```

void _xassert (
    const char * expr_str,
    bool expr,
    const char * file,
    int line,
    const char * function,
    std::stringstream && msg ) [inline]

```

8.18 Référence du fichier README.md

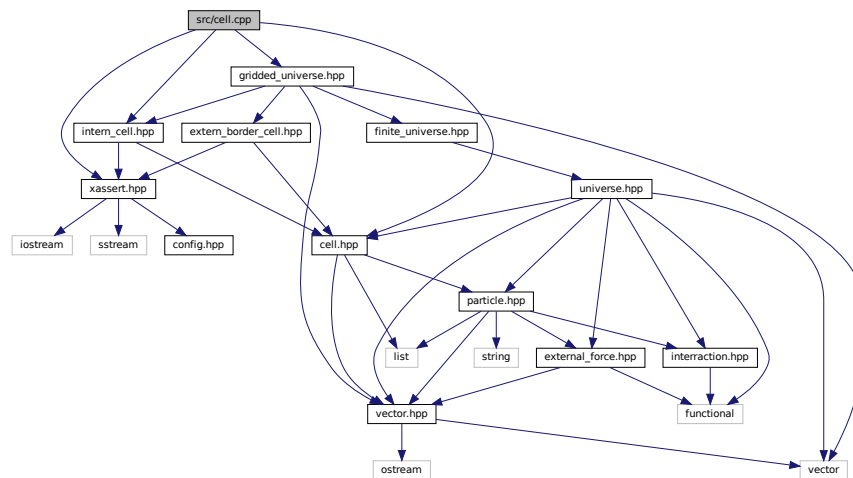
8.19 Référence du fichier src/cell.cpp

```

#include "cell.hpp"
#include "xassert.hpp"
#include "gridded_universe.hpp"
#include "intern_cell.hpp"

```

Graphe des dépendances par inclusion de cell.cpp:



8.20 Référence du fichier src/extern_border_cell.cpp

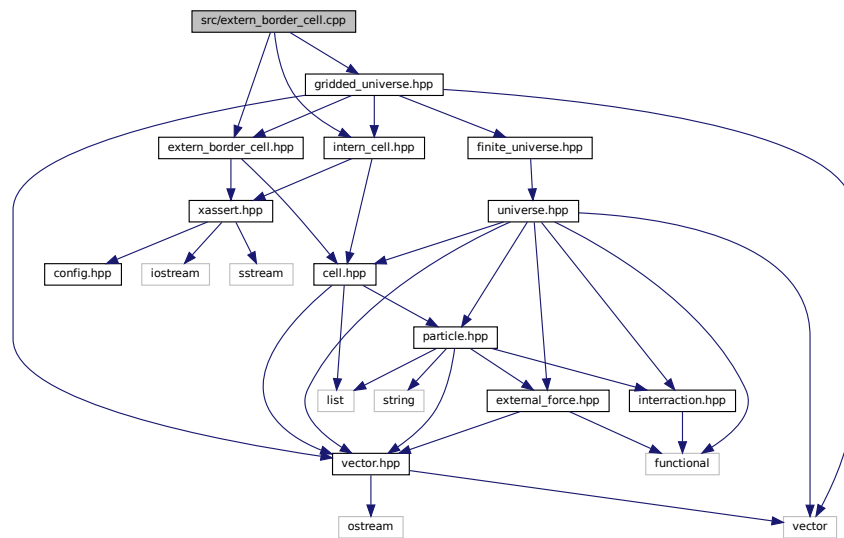
```

#include "extern_border_cell.hpp"
#include "intern_cell.hpp"

```

```
#include "gridded_universe.hpp"
```

Graphe des dépendances par inclusion de `extern_border_cell.cpp`:



8.21 Référence du fichier `src/finite_universe.cpp`

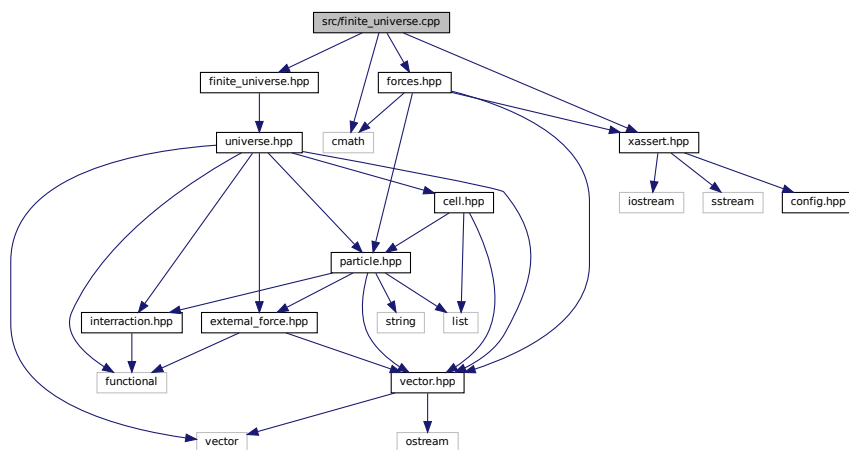
```
#include <finite_universe.hpp>
```

```
#include <xassert.hpp>
```

```
#include <cmath>
```

```
#include "forces.hpp"
```

Graphe des dépendances par inclusion de `finite_universe.cpp`:



Fonctions

— `std::ostream & operator<< (std::ostream &strm, FiniteUniverse universe)`

8.21.1 Documentation des fonctions

8.21.1.1 operator<<()

```
std::ostream& operator<< (
    std::ostream & strm,
    FiniteUniverse universe )
```

Paramètres

<i>strm</i>	
<i>universe</i>	

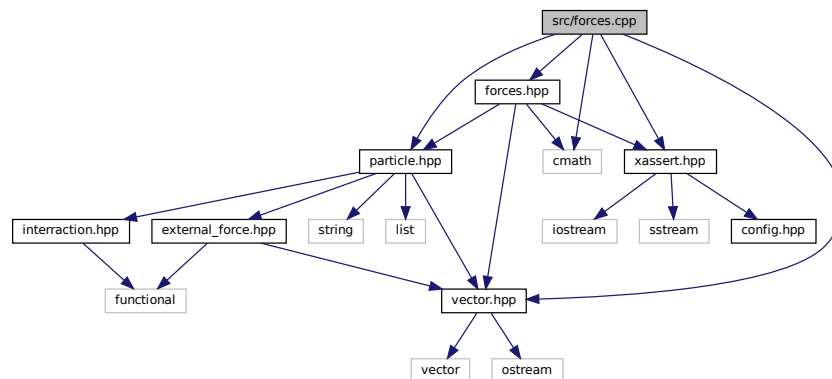
Renvoie

std::ostream&

8.22 Référence du fichier src/forces.cpp

```
#include "forces.hpp"
#include <vector.hpp>
#include <particle.hpp>
#include <cmath>
#include <xassert.hpp>
```

Graphe des dépendances par inclusion de forces.cpp:



Fonctions

- void [gravitationalInteraction](#) (const [Particle](#) &source, [Particle](#) &target)
Computes the gravitational force applied by source on target particle.
- void [lennardJonesInteraction](#) (const [Particle](#) &source, [Particle](#) &target, double epsilon, double sigma)
Computes the Lennard Jones force applied by source on target particle.
- void [gravitationalForce](#) ([Particle](#) &target, double G)
Adds the gravitational force applied on a particle to the existing force. The gravitational field is applied on the last dimension of the particle.
- void [wallsForce](#) ([Particle](#) &target, [Vector](#) lowerBound, [Vector](#) upperBound, double epsilon, double sigma)
Applies on the particle the strength of the walls (c.f. TP6)

8.22.1 Documentation des fonctions

8.22.1.1 gravitationalForce()

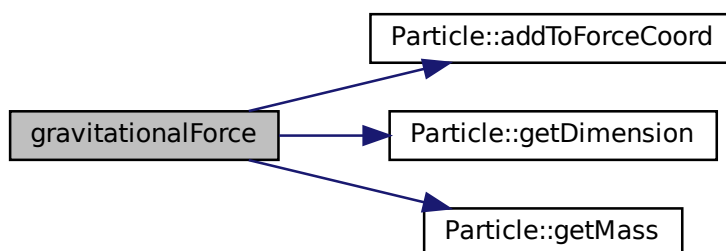
```
void gravitationalForce (
    Particle & target,
    double G )
```

Adds the gravitational force applied on a particle to the existing force. The gravitational field is applied on the last dimension of the particle.

