

# Tactics Arena

Généré par Doxygen 1.8.9.1

Vendredi 18 Décembre 2015 10 :22 :01



# Table des matières

<b>1</b>	<b>Page principale</b>	<b>1</b>
<b>2</b>	<b>Index des classes</b>	<b>3</b>
2.1	Liste des classes . . . . .	3
<b>3</b>	<b>Index des fichiers</b>	<b>5</b>
3.1	Liste des fichiers . . . . .	5
<b>4</b>	<b>Documentation des classes</b>	<b>7</b>
4.1	Référence de la structure elem . . . . .	7
4.1.1	Description détaillée . . . . .	7
4.2	Référence de la structure element . . . . .	8
4.2.1	Description détaillée . . . . .	8
4.3	Référence de la structure targetStat . . . . .	8
4.3.1	Description détaillée . . . . .	9
4.4	Référence de la structure unit . . . . .	9
4.4.1	Description détaillée . . . . .	9
4.5	Référence de la structure unitStat . . . . .	10
4.5.1	Description détaillée . . . . .	10
4.6	Référence de la structure vector . . . . .	10
4.6.1	Description détaillée . . . . .	11
<b>5</b>	<b>Documentation des fichiers</b>	<b>13</b>
5.1	Référence du fichier include/controller/manageSignal.h . . . . .	13
5.1.1	Description détaillée . . . . .	13
5.1.2	Documentation des fonctions . . . . .	14
5.1.2.1	checkSignal . . . . .	14
5.1.2.2	freeAll . . . . .	14
5.2	Référence du fichier include/controller/manageString.h . . . . .	14
5.2.1	Description détaillée . . . . .	15
5.2.2	Documentation des fonctions . . . . .	15
5.2.2.1	clearBuffer . . . . .	15
5.2.2.2	get2Char . . . . .	15

5.2.2.3	getCoordS	15
5.2.2.4	getDirectionUnit	15
5.2.2.5	getNameEffect	16
5.2.2.6	getNameUnit	16
5.2.2.7	printNameUnit	16
5.2.2.8	readDouble	16
5.2.2.9	readLong	17
5.2.2.10	readS	17
5.2.2.11	rightSide	17
5.3	Référence du fichier include/controller/saveNLoad.h	18
5.3.1	Description détaillée	18
5.3.2	Documentation des fonctions	18
5.3.2.1	load	18
5.3.2.2	save	19
5.4	Référence du fichier include/controller/terminal.h	19
5.4.1	Description détaillée	19
5.4.2	Documentation des fonctions	20
5.4.2.1	color	20
5.4.2.2	fontColor	20
5.4.2.3	reinitColor	20
5.5	Référence du fichier include/display/grid.h	20
5.5.1	Description détaillée	21
5.5.2	Documentation des fonctions	21
5.5.2.1	gridDisp	21
5.6	Référence du fichier include/display/menu.h	21
5.6.1	Description détaillée	22
5.6.2	Documentation des fonctions	22
5.6.2.1	dispDirection	22
5.6.2.2	gameMenu	22
5.6.2.3	helpUnit	22
5.6.2.4	mainMenu	22
5.6.2.5	unitList	23
5.6.2.6	unitMenu	23
5.7	Référence du fichier include/game/engine.h	23
5.7.1	Description détaillée	25
5.7.2	Documentation des fonctions	25
5.7.2.1	attackable	25
5.7.2.2	gameInit	25
5.7.2.3	isSurrounded	26
5.7.2.4	launchAttack	26

5.7.2.5	lineOfSight	26
5.7.2.6	movable	26
5.7.2.7	pathFind	26
5.7.2.8	possiblePath	27
5.7.2.9	selectUnit	27
5.7.2.10	setTarget	27
5.7.2.11	tileWalkable	27
5.7.3	Documentation des variables	28
5.7.3.1	grid	28
5.7.3.2	noPlayer	28
5.8	Référence du fichier include/game/listes.h	28
5.8.1	Description détaillée	29
5.8.2	Documentation des fonctions	29
5.8.2.1	addTarget	29
5.8.2.2	addUnit	30
5.8.2.3	countUnits	30
5.8.2.4	destroyUnit	30
5.8.2.5	dumpAllLists	30
5.8.2.6	dumpList	30
5.8.2.7	printList	31
5.8.2.8	searchTarget	31
5.8.3	Documentation des variables	31
5.8.3.1	targetList	31
5.9	Référence du fichier include/game/pathList.h	31
5.9.1	Description détaillée	32
5.9.2	Documentation des fonctions	32
5.9.2.1	addCloseList	32
5.9.2.2	addOpenList	33
5.9.2.3	dumpAllPaths	33
5.9.2.4	dumpPath	33
5.9.2.5	emptyPath	33
5.9.2.6	eraseTile	33
5.9.2.7	getCurrentNode	34
5.9.2.8	getTile	34
5.9.2.9	initPath	34
5.9.2.10	next	34
5.9.2.11	outPath	34
5.9.2.12	pathHead	35
5.9.2.13	pathTail	35
5.9.2.14	previous	35

5.9.2.15	searchTile	35
5.9.2.16	setTile	35
5.9.2.17	toRightPath	36
5.10	Référence du fichier include/game/pawns.h	36
5.10.1	Description détaillée	36
5.10.2	Documentation des fonctions	37
5.10.2.1	makePawns	37
5.10.3	Documentation des variables	37
5.10.3.1	pawns	37
5.11	Référence du fichier include/game/turn.h	37
5.11.1	Description détaillée	38
5.11.2	Documentation des fonctions	38
5.11.2.1	changeDirection	38
5.11.2.2	hasPlay	38
5.11.2.3	passTurn	39
5.11.2.4	playAttack	39
5.11.2.5	playMove	39
5.11.2.6	playTurn	39
5.12	Référence du fichier include/units/unit.h	39
5.12.1	Description détaillée	40
5.12.2	Documentation des fonctions	40
5.12.2.1	addEffect	40
5.12.2.2	allStatic	41
5.12.2.3	attack	41
5.12.2.4	canAttack	41
5.12.2.5	canBlock	41
5.12.2.6	canGetPassed	42
5.12.2.7	canMove	42
5.12.2.8	canTeleport	42
5.12.2.9	copy	42
5.12.2.10	erase	43
5.12.2.11	getSideAttacked	43
5.12.2.12	heal	43
5.12.2.13	isSleeping	43
5.12.2.14	minusEffect	44
5.12.2.15	move	44
5.12.2.16	poison	44
5.12.2.17	powerBonus	44
5.12.2.18	recover	44
5.12.2.19	setDirection	44

5.12.2.20 sleep . . . . .	44
5.12.2.21 unitInit . . . . .	46
5.13 Référence du fichier src/controller/manageSignal.c . . . . .	46
5.13.1 Description détaillée . . . . .	46
5.13.2 Documentation des fonctions . . . . .	47
5.13.2.1 checkSignal . . . . .	47
5.13.2.2 freeAll . . . . .	47
5.13.2.3 interrupt . . . . .	47
5.13.2.4 terminator . . . . .	47
5.13.2.5 timeDown . . . . .	47
5.14 Référence du fichier src/controller/manageString.c . . . . .	48
5.14.1 Description détaillée . . . . .	48
5.14.2 Documentation des fonctions . . . . .	49
5.14.2.1 clearBuffer . . . . .	49
5.14.2.2 correctCoord . . . . .	49
5.14.2.3 get2Char . . . . .	49
5.14.2.4 getCoordS . . . . .	49
5.14.2.5 getDirectionUnit . . . . .	50
5.14.2.6 getNameEffect . . . . .	50
5.14.2.7 getNameUnit . . . . .	50
5.14.2.8 isOutGrid . . . . .	50
5.14.2.9 printNameUnit . . . . .	51
5.14.2.10 readDouble . . . . .	51
5.14.2.11 readLong . . . . .	51
5.14.2.12 readS . . . . .	51
5.14.2.13 rightSide . . . . .	52
5.15 Référence du fichier src/controller/saveNLoad.c . . . . .	53
5.15.1 Description détaillée . . . . .	53
5.15.2 Documentation des fonctions . . . . .	54
5.15.2.1 checkDecrypt . . . . .	54
5.15.2.2 crypt . . . . .	54
5.15.2.3 decrypt . . . . .	54
5.15.2.4 getCharKey . . . . .	54
5.15.2.5 getKey . . . . .	55
5.15.2.6 load . . . . .	56
5.15.2.7 save . . . . .	56
5.16 Référence du fichier src/controller/terminal.c . . . . .	56
5.16.1 Description détaillée . . . . .	57
5.16.2 Documentation des fonctions . . . . .	57
5.16.2.1 color . . . . .	57

5.16.2.2	fontColor	57
5.16.2.3	getColor	57
5.16.2.4	reinitColor	58
5.17	Référence du fichier src/display/grid.c	58
5.17.1	Description détaillée	58
5.17.2	Documentation des fonctions	59
5.17.2.1	borderRight	59
5.17.2.2	gridDisp	59
5.18	Référence du fichier src/display/menu.c	59
5.18.1	Description détaillée	60
5.18.2	Documentation des fonctions	60
5.18.2.1	dispDirection	60
5.18.2.2	gameMenu	60
5.18.2.3	helpUnit	60
5.18.2.4	mainMenu	60
5.18.2.5	unitList	61
5.18.2.6	unitMenu	61
5.19	Référence du fichier src/game/engine.c	61
5.19.1	Description détaillée	62
5.19.2	Documentation des fonctions	62
5.19.2.1	askCoord	62
5.19.2.2	askUnit	63
5.19.2.3	attackable	63
5.19.2.4	gameInit	63
5.19.2.5	isSurrounded	63
5.19.2.6	launchAttack	63
5.19.2.7	lineOfSight	64
5.19.2.8	movable	64
5.19.2.9	pathFind	64
5.19.2.10	playerAddUnit	64
5.19.2.11	possiblePath	64
5.19.2.12	selectUnit	65
5.19.2.13	setTarget	65
5.19.2.14	specialBoons	65
5.19.2.15	tileWalkable	65
5.19.2.16	tooMuchUnit	66
5.19.2.17	updateLimits	66
5.19.3	Documentation des variables	66
5.19.3.1	grid	66
5.19.3.2	noPlayer	66



5.20	Référence du fichier src/game/listes.c	66
5.20.1	Description détaillée	68
5.20.2	Documentation des fonctions	68
5.20.2.1	addTarget	68
5.20.2.2	addUnit	68
5.20.2.3	countUnits	69
5.20.2.4	destroyUnit	69
5.20.2.5	dumpAllLists	69
5.20.2.6	dumpList	69
5.20.2.7	printList	69
5.20.2.8	searchTarget	70
5.20.3	Documentation des variables	70
5.20.3.1	targetList	70
5.21	Référence du fichier src/game/pathList.c	70
5.21.1	Description détaillée	71
5.21.2	Documentation des fonctions	72
5.21.2.1	addCloseList	72
5.21.2.2	addOpenList	72
5.21.2.3	dumpAllPaths	72
5.21.2.4	dumpPath	72
5.21.2.5	emptyPath	72
5.21.2.6	eraseTile	73
5.21.2.7	getCurrentNode	73
5.21.2.8	getTile	73
5.21.2.9	initPath	73
5.21.2.10	next	73
5.21.2.11	outPath	74
5.21.2.12	pathHead	74
5.21.2.13	pathTail	74
5.21.2.14	previous	74
5.21.2.15	searchTile	74
5.21.2.16	setTile	75
5.21.2.17	toRightPath	75
5.22	Référence du fichier src/game/pawns.c	75
5.22.1	Description détaillée	76
5.22.2	Documentation des fonctions	76
5.22.2.1	checkPawn	76
5.22.2.2	createPawn	76
5.22.2.3	initPawn	77
5.22.2.4	makePawns	77

5.22.3	Documentation des variables	77
5.22.3.1	pawns	77
5.23	Référence du fichier src/game/turn.c	77
5.23.1	Description détaillée	78
5.23.2	Documentation des fonctions	78
5.23.2.1	changeDirection	78
5.23.2.2	endTurn	78
5.23.2.3	hasPlay	79
5.23.2.4	passTurn	79
5.23.2.5	playAttack	79
5.23.2.6	playMove	79
5.23.2.7	playTurn	79
5.23.2.8	setAction	79
5.24	Référence du fichier src/main.c	80
5.24.1	Description détaillée	80
5.25	Référence du fichier src/units/unit.c	80
5.25.1	Description détaillée	82
5.25.2	Documentation des fonctions	82
5.25.2.1	addEffect	82
5.25.2.2	allStatic	82
5.25.2.3	attack	82
5.25.2.4	canAttack	83
5.25.2.5	canBlock	84
5.25.2.6	canGetPassed	84
5.25.2.7	canMove	84
5.25.2.8	canTeleport	84
5.25.2.9	copy	85
5.25.2.10	erase	85
5.25.2.11	getSideAttacked	85
5.25.2.12	heal	85
5.25.2.13	isSleeping	86
5.25.2.14	minusEffect	86
5.25.2.15	move	86
5.25.2.16	poison	86
5.25.2.17	powerBonus	86
5.25.2.18	recover	87
5.25.2.19	setDirection	87
5.25.2.20	sleep	87
5.25.2.21	unitInit	87

Index	89
-------	----



# Chapitre 1

## Page principale

Tactics Arena est un jeu de stratégie au tour par tour se jouant sur un plateau. Chaque joueur déplace ses unités respectant des règles de déplacement propre à chaque unité, le but du jeu est d'éliminer ou paralyser toutes les unités ennemies.

### Règles

Les conditions d'égalité sont :

- **Toutes les unités actives sont détruites ou immobilisée**
- **Les joueurs passent leurs tours 3 fois d'affilé**
- **Il n'y a eu aucune rencontre entre les deux unités ennemies depuis les 30 derniers tours.**

Le joueur qui commence est défini aléatoirement par l'ordinateur au début de chaque partie.

Il n'est possible de déplacer qu'une seule unité par tour soit pour :

1. **Se déplacer**
2. **Et / ou attaquer**
3. **Et / ou changer de direction**

Le changement de direction impose d'être fait en dernier.

Chaque tour dure 1 à 2 minutes, dépendant du nombre d'unité restante. Une fois ce temps expiré :

- **Le tour se termine automatiquement si au moins une commande a été effectué**
- **Si aucune commande n'a été effectuée alors abandon automatique**

Important : toute action est irréversible.