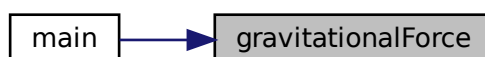
Paramètres

<i>source</i>	
<i>target</i>	

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



8.22.1.2 gravitationalInteraction()

```
void gravitationalInteraction (
    const Particle & source,
    Particle & target )
```

Computes the gravitational force applied by source on target particle.

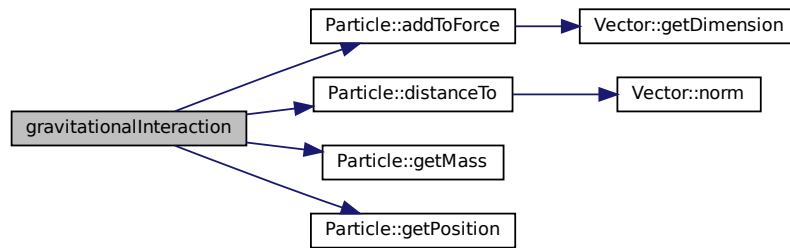
Paramètres

<i>source</i>	
<i>target</i>	

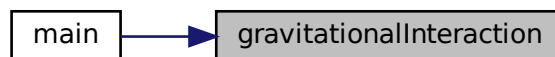
Renvoie

[Vector](#)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



8.22.1.3 lennardJonesInteraction()

```
void lennardJonesInteraction (
    const Particle & source,
    Particle & target,
    double epsilon,
    double sigma )
```

Computes the Lennard Jones force applied by source on target particle.

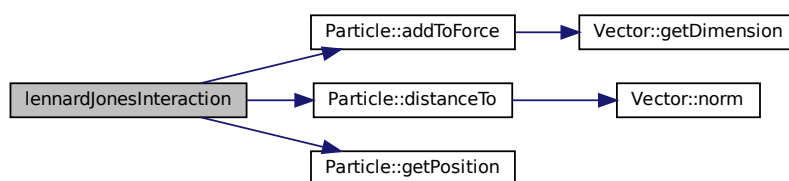
Paramètres

<i>source</i>	
<i>target</i>	
<i>epsilon</i>	
<i>sigma</i>	

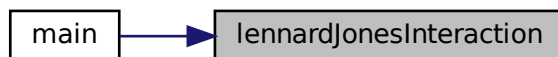
Renvoie

[Vector](#)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



8.22.1.4 wallsForce()

```

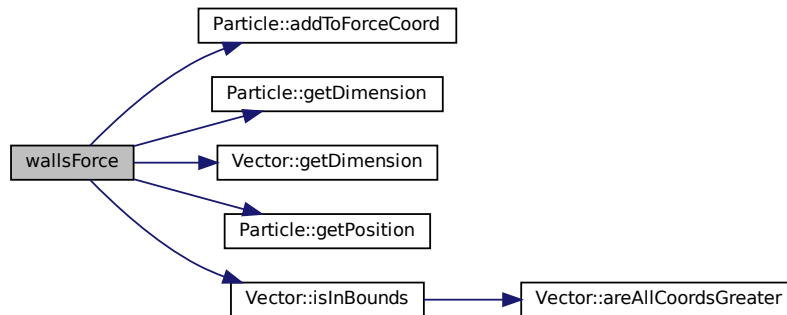
void wallsForce (
    Particle & target,
    Vector lowerBound,
    Vector upperBound,
    double epsilon,
    double sigma )
  
```

Applies on the particle the strength of the walls (c.f. TP6)

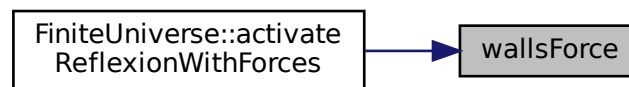
Paramètres

<i>target</i>	
<i>lowerBound</i>	
<i>upperBound</i>	

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



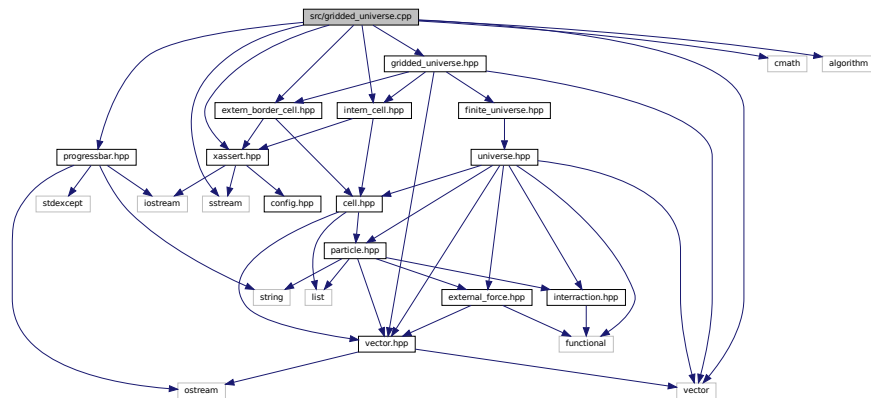
8.23 Référence du fichier src/gridded_universe.cpp

```

#include <xassert.hpp>
#include <cmath>
#include <vector>
#include <algorithm>
#include <sstream>
#include "gridded_universe.hpp"
#include "progressbar.hpp"
#include "intern_cell.hpp"
#include "extern_border_cell.hpp"

```

Graphe des dépendances par inclusion de gridded_universe.cpp:



Fonctions

- bool [isExternCoord](#) (const std::vector< int > &coords, const std::vector< int > &dimensions)
- bool [isInternCoord](#) (const std::vector< int > &coords, const std::vector< int > &dimensions)
- std::ostream & [operator<<](#) (std::ostream &strm, [GriddedUniverse](#) universe)

8.23.1 Documentation des fonctions

8.23.1.1 isExternCoord()

```
bool isExternCoord (
    const std::vector< int > & coords,
    const std::vector< int > & dimensions )
```

8.23.1.2 isInternCoord()

```
bool isInternCoord (
    const std::vector< int > & coords,
    const std::vector< int > & dimensions )
```