### Sommaire

#### Tactics Arena

1. **[\\*\\*Règles\\*\\*](#)**
2. **[\\*\\*Changelog\\*\\*](#)**
  - **[\\*\\*Version 0.1\\*\\*](#)**
3. **[\\*\\*A faire\\*\\*](#)**
4. **[\\*\\*Instructions de compilation\\*\\*](#)**
5. **[\\*\\*Utilisation\\*\\*](#)**
6. **[\\*\\*Documentation\\*\\*](#)**

## A faire

- **[x]** Analyse préliminaire
- **[ ]** Compte rendu mécaniques du jeu
- **[x]** Analyse conceptuelle
- **[ ]** Implémentation
  - **[x]** Grille
  - **[x]** Menu
  - **[x]** Unités
  - **[x]** Statistiques
  - **[ ]** Mouvements
  - **[ ]** Gestion des tours
  - **[ ]** Gestion du temps
  - **[ ]** Capacités spéciales
  - **[ ]** Etat unités
- **[ ]** Test unitaires
- **[ ]** Test d'intégration
- **[ ]** Ajout de fonctionnalités
  - **[ ]** Intégration IA
  - **[ ]** GUI

## Changelog

### Version 0.1

- Mise en place sommaire des TAD

## Instructions de compilation

## Utilisation

## Documentation

## Chapitre 2

# Index des classes

### 2.1 Liste des classes

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

<a href="#">elem</a>	Représente une case de la grille . . . . .	7
<a href="#">element</a>	Représente un élément de la liste . . . . .	8
<a href="#">targetStat</a>	Représente les informations liées aux cibles . . . . .	8
<a href="#">unit</a>	Représente une unité . . . . .	9
<a href="#">unitStat</a>	Représente les statistiques d'une unité . . . . .	10
<a href="#">vector</a>	Représente les coordonnées d'un vecteur . . . . .	10





## Chapitre 3

# Index des fichiers

### 3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

include/controller/ <a href="#">manageSignal.h</a>	
En-tête gestion des signaux . . . . .	13
include/controller/ <a href="#">manageString.h</a>	
En-tête gestion des chaînes de caractères . . . . .	14
include/controller/ <a href="#">saveNLoad.h</a>	
En-tête gestion de la sauvegarde et chargement . . . . .	18
include/controller/ <a href="#">terminal.h</a>	
En-tête gestion du terminal . . . . .	19
include/display/ <a href="#">grid.h</a>	
En-tête gestion de la grille . . . . .	20
include/display/ <a href="#">menu.h</a>	
En-tête gestion des menus . . . . .	21
include/game/ <a href="#">engine.h</a>	
En-tête moteur de jeu . . . . .	23
include/game/ <a href="#">listes.h</a>	
En-tête listes de vecteurs . . . . .	28
include/game/ <a href="#">pathList.h</a>	
En-tête listes des chemins . . . . .	31
include/game/ <a href="#">pawns.h</a>	
En-tête gestions des pions . . . . .	36
include/game/ <a href="#">turn.h</a>	
En-tête gestion des tours . . . . .	37
include/units/ <a href="#">unit.h</a>	
En-tête gestion des unités . . . . .	39
src/ <a href="#">main.c</a>	
Programme principal . . . . .	80
src/controller/ <a href="#">manageSignal.c</a>	
Gestion des signaux . . . . .	46
src/controller/ <a href="#">manageString.c</a>	
Gestions des chaînes de caractères . . . . .	48
src/controller/ <a href="#">saveNLoad.c</a>	
Gestion de la sauvegarde et du chargement . . . . .	53
src/controller/ <a href="#">terminal.c</a>	
Gestion du terminal . . . . .	56
src/display/ <a href="#">grid.c</a>	
Gestion de la grille . . . . .	58
src/display/ <a href="#">menu.c</a>	
Gestion des menus . . . . .	59

src/game/ <a href="#">engine.c</a>	
Moteur de jeu . . . . .	61
src/game/ <a href="#">listes.c</a>	
Listes de vecteurs . . . . .	66
src/game/ <a href="#">pathList.c</a>	
Listes de cases sur un chemin définis . . . . .	70
src/game/ <a href="#">pawns.c</a>	
Gestion des pions . . . . .	75
src/game/ <a href="#">turn.c</a>	
Gestion des tours . . . . .	77
src/units/ <a href="#">unit.c</a>	
Gestion des unités . . . . .	80

## Chapitre 4

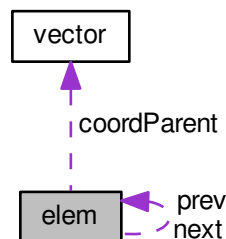
# Documentation des classes

### 4.1 Référence de la structure elem

Représente une case de la grille.

```
#include <pathList.c>
```

Graphe de collaboration de elem :



#### Attributs publics

- `vector coordParent`  
*Coordonnées de la case.*
- `int F`  
*Poids de la case.*
- `struct elem * prev`  
*Élément précédent.*
- `struct elem * next`  
*Élément suivant.*

#### 4.1.1 Description détaillée

Représente une case de la grille.

Définition à la ligne 16 du fichier pathList.c.

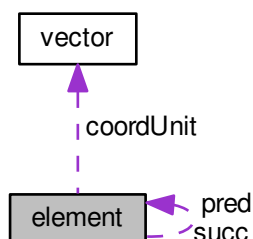
La documentation de cette structure a été générée à partir du fichier suivant :

- `src/game/pathList.c`

## 4.2 Référence de la structure element

Représente un élément de la liste.

Graphe de collaboration de element :



### Attributs publics

- `vector coordUnit`  
*Coordonnées de l'unité*
- `struct element * pred`  
*Élément précédent.*
- `struct element * succ`  
*Élément suivant.*

### 4.2.1 Description détaillée

Représente un élément de la liste.

Définition à la ligne 19 du fichier `listes.c`.

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

- `src/game/listes.c`

## 4.3 Référence de la structure targetStat

Représente les informations liées aux cibles.

```
#include <engine.h>
```

### Attributs publics

- `short vertRange`  
*Portée verticale.*
- `short horizRange`  
*Portée horizontale.*
- `short ringSize`  
*Taille de l'anneau.*
- `short line`  
*Ciblage en ligne.*

### 4.3.1 Description détaillée

Représente les informations liées aux cibles.

Définition à la ligne 54 du fichier engine.h.

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

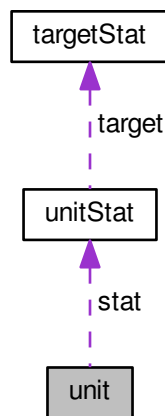
— [include/game/engine.h](#)

## 4.4 Référence de la structure unit

Représente une unité

```
#include <engine.h>
```

Graphe de collaboration de unit :



### Attributs publics

- [unitName name](#)  
*Nom de l'unité*
- [unitStat stat](#)  
*Statistiques de l'unité*
- [unitEffect effect \[NB\\_MAX\\_EFFECT\]](#)  
*Effets sur l'unité*
- [cardinal direct](#)  
*Direction de l'unité*
- int [noPlayer](#)  
*Propriétaire de l'unité*
- int [unitColor](#)  
*Couleur de l'unité*

### 4.4.1 Description détaillée

Représente une unité

Définition à la ligne 88 du fichier engine.h.

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

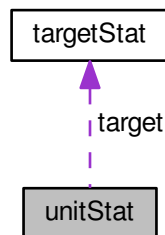
— include/game/engine.h

## 4.5 Référence de la structure unitStat

Représente les statistiques d'une unité

```
#include <engine.h>
```

Graphe de collaboration de unitStat :



### Attributs publics

- int **HP**  
*Points de vie.*
- int **POWER**  
*Puissance.*
- float **ARMOR**  
*Armure.*
- int **RECOVERY**  
*Repos.*
- float **BLOCK** [3]  
*Blocage.*
- **targetStat** target  
*Ciblage.*
- int **MOVE\_RANGE**  
*Portée mouvement.*

### 4.5.1 Description détaillée

Représente les statistiques d'une unité

Définition à la ligne 65 du fichier engine.h.

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

— include/game/engine.h

## 4.6 Référence de la structure vector

Représente les coordonnées d'un vecteur.

```
#include <engine.h>
```

### Attributs publics

- int [x](#)  
*Position x.*
- int [y](#)  
*Position y.*

#### 4.6.1 Description détaillée

Représente les coordonnées d'un vecteur.

Définition à la ligne 79 du fichier engine.h.

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

- include/game/[engine.h](#)





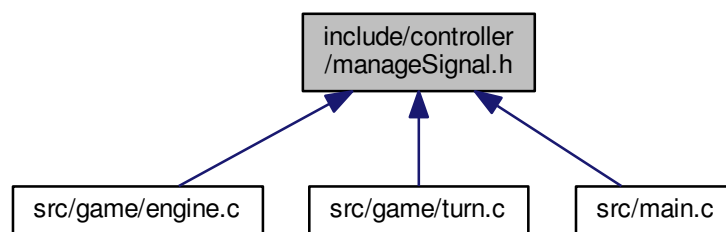
## Chapitre 5

# Documentation des fichiers

### 5.1 Référence du fichier include/controller/manageSignal.h

En-tête gestion des signaux.

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



#### Fonctions

- void `checkSignal` ()  
*Vérifie les signaux du programme.*
- void `timeDown` ()  
*Message lors du temps écoulé*
- void `freeAll` ()  
*Libère toute la mémoire.*

#### 5.1.1 Description détaillée

En-tête gestion des signaux.

##### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

##### Version

v1.00

## Date

18/12/2015

## 5.1.2 Documentation des fonctions

## 5.1.2.1 void checkSignal ( )

Vérifie les signaux du programme.

Vérifie les signaux du programme.

Définition à la ligne 77 du fichier manageSignal.c.

## 5.1.2.2 void freeAll ( )

Libère toute la mémoire.

Libère toute la mémoire.

Définition à la ligne 21 du fichier manageSignal.c.

## 5.2 Référence du fichier include/controller/manageString.h

En-tête gestion des chaînes de caractères.

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



## Fonctions

- char \* **get2Char** (char name[])  
*Récupère 2 caractères du nom de l'unité*
- char \* **getNameUnit** (unitName unit)  
*Récupère le nom de l'unité*
- void **printNameUnit** (unitName unit)  
*Affiche le nom de l'unité*
- void **getCoordS** (char coordString[], vector \*coordUnit)  
*Récupère les coordonnées d'une chaîne de caractères.*
- bool **correctCoord** (char \*coordString)  
*Vérifie l'authenticité des coordonnées d'une chaîne de caractères.*
- bool **rightSide** (char \*coordString)  
*Vérifie que l'unité est dans le bon camp.*
- int **readS** (char \*string)  
*Lecture sécurisée d'une chaîne.*
- long **readLong** ()  
*Lecture sécurisée d'un long.*
- double **readDouble** ()  
*Lecture sécurisée d'un double.*
- void **clearBuffer** ()  
*Vide la mémoire tampon.*
- char \* **getDirectionUnit** (cardinal direct)  
*Récupère le nom de la direction.*
- char \* **getNameEffect** (unitEffect effect)  
*Récupère le nom de l'effet.*

### 5.2.1 Description détaillée

En-tête gestion des chaînes de caractères.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.2.2 Documentation des fonctions

#### 5.2.2.1 void clearBuffer ( )

Vide la mémoire tampon.

Vide la mémoire tampon.

Définition à la ligne 253 du fichier manageString.c.

#### 5.2.2.2 char\* get2Char ( char *name*[ ] )

Récupère 2 caractères du nom de l'unité

##### Paramètres

<i>name</i>	Nom de l'unité
-------------	----------------

##### Renvoie

Retourne 2 caractères liés au nom de l'unité

Définition à la ligne 47 du fichier manageString.c.

#### 5.2.2.3 void getCoordS ( char *coordString*[ ], vector \* *coordUnit* )

Récupère les coordonnées d'une chaîne de caractères.

Récupère les coordonnées d'une chaîne de caractères.

##### Paramètres

<i>coordString</i>	Coordonnées saisie par l'utilisateur
<i>coordUnit</i>	Coordonnées de l'unité récupérées de la saisie utilisateur

Définition à la ligne 19 du fichier manageString.c.

#### 5.2.2.4 char\* getDirectionUnit ( cardinal *direct* )

Récupère le nom de la direction.

Récupère le nom de la direction.

**Paramètres**

<i>direct</i>	Direction de l'unité
---------------	----------------------

**Renvoie**

Retourne La direction sous forme de chaîne

Définition à la ligne 235 du fichier manageString.c.

**5.2.2.5 char\* getNameEffect ( unitEffect effect )**

Récupère le nom de l'effet.

Récupère le nom de l'effet.

**Paramètres**

<i>effect</i>	Effet provenant de la liste énumérée
---------------	--------------------------------------

**Renvoie**

Nom de l'effet sous forme de chaîne de caractère

Définition à la ligne 220 du fichier manageString.c.

**5.2.2.6 char\* getNameUnit ( unitName name )**

Récupère le nom de l'unité

Récupère le nom de l'unité

**Paramètres**

<i>name</i>	Nom de l'unité provenant de la liste énumérée
-------------	---

**Renvoie**

Nom de l'unité sous forme de chaîne

Définition à la ligne 189 du fichier manageString.c.

**5.2.2.7 void printNameUnit ( unitName unit )**

Affiche le nom de l'unité

**Paramètres**

<i>unit</i>	Nom de l'unité provenant de la liste énumérée
-------------	---

Définition à la ligne 246 du fichier manageString.c.

**5.2.2.8 double readDouble ( )**

Lecture sécurisée d'un double.

Lecture sécurisée d'un double.

**Renvoie**

Retourne le double saisie ou 0.0 en cas d'erreur

Définition à la ligne 305 du fichier manageString.c.

### 5.2.2.9 long readLong ( )

Lecture sécurisée d'un long.

Lecture sécurisée d'un long.

#### Renvoie

Retourne le long saisie ou 0 en cas d'erreur

Définition à la ligne 291 du fichier manageString.c.

### 5.2.2.10 int readS ( char \* *string* )

Lecture sécurisée d'une chaîne.

Lecture sécurisée d'une chaîne.

#### Paramètres

<i>string</i>	Chaîne de caractère à vérifier
---------------	--------------------------------

#### Renvoie

Retourne 1 si chaîne correcte

Définition à la ligne 265 du fichier manageString.c.

### 5.2.2.11 bool rightSide ( char \* *coordString* )

Vérifie que l'unité est dans le bon camp.

Vérifie que l'unité est dans le bon camp.

#### Paramètres

<i>coordString</i>	Coordonnées sous forme de chaîne
--------------------	----------------------------------

#### Renvoie

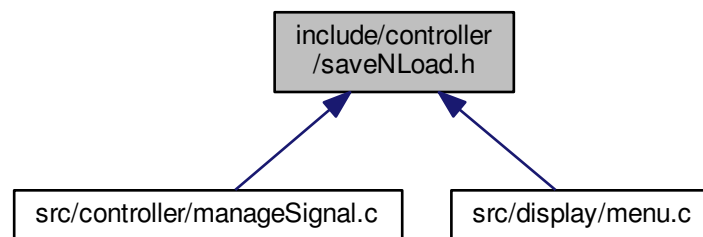
Retourne vrai si du bon côté

Définition à la ligne 165 du fichier manageString.c.

### 5.3 Référence du fichier include/controller/saveNLoad.h

En-tête gestion de la sauvegarde et chargement.

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



#### Fonctions

- void **save** ()  
*Sauvegarde la partie.*
- void **load** ()  
*Charge une partie.*

#### 5.3.1 Description détaillée

En-tête gestion de la sauvegarde et chargement.

##### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

##### Version

v1.00

##### Date

18/12/2015

#### 5.3.2 Documentation des fonctions

##### 5.3.2.1 void load ( )

Charge une partie.

Charge une partie.

Définition à la ligne 201 du fichier saveNLoad.c.

## 5.3.2.2 void save ( )

Sauvegarde la partie.

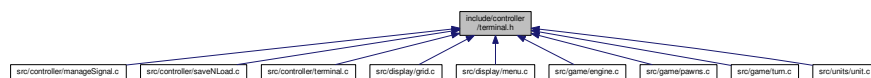
Sauvegarde la partie.

Définition à la ligne 158 du fichier saveNLoad.c.

## 5.4 Référence du fichier include/controller/terminal.h

En-tête gestion du terminal.

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



### Énumérations

- enum `terminal` {  
**black, red, green, yellow,**  
**blue, magenta, cyan, white** }  
[terminal.h](#)
- enum `screen` { **reinit** = 0, **blink** = 5, **invertColor** = 7 }  
[terminal.h](#)

### Fonctions

- void `color` (int color, char type[])  
*Met en couleurs le texte ou l'écran.*
- void `fontColor` (int color)  
*Met en couleurs la police.*
- void `clearScreen` ()  
*Efface l'écran.*
- void `reinitColor` ()  
*Réinitialise les couleurs.*

#### 5.4.1 Description détaillée

En-tête gestion du terminal.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

## 5.4.2 Documentation des fonctions

### 5.4.2.1 void color ( int color, char type[] )

Met en couleurs le texte ou l'écran.

Met en couleurs le texte ou l'écran.

Paramètres

<i>color</i>	Couleur à utiliser
<i>type</i>	Texte à changer de couleur ou arrière plan

Définition à la ligne 51 du fichier terminal.c.

### 5.4.2.2 void fontColor ( int color )

Met en couleurs la police.

Met en couleurs la police.

Paramètres

<i>color</i>	Couleur à utiliser
--------------	--------------------

Définition à la ligne 76 du fichier terminal.c.

### 5.4.2.3 void reinitColor ( )

Réinitialise les couleurs.

Réinitialise les couleurs.

Définition à la ligne 67 du fichier terminal.c.

## 5.5 Référence du fichier include/display/grid.h

En-tête gestion de la grille.

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



## Macros

- #define **RB** "\e(0\x6a\e(B"  
188 Right Bottom corner
- #define **RT** "\e(0\x6b\e(B"  
187 Right Top corner
- #define **LT** "\e(0\x6c\e(B"  
201 Left Top corner
- #define **LB** "\e(0\x6d\e(B"  
200 Left Bottom corner
- #define **VL** "\e(0\x78\e(B"



186 Vertical Line  
— #define HL "\e(0\x71\e(B"  
205 Horizontal Line

## Fonctions

— void gridDisp ()  
Affiche la grille.

### 5.5.1 Description détaillée

En-tête gestion de la grille.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.5.2 Documentation des fonctions

#### 5.5.2.1 void gridDisp ( )

Affiche la grille.

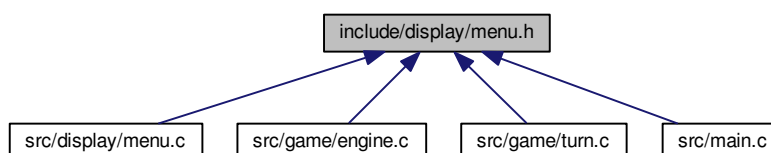
Affiche la grille.

Définition à la ligne 78 du fichier grid.c.

## 5.6 Référence du fichier include/display/menu.h

En-tête gestion des menus.

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



## Fonctions

— void mainMenu ()  
Menu principal.  
— void gameMenu ()

- Menu de jeu.*  
— void `unitMenu` (int choice)
- Menu de sélection de l'unité*  
— void `unitList` ()
- Listes des unités du jeu.*  
— void `mainHelp` ()
- Menu d'aide principal.*  
— void `helpUnit` ()
- Menu d'aide des unités.*  
— void `dispDirection` ()  
*Affiche la liste des directions.*

### 5.6.1 Description détaillée

En-tête gestion des menus.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.6.2 Documentation des fonctions

#### 5.6.2.1 void `dispDirection` ( )

Affiche la liste des directions.

Affiche la liste des directions.

Définition à la ligne 458 du fichier menu.c.

#### 5.6.2.2 void `gameMenu` ( )

Menu de jeu.

Menu de jeu.

Définition à la ligne 470 du fichier menu.c.

#### 5.6.2.3 void `helpUnit` ( )

Menu d'aide des unités.

Menu d'aide des unités.

Définition à la ligne 222 du fichier menu.c.

#### 5.6.2.4 void `mainMenu` ( )

Menu principal.

Menu principal.

Définition à la ligne 24 du fichier menu.c.

## 5.6.2.5 void unitList ( )

Listes des unités du jeu.

Listes des unités du jeu.

Définition à la ligne 593 du fichier menu.c.

## 5.6.2.6 void unitMenu ( int choice )

Menu de sélection de l'unité

Paramètres

choice	Choix de l'action pour l'unité
--------	--------------------------------

Définition à la ligne 517 du fichier menu.c.

## 5.7 Référence du fichier include/game/engine.h

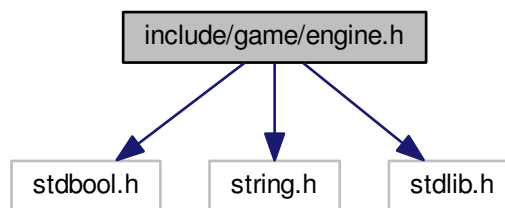
En-tête moteur de jeu.

```
#include <stdbool.h>
```

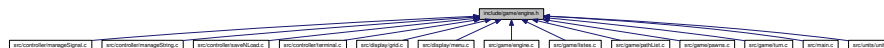
```
#include <string.h>
```

```
#include <stdlib.h>
```

Graphe des dépendances par inclusion de engine.h :



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



## Classes

- struct [targetStat](#)  
*Représente les informations liées aux cibles.*
- struct [unitStat](#)  
*Représente les statistiques d'une unité*
- struct [vector](#)  
*Représente les coordonnées d'un vecteur.*
- struct [unit](#)  
*Représente une unité*

## Macros

- #define **N** 11  
*Taille de la grille.*
- #define **NB\_LISTS\_ENGINE** 2  
*Nombre de liste additionnelles nécessaires pour le jeu.*
- #define **NB\_PLAYERS** 2  
*Nombre de joueurs.*
- #define **NB\_LINES** 3  
*Limite du camp du joueur.*
- #define **NB\_UNITS** 21  
*Nombre d'unités dans le jeu.*
- #define **NB\_MAX\_EFFECT** 6  
*Nombre total de status différent.*
- #define **MANDATORY\_STATS** 12  
*Nombre de stats obligatoire.*
- #define **FIRST\_PLAYER** 0  
*Définis la valeur du premier joueur.*
- #define **NB\_MAX\_KN** 3  
*Nombre max de Guerrier par joueur.*
- #define **NB\_MAX\_SC** 2  
*Nombre max de Recrue par joueur.*
- #define **NB\_MAX\_SG** 2  
*Nombre max de Golem de pierre par joueur.*
- #define **NB\_MAX\_LT** 1  
*Nombre max de Lightning totem par joueur.*
- #define **NB\_MAX\_DR** 2  
*Nombre max de Dragon par joueur.*
- #define **NB\_MAX\_FU** 2  
*Nombre max de Furgon par joueur.*
- #define **NB\_MAX\_UNIT** 5  
*Nombre max d'unité par joueur.*
- #define **NB\_MAX\_DECOR** 7  
*Nombre max de décor.*

## Énumérations

- enum **cardinal** { **north, east, south, west** }  
*engine.h*
- enum **unitName** {  
**empty, decors, knight, scout,**  
**assassin, cleric, pyromancer, enchantress,**  
**dragonborn, darkWitch, lightningTotem, barrierTotem,**  
**mudGolem, golemAmbusher, frostGolem, stoneGolem,**  
**dragonTyrant, berserker, beastRider, poisonWisp,**  
**furgon** }  
*engine.h*
- enum **unitEffect** {  
**none, POWER\_BONUS, ARMOR\_BONUS, BARRIER,**  
**POISON, PARALYSE, FOCUS** }  
*engine.h*

## Fonctions

- bool **isSurrounded** (**vector** currentUnit)  
*Vérifie si une unité est entourée.*
- void **gameInit** ()  
*Initialise le jeu.*
- bool **selectUnit** (**vector** \*coordUnit)

- *Sélectionne une unité*  
void **setTarget** (unitName name, **vector** coordUnit, int colorDisp)
- *Définit les cibles.*  
void **launchAttack** (**vector** coordSource, **vector** coordTarget)
- *Lance une attaque.*  
void **movable** (int colorDisp)
- *Fait la liste des unités déplaçables.*  
void **attackable** (int colorDisp)
- *Fait la liste des unités pouvant attaquer.*  
void **tileWalkable** (**vector** coordUnit, int colorDisp)
- *Fait la liste des cases atteignables par l'unité*  
bool **possiblePath** (**vector** coordUnit)
- *Vérifie qu'un chemin est possible.*  
bool **pathFind** (**vector**, **vector**)
- *Trouve une chemin vers la destination.*  
void **startGame** ()
- *Débute la partie.*  
void **lineOfSight** (**vector**, **vector**)
- *Ligne de vue de l'archer.*

## Variables

- **unit grid** [N][N]  
*Représentation d'une grille d'unité globale.*
- int **noPlayer**  
*Représentation du joueur.*

### 5.7.1 Description détaillée

En-tête moteur de jeu.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.7.2 Documentation des fonctions

#### 5.7.2.1 void attackable ( int colorDisp )

Fait la liste des unités pouvant attaquer.

#### Paramètres

<i>colorDisp</i>	Couleur d'affichage
------------------	---------------------

Définition à la ligne 164 du fichier engine.c.

#### 5.7.2.2 void gamelnit ( )

Initialise le jeu.

Initialise le jeu.

Définition à la ligne 750 du fichier engine.c.

### 5.7.2.3 bool isSurrounded ( vector currentUnit )

Vérifie si une unité est entourée.

Vérifie si une unité est entourée.

#### Paramètres

<i>currentUnit</i>	Unité courante
--------------------	----------------

#### Renvoie

Retourne vrai si unité entourée

Définition à la ligne 215 du fichier engine.c.

### 5.7.2.4 void launchAttack ( vector coordSource, vector coordTarget )

Lance une attaque.

Lance une attaque.

#### Paramètres

<i>coordSource</i>	Nom de l'unité source
<i>coordTarget</i>	Coordonnées de la cible

Définition à la ligne 336 du fichier engine.c.

### 5.7.2.5 void lineOfSight ( vector coordSource, vector coordTarget )

Ligne de vue de l'archer.

#### Paramètres

<i>coordSource</i>	Coordonnées de l'unité attaquante
<i>coordTarget</i>	Coordonnées de la cible

Définition à la ligne 117 du fichier engine.c.

### 5.7.2.6 void movable ( int colorDisp )

Fait la liste des unités déplaçables.

#### Paramètres

<i>colorDisp</i>	Couleur d'affichage
------------------	---------------------

Définition à la ligne 141 du fichier engine.c.

### 5.7.2.7 bool pathFind ( vector coordUnit, vector coordTarget )

Trouve une chemin vers la destination.

Trouve une chemin vers la destination.

#### Paramètres

<i>coordUnit</i>	Coordonnées de l'unité
<i>coordTarget</i>	Coordonnées de la cible

**Renvoie**

Retourne vrai si chemin trouvé

Définition à la ligne 60 du fichier engine.c.

**5.7.2.8 bool possiblePath ( vector *coordUnit* )**

Vérifie qu'un chemin est possible.

Vérifie qu'un chemin est possible.

**Paramètres**

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

**Renvoie**

Retourne Vrai si chemin possible vers une quelconque position

Définition à la ligne 31 du fichier engine.c.

**5.7.2.9 bool selectUnit ( vector \* *coordUnit* )**

Sélectionne une unité

**Paramètres**

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

**Renvoie**

Retourne vrai si unité bien sélectionnée

Définition à la ligne 416 du fichier engine.c.

**5.7.2.10 void setTarget ( unitName *name*, vector *coordUnit*, int *colorDisp* )**

Définis les cibles.

Définis les cibles.

**Paramètres**

<i>name</i>	Nom du pion
<i>coordUnit</i>	Coordonnées de l'unité
<i>colorDisp</i>	Couleur d'affichage

Définition à la ligne 252 du fichier engine.c.

**5.7.2.11 void tileWalkable ( vector *coordUnit*, int *colorDisp* )**

Fait la liste des cases atteignables par l'unité

Fait la liste des cases atteignables par l'unité

## Paramètres

<i>coordUnit</i>	Coordonnées de l'unité
<i>colorDisp</i>	Couleur d'affichage

Définition à la ligne 188 du fichier engine.c.

### 5.7.3 Documentation des variables

#### 5.7.3.1 unit grid[N][N]

Représentation d'une grille d'unité globale.

Représentation d'une grille d'unité globale.

Définition à la ligne 23 du fichier engine.c.

#### 5.7.3.2 int noPlayer

Représentation du joueur.

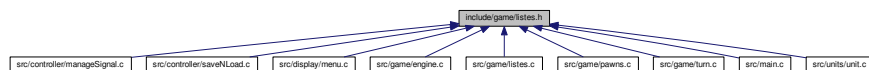
Représentation du joueur.

Définition à la ligne 24 du fichier engine.c.

## 5.8 Référence du fichier include/game/listes.h

En-tête listes de vecteurs.

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



## Macros

- `#define MAX_JOUEUR 3`  
Nombre max de listes.

## Fonctions

- void `init_liste` (int)  
Initialise une liste.
- void `initLists` ()  
Initialise les listes.
- int `liste_vide` (int)  
Vérifie si une liste est vide.
- int `hors_liste` (int)  
Vérifie si l'élément courant est hors de la liste.
- void `en_tete` (int)  
Se met en tête de la liste.
- void `en_queue` (int)  
Se met en queue de la liste.
- void `precedent` (int)



- Se positionne sur l'élément précédent.*
- void `suivant` (int)
- Se positionne sur l'élément suivant.*
- void `valeur_elt` (int, `vector` \*v)
- Récupère la valeur de l'élément.*
- void `modif_elt` (int, `vector` v)
- Modifie la valeur de l'élément.*
- void `oter_elt` (int)
- Supprime l'élément.*
- void `ajout_droit` (int, `vector` v)
- Ajoute à droite l'élément.*
- void `ajout_gauche` (int, `vector` v)
- Ajoute à gauche l'élément.*
- void `dumpList` (short nbList)
- Vide une liste.*
- void `dumpAllLists` ()
- Vide les listes.*
- void `addUnit` (`vector` coordUnit)
- Ajoute une unité*
- void `destroyUnit` (`vector` coordUnit)
- Détruit une unité*
- void `printList` (short numList)
- Affiche la liste.*
- int `countUnits` ()
- Compte le nombre d'unités.*
- void `addTarget` (`unitName` name, `vector` coordUnit)
- Ajoute une cible.*
- bool `searchTarget` (int numList, `vector` coordTarget)
- Cherche une cible.*

## Variables

- int `targetList`
- Cibles potentielles.*

### 5.8.1 Description détaillée

En-tête listes de vecteurs.

Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

Version

v1.00

Date

18/12/2015

### 5.8.2 Documentation des fonctions

#### 5.8.2.1 void addTarget ( unitName name, vector coordUnit )

Ajoute une cible.

Ajoute une cible.

**Paramètres**

<i>name</i>	Nom de l'unité
<i>coordUnit</i>	Coordonnées de l'unité

Définition à la ligne 211 du fichier listes.c.

**5.8.2.2 void addUnit ( vector coordUnit )**

Ajoute une unité

Ajoute une unité

**Paramètres**

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

Définition à la ligne 179 du fichier listes.c.

**5.8.2.3 int countUnits ( )**

Compte le nombre d'unités.

Compte le nombre d'unités.

**Renvoie**

Retourne le nombre d'unité

Définition à la ligne 280 du fichier listes.c.

**5.8.2.4 void destroyUnit ( vector coordUnit )**

Détruit une unité

Détruit une unité

**Paramètres**

<i>coordUnit</i>	Coordonnées de l'unité à détruire
------------------	-----------------------------------

Définition à la ligne 260 du fichier listes.c.

**5.8.2.5 void dumpAllLists ( )**

Vide les listes.

Vide les listes.

Définition à la ligne 167 du fichier listes.c.

**5.8.2.6 void dumpList ( short nbList )**

Vide une liste.

**Paramètres**

<i>nbList</i>	Numéro de la liste à vider
---------------	----------------------------

Définition à la ligne 155 du fichier listes.c.

#### 5.8.2.7 void printList ( short numList )

Affiche la liste.

Affiche la liste.

Paramètres

<i>numList</i>	Numéro de liste
----------------	-----------------

Définition à la ligne 222 du fichier listes.c.

#### 5.8.2.8 bool searchTarget ( int numList, vector coordTarget )

Cherche une cible.