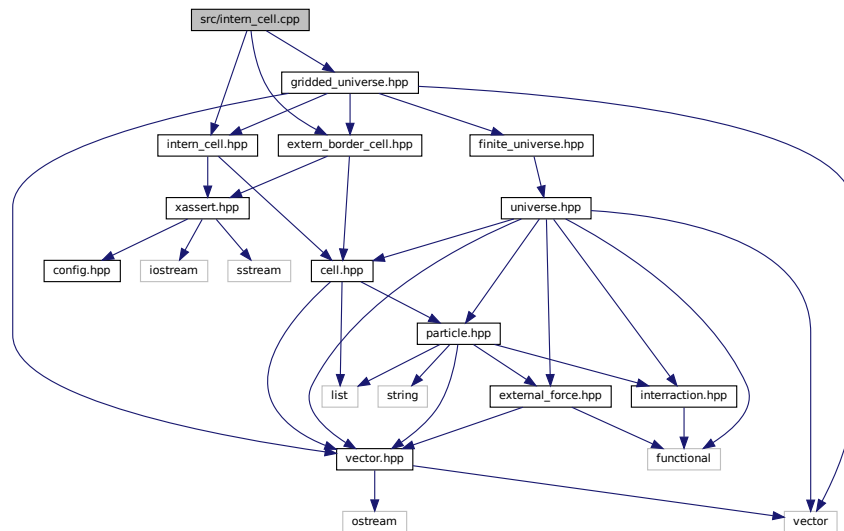
8.23.1.3 operator<<()

```
std::ostream& operator<< (
    std::ostream & strm,
    GriddedUniverse universe )
```

8.24 Référence du fichier src/intern_cell.cpp

```
#include "intern_cell.hpp"
#include "extern_border_cell.hpp"
#include "gridded_universe.hpp"
```

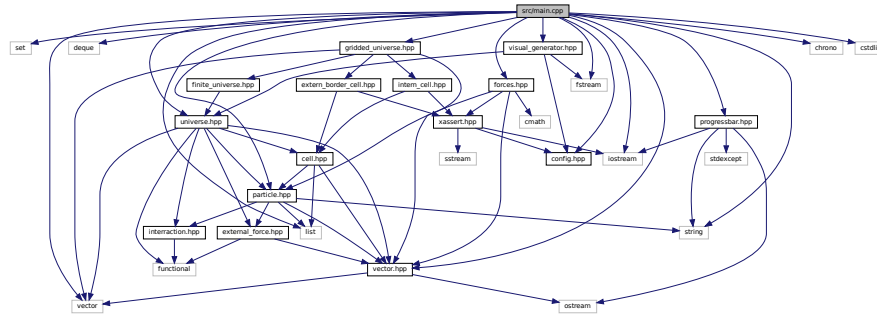
Graphe des dépendances par inclusion de intern_cell.cpp:



8.25 Référence du fichier src/main.cpp

```
#include <set>
#include <list>
#include <deque>
#include <vector>
#include <particle.hpp>
#include <vector.hpp>
#include <universe.hpp>
#include <gridded_universe.hpp>
#include <forces.hpp>
#include <progressbar.hpp>
#include <config.hpp>
#include <visual_generator.hpp>
#include <chrono>
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
```

Graphe des dépendances par inclusion de main.cpp:



Fonctions

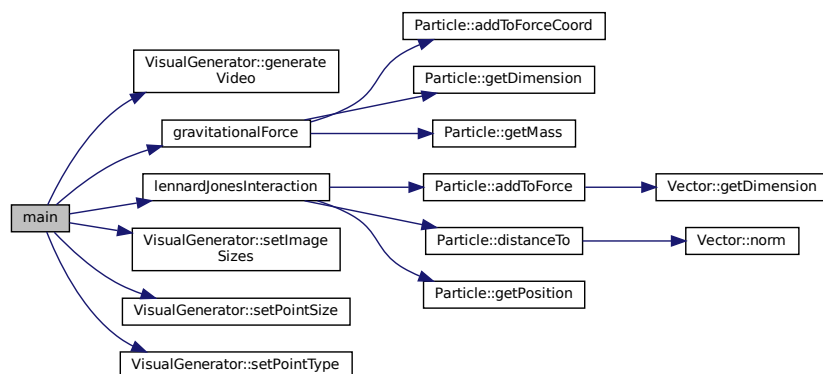
— int [main](#) (int argc, char **argv)

8.25.1 Documentation des fonctions

8.25.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

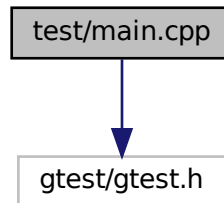
Voici le graphe d'appel pour cette fonction :



8.26 Référence du fichier test/main.cpp

```
#include <gtest/gtest.h>
```

Graphe des dépendances par inclusion de main.cpp:



Fonctions

— `int main (int argc, char **argv)`

8.26.1 Documentation des fonctions

8.26.1.1 main()

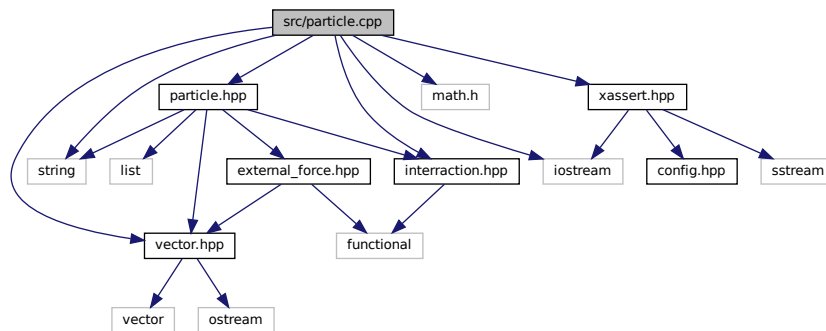
```
int main (  
    int argc,  
    char ** argv )
```

8.27 Référence du fichier src/particle.cpp

```
#include <particle.hpp>  
#include <vector.hpp>  
#include <iostream>  
#include <string>  
#include <math.h>  
#include <xassert.hpp>
```

```
#include <interaction.hpp>
```

Graphe des dépendances par inclusion de particle.cpp:



Fonctions

— `std::ostream & operator<< (std::ostream &strm, const Particle &p)`

8.27.1 Documentation des fonctions

8.27.1.1 operator<<()

```
std::ostream& operator<< (
    std::ostream & strm,
    const Particle & p )
```

Paramètres

<i>strm</i>	
<i>p</i>	

Renvoie

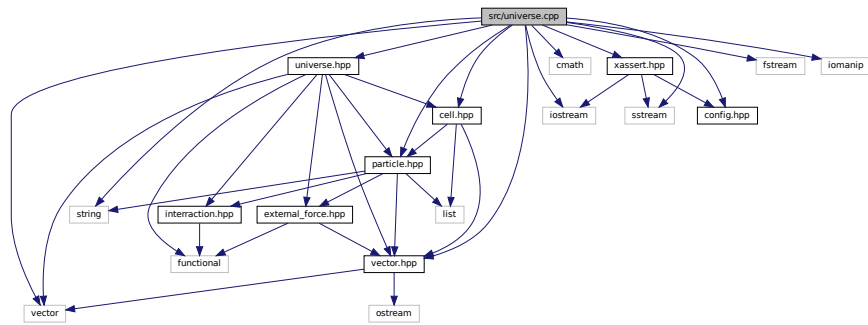
`std::ostream&`

8.28 Référence du fichier src/universe.cpp

```
#include <iostream>
#include <string>
#include <universe.hpp>
#include <particle.hpp>
#include <cmath>
```

```
#include <vector.hpp>
#include <vector>
#include <xassert.hpp>
#include <cell.hpp>
#include <config.hpp>
#include <fstream>
#include <iomanip>
#include <sstream>
```

Graphe des dépendances par inclusion de universe.cpp:



Fonctions

- void [writeData](#) (std::ofstream &dataFile, const std::list< [Particle](#) > &particles)
- void [writeDataVTK](#) (std::ofstream &dataFile, const std::list< [Particle](#) > &particles, const size_t dimension)
- std::ostream & [operator<<](#) (std::ostream &strm, const [Universe](#) &univers)