Cherche une cible.

Paramètres

<i>numList</i>	Numéro de la liste
<i>coordTarget</i>	Coordonnées de la cible

Renvoie

Retourne vrai si cible trouvée

Définition à la ligne 301 du fichier listes.c.

### 5.8.3 Documentation des variables

#### 5.8.3.1 int targetList

Cibles potentielles.

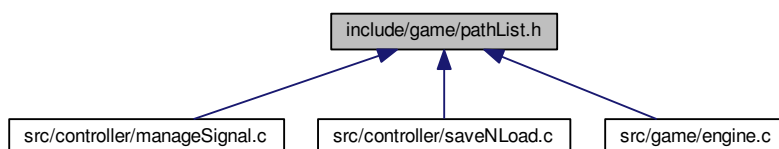
Cibles potentielles.

Définition à la ligne 28 du fichier listes.c.

## 5.9 Référence du fichier include/game/pathList.h

En-tête listes des chemins.

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



## Fonctions

— void [initPath](#) (int)

- *Initialise le chemin.*  
void `initPaths` ()
- *Initialise les chemins.*  
int `emptyPath` (int n)
- *Vérifie si chemin vide.*  
int `outPath` (int n)
- *Vérifie si l'élément courant est en dehors de la liste.*  
void `pathHead` (int n)
- *Se met en tête de la liste.*  
void `pathTail` (int n)
- *Se met en queue de la liste.*  
void `previous` (int n)
- *Se positionne sur l'élément précédent.*  
void `next` (int n)
- *Se positionne sur l'élément suivant.*  
void `getTile` (int n, `vector` \*v, int \*F)
- *Récupère la valeur d'une case.*  
void `setTile` (int n, `vector` v, int F)
- *Modifie la valeur d'une case.*  
void `eraseTile` (int n)
- *Efface une case.*  
void `toRightPath` (int n, `vector` v, int F)
- *Ajoute une case à droite.*  
void `dumpPath` (short nbList)
- *Vide le chemin.*  
void `dumpAllPaths` ()
- *Vide les chemins.*  
bool `searchTile` (int n, `vector`)
- *Cherche une case.*  
`vector` `getCurrentNode` (int n)
- *Récupère la case ayant le plus petit poids.*  
void `addCloseList` (`vector`, int)
- *Ajoute la case à la liste fermée.*  
void `addOpenList` (`vector`, int)
- *Ajoute la case à la liste ouverte.*

### 5.9.1 Description détaillée

En-tête listes des chemins.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.9.2 Documentation des fonctions

#### 5.9.2.1 void addCloseList ( `vector` *current*, int *F* )

Ajoute la case à la liste fermée.

Ajoute la case à la liste fermée.

## Paramètres

<i>current</i>	Destination
<i>F</i>	Poids de la case

Définition à la ligne 262 du fichier pathList.c.

5.9.2.2 void addOpenList ( vector *current*, int *F* )

Ajoute la case à la liste ouverte.

Ajoute la case à la liste ouverte.

## Paramètres

<i>current</i>	Destination
<i>F</i>	Poids de la case

Définition à la ligne 285 du fichier pathList.c.

## 5.9.2.3 void dumpAllPaths ( )

Vide les chemins.

Vide les chemins.

Définition à la ligne 226 du fichier pathList.c.

5.9.2.4 void dumpPath ( short *nbList* )

Vide le chemin.

Vide le chemin.

## Paramètres

<i>nbList</i>	Numéro de la liste à vider
---------------	----------------------------

Définition à la ligne 214 du fichier pathList.c.

5.9.2.5 int emptyPath ( int *n* )

Vérifie si chemin vide.

Vérifie si chemin vide.

## Paramètres

<i>n</i>	Chemin
----------	--------

## Renvoie

Retourne le drapeau si pas vide

Définition à la ligne 54 du fichier pathList.c.

5.9.2.6 void eraseTile ( int *n* )

Efface une case.

Efface une case.

**Paramètres**

<i>n</i>	Liste
----------	-------

Définition à la ligne 170 du fichier pathList.c.

**5.9.2.7 vector getCurrentNode ( int *n* )**

Récupère la case ayant le plus petit poids.

Récupère la case ayant le plus petit poids.

**Paramètres**

<i>n</i>	Liste dans laquelle chercher
----------	------------------------------

Définition à la ligne 130 du fichier pathList.c.

**5.9.2.8 void getTile ( int *n*, vector \* *v*, int \* *F* )**

Récupère la valeur d'une case.

Récupère la valeur d'une case.

**Paramètres**

<i>n</i>	Chemin
<i>v</i>	Position de la case
<i>F</i>	Poids de la case

Définition à la ligne 117 du fichier pathList.c.

**5.9.2.9 void initPath ( int *n* )**

Initialise le chemin.

**Paramètres**

<i>n</i>	Chemin
----------	--------

Définition à la ligne 31 du fichier pathList.c.

**5.9.2.10 void next ( int *n* )**

Se positionne sur l'élément suivant.

Se positionne sur l'élément suivant.

**Paramètres**

<i>n</i>	Chemin
----------	--------

Définition à la ligne 105 du fichier pathList.c.

**5.9.2.11 int outPath ( int *n* )**

Vérifie si l'élément courant est en dehors de la liste.

Vérifie si l'élément courant est en dehors de la liste.

## Paramètres

<i>n</i>	Liste
----------	-------

## Renvoie

Retourne l'élément courant si pas hors liste

Définition à la ligne 65 du fichier pathList.c.

5.9.2.12 void pathHead ( int *n* )

Se met en tête de la liste.

Se met en tête de la liste.

## Paramètres

<i>n</i>	Chemin
----------	--------

Définition à la ligne 75 du fichier pathList.c.

5.9.2.13 void pathTail ( int *n* )

Se met en queue de la liste.

Se met en queue de la liste.

## Paramètres

<i>n</i>	Chemin
----------	--------

Définition à la ligne 85 du fichier pathList.c.

5.9.2.14 void previous ( int *n* )

Se positionne sur l'élément précédent.

## Paramètres

<i>n</i>	Chemin
----------	--------

Définition à la ligne 95 du fichier pathList.c.

5.9.2.15 bool searchTile ( int *n*, vector *coordTile* )

Cherche une case.

Cherche une case.

## Paramètres

<i>n</i>	Liste
<i>coordTile</i>	Position de la case

Définition à la ligne 239 du fichier pathList.c.

5.9.2.16 void setTile ( int *n*, vector *v*, int *F* )

Modifie la valeur d'une case.

Modifie la valeur d'une case.

## Paramètres

$n$	Liste
$v$	Nouvelle position
$F$	Nouveau poids

Définition à la ligne 157 du fichier pathList.c.

5.9.2.17 void toRightPath ( int  $n$ , vector  $v$ , int  $F$  )

Ajoute une case à droite.

## Paramètres

$n$	Liste
$v$	Position à ajouter
$F$	Poids à ajouter

Définition à la ligne 191 du fichier pathList.c.

## 5.10 Référence du fichier include/game/pawns.h

En-tête gestions des pions.

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



## Fonctions

- void [makePawns](#) ()  
Crée les pions.

## Variables

- [unit](#) \* [pawns](#)  
Tableaux des pions.
- int [sizePawns](#)  
Nombre de pions.

## 5.10.1 Description détaillée

En-tête gestions des pions.

## Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

## Version

v1.00



## Date

18/12/2015

## 5.10.2 Documentation des fonctions

## 5.10.2.1 void makePawns ( )

Crée les pions.

Crée les pions.

Définition à la ligne 109 du fichier pawns.c.

## 5.10.3 Documentation des variables

## 5.10.3.1 unit\* pawns

Tableaux des pions.

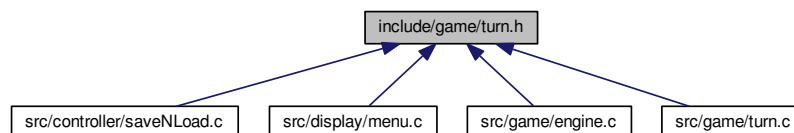
Tableaux des pions.

Définition à la ligne 16 du fichier pawns.c.

## 5.11 Référence du fichier include/game/turn.h

En-tête gestion des tours.

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



## Macros

- #define **TIME\_BY\_UNIT** 12  
*Temps par unité possédée par le joueur.*
- #define **MIN\_TIME** 60  
*Temps minimum par tour.*
- #define **MAX\_TIME** 120  
*Temps maximum par tour.*

## Fonctions

- void **playTurn** ()  
*Joue un tour.*
- void **playAttack** ()  
*Joue une attaque.*
- void **playMove** ()  
*Joue une mouvement.*
- void **passTurn** ()

- Passe le tour.*
  - void `changeDirection` ()
- Change la direction.*
  - bool `hasPlay` ()
- Vérifie si une action a été effectuée.*
  - void `surrender` ()
- Abandonne la partie.*

## Variables

- int `hasMoved`
  - Joueur a joué*
- int `hasAttacked`
  - Joueur a attaqué*
- int `hasPassed`
  - Joueur a passé son tour.*
- int `hasSurrender`
  - Joueur a abandonné la partie.*

### 5.11.1 Description détaillée

En-tête gestion des tours.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.11.2 Documentation des fonctions

#### 5.11.2.1 void `changeDirection` ( )

Change la direction.

Change la direction.

Définition à la ligne 57 du fichier turn.c.

#### 5.11.2.2 bool `hasPlay` ( )

Vérifie si une action a été effectuée.

Vérifie si une action a été effectuée.

#### Renvoie

Retourne vrai si le joueur a joué sinon faux

Définition à la ligne 245 du fichier turn.c.

### 5.11.2.3 void passTurn ( )

Passe le tour.

Passe le tour.

Définition à la ligne 235 du fichier turn.c.

### 5.11.2.4 void playAttack ( )

Joue une attaque.

Joue une attaque.

Définition à la ligne 104 du fichier turn.c.

### 5.11.2.5 void playMove ( )

Joue une mouvement.

Joue une mouvement.

Définition à la ligne 177 du fichier turn.c.

### 5.11.2.6 void playTurn ( )

Joue un tour.

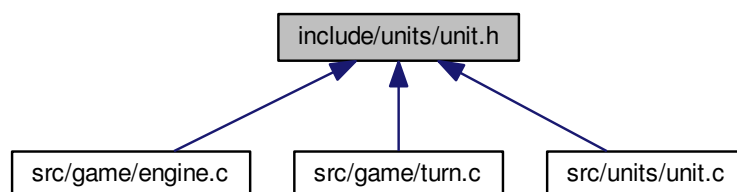
Joue un tour.

Définition à la ligne 272 du fichier turn.c.

## 5.12 Référence du fichier include/units/unit.h

En-tête gestion des unités.

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



## Fonctions

- bool **canGetPassed** (unit \*target)  
*Vérifie si le passage est autorisé*
- bool **canBlock** (unit \*target)  
*Vérifie si l'unité peut bloquer.*
- bool **canAttack** (unit \*target)

- *Vérifie si l'unité peut attaquer.*  
bool `canMove` (`unit` \*target)
- *Vérifie si l'unité peut bouger.*  
bool `canTeleport` (`unitName` name)
- *Vérifie si l'unité peut se téléporter.*  
int `getSideAttacked` (`vector` source, `vector` target)
- *Récupère le côté attaqué*  
void `heal` (`unitName` name)
- *Soigne les unités.*  
void `attack` (`vector` source, `vector` target)
- *Attaque une unité*  
bool `copy` (`unit` \*destination, `unit` \*source)
- *Copie la structure source vers destination.*  
void `move` (`vector` destination, `vector` source)
- *Bouge une unité*  
void `addEffect` (`vector` target, `unitEffect` effect)
- *Ajoute un effet à une unité*  
void `unitInit` (short `noPlayer`, `vector` coordUnit)
- *Initialise une unité*  
void `setDirection` (`vector` coordUnit, int dir)
- *Définit la direction d'une unité*  
void `erase` (`unit` \*)
- *Efface une unité de la grille.*  
bool `isSleeping` (`vector`)
- *Vérifie si une unité est endormie.*  
void `recover` ()
- *Se réveille tour par tour.*  
bool `allStatic` (int numPlayer)
- *Vérifie si toutes les unités sont immobilisées.*  
void `minusEffect` ()
- *Diminue le temps de l'effet.*  
void `poison` ()
- *Empoisonne une unité*  
void `powerBonus` ()
- *Octroie un bonus de puissance selon certaines conditions.*  
void `sleep` (`vector`)
- *Endors une unité*

### 5.12.1 Description détaillée

En-tête gestion des unités.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.12.2 Documentation des fonctions

#### 5.12.2.1 void addEffect ( `vector` target, `unitEffect` effect )

Ajoute un effet à une unité

Ajoute un effet à une unité

## Paramètres

<i>target</i>	Position unité cible
<i>effect</i>	Effet à appliquer sur la cible

Définition à la ligne 437 du fichier unit.c.

5.12.2.2 `bool allStatic ( int numPlayer )`

Vérifie si toutes les unités sont immobilisées.

Vérifie si toutes les unités sont immobilisées.

## Paramètres

<i>numPlayer</i>	Numéro du joueur
------------------	------------------

## Renvoie

Retourne vrai si toutes les unités sont immobilisés

Définition à la ligne 529 du fichier unit.c.

5.12.2.3 `void attack ( vector source, vector target )`

Attaque une unité

Attaque une unité

## Paramètres

<i>source</i>	Unité attaquante
<i>target</i>	Unité cible

Définition à la ligne 184 du fichier unit.c.

5.12.2.4 `bool canAttack ( unit * target )`

Vérifie si l'unité peut attaquer.

Vérifie si l'unité peut attaquer.

## Paramètres

<i>target</i>	Unité à analyser
---------------	------------------

## Renvoie

Retourne vrai si l'unité peut attaquer

Définition à la ligne 86 du fichier unit.c.

5.12.2.5 `bool canBlock ( unit * target )`

Vérifie si l'unité peut bloquer.

Vérifie si l'unité peut bloquer.

## Paramètres

<i>target</i>	Unité à analyser
---------------	------------------

## Renvoie

Retourne vrai si l'unité peut bloquer

Définition à la ligne 66 du fichier unit.c.

5.12.2.6 **bool canGetPassed ( unit \* target )**

Vérifie si le passage est autorisé

Vérifie si le passage est autorisé

## Paramètres

<i>target</i>	Unité à analyser
---------------	------------------

## Renvoie

Retourne vrai si passage autorisé

Définition à la ligne 46 du fichier unit.c.

5.12.2.7 **bool canMove ( unit \* target )**

Vérifie si l'unité peut bouger.

Vérifie si l'unité peut bouger.

## Paramètres

<i>target</i>	Unité à analyser
---------------	------------------

## Renvoie

Retourne vraie si l'unité peut se mouvoir

Définition à la ligne 106 du fichier unit.c.

5.12.2.8 **bool canTeleport ( unitName name )**

Vérifie si l'unité peut se téléporter.

Vérifie si l'unité peut se téléporter.

## Paramètres

<i>name</i>	Nom de l'unité
-------------	----------------

## Renvoie

Retourne vraie si l'unité peut se déplacer

Définition à la ligne 125 du fichier unit.c.

5.12.2.9 **bool copy ( unit \* destination, unit \* source )**

Copie la structure source vers destination.

Copie la structure source vers destination.

## Paramètres

<i>destination</i>	Structure destination
<i>source</i>	Structure source

## Renvoie

Retourne vrai si copie bien déroulée

Définition à la ligne 236 du fichier unit.c.

## 5.12.2.10 void erase ( unit \* source )

Efface une unité de la grille.

## Paramètres

<i>source</i>	Source à effacer
---------------	------------------

Définition à la ligne 271 du fichier unit.c.

## 5.12.2.11 int getSideAttacked ( vector source, vector target )

Récupère le côté attaqué

Récupère le côté attaqué

## Paramètres

<i>source</i>	Position unité source
<i>target</i>	Position unité cible

## Renvoie

Le côté attaqué

Définition à la ligne 168 du fichier unit.c.

## 5.12.2.12 void heal ( unitName name )

Soigne les unités.

## Paramètres

<i>name</i>	Nom de l'unité soignant
-------------	-------------------------

Définition à la ligne 137 du fichier unit.c.

## 5.12.2.13 bool isSleeping ( vector pos )

Vérifie si une unité est endormie.

Vérifie si une unité est endormie.

## Paramètres

<i>pos</i>	Position de l'unité
------------	---------------------

## Renvoie

Retourne vrai si endormi

Définition à la ligne 490 du fichier unit.c.

**5.12.2.14 void minusEffect ( )**

Diminue le temps de l'effet.

Diminue le temps de l'effet.

Définition à la ligne 455 du fichier unit.c.

**5.12.2.15 void move ( vector destination, vector source )**

Bouge une unité

Bouge une unité

**Paramètres**

<i>destination</i>	Destination souhaitée
<i>source</i>	Position de l'unité

Définition à la ligne 351 du fichier unit.c.

**5.12.2.16 void poison ( )**

Empoisonne une unité

Empoisonne une unité

Définition à la ligne 386 du fichier unit.c.

**5.12.2.17 void powerBonus ( )**

Octroie un bonus de puissance selon certaines conditions.

Octroie un bonus de puissance selon certaines conditions.

Définition à la ligne 286 du fichier unit.c.

**5.12.2.18 void recover ( )**

Se réveille tour par tour.

Se réveille tour par tour.

Définition à la ligne 503 du fichier unit.c.

**5.12.2.19 void setDirection ( vector source, int dir )**

Définis la direction d'une unité

Définis la direction d'une unité

**Paramètres**

<i>source</i>	Unité à tourner
<i>dir</i>	Direction dans laquelle tourner

Définition à la ligne 377 du fichier unit.c.

**5.12.2.20 void sleep ( vector pos )**

Endors une unité



Endors une unité

## Paramètres

<i>pos</i>	Position de l'unité
------------	---------------------

Définition à la ligne 480 du fichier unit.c.

5.12.2.21 void unitInit ( short *noP*, vector *coordUnit* )

Initialise une unité

Initialise une unité

## Paramètres

<i>noP</i>	Numéro du joueur
<i>coordUnit</i>	Coordonnées de l'unité

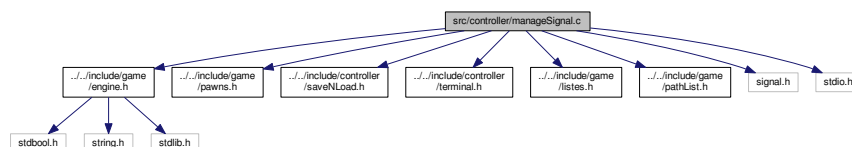
Définition à la ligne 24 du fichier unit.c.

## 5.13 Référence du fichier src/controller/manageSignal.c

Gestion des signaux.

```
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/controller/saveNLoad.h"
#include "../include/controller/terminal.h"
#include "../include/game/lists.h"
#include "../include/game/pathList.h"
#include <signal.h>
#include <stdio.h>
```

Graphe des dépendances par inclusion de manageSignal.c :



## Fonctions

- void **freeAll** ()  
*Libère tout ce qui reste encore en mémoire.*
- void **interrupt** (int signal)  
*A l'interruption du programme libère la mémoire et sauvegarde du jeu.*
- void **timeDown** (int signal)  
*Petit message sympa lors du temps écoulé*
- void **terminator** (int signal)  
*Libération de la mémoire lors de la fin du programme.*
- void **checkSignal** ()  
*Vérifie le type de signal envoyé au terminal.*

## 5.13.1 Description détaillée

Gestion des signaux.

**Auteur**

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

**Version**

v1.00

**Date**

18/12/2015

**5.13.2 Documentation des fonctions****5.13.2.1 void checkSignal ( )**

Vérifie le type de signal envoyé au terminal.

Vérifie les signaux du programme.

Définition à la ligne 77 du fichier manageSignal.c.

**5.13.2.2 void freeAll ( )**

Libère tout ce qui reste encore en mémoire.

Libère toute la mémoire.

Définition à la ligne 21 du fichier manageSignal.c.

**5.13.2.3 void interrupt ( int *signal* )**

A l'interruption du programme libère la mémoire et sauvegarde du jeu.

**Paramètres**

<i>signal</i>	Signal d'interruption
---------------	-----------------------

Définition à la ligne 35 du fichier manageSignal.c.

**5.13.2.4 void terminator ( int *signal* )**

Libération de la mémoire lors de la fin du programme.

**Paramètres**

<i>signal</i>	Signal d'interruption
---------------	-----------------------

Définition à la ligne 68 du fichier manageSignal.c.

**5.13.2.5 void timeDown ( int *signal* )**

Petit message sympa lors du temps écoulé

**Paramètres**

<i>signal</i>	Signal d'interruption
---------------	-----------------------

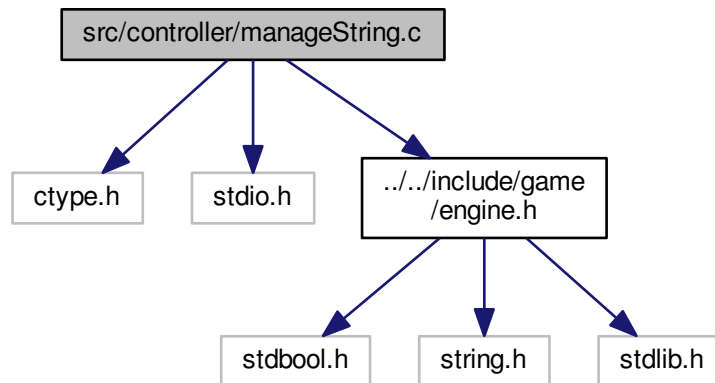
Définition à la ligne 58 du fichier manageSignal.c.

## 5.14 Référence du fichier src/controller/manageString.c

Gestions des chaînes de caractères.

```
#include <ctype.h>
#include <stdio.h>
#include "../include/game/engine.h"
```

Graphe des dépendances par inclusion de manageString.c :



### Fonctions

- void **getCoordS** (char coordString[ ], **vector** \*coordUnit)  
*Récupère les coordonnées d'une chaîne de caractère sous forme de vecteur x et y.*
- char \* **get2Char** (char name[ ])  
*Récupère 2 caractères du nom de l'unité*
- bool **isOutGrid** (char coordString[ ])  
*Vérifie que les coordonnées sont dans la grille.*
- bool **correctCoord** (char coordString[ ])  
*Sélectionne une coordonnée et vérifie son format.*
- bool **rightSide** (char \*coordString)  
*Vérifie que l'unité est du bon côté*
- char \* **getNameUnit** (**unitName** name)  
*Récupère le nom de l'unité à partir de la liste énumérée.*
- char \* **getNameEffect** (**unitEffect** effect)  
*Récupère le nom de l'effet à partir de la liste énumérée.*
- char \* **getDirectionUnit** (**cardinal** direct)  
*Retourne la direction.*
- void **printNameUnit** (**unitName** unit)  
*Affiche le nom de l'unité*
- void **clearBuffer** ()  
*Vide le tampon mémoire.*
- int **readS** (char \*string)  
*Lit la chaîne de caractère passée en paramètre et vérifie qu'elle ne soit pas trop longue.*
- long **readLong** ()  
*Lit un long de manière sécurisée.*
- double **readDouble** ()  
*Lit un double de manière sécurisée.*

### 5.14.1 Description détaillée

Gestions des chaînes de caractères.

**Auteur**

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

**Version**

v1.00

**Date**

18/12/2015

**5.14.2 Documentation des fonctions****5.14.2.1 void clearBuffer ( )**

Vide le tampon mémoire.

Vide la mémoire tampon.

Définition à la ligne 253 du fichier manageString.c.

**5.14.2.2 bool correctCoord ( char coordString[ ] )**

Sélectionne une coordonnée et vérifie son format.

**Paramètres**

<i>coordString</i>	Chaîne de caractère à vérifier
--------------------	--------------------------------

**Renvoie**

Vraie si les coordonnées saisies sont correctes

Définition à la ligne 138 du fichier manageString.c.

**5.14.2.3 char\* get2Char ( char name[ ] )**

Récupère 2 caractères du nom de l'unité

**Paramètres**

<i>name</i>	Nom de l'unité
-------------	----------------

**Renvoie**

Retourne 2 caractères liés au nom de l'unité

Définition à la ligne 47 du fichier manageString.c.

**5.14.2.4 void getCoordS ( char coordString[ ], vector \* coordUnit )**

Récupère les coordonnées d'une chaîne de caractère sous forme de vecteur x et y.

Récupère les coordonnées d'une chaîne de caractères.

**Paramètres**

<i>coordString</i>	Coordonnées saisie par l'utilisateur
<i>coordUnit</i>	Coordonnées de l'unité récupérées de la saisie utilisateur

Définition à la ligne 19 du fichier manageString.c.

#### 5.14.2.5 `char* getDirectionUnit ( cardinal direct )`

Retourne la direction.

Récupère le nom de la direction.

##### Paramètres

<i>direct</i>	Direction de l'unité
---------------	----------------------

##### Renvoie

Retourne La direction sous forme de chaîne

Définition à la ligne 235 du fichier manageString.c.

#### 5.14.2.6 `char* getNameEffect ( unitEffect effect )`

Récupère le nom de l'effet à partir de la liste énumérée.

Récupère le nom de l'effet.

##### Paramètres

<i>effect</i>	Effet provenant de la liste énumérée
---------------	--------------------------------------

##### Renvoie

Nom de l'effet sous forme de chaîne de caractère

Définition à la ligne 220 du fichier manageString.c.

#### 5.14.2.7 `char* getNameUnit ( unitName name )`

Récupère le nom de l'unité à partir de la liste énumérée.

Récupère le nom de l'unité

##### Paramètres

<i>name</i>	Nom de l'unité provenant de la liste énumérée
-------------	---

##### Renvoie

Nom de l'unité sous forme de chaîne

Définition à la ligne 189 du fichier manageString.c.

#### 5.14.2.8 `bool isOutGrid ( char coordString[ ] )`

Vérifie que les coordonnées sont dans la grille.

## Paramètres

<i>coordString</i>	Coordonnées sous forme de chaîne
--------------------	----------------------------------

## Renvoie

Retourne false si pas en dehors

Définition à la ligne 114 du fichier manageString.c.

## 5.14.2.9 void printNameUnit ( unitName unit )

Affiche le nom de l'unité

## Paramètres

<i>unit</i>	Nom de l'unité provenant de la liste énumérée
-------------	---

Définition à la ligne 246 du fichier manageString.c.

## 5.14.2.10 double readDouble ( )

Lit un double de manière sécurisée.

Lecture sécurisée d'un double.

## Renvoie

Retourne le double saisie ou 0.0 en cas d'erreur

Définition à la ligne 305 du fichier manageString.c.

## 5.14.2.11 long readLong ( )

Lit un long de manière sécurisée.

Lecture sécurisée d'un long.

## Renvoie

Retourne le long saisie ou 0 en cas d'erreur

Définition à la ligne 291 du fichier manageString.c.

## 5.14.2.12 int readS ( char \* string )

Lit la chaîne de caractère passée en paramètre et vérifie qu'elle ne soit pas trop longue.

Lecture sécurisée d'une chaîne.

## Paramètres

<i>string</i>	Chaîne de caractère à vérifier
---------------	--------------------------------

## Renvoie

Retourne 1 si chaîne correcte

Définition à la ligne 265 du fichier manageString.c.

#### 5.14.2.13 `bool rightSide ( char * coordString )`

Vérifie que l'unité est du bon côté

Vérifie que l'unité est dans le bon camp.



## Paramètres

<code>coordString</code>	Coordonnées sous forme de chaîne
--------------------------	----------------------------------

## Renvoi

Retourne vrai si du bon côté

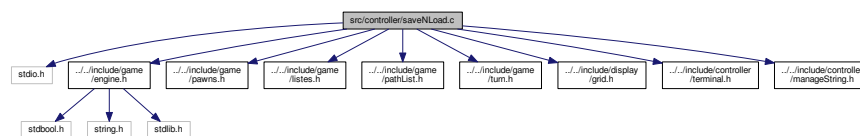
Définition à la ligne 165 du fichier manageString.c.

## 5.15 Référence du fichier src/controller/saveNLoad.c

Gestion de la sauvegarde et du chargement.

```
#include <stdio.h>
#include "../..../include/game/engine.h"
#include "../..../include/game/pawns.h"
#include "../..../include/game/listes.h"
#include "../..../include/game/pathList.h"
#include "../..../include/game/turn.h"
#include "../..../include/display/grid.h"
#include "../..../include/controller/terminal.h"
#include "../..../include/controller/manageString.h"
```

Graphe des dépendances par inclusion de saveNLoad.c :



## Fonctions

- char `getCharKey` (char \*dynamicKey, int \*pos)  
*Récupère un caractère de la clé*
- char \* `getKey` (unitName name)  
*Récupère une clé de décryptage.*
- void `crypt` (unit \*unitSaved)  
*Crypte les statistiques de l'unité pour la sauvegarde.*
- void `decrypt` (unit \*unitLoaded)  
*Décrypte les statistiques de l'unité pour le chargement.*
- bool `checkDecrypt` (unit \*unitLoaded)  
*Vérifie que le décryptage s'est bien passé*
- void `save` ()  
*Sauvegarde la partie avec l'état actuel de la grille et les données liées au joueur en cours.*
- void `load` ()  
*Charge la partie précédemment enregistré*

## Variables

- char `key` [] = "SPIBCTBEC"  
*Clé statique à utiliser pour le cryptage.*

## 5.15.1 Description détaillée

Gestion de la sauvegarde et du chargement.

**Auteur**

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

**Version**

v1.00

**Date**

18/12/2015

**5.15.2 Documentation des fonctions****5.15.2.1 bool checkDecrypt ( unit \* unitLoaded )**

Vérifie que le décryptage s'est bien passé

**Paramètres**

<i>unitLoaded</i>	Unité chargée
-------------------	---------------

**Renvoie**

Retourne vrai si données correctes sinon faux

Définition à la ligne 133 du fichier saveNLoad.c.

**5.15.2.2 void crypt ( unit \* unitSaved )**

Crypte les statistiques de l'unité pour la sauvegarde.

**Paramètres**

<i>unitSaved</i>	Unité sauvegardée
------------------	-------------------

Définition à la ligne 70 du fichier saveNLoad.c.

**5.15.2.3 void decrypt ( unit \* unitLoaded )**

Décrypte les statistiques de l'unité pour le chargement.

**Paramètres**

<i>unitLoaded</i>	Unité chargée
-------------------	---------------

Définition à la ligne 101 du fichier saveNLoad.c.

**5.15.2.4 char getCharKey ( char \* dynamicKey, int \* pos )**

Récupère un caractère de la clé

**Paramètres**

<i>dynamicKey</i>	Clé dynamique
<i>pos</i>	Position dans la clé

**Renvoie**

Retourne un caractère de la clé

Définition à la ligne 27 du fichier saveNLoad.c.

#### 5.15.2.5 char\* getKey ( unitName *name* )

Récupère une clé de décryptage.

## Paramètres

<i>name</i>	Nom de l'unité
-------------	----------------

## Renvoie

Retourne une clé

Définition à la ligne 39 du fichier saveNLoad.c.