8.28.1 Documentation des fonctions

8.28.1.1 operator<<()

```
std::ostream& operator<< (
    std::ostream & strm,
    const Universe & univers )
```

Paramètres

<i>strm</i>	Le flux de sortie
<i>univers</i>	L'univers à afficher

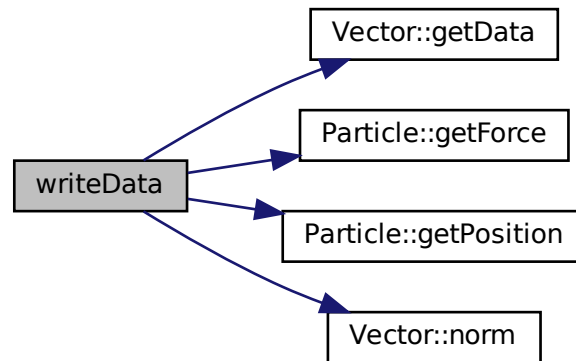
Renvoie

std::ostream&, le flux de sortie

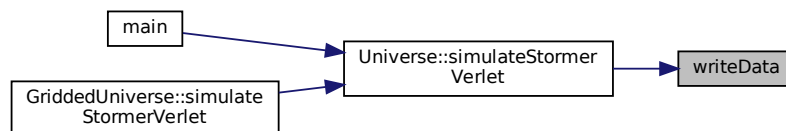
8.28.1.2 writeData()

```
void writeData (
    std::ofstream & dataFile,
    const std::list< Particle > & particles )
```

Voici le graphe d'appel pour cette fonction :



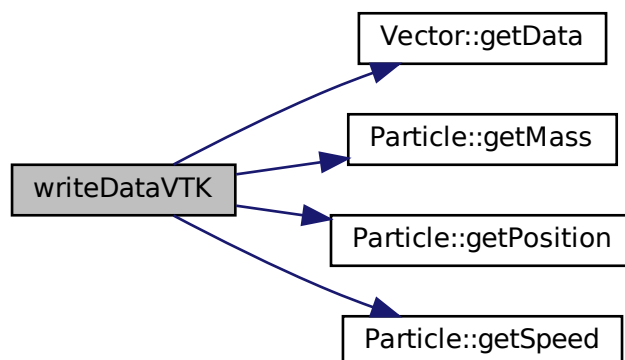
Voici le graphe des appelants de cette fonction :



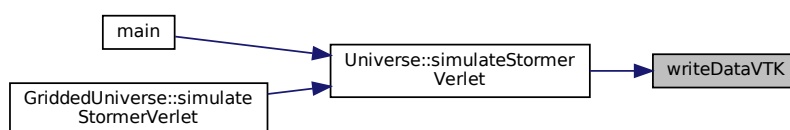
8.28.1.3 writeDataVTK()

```
void writeDataVTK (
    std::ofstream & dataFile,
    const std::list< Particle > & particles,
    const size_t dimension )
```


Voici le graphe d'appel pour cette fonction :



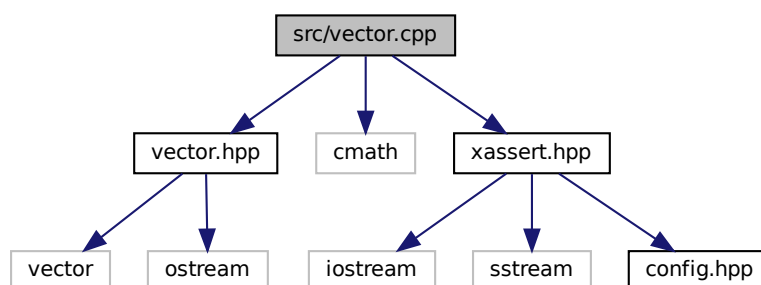
Voici le graphe des appelants de cette fonction :



8.29 Référence du fichier src/vector.cpp

```
#include <vector.hpp>
#include <cmath>
#include <xassert.hpp>
```

Graphe des dépendances par inclusion de vector.cpp:



Fonctions

- `Vector min (Vector v1, Vector v2)`
- `Vector max (Vector v1, Vector v2)`
- `std::ostream & operator<< (std::ostream &strm, const Vector &v)`

8.29.1 Documentation des fonctions

8.29.1.1 max()

```
Vector max (
    Vector v1,
    Vector v2 )
```

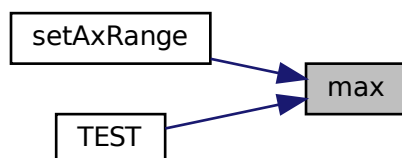
Paramètres

v1	
v2	

Renvoie

`Vector`

Voici le graphe des appelants de cette fonction :



8.29.1.2 min()

```
Vector min (
    Vector v1,
    Vector v2 )
```

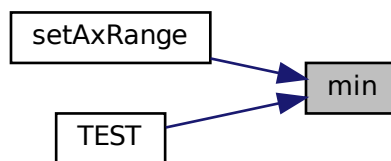
Paramètres

<i>v1</i>	
<i>v2</i>	

Renvoie

[Vector](#)

Voici le graphe des appelants de cette fonction :



8.29.1.3 operator<<()

```
std::ostream& operator<< (
    std::ostream & strm,
    const Vector & v )
```

Paramètres

<i>strm</i>	
<i>v</i>	

Renvoie

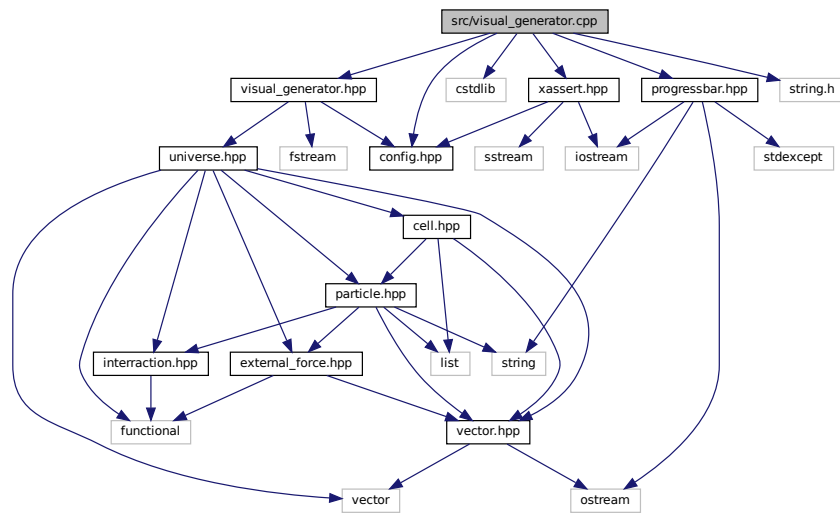
std::ostream&

8.30 Référence du fichier src/visual_generator.cpp

```
#include <visual_generator.hpp>
#include <cstdlib>
#include <xassert.hpp>
#include <config.hpp>
#include <progressbar.hpp>
```

```
#include <string.h>
```

Graphe des dépendances par inclusion de visual_generator.cpp:



Fonctions

- void **setAxRange** (std::ofstream &scriptFile, std::string axName, double min, double max)
Write the range for one axe in the script.