## 5.15.2.6 void load ( )

Charge la partie précédemment enregistré

Charge une partie.

Définition à la ligne 201 du fichier saveNLoad.c.

## 5.15.2.7 void save ( )

Sauvegarde la partie avec l'état actuel de la grille et les données liées au joueur en cours.

Sauvegarde la partie.

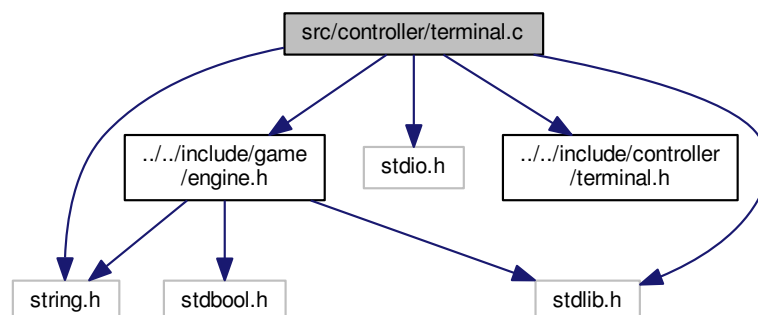
Définition à la ligne 158 du fichier saveNLoad.c.

## 5.16 Référence du fichier src/controller/terminal.c

Gestion du terminal.

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "../../include/controller/terminal.h"
#include "../../include/game/engine.h"
```

Graphe des dépendances par inclusion de terminal.c :



## Fonctions

- char \* `getColor` (int `color`, char type[])  
Récupère le code correspondant à la couleur.

- void `color` (int `color`, char `type`[ ])   
 *Change la couleur de l'écran ou de la police.*
- void `reinitColor` ()   
 *Réinitialise la couleur de l'écran.*
- void `fontColor` (int `color`)   
 *Change la couleur de la police.*
- void `clearScreen` ()   
 *Efface l'écran.*

### 5.16.1 Description détaillée

Gestion du terminal.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.16.2 Documentation des fonctions

#### 5.16.2.1 void `color` ( int `color`, char `type`[ ] )

Change la couleur de l'écran ou de la police.

Met en couleurs le texte ou l'écran.

##### Paramètres

<code>color</code>	Couleur à utiliser
<code>type</code>	Texte à changer de couleur ou arrière plan

Définition à la ligne 51 du fichier terminal.c.

#### 5.16.2.2 void `fontColor` ( int `color` )

Change la couleur de la police.

Met en couleurs la police.

##### Paramètres

<code>color</code>	Couleur à utiliser
--------------------	--------------------

Définition à la ligne 76 du fichier terminal.c.

#### 5.16.2.3 char\* `getColor` ( int `color`, char `type`[ ] )

Récupère le code correspondant à la couleur.

##### Paramètres

<i>color</i>	Couleur demandée
<i>type</i>	Type d'objet à modifier

#### Renvoie

Code correspondant, écran blanc si code incorrect

Définition à la ligne 21 du fichier terminal.c.

#### 5.16.2.4 void reinitColor ( )

Réinitialise la couleur de l'écran.

Réinitialise les couleurs.

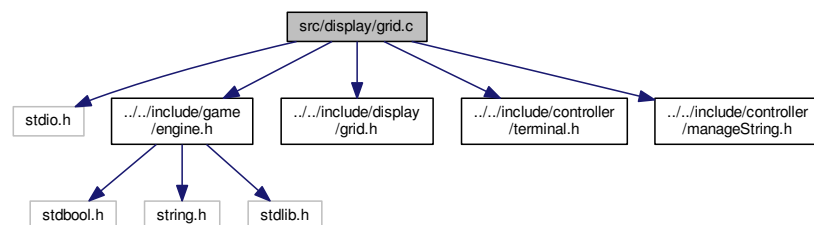
Définition à la ligne 67 du fichier terminal.c.

## 5.17 Référence du fichier src/display/grid.c

Gestion de la grille.

```
#include <stdio.h>
#include "../..../include/game/engine.h"
#include "../..../include/display/grid.h"
#include "../..../include/controller/terminal.h"
#include "../..../include/controller/manageString.h"
```

Graphe des dépendances par inclusion de grid.c :



### Fonctions

- void **borderRight** (short row)  
*Affiche une bordure sur le côté droit sur les lignes vides et utiles.*
- void **borderHoriz** ()  
*Affiche une bordure horizontale.*
- void **dispX** ()  
*Affiche les coordonnées verticales.*
- void **gridDisp** ()  
*Affiche la grille avec les coordonnées.*

#### 5.17.1 Description détaillée

Gestion de la grille.

**Auteur**

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

**Version**

v1.00

**Date**

18/12/2015

**5.17.2 Documentation des fonctions****5.17.2.1 void borderRight ( short row )**

Affiche une bordure sur le côté droit sur les lignes vides et utiles.

**Paramètres**

<i>row</i>	Ligne actuelle
------------	----------------

Définition à la ligne 19 du fichier grid.c.

**5.17.2.2 void gridDisp ( )**

Affiche la grille avec les coordonnées.

Affiche la grille.

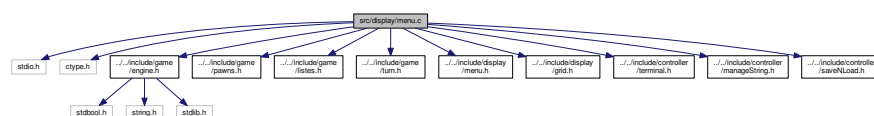
Définition à la ligne 78 du fichier grid.c.

**5.18 Référence du fichier src/display/menu.c**

Gestion des menus.

```
#include <stdio.h>
#include <ctype.h>
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/game/listes.h"
#include "../include/game/turn.h"
#include "../include/display/menu.h"
#include "../include/display/grid.h"
#include "../include/controller/terminal.h"
#include "../include/controller/manageString.h"
#include "../include/controller/saveNLoad.h"
```

Graphe des dépendances par inclusion de menu.c :

**Fonctions**— void [mainMenu](#) ()

- *Menu principal du jeu.*  
void `mainHelp` ()
- *Menu d'aide principal.*  
void `helpUnit` ()
- *Menu d'aide spécifique aux unités.*  
void `dispDirection` ()
- *Afficher les directions.*  
void `gameMenu` ()
- *Menu du joueur lors de la partie.*  
void `unitMenu` (int choice)
- *Menu de sélection de l'unité*  
void `unitList` ()
- *Affiche la liste des unités inclus dans le jeu.*

### 5.18.1 Description détaillée

Gestion des menus.

Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

Version

v1.00

Date

18/12/2015

### 5.18.2 Documentation des fonctions

#### 5.18.2.1 void `dispDirection` ( )

Afficher les directions.

Affiche la liste des directions.

Définition à la ligne 458 du fichier menu.c.

#### 5.18.2.2 void `gameMenu` ( )

Menu du joueur lors de la partie.

Menu de jeu.

Définition à la ligne 470 du fichier menu.c.

#### 5.18.2.3 void `helpUnit` ( )

Menu d'aide spécifique aux unités.

Menu d'aide des unités.

Définition à la ligne 222 du fichier menu.c.

#### 5.18.2.4 void `mainMenu` ( )

Menu principal du jeu.

Menu principal.

Définition à la ligne 24 du fichier menu.c.



## 5.18.2.5 void unitList ( )

Affiche la liste des unités inclus dans le jeu.

Listes des unités du jeu.

Définition à la ligne 593 du fichier menu.c.

## 5.18.2.6 void unitMenu ( int choice )

Menu de sélection de l'unité

Paramètres

choice	Choix de l'action pour l'unité
--------	--------------------------------

Définition à la ligne 517 du fichier menu.c.

## 5.19 Référence du fichier src/game/engine.c

Moteur de jeu.

```
#include <stdio.h>
#include <time.h>
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/game/pathList.h"
#include "../include/game/listes.h"
#include "../include/game/turn.h"
#include "../include/display/grid.h"
#include "../include/display/menu.h"
#include "../include/controller/terminal.h"
#include "../include/controller/manageString.h"
#include "../include/controller/manageSignal.h"
#include "../include/units/unit.h"
```

Graphe des dépendances par inclusion de engine.c :



## Fonctions

- bool possiblePath (vector coordUnit)  
*Trouve un chemin vers une quelconque position pour l'unité*
- bool pathFind (vector coordUnit, vector coordTarget)  
*Trouve un chemin vers la position désirée.*
- void lineOfSight (vector coordSource, vector coordTarget)  
*Ligne de vue de l'archer.*
- void movable (int colorDisp)  
*Fait la liste des unités déplaçables.*
- void attackable (int colorDisp)  
*Fait la liste des unités pouvant attaquer.*
- void tileWalkable (vector coordUnit, int colorDisp)  
*Fait la liste des cases pouvant être atteintes par l'unité*
- bool isSurrounded (vector currentUnit)  
*Permet de savoir si l'unité courante est entourée.*
- void setTarget (unitName name, vector coordUnit, int colorDisp)

- *Initialise les cibles du pion.*  
void `specialBoons` (`vector` coordSource, `vector` coordTarget)
- *Déclenche une capacité spéciale selon l'unité*  
void `launchAttack` (`vector` coordSource, `vector` coordTarget)
- *Lance une attaque selon l'unité*  
bool `selectUnit` (`vector` \*coordUnit)  
*Sélectionne une unité*
- void `gridInit` ()  
*Initialise la grille.*
- bool `tooMuchUnit` (int unitSelected, int limitUnits[])  
*Vérifie que l'unité sélectionnée n'est pas en trop grand nombre sur le plateau.*
- void `updateLimits` (int unitSelected, int limitUnits[], `vector` coordUnit)  
*Met à jour les limites d'unités.*
- void `askUnit` (int \*unitSelected, int limitUnits[])  
*Demande de choisir une unité*
- void `askCoord` (char coordString[])  
*Demande les coordonnées de l'unité à placer.*
- void `playerAddUnit` (int limitUnits[], int \*nbUnit)  
*Placement des unités par le joueur.*
- void `playerInit` ()  
*Initialise la liste des unités du joueur.*
- bool `endGame` ()  
*Fin de la partie.*
- void `startGame` ()  
*Début la partie.*
- void `gameInit` ()  
*Initialise la partie.*

## Variables

- `unit` `grid` [N][N]  
*Grille d'unités.*
- int `noPlayer` = `FIRST_PLAYER`  
*Joueur en cours.*

### 5.19.1 Description détaillée

Moteur de jeu.

Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

Version

v1.00

Date

18/12/2015

### 5.19.2 Documentation des fonctions

#### 5.19.2.1 void askCoord ( char coordString[] )

Demande les coordonnées de l'unité à placer.

## Paramètres

<i>coordString</i>	Coordonnées sous forme de chaîne
--------------------	----------------------------------

Définition à la ligne 579 du fichier engine.c.

5.19.2.2 void askUnit ( int \* *unitSelected*, int *limitUnits*[ ] )

Demande de choisir une unité

## Paramètres

<i>unitSelected</i>	Unité sélectionnée
<i>limitUnits</i>	Nombre limite pour chaque unité

Définition à la ligne 551 du fichier engine.c.

5.19.2.3 void attackable ( int *colorDisp* )

Fait la liste des unités pouvant attaquer.

## Paramètres

<i>colorDisp</i>	Couleur d'affichage
------------------	---------------------

Définition à la ligne 164 du fichier engine.c.

## 5.19.2.4 void gamelnit ( )

Initialise la partie.

Initialise le jeu.

Définition à la ligne 750 du fichier engine.c.

5.19.2.5 bool isSurrounded ( vector *currentUnit* )

Permet de savoir si l'unité courante est entourée.

Vérifie si une unité est entourée.

## Paramètres

<i>currentUnit</i>	Unité courante
--------------------	----------------

## Renvoie

Retourne vrai si unité entourée

Définition à la ligne 215 du fichier engine.c.

5.19.2.6 void launchAttack ( vector *coordSource*, vector *coordTarget* )

Lance une attaque selon l'unité

Lance une attaque.

**Paramètres**

<i>coordSource</i>	Nom de l'unité source
<i>coordTarget</i>	Coordonnées de la cible

Définition à la ligne 336 du fichier engine.c.

**5.19.2.7 void lineOfSight ( vector *coordSource*, vector *coordTarget* )**

Ligne de vue de l'archer.

**Paramètres**

<i>coordSource</i>	Coordonnées de l'unité attaquante
<i>coordTarget</i>	Coordonnées de la cible

Définition à la ligne 117 du fichier engine.c.

**5.19.2.8 void movable ( int *colorDisp* )**

Fait la liste des unités déplaçables.

**Paramètres**

<i>colorDisp</i>	Couleur d'affichage
------------------	---------------------

Définition à la ligne 141 du fichier engine.c.

**5.19.2.9 bool pathFind ( vector *coordUnit*, vector *coordTarget* )**

Trouve un chemin vers la position désirée.

Trouve une chemin vers la destination.

**Paramètres**

<i>coordUnit</i>	Coordonnées de l'unité
<i>coordTarget</i>	Coordonnées de la cible

**Renvoie**

Retourne vrai si chemin trouvé

Définition à la ligne 60 du fichier engine.c.

**5.19.2.10 void playerAddUnit ( int *limitUnits*[], int \* *nbUnit* )**

Placement des unités par le joueur.

**Paramètres**

<i>limitUnits</i>	Limites d'unités
<i>nbUnit</i>	Nombre d'unités restantes à placer

Définition à la ligne 608 du fichier engine.c.

**5.19.2.11 bool possiblePath ( vector *coordUnit* )**

Trouve un chemin vers une quelconque position pour l'unité

Vérifie qu'un chemin est possible.

## Paramètres

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

## Renvoie

Retourne Vrai si chemin possible vers une quelconque position

Définition à la ligne 31 du fichier engine.c.

5.19.2.12 **bool selectUnit ( vector \* coordUnit )**

Sélectionne une unité

## Paramètres

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

## Renvoie

Retourne vrai si unité bien sélectionnée

Définition à la ligne 416 du fichier engine.c.

5.19.2.13 **void setTarget ( unitName name, vector coordUnit, int colorDisp )**

Initialise les cibles du pion.

Définis les cibles.

## Paramètres

<i>name</i>	Nom du pion
<i>coordUnit</i>	Coordonnées de l'unité
<i>colorDisp</i>	Couleur d'affichage

Définition à la ligne 252 du fichier engine.c.

5.19.2.14 **void specialBoons ( vector coordSource, vector coordTarget )**

Déclenche une capacité spéciale selon l'unité

## Paramètres

<i>coordSource</i>	Coordonnées de l'unité source (déclenchant une capacité)
<i>coordTarget</i>	Coordonnées de l'unité affectée par la capacité spéciale

Définition à la ligne 299 du fichier engine.c.

5.19.2.15 **void tileWalkable ( vector coordUnit, int colorDisp )**

Fait la liste des cases pouvant être atteintes par l'unité

Fait la liste des cases atteignables par l'unité

## Paramètres

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

<i>colorDisp</i>	Couleur d'affichage
------------------	---------------------

Définition à la ligne 188 du fichier engine.c.

#### 5.19.2.16 `bool tooMuchUnit ( int unitSelected, int limitUnits[ ] )`

Vérifie que l'unité sélectionnée n'est pas en trop grand nombre sur le plateau.

Paramètres

<i>unitSelected</i>	Unité sélectionnée
<i>limitUnits</i>	Tableau des limites pour chaque unité

Renvoie

Retourne vrai si unité en surnombre

Définition à la ligne 468 du fichier engine.c.

#### 5.19.2.17 `void updateLimits ( int unitSelected, int limitUnits[ ], vector coordUnit )`

Met à jour les limites d'unités.

Paramètres

<i>unitSelected</i>	Unité sélectionnée
<i>limitUnits</i>	Limites liées aux unités sur le plateau
<i>coordUnit</i>	Coordonnées de l'unité

Définition à la ligne 515 du fichier engine.c.

### 5.19.3 Documentation des variables

#### 5.19.3.1 `unit grid[N][N]`

Grille d'unités.

Représentation d'une grille d'unité globale.

Définition à la ligne 23 du fichier engine.c.

#### 5.19.3.2 `int noPlayer = FIRST_PLAYER`

Joueur en cours.

Représentation du joueur.

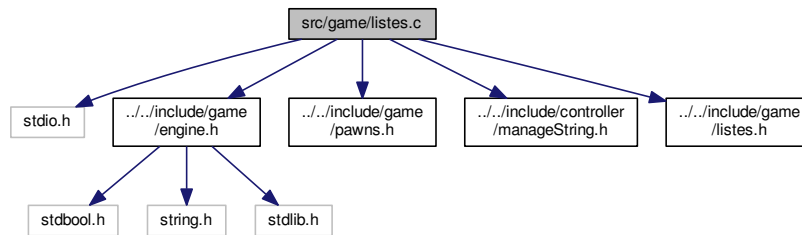
Définition à la ligne 24 du fichier engine.c.

## 5.20 Référence du fichier src/game/listes.c

Listes de vecteurs.