8.30.1 Documentation des fonctions

8.30.1.1 setAxRange()

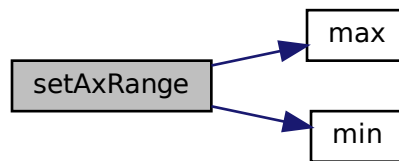
```
void setAxRange (
    std::ofstream & scriptFile,
    std::string axName,
    double min,
    double max )
```

Write the range for one axe in the script.

Paramètres

<i>scriptFile</i>	
<i>axName</i>	
<i>min</i>	
<i>max</i>	

Voici le graphe d'appel pour cette fonction :



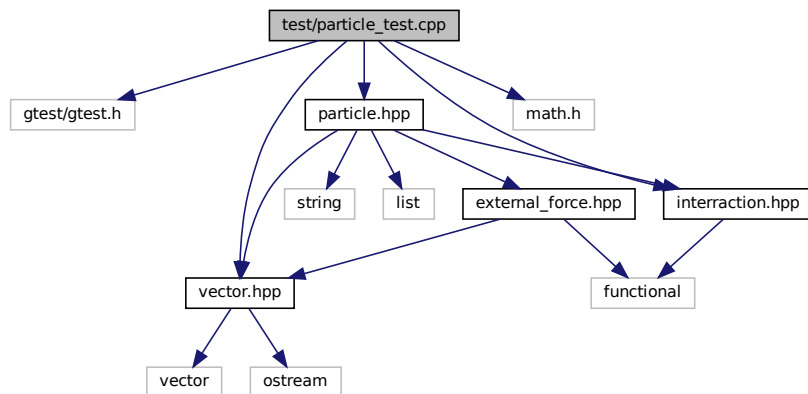
8.31 Référence du fichier test/particle_test.cpp

```

#include <gtest/gtest.h>
#include <particle.hpp>
#include <vector.hpp>
#include <interaction.hpp>
#include <math.h>

```

Graphe des dépendances par inclusion de `particle_test.cpp`:



Fonctions

- **TEST** (ParticleTest, ConstructorAndOutputOperator)
Test the constructor and output operator.
- **TEST** (ParticleTest, DistanceTo)
Test the distanceTo function.
- **TEST** (ParticleTest, InvertSpeed)
Test the invertSpeed function.
- **TEST** (ParticleTest, SetPosCoord)
Test the setPosCoord function.
- **TEST** (ParticleTest, AddToForceCoord)
Test the addToForceCoord function.
- **TEST** (ParticleTest, AddToForce)

- **TEST** (ParticleTest, AddToPosition)
Test the addToForce function.
- **TEST** (ParticleTest, AddToSpeed)
Test the addToPosition function.
- **TEST** (ParticleTest, SetForceToZero)
Test the addToSpeed function.
- **TEST** (ParticleTest, SetForceToZero)
Test the setForceToZero function.

Variables

- **Vector pos** ({1.0, 2.0, 3.0})
Test particle setup for all tests.
- **Vector speed** ({0.1, 0.2, 0.3})
- double **mass** = 2.5
- std::string **name** = "TestParticle"
- **Particle p** (pos, speed, mass, name)

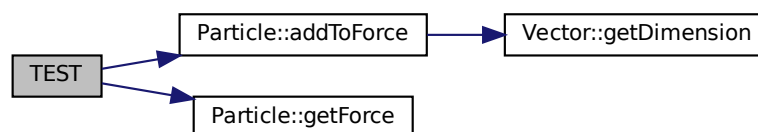
8.31.1 Documentation des fonctions

8.31.1.1 TEST() [1/9]

```
TEST (
    ParticleTest ,
    AddToForce )
```

Test the addToForce function.

This test checks that the addToForce function correctly adds the specified vector to the particle's force. Voici le graphe d'appel pour cette fonction :

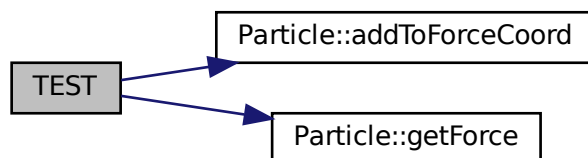


8.31.1.2 TEST() [2/9]

```
TEST (
    ParticleTest ,
    AddToForceCoord )
```

Test the addToForceCoord function.

This test checks that the addToForceCoord function correctly adds the specified value to the force coordinate at the specified index. Voici le graphe d'appel pour cette fonction :

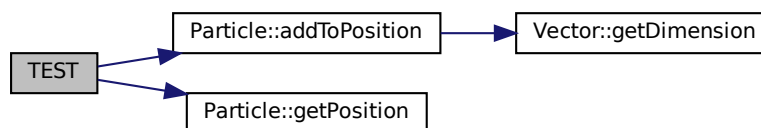


8.31.1.3 TEST() [3/9]

```
TEST (
    ParticleTest ,
    AddToPosition )
```

Test the addToPosition function.

This test checks that the addToPosition function correctly adds the specified vector to the particle's position. Voici le graphe d'appel pour cette fonction :

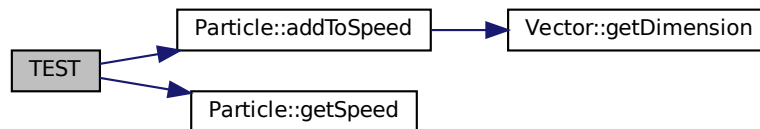


8.31.1.4 TEST() [4/9]

```
TEST (
    ParticleTest ,
    AddToSpeed )
```

Test the addToSpeed function.

This test checks that the addToSpeed function correctly adds the specified vector to the particle's speed. Voici le graphe d'appel pour cette fonction :



8.31.1.5 TEST() [5/9]

```
TEST (
    ParticleTest ,
    ConstructorAndOutputOperator )
```

Test the constructor and output operator.

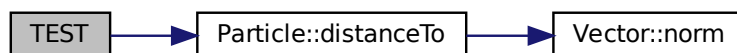
This test checks that the [Particle](#) constructor initializes the particle correctly and that the output operator prints the expected output.

8.31.1.6 TEST() [6/9]

```
TEST (
    ParticleTest ,
    DistanceTo )
```

Test the distanceTo function.

This test checks that the distanceTo function calculates the correct distance between two particles. Voici le graphe d'appel pour cette fonction :

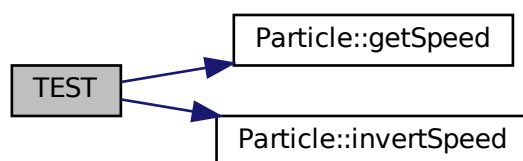


8.31.1.7 TEST() [7/9]

```
TEST (
    ParticleTest ,
    InvertSpeed )
```

Test the invertSpeed function.

This test checks that the invertSpeed function correctly inverts the speed in the specified dimension. Voici le graphe d'appel pour cette fonction :

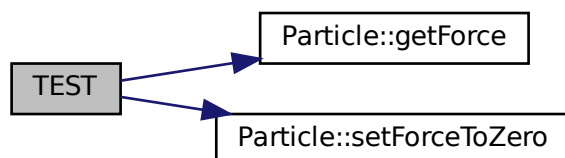


8.31.1.8 TEST() [8/9]

```
TEST (
    ParticleTest ,
    SetForceToZero )
```

Test the setForceToZero function.

This test checks that the setForceToZero function correctly resets the particle's force to zero. Voici le graphe d'appel pour cette fonction :

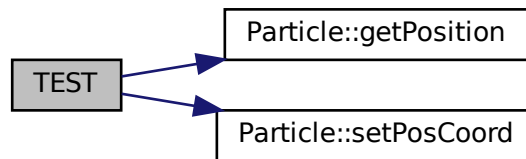


8.31.1.9 TEST() [9/9]

```
TEST (
    ParticleTest ,
    SetPosCoord )
```

Test the setPosCoord function.

This test checks that the setPosCoord function correctly sets the position coordinate at the specified index. Voici le graphe d'appel pour cette fonction :



8.31.2 Documentation des variables

8.31.2.1 mass

```
double mass = 2.5
```

8.31.2.2 name

```
std::string name = "TestParticle"
```

8.31.2.3 p

```
Particle p(pos, speed, mass, name) (
    pos ,
    speed ,
    mass ,
    name )
```

8.31.2.4 pos

```
Vector pos({1.0, 2.0, 3.0}) (
    {1.0, 2.0, 3.0} )
```

Test particle setup for all tests.

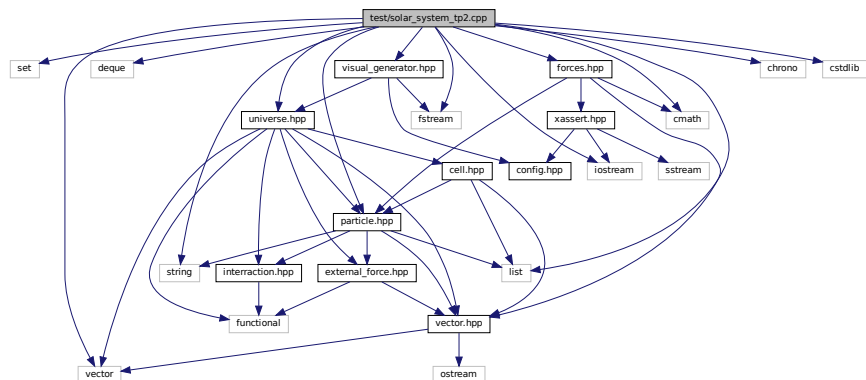
8.31.2.5 speed

```
Vector speed({0.1, 0.2, 0.3}) (
    {0.1, 0.2, 0.3} )
```

8.32 Référence du fichier test/solar_system_tp2.cpp

```
#include <set>
#include <list>
#include <deque>
#include <vector>
#include <particle.hpp>
#include <universe.hpp>
#include <forces.hpp>
#include <visual_generator.hpp>
#include <chrono>
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <cmath>
```

Graphe des dépendances par inclusion de solar_system_tp2.cpp:



Fonctions

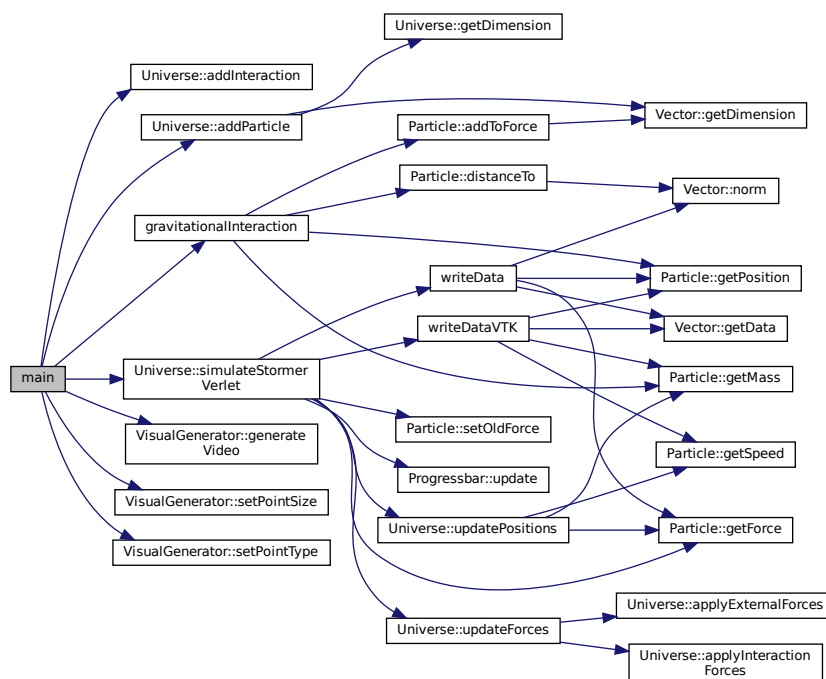
— int `main` (int argc, char **argv)

8.32.1 Documentation des fonctions

8.32.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

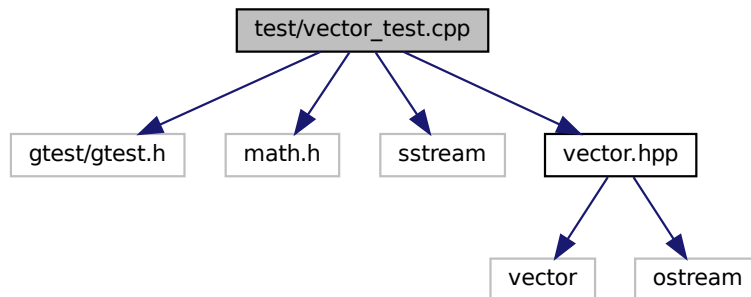
Voici le graphe d'appel pour cette fonction :



8.33 Référence du fichier test/vector_test.cpp

```
#include <gtest/gtest.h>
#include <math.h>
#include <sstream>
#include <vector.hpp>
```

Graphe des dépendances par inclusion de vector_test.cpp:



Fonctions

- **TEST** (VectorTest, Addition)
Test vector addition.
- **TEST** (VectorTest, Subtraction)
Test vector subtraction.
- **TEST** (VectorTest, MultScalar)
Test vector scalar multiplication.
- **TEST** (VectorTest, Equality)
Test vector equality operator.
- **TEST** (VectorTest, Norm)
Test norm calculation.
- **TEST** (VectorTest, MinFunction)
Test min function.
- **TEST** (VectorTest, MaxFunction)
Test max function.
- **TEST** (VectorTest, OutputOperator)
Test output stream operator.

8.33.1 Documentation des fonctions

8.33.1.1 TEST() [1/8]

```
TEST (
    VectorTest ,
    Addition )
```

Test vector addition.

This test checks that the vector addition operator correctly adds two vectors element-wise.

8.33.1.2 TEST() [2/8]

```
TEST (
    VectorTest ,
    Equality )
```

Test vector equality operator.

This test checks that the equality operator correctly determines if two vectors are equal element-wise.

8.33.1.3 TEST() [3/8]

```
TEST (
    VectorTest ,
    MaxFunction )
```

Test max function.

This test checks that the max function correctly returns a vector consisting of the element-wise maximum values. Voici le graphe d'appel pour cette fonction :



8.33.1.4 TEST() [4/8]

```
TEST (
    VectorTest ,
    MinFunction )
```

Test min function.

This test checks that the min function correctly returns a vector consisting of the element-wise minimum values. Voici le graphe d'appel pour cette fonction :



8.33.1.5 TEST() [5/8]

```
TEST (
    VectorTest ,
    MultScalar )
```

Test vector scalar multiplication.

This test checks that the scalar multiplication operator correctly multiplies each element of the vector by the scalar.

8.33.1.6 TEST() [6/8]

```
TEST (
    VectorTest ,
    Norm )
```

Test norm calculation.

This test checks that the norm function correctly calculates the Euclidean norm of the vector.

8.33.1.7 TEST() [7/8]

```
TEST (
    VectorTest ,
    OutputOperator )
```

Test output stream operator.

This test checks that the output stream operator correctly prints the vector in the expected format.