```
#include <stdio.h>
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/controller/manageString.h"
#include "../include/game/listes.h"
```

Graphe des dépendances par inclusion de lists.c :



## Classes

- struct `element`  
*Représente un élément de la liste.*

## Définitions de type

- typedef struct `element` `t_element`  
*Définis un élément.*

## Fonctions

- void `init_liste` (int n)  
*Initialise une liste.*
- void `initLists` ()  
*Initialise les listes.*
- int `liste_vide` (int n)  
*Vérifie si une liste est vide.*
- int `hors_liste` (int n)  
*Vérifie si l'élément courant est hors de la liste.*
- void `en_tete` (int n)  
*Se met en tête de la liste.*
- void `en_queue` (int n)  
*Se met en queue de la liste.*
- void `precedent` (int n)  
*Se positionne sur l'élément précédent.*
- void `suivant` (int n)  
*Se positionne sur l'élément suivant.*
- void `valeur_elt` (int n, `vector` \*v)  
*Récupère la valeur de l'élément.*
- void `modif_elt` (int n, `vector` v)  
*Modifie la valeur de l'élément.*
- void `oter_elt` (int n)  
*Supprime l'élément.*
- void `ajout_droit` (int n, `vector` v)  
*Ajoute à droite l'élément.*
- void `ajout_gauche` (int n, `vector` v)  
*Ajoute à gauche l'élément.*
- void `dumpList` (short nbList)  
*Vide une liste.*
- void `dumpAllLists` ()  
*Vide toutes les listes.*
- void `addUnit` (`vector` coordUnit)

- *Ajoute une unité dans la liste des unités du joueur.*  
void `addTarget` (`unitName` name, `vector` coordUnit)
- *Ajoute une cible pour une unité*  
void `printList` (short numList)
- *Affiche la liste désirée.*  
void `destroyUnit` (`vector` coordUnit)
- *Détruit une unité dans la liste.*  
int `countUnits` ()
- *Compte le nombre d'unité*  
bool `searchTarget` (int numList, `vector` coordTarget)
- *Cherche une cible dans la liste.*

## Variables

- `t_element` \* `drapeau` [NB\_UNITS]  
*Tableaux de drapeau.*
- `t_element` \* `ec` [NB\_UNITS]  
*Tableau d'élément courant.*
- int `targetList` = NB\_PLAYERS + 1  
*Numéro de la liste des cibles.*

### 5.20.1 Description détaillée

Listes de vecteurs.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.20.2 Documentation des fonctions

#### 5.20.2.1 void addTarget ( unitName name, vector coordUnit )

Ajoute une cible pour une unité

Ajoute une cible.

#### Paramètres

<i>name</i>	Nom de l'unité
<i>coordUnit</i>	Coordonnées de l'unité

Définition à la ligne 211 du fichier listes.c.

#### 5.20.2.2 void addUnit ( vector coordUnit )

Ajoute une unité dans la liste des unités du joueur.

Ajoute une unité



## Paramètres

<i>coordUnit</i>	Coordonnées de l'unité
------------------	------------------------

Définition à la ligne 179 du fichier listes.c.

## 5.20.2.3 int countUnits ( )

Compte le nombre d'unité

Compte le nombre d'unités.

## Renvoie

Retourne le nombre d'unité

Définition à la ligne 280 du fichier listes.c.

5.20.2.4 void destroyUnit ( vector *coordUnit* )

Détruit une unité dans la liste.

Détruit une unité

## Paramètres

<i>coordUnit</i>	Coordonnées de l'unité à détruire
------------------	-----------------------------------

Définition à la ligne 260 du fichier listes.c.

## 5.20.2.5 void dumpAllLists ( )

Vide toutes les listes.

Vide les listes.

Définition à la ligne 167 du fichier listes.c.

5.20.2.6 void dumpList ( short *nbList* )

Vide une liste.

## Paramètres

<i>nbList</i>	Numéro de la liste à vider
---------------	----------------------------

Définition à la ligne 155 du fichier listes.c.

5.20.2.7 void printList ( short *numList* )

Affiche la liste désirée.

Affiche la liste.

## Paramètres

<i>numList</i>	Numéro de liste
----------------	-----------------

Définition à la ligne 222 du fichier listes.c.

#### 5.20.2.8 bool searchTarget ( int numList, vector coordTarget )

Cherche une cible dans la liste.

Cherche une cible.

##### Paramètres

<i>numList</i>	Numéro de la liste
<i>coordTarget</i>	Coordonnées de la cible

##### Renvoie

Retourne vrai si cible trouvée

Définition à la ligne 301 du fichier listes.c.

### 5.20.3 Documentation des variables

#### 5.20.3.1 int targetList = NB\_PLAYERS + 1

Numéro de la liste des cibles.

Cibles potentielles.

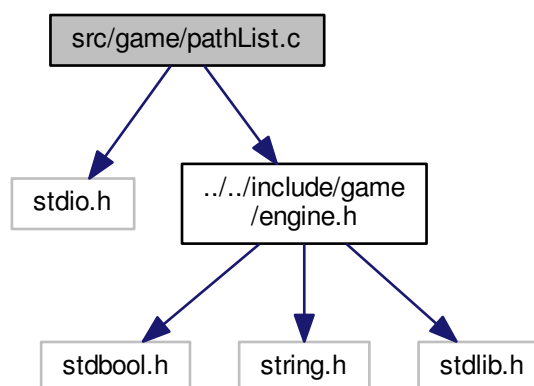
Définition à la ligne 28 du fichier listes.c.

## 5.21 Référence du fichier src/game/pathList.c

Listes de cases sur un chemin définis.

```
#include <stdio.h>
#include "../..../include/game/engine.h"
```

Graphe des dépendances par inclusion de pathList.c :



## Classes

- struct [elem](#)  
Représente une case de la grille.

## Définitions de type

- typedef struct `elem t_tile`  
*Définis la case.*

## Fonctions

- void `initPath` (int n)  
*Initialise le chemin.*
- void `initPaths` ()  
*Initialise les chemins.*
- int `emptyPath` (int n)  
*Vérifie si un chemin est vide.*
- int `outPath` (int n)  
*Vérifie si l'élément courant est hors liste.*
- void `pathHead` (int n)  
*Se met en tête du chemin.*
- void `pathTail` (int n)  
*Se met en queue du chemin.*
- void `previous` (int n)  
*Se positionne sur l'élément précédent.*
- void `next` (int n)  
*Se positionne sur l'élément suivant.*
- void `getTile` (int n, `vector *v`, int \*F)  
*Récupère une case du chemin.*
- `vector` `getCurrentNode` (int n)  
*Récupère le noeud ayant le plus petit F.*
- void `setTile` (int n, `vector v`, int F)  
*Modifie les infos d'une case dans la liste.*
- void `eraseTile` (int n)  
*Efface une case de la grille.*
- void `toRightPath` (int n, `vector v`, int F)  
*Ajoute une case à droite.*
- void `dumpPath` (short nbList)  
*Vide une liste.*
- void `dumpAllPaths` ()  
*Vide toutes les listes.*
- bool `searchTile` (int n, `vector coordTile`)  
*Cherche si un vecteur est présent dans la liste fermée ou ouverte.*
- void `addCloseList` (`vector current`, int F)  
*Ajoute à la liste fermée.*
- void `addOpenList` (`vector current`, int F)  
*Ajoute à la liste ouverte.*

## Variables

- `t_tile * flag` [2]  
*Tableau de drapeaux.*
- `t_tile * elemC` [2]  
*Tableau d'éléments courants.*

### 5.21.1 Description détaillée

Listes de cases sur un chemin définis.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

Date

18/12/2015

## 5.21.2 Documentation des fonctions

### 5.21.2.1 void addCloseList ( vector *current*, int *F* )

Ajoute à la liste fermée.

Ajoute la case à la liste fermée.

Paramètres

<i>current</i>	Destination
<i>F</i>	Poids de la case

Définition à la ligne 262 du fichier pathList.c.

### 5.21.2.2 void addOpenList ( vector *current*, int *F* )

Ajoute à la liste ouverte.

Ajoute la case à la liste ouverte.

Paramètres

<i>current</i>	Destination
<i>F</i>	Poids de la case

Définition à la ligne 285 du fichier pathList.c.

### 5.21.2.3 void dumpAllPaths ( )

Vide toutes les listes.

Vide les chemins.

Définition à la ligne 226 du fichier pathList.c.

### 5.21.2.4 void dumpPath ( short *nbList* )

Vide une liste.

Vide le chemin.

Paramètres

<i>nbList</i>	Numéro de la liste à vider
---------------	----------------------------

Définition à la ligne 214 du fichier pathList.c.

### 5.21.2.5 int emptyPath ( int *n* )

Vérifie si un chemin est vide.

Vérifie si chemin vide.

## Paramètres

$n$	Chemin
-----	--------

## Renvoie

Retourne le drapeau si pas vide

Définition à la ligne 54 du fichier pathList.c.

5.21.2.6 void eraseTile ( int  $n$  )

Efface une case de la grille.

Efface une case.

## Paramètres

$n$	Liste
-----	-------

Définition à la ligne 170 du fichier pathList.c.

5.21.2.7 vector getCurrentNode ( int  $n$  )

Récupère le noeud ayant le plus petit F.

Récupère la case ayant le plus petit poids.

## Paramètres

$n$	Liste dans laquelle chercher
-----	------------------------------

Définition à la ligne 130 du fichier pathList.c.

5.21.2.8 void getTile ( int  $n$ , vector \*  $v$ , int \*  $F$  )

Récupère une case du chemin.

Récupère la valeur d'une case.

## Paramètres

$n$	Chemin
$v$	Position de la case
$F$	Poids de la case

Définition à la ligne 117 du fichier pathList.c.

5.21.2.9 void initPath ( int  $n$  )

Initialise le chemin.

## Paramètres

$n$	Chemin
-----	--------

Définition à la ligne 31 du fichier pathList.c.

5.21.2.10 void next ( int  $n$  )

Se positionne sur l'élément suivant.

Se positionne sur l'élément suivant.

**Paramètres**

<i>n</i>	Chemin
----------	--------

Définition à la ligne 105 du fichier pathList.c.

**5.21.2.11 int outPath ( int *n* )**

Vérifie si l'élément courant est hors liste.

Vérifie si l'élément courant est en dehors de la liste.

**Paramètres**

<i>n</i>	Liste
----------	-------

**Renvoie**

Retourne l'élément courant si pas hors liste

Définition à la ligne 65 du fichier pathList.c.

**5.21.2.12 void pathHead ( int *n* )**

Se met en tête du chemin.

Se met en tête de la liste.

**Paramètres**

<i>n</i>	Chemin
----------	--------

Définition à la ligne 75 du fichier pathList.c.

**5.21.2.13 void pathTail ( int *n* )**

Se met en queue du chemin.

Se met en queue de la liste.

**Paramètres**

<i>n</i>	Chemin
----------	--------

Définition à la ligne 85 du fichier pathList.c.

**5.21.2.14 void previous ( int *n* )**

Se positionne sur l'élément précédent.

**Paramètres**

<i>n</i>	Chemin
----------	--------

Définition à la ligne 95 du fichier pathList.c.

**5.21.2.15 bool searchTile ( int *n*, vector *coordTile* )**

Cherche si un vecteur est présent dans la liste fermée ou ouverte.

Cherche une case.

## Paramètres

<i>n</i>	Liste
<i>coordTile</i>	Position de la case

Définition à la ligne 239 du fichier pathList.c.

5.21.2.16 void setTile ( int *n*, vector *v*, int *F* )

Modifie les infos d'une case dans la liste.

Modifie la valeur d'une case.

## Paramètres

<i>n</i>	Liste
<i>v</i>	Nouvelle position
<i>F</i>	Nouveau poids

Définition à la ligne 157 du fichier pathList.c.

5.21.2.17 void toRightPath ( int *n*, vector *v*, int *F* )

Ajoute une case à droite.

## Paramètres

<i>n</i>	Liste
<i>v</i>	Position à ajouter
<i>F</i>	Poids à ajouter

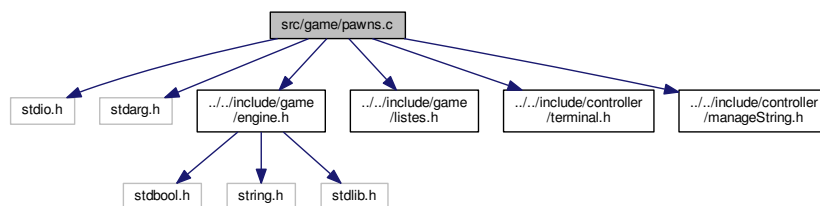
Définition à la ligne 191 du fichier pathList.c.

## 5.22 Référence du fichier src/game/pawns.c

Gestion des pions.

```
#include <stdio.h>
#include <stdarg.h>
#include "../..//include/game/engine.h"
#include "../..//include/game/listes.h"
#include "../..//include/controller/terminal.h"
#include "../..//include/controller/manageString.h"
```

Graphe des dépendances par inclusion de pawns.c :



## Fonctions

- void `initPawn` (`unit *pawn`)  
*Initialise un pion.*
- bool `checkPawn` (`unit *pawn`)  
*Vérifie que le pion est correct.*
- void `createPawn` (int nbParams, `unitName` name,...)  
*Crée un pion pour le jeu.*
- void `makePawns` ()  
*Crée tout les pions utiles au jeu.*

## Variables

- `unit * pawns` = NULL  
*Pions initialisés avec leur valeurs.*
- int `sizePawns` = 0  
*Nombre de pions.*

### 5.22.1 Description détaillée

Gestion des pions.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.22.2 Documentation des fonctions

#### 5.22.2.1 bool `checkPawn` ( `unit * pawn` )

Vérifie que le pion est correct.

##### Paramètres

<code>pawn</code>	Pion à vérifier
-------------------	-----------------

##### Renvoie

Retourne vrai si le pion est correct

Définition à la ligne 37 du fichier `pawns.c`.

#### 5.22.2.2 void `createPawn` ( int `nbParams`, `unitName` `name`, ... )

Crée un pion pour le jeu.

##### Paramètres

<code>nbParams</code>	Nombre de stats pour le pion
-----------------------	------------------------------



<i>name</i>	Nom du pion
-------------	-------------

Définition à la ligne 51 du fichier pawns.c.

#### 5.22.2.3 void initPawn ( unit \* pawn )

Initialise un pion.

Paramètres

<i>pawn</i>	Pion à initialiser
-------------	--------------------

Définition à la ligne 23 du fichier pawns.c.

#### 5.22.2.4 void makePawns ( )

Crée tout les pions utiles au jeu.

Crée les pions.

Définition à la ligne 109 du fichier pawns.c.

### 5.22.3 Documentation des variables

#### 5.22.3.1 unit\* pawns = NULL

Pions initialisés avec leur valeurs.

Tableaux des pions.

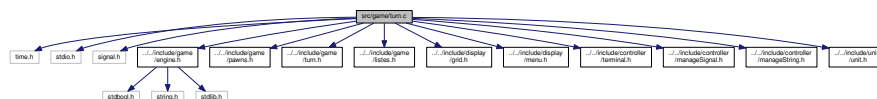
Définition à la ligne 16 du fichier pawns.c.

## 5.23 Référence du fichier src/game/turn.c

Gestion des tours.

```
#include <time.h>
#include <stdio.h>
#include <signal.h>
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/game/turn.h"
#include "../include/game/listes.h"
#include "../include/display/grid.h"
#include "../include/display/menu.h"
#include "../include/controller/terminal.h"
#include "../include/controller/manageSignal.h"
#include "../include/controller/manageString.h"
#include "../include/units/unit.h"
```

Graphe des dépendances par inclusion de turn.c :



## Fonctions

- int `startTurn` ()  
*Début le tour du joueur.*
- int `endTurn` (time\_t start, int totalTime)  
*Renvoie le temps restant avant la fin du tour.*
- void `changeDirection` ()  
*Change la direction de l'unité*
- void `playAttack` ()  
*Jouer une attaque.*
- void `playMove` ()  
*Jouer un mouvement.*
- void `passTurn` ()  
*Passer tour.*
- bool `hasPlay` ()  
*Définis si le joueur a joué*
- void `surrender` ()  
*Abandonne la partie.*
- void `setAction` (int timeLeft)  
*Définis si l'action est à prendre en compte.*
- void `playTurn` ()  
*Jouer un tour.*

## Variables

- int `hasMoved` = 0  
*Joueur a joué*
- int `hasAttacked` = 0  
*Joueur a attaqué*
- int `hasPassed` = 0  
*Joueur a passé son tour.*
- int `hasSurrender` = 0  
*Joueur a abandonné la partie.*

### 5.23.1 Description détaillée

Gestion des tours.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.23.2 Documentation des fonctions

#### 5.23.2.1 void changeDirection ( )

Change la direction de l'unité

Change la direction.

Définition à la ligne 57 du fichier turn.c.

#### 5.23.2.2 int endTurn ( time\_t start, int totalTime )

Renvoie le temps restant avant la fin du tour.

## Paramètres

<i>start</i>	Temps de début du tour
<i>totalTime</i>	Temps total du tour

Définition à la ligne 44 du fichier turn.c.

## 5.23.2.3 bool hasPlay ( )

Définis si le joueur a joué

Vérifie si une action a été effectuée.

## Renvoie

Retourne vrai si le joueur a joué sinon faux

Définition à la ligne 245 du fichier turn.c.

## 5.23.2.4 void passTurn ( )

Passer tour.

Passe le tour.

Définition à la ligne 235 du fichier turn.c.

## 5.23.2.5 void playAttack ( )

Jouer une attaque.

Joue une attaque.

Définition à la ligne 104 du fichier turn.c.

## 5.23.2.6 void playMove ( )

Jouer un mouvement.

Joue une mouvement.

Définition à la ligne 177 du fichier turn.c.

## 5.23.2.7 void playTurn ( )

Jouer un tour.

Joue un tour.

Définition à la ligne 272 du fichier turn.c.

5.23.2.8 void setAction ( int *timeLeft* )

Définis si l'action est à prendre en compte.

## Paramètres

<i>timeLeft</i>	Temps restant pour le tour
-----------------	----------------------------

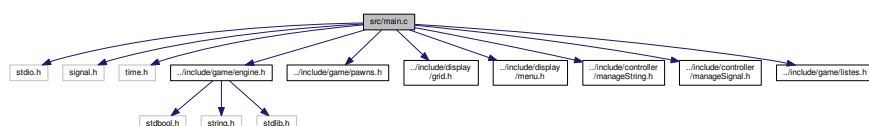
Définition à la ligne 261 du fichier turn.c.

## 5.24 Référence du fichier src/main.c

Programme principal.

```
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/display/grid.h"
#include "../include/display/menu.h"
#include "../include/controller/manageString.h"
#include "../include/controller/manageSignal.h"
#include "../include/game/listes.h"
```

Graphe des dépendances par inclusion de main.c :



## Fonctions

— int [main](#) ()

*Programme principal.*

### 5.24.1 Description détaillée

Programme principal.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

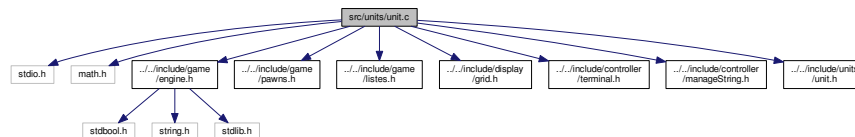
18/12/2015

## 5.25 Référence du fichier src/units/unit.c

Gestion des unités.

```
#include <stdio.h>
#include <math.h>
#include "../include/game/engine.h"
#include "../include/game/pawns.h"
#include "../include/game/listes.h"
#include "../include/display/grid.h"
#include "../include/controller/terminal.h"
#include "../include/controller/manageString.h"
#include "../include/units/unit.h"
```

Graphes des dépendances par inclusion de unit.c :



## Fonctions

- void **unitInit** (short noP, **vector** coordUnit)  
*Initialise l'unité courante.*
- bool **canGetPassed** (**unit** \*target)  
*Définis si le passage est permis.*
- bool **canBlock** (**unit** \*target)  
*Définis si une unité peut bloquer.*
- bool **canAttack** (**unit** \*target)  
*Définis si une unité peut attaquer.*
- bool **canMove** (**unit** \*target)  
*Définis si une unité peut bouger.*
- bool **canTeleport** (**unitName** name)  
*Détermine si une unité peut se téléporter.*
- void **heal** (**unitName** name)  
*Soigne les unités.*
- int **getSideAttacked** (**vector** source, **vector** target)  
*Retourne le côté attaqué*
- void **attack** (**vector** source, **vector** target)  
*Attaque une unité en prenant en compte son taux de blocage.*
- bool **copy** (**unit** \*destination, **unit** \*source)  
*Copie la structure source vers la structure destination.*
- void **erase** (**unit** \*source)  
*Efface une unité de la grille.*
- void **powerBonus** ()  
*Octroie un bonus de puissance.*
- void **move** (**vector** destination, **vector** source)  
*Déplace une unité vers la destination.*
- void **setDirection** (**vector** source, int dir)  
*Définis la direction de l'unité*
- void **poison** ()  
*Gère le statut empoisonnement.*
- void **addEffect** (**vector** target, **unitEffect** effect)  
*Ajoute un effet sur l'unité cible.*
- void **minusEffect** ()  
*Décrémente les effets.*
- void **sleep** (**vector** pos)  
*Endors l'unité*
- bool **isSleeping** (**vector** pos)  
*Vérifie si l'unité est endormie.*
- void **recover** ()  
*Réveille l'unité tour par tour.*
- bool **allStatic** (int numPlayer)  
*Vérifie si toutes les unités sont immobilisés.*

### 5.25.1 Description détaillée

Gestion des unités.

#### Auteur

Cousin Brandon Chaudemanche Ewen Biardeau Tristan

#### Version

v1.00

#### Date

18/12/2015

### 5.25.2 Documentation des fonctions

#### 5.25.2.1 void addEffect ( *vector target*, *unitEffect effect* )

Ajoute un effet sur l'unité cible.

Ajoute un effet à une unité

##### Paramètres

<i>target</i>	Position unité cible
<i>effect</i>	Effet à appliquer sur la cible

Définition à la ligne 437 du fichier unit.c.

#### 5.25.2.2 bool allStatic ( *int numPlayer* )

Vérifie si toutes les unités sont immobilisés.

Vérifie si toutes les unités sont immobilisées.

##### Paramètres

<i>numPlayer</i>	Numéro du joueur
------------------	------------------

#### Renvoie

Retourne vrai si toutes les unités sont immobilisés

Définition à la ligne 529 du fichier unit.c.

#### 5.25.2.3 void attack ( *vector source*, *vector target* )

Attaque une unité en prenant en compte son taux de blocage.

Attaque une unité

##### Paramètres

<i>source</i>	Unité attaquante
<i>target</i>	Unité cible

Définition à la ligne 184 du fichier unit.c.

#### 5.25.2.4 bool canAttack ( unit \* target )

Définis si une unité peut attaquer.

Vérifie si l'unité peut attaquer.

**Paramètres**

<i>target</i>	Unité à analyser
---------------	------------------

**Renvoie**

Retourne vrai si l'unité peut attaquer

Définition à la ligne 86 du fichier unit.c.

**5.25.2.5 bool canBlock ( unit \* target )**

Définis si une unité peut bloquer.

Vérifie si l'unité peut bloquer.

**Paramètres**

<i>target</i>	Unité à analyser
---------------	------------------

**Renvoie**

Retourne vrai si l'unité peut bloquer

Définition à la ligne 66 du fichier unit.c.

**5.25.2.6 bool canGetPassed ( unit \* target )**

Définis si le passage est permis.

Vérifie si le passage est autorisé

**Paramètres**

<i>target</i>	Unité à analyser
---------------	------------------

**Renvoie**

Retourne vrai si passage autorisé

Définition à la ligne 46 du fichier unit.c.

**5.25.2.7 bool canMove ( unit \* target )**

Définis si une unité peut bouger.

Vérifie si l'unité peut bouger.

**Paramètres**

<i>target</i>	Unité à analyser
---------------	------------------

**Renvoie**

Retourne vraie si l'unité peut se mouvoir

Définition à la ligne 106 du fichier unit.c.

**5.25.2.8 bool canTeleport ( unitName name )**

Détermine si une unité peut se téléporter.

Vérifie si l'unité peut se téléporter.



## Paramètres

<i>name</i>	Nom de l'unité
-------------	----------------

## Renvoie

Retourne vraie si l'unité peut se déplacer

Définition à la ligne 125 du fichier unit.c.

## 5.25.2.9 bool copy ( unit \* destination, unit \* source )

Copie la structure source vers la structure destination.

Copie la structure source vers destination.

## Paramètres

<i>destination</i>	Structure destination
<i>source</i>	Structure source

## Renvoie

Retourne vrai si copie bien déroulée

Définition à la ligne 236 du fichier unit.c.

## 5.25.2.10 void erase ( unit \* source )

Efface une unité de la grille.

## Paramètres

<i>source</i>	Source à effacer
---------------	------------------

Définition à la ligne 271 du fichier unit.c.

## 5.25.2.11 int getSideAttacked ( vector source, vector target )

Retourne le côté attaqué

Récupère le côté attaqué

## Paramètres

<i>source</i>	Position unité source
<i>target</i>	Position unité cible

## Renvoie

Le côté attaqué

Définition à la ligne 168 du fichier unit.c.

## 5.25.2.12 void heal ( unitName name )

Soigne les unités.

**Paramètres**

<i>name</i>	Nom de l'unité soignant
-------------	-------------------------

Définition à la ligne 137 du fichier unit.c.

**5.25.2.13 bool isSleeping ( vector pos )**

Vérifie si l'unité est endormie.

Vérifie si une unité est endormie.

**Paramètres**

<i>pos</i>	Position de l'unité
------------	---------------------

**Renvoie**

Retourne vrai si endormi

Définition à la ligne 490 du fichier unit.c.

**5.25.2.14 void minusEffect ( )**

Décrémente les effets.

Diminue le temps de l'effet.

Définition à la ligne 455 du fichier unit.c.

**5.25.2.15 void move ( vector destination, vector source )**

Déplace une unité vers la destination.

Bouge une unité

**Paramètres**

<i>destination</i>	Destination souhaitée
<i>source</i>	Position de l'unité

Définition à la ligne 351 du fichier unit.c.

**5.25.2.16 void poison ( )**

Gère le statut empoisonnement.

Empoisonne une unité

Définition à la ligne 386 du fichier unit.c.

**5.25.2.17 void powerBonus ( )**

Octroie un bonus de puissance.

Octroie un bonus de puissance selon certaines conditions.

Définition à la ligne 286 du fichier unit.c.

**5.25.2.18 void recover ( )**

Réveille l'unité tour par tour.

Se réveille tour par tour.

Définition à la ligne 503 du fichier unit.c.

**5.25.2.19 void setDirection ( vector source, int dir )**

Définis la direction de l'unité

Définis la direction d'une unité

**Paramètres**

<i>source</i>	Unité à tourner
<i>dir</i>	Direction dans laquelle tourner

Définition à la ligne 377 du fichier unit.c.

**5.25.2.20 void sleep ( vector pos )**

Endors l'unité

Endors une unité

**Paramètres**

<i>pos</i>	Position de l'unité
------------	---------------------

Définition à la ligne 480 du fichier unit.c.

**5.25.2.21 void unitInit ( short noP, vector coordUnit )**

Initialise l'unité courante.

Initialise une unité

**Paramètres**

<i>noP</i>	Numéro du joueur
<i>coordUnit</i>	Coordonnées de l'unité

Définition à la ligne 24 du fichier unit.c.



# Index

- addCloseList
  - pathList.c, [72](#)
  - pathList.h, [32](#)
- addEffect
  - unit.c, [82](#)
  - unit.h, [40](#)
- addOpenList
  - pathList.c, [72](#)
  - pathList.h, [33](#)
- addTarget
  - listes.c, [68](#)
  - listes.h, [29](#)
- addUnit
  - listes.c, [68](#)
  - listes.h, [30](#)
- allStatic
  - unit.c, [82](#)
  - unit.h, [41](#)
- askCoord
  - engine.c, [62](#)
- askUnit
  - engine.c, [63](#)
- attack
  - unit.c, [82](#)
  - unit.h, [41](#)
- attackable
  - engine.c, [63](#)
  - engine.h, [25](#)
- borderRight
  - grid.c, [59](#)
- canAttack
  - unit.c, [82](#)
  - unit.h, [41](#)
- canBlock
  - unit.c, [84](#)
  - unit.h, [41](#)
- canGetPassed
  - unit.c, [84](#)
  - unit.h, [42](#)
- canMove
  - unit.c, [84](#)
  - unit.h, [42](#)
- canTeleport
  - unit.c, [84](#)
  - unit.h, [42](#)
- changeDirection
  - turn.c, [78](#)
  - turn.h, [38](#)
- checkDecrypt
  - saveNLoad.c, [54](#)
- checkPawn
  - pawns.c, [76](#)
- checkSignal
  - manageSignal.c, [47](#)
  - manageSignal.h, [14](#)
- clearBuffer
  - manageString.c, [49](#)
  - manageString.h, [15](#)
- color
  - terminal.c, [57](#)
  - terminal.h, [20](#)
- copy
  - unit.c, [85](#)
  - unit.h, [42](#)
- correctCoord
  - manageString.c, [49](#)
- countUnits
  - listes.c, [69](#)
  - listes.h, [30](#)
- createPawn
  - pawns.c, [76](#)
- crypt
  - saveNLoad.c, [54](#)
- decrypt
  - saveNLoad.c, [54](#)
- destroyUnit
  - listes.c, [69](#)
  - listes.h, [30](#)
- dispDirection
  - menu.c, [60](#)
  - menu.h, [22](#)
- dumpAllLists
  - listes.c, [69](#)
  - listes.h, [30](#)
- dumpAllPaths
  - pathList.c, [72](#)
  - pathList.h, [33](#)
- dumpList
  - listes.c, [69](#)
  - listes.h, [