8.33.1.8 TEST() [8/8]

```
TEST (
    VectorTest ,
    Subtraction )
```

Test vector subtraction.

This test checks that the vector subtraction operator correctly subtracts one vector from another element-wise.

Index

- `_xassert`
 - `xassert.hpp`, [110](#)
- `~Progressbar`
- `Progressbar`, [61](#)
- ABSORPTION**
 - `finite_universe.hpp`, [95](#)
- `activateReflexionWithForces`
 - `FiniteUniverse`, [28](#)
- `addExternalForce`
 - `Universe`, [67](#)
- `addInteraction`
 - `Universe`, [67](#)
- `addNeighbour`
 - `Cell`, [17](#)
- `addParticle`
 - `FiniteUniverse`, [28–30](#)
 - `InternCell`, [46](#)
 - `Universe`, [67–69](#)
- `addToForce`
 - `Particle`, [50](#)
- `addToForceCoord`
 - `Particle`, [51](#)
- `addToPosition`
 - `Particle`, [51](#)
- `addToSpeed`
 - `Particle`, [52](#)
- `applyExternalForces`
 - `FiniteUniverse`, [31](#)
 - `Particle`, [53](#)
 - `Universe`, [70](#)
- `applyForceOnNeighbours`
 - `Cell`, [17](#)
 - `ExternBorderCell`, [23](#)
 - `InternCell`, [47](#)
- `applyInteractionForces`
 - `FiniteUniverse`, [31](#)
 - `Universe`, [70](#)
- `applyInteractionForcesOn`
 - `Particle`, [53](#)
- `applyInternInteractionsForces`
 - `FiniteUniverse`, [31](#)
- `applyLimitinterractionForces`
 - `FiniteUniverse`, [32](#)
- `applyOn`
 - `ExternalForce`, [20](#)
- `areAllCoordsGreater`
 - `Vector`, [78](#)
- BOX**
 - `visual_generator.hpp`, [109](#)
- BOX_F**
 - `visual_generator.hpp`, [109](#)
- Cell**, [15](#)
 - `addNeighbour`, [17](#)
 - `applyForceOnNeighbours`, [17](#)
 - `Cell`, [16](#)
 - `clearParticles`, [17](#)
 - `getCoordinates`, [17](#)
 - `getDimension`, [17](#)
 - `getNbNeighbours`, [18](#)
 - `getNeighbours`, [18](#)
 - `getUniverse`, [18](#)
- CIRCLE**
 - `visual_generator.hpp`, [109](#)
- CIRCLE_F**
 - `visual_generator.hpp`, [109](#)
- `clearParticles`
 - `Cell`, [17](#)
 - `ExternBorderCell`, [23](#)
 - `InternCell`, [47](#)
- `computeInternInterractions`
 - `InternCell`, [47](#)
- `config.hpp`
 - `NDEBUG`, [91](#)
 - `PNG_OUTPUT`, [91](#)
 - `SHOW_PROGRESS_INFOS`, [91](#)
 - `XML_OUTPUT`, [92](#)
- `copyParticles`
 - `ExternBorderCell`, [23](#)
- CROSS**
 - `visual_generator.hpp`, [109](#)
- `distanceTo`
 - `Particle`, [53](#)
- `docs/conception.md`, [89](#)
- `docs/TO DO.md`, [89](#)
- ExternalForce**, [19](#)
 - `applyOn`, [20](#)
 - `ExternalForce`, [20](#)
 - `setForceFunction`, [20](#)
- ExternBorderCell**, [21](#)
 - `applyForceOnNeighbours`, [23](#)
 - `clearParticles`, [23](#)
 - `copyParticles`, [23](#)
 - `ExternBorderCell`, [22](#)
- `finite_universe.cpp`

- operator<<, 113
- finite_universe.hpp
 - ABSORPTION, 95
 - OOBBehavior, 95
 - PERIODIC, 95
 - REFLEXION, 95
- FiniteUniverse, 24
 - activateReflexionWithForces, 28
 - addParticle, 28–30
 - applyExternalForces, 31
 - applyInteractionForces, 31
 - applyInternInteractionsForces, 31
 - applyLimitInterractionForces, 32
 - FiniteUniverse, 27
 - getBounds, 32
 - getLowerBound, 33
 - getoobbehavior, 33
 - getUpperBound, 33
 - handleOutOfBoundsParticles, 33
 - isInBounds, 34
 - operator<<, 37
 - reflectOutOfBoundsParticles, 35
 - removeOutOfBoundsParticles, 35
 - setApplyWallsForces, 36
 - setOOBBehavior, 36
 - teleportOutOfBoundsParticles, 36
 - updatePositions, 37
- forces.cpp
 - gravitationalForce, 114
 - gravitationalInteraction, 114
 - lennardJonesInteraction, 115
 - wallsForce, 116
- forces.hpp
 - gravitationalForce, 97
 - gravitationalInteraction, 98
 - lennardJonesInteraction, 99
 - wallsForce, 100
- generatePhoto
 - VisualGenerator, 85
- generateVideo
 - VisualGenerator, 85
- getBounds
 - FiniteUniverse, 32
 - Universe, 71
- getCoordinates
 - Cell, 17
- getData
 - Vector, 79
- getDimension
 - Cell, 17
 - Particle, 54
 - Universe, 71
 - Vector, 79
- getDimensions
 - GriddedUniverse, 41
- getForce
 - Particle, 54
- getInteractions
 - Universe, 71
- getLastAddedParticlePointer
 - Universe, 72
- getLowerBound
 - FiniteUniverse, 33
- getMass
 - Particle, 55
- getMaxForce
 - Universe, 72
- getName
 - Particle, 55
- getNbNeighbours
 - Cell, 18
- getNbParticles
 - Universe, 72
- getNbPastStates
 - Universe, 72
- getNeighbours
 - Cell, 18
- getOldForce
 - Particle, 55
- getoobbehavior
 - FiniteUniverse, 33
- getParticleCount
 - Particle, 56
- getParticles
 - InternCell, 47
 - Universe, 73
- getPosition
 - Particle, 56
- getSpeed
 - Particle, 56
- getUniverse
 - Cell, 18
- getUpperBound
 - FiniteUniverse, 33
- gravitationalForce
 - forces.cpp, 114
 - forces.hpp, 97
- gravitationalInteraction
 - forces.cpp, 114
 - forces.hpp, 98
- gridded_universe.cpp
 - isExternCoord, 118
 - isInternCoord, 118
 - operator<<, 118
- GriddedUniverse, 38
 - getDimensions, 41
 - GriddedUniverse, 41
 - operator<<, 42
 - simulateStormerVerlet, 42
- handleOutOfBoundsParticles
 - FiniteUniverse, 33
- include/cell.hpp, 89
- include/config.hpp, 90
- include/extern_border_cell.hpp, 92
- include/external_force.hpp, 93

- include/finite_universe.hpp, 94
- include/forces.hpp, 96
- include/gridded_universe.hpp, 101
- include/intern_cell.hpp, 102
- include/interaction.hpp, 103
- include/particle.hpp, 104
- include/progressbar.hpp, 104
- include/universe.hpp, 105
- include/vector.hpp, 106
- include/visual_generator.hpp, 108
- include/xassert.hpp, 109
- Interaction, 43
 - Interaction, 43
 - operator(), 43
- InternCell, 44
 - addParticle, 46
 - applyForceOnNeighbours, 47
 - clearParticles, 47
 - computeInternInteractions, 47
 - getParticles, 47
 - InternCell, 46
- invertSpeed
 - Particle, 56
- isExternCoord
 - gridded_universe.cpp, 118
- isInBounds
 - FiniteUniverse, 34
 - Vector, 80
- isInternCoord
 - gridded_universe.cpp, 118
- lennardJonesInteraction
 - forces.cpp, 115
 - forces.hpp, 99
- main
 - main.cpp, 120, 121
 - solar_system_tp2.cpp, 136
- main.cpp
 - main, 120, 121
- mass
 - particle_test.cpp, 134
- max
 - Vector, 83
 - vector.cpp, 126
- min
 - Vector, 83
 - vector.cpp, 126
- multiplySpeed
 - Particle, 58
- name
 - particle_test.cpp, 134
- NDEBUG
 - config.hpp, 91
- NO_SYMBOL
 - visual_generator.hpp, 109
- norm
 - Vector, 81
- OOBBehavior
 - finite_universe.hpp, 95
- operator!=
 - Vector, 81
- operator<<
 - finite_universe.cpp, 113
 - FiniteUniverse, 37
 - gridded_universe.cpp, 118
 - GriddedUniverse, 42
 - Particle, 60
 - particle.cpp, 122
 - Universe, 76
 - universe.cpp, 123
 - Vector, 84
 - vector.cpp, 127
- operator*=
 - Vector, 82
- operator()
 - Interaction, 43
- operator+=
 - Vector, 82
- operator-=
 - Vector, 82
- operator=
 - Progressbar, 62
- operator==
 - Vector, 82
- operator[]
 - Vector, 82, 83
- p
 - particle_test.cpp, 134
- Particle, 48
 - addToForce, 50
 - addToForceCoord, 51
 - addPosition, 51
 - addSpeed, 52
 - applyExternalForces, 53
 - applyInteractionForcesOn, 53
 - distanceTo, 53
 - getDimension, 54
 - getForce, 54
 - getMass, 55
 - getName, 55
 - getOldForce, 55
 - getParticleCount, 56
 - getPosition, 56
 - getSpeed, 56
 - invertSpeed, 56
 - multiplySpeed, 58
 - operator<<, 60
 - Particle, 49
 - setForce, 58
 - setForceToZero, 58
 - setOldForce, 59
 - setPosCoord, 59
 - setPosition, 59
 - setSpeed, 59
- particle.cpp

- operator<<, 122
- particle_test.cpp
 - mass, 134
 - name, 134
 - p, 134
 - pos, 134
 - speed, 135
 - TEST, 130–133
- PERIODIC
 - finite_universe.hpp, 95
- PLUS
 - visual_generator.hpp, 109
- PNG_OUTPUT
 - config.hpp, 91
- PointType
 - visual_generator.hpp, 109
- pos
 - particle_test.cpp, 134
- Progressbar, 60
 - ~Progressbar, 61
 - operator=, 62
 - Progressbar, 61
 - reset, 62
 - set_closing_bracket_char, 62
 - set_done_char, 62
 - set_niter, 62
 - set_opening_bracket_char, 63
 - set_output_stream, 63
 - set_todo_char, 63
 - show_bar, 63
 - update, 63
- README.md, 111
- reflectOutOfBoundsParticles
 - FiniteUniverse, 35
- REFLEXION
 - finite_universe.hpp, 95
- removeOutOfBoundsParticles
 - FiniteUniverse, 35
- reset
 - Progressbar, 62
- set_closing_bracket_char
 - Progressbar, 62
- set_done_char
 - Progressbar, 62
- set_niter
 - Progressbar, 62
- set_opening_bracket_char
 - Progressbar, 63
- set_output_stream
 - Progressbar, 63
- set_todo_char
 - Progressbar, 63
- setApplyWallsForces
 - FiniteUniverse, 36
- setAxRange
 - visual_generator.cpp, 128
- setCineticEnergyLimit
 - Universe, 73
- setForce
 - Particle, 58
- setForceFunction
 - ExternalForce, 20
- setForceToZero
 - Particle, 58
- setImageSizes
 - VisualGenerator, 86
- setOldForce
 - Particle, 59
- setOOBBehavior
 - FiniteUniverse, 36
- setPointSize
 - VisualGenerator, 86
- setPointType
 - VisualGenerator, 87
- setPosCoord
 - Particle, 59
- setPosition
 - Particle, 59
- setSpeed
 - Particle, 59
- show_bar
 - Progressbar, 63
- SHOW_PROGRESS_INFOS
 - config.hpp, 91
- simulateStormerVerlet
 - GriddedUniverse, 42
 - Universe, 73
- solar_system_tp2.cpp
 - main, 136
- speed
 - particle_test.cpp, 135
- src/cell.cpp, 111
- src/extern_border_cell.cpp, 111
- src/finite_universe.cpp, 112
- src/forces.cpp, 113
- src/gridded_universe.cpp, 117
- src/intern_cell.cpp, 119
- src/main.cpp, 119
- src/particle.cpp, 121
- src/universe.cpp, 122
- src/vector.cpp, 125
- src/visual_generator.cpp, 127
- STAR
 - visual_generator.hpp, 109
- teleportOutOfBoundsParticles
 - FiniteUniverse, 36
- TEST
 - particle_test.cpp, 130–133
 - vector_test.cpp, 137–139
- test/main.cpp, 121
- test/particle_test.cpp, 129
- test/solar_system_tp2.cpp, 135
- test/vector_test.cpp, 136
- Universe, 64

- addExternalForce, 67
- addInteraction, 67
- addParticle, 67–69
- applyExternalForces, 70
- applyInteractionForces, 70
- getBounds, 71
- getDimension, 71
- getInteractions, 71
- getLastAddedParticlePointer, 72
- getMaxForce, 72
- getNbParticles, 72
- getNbPastStates, 72
- getParticles, 73
- operator<<, 76
- setCineticEnergyLimit, 73
- simulateStormerVerlet, 73
- Universe, 66
- updateForces, 74
- updatePositions, 75
- VisualGenerator, 76
- universe.cpp
 - operator<<, 123
 - writeData, 123
 - writeDataVTK, 124
- update
 - Progressbar, 63
- updateForces
 - Universe, 74
- updatePositions
 - FiniteUniverse, 37
 - Universe, 75
- Vector, 76
 - areAllCoordsGreater, 78
 - getData, 79
 - getDimension, 79
 - isInBounds, 80
 - max, 83
 - min, 83
 - norm, 81
 - operator!=, 81
 - operator<<, 84
 - operator*=:, 82
 - operator+=, 82
 - operator-=, 82
 - operator==, 82
 - operator[], 82, 83
 - Vector, 78
- vector.cpp
 - max, 126
 - min, 126
 - operator<<, 127
- vector_test.cpp
 - TEST, 137–139
- visual_generator.cpp
 - setAxRange, 128
- visual_generator.hpp
 - BOX, 109
 - BOX_F, 109
 - CIRCLE, 109
 - CIRCLE_F, 109
 - CROSS, 109
 - NO_SYMBOL, 109
 - PLUS, 109
 - PointType, 109
 - STAR, 109
- VisualGenerator, 84
 - generatePhoto, 85
 - generateVideo, 85
 - setImageSizes, 86
 - setPointSize, 86
 - setPointType, 87
 - Universe, 76
 - VisualGenerator, 85
- wallsForce
 - forces.cpp, 116
 - forces.hpp, 100
- writeData
 - universe.cpp, 123
- writeDataVTK
 - universe.cpp, 124
- xassert
 - xassert.hpp, 110
- xassert.hpp
 - _xassert, 110
 - xassert, 110
- XML_OUTPUT
 - config.hpp, 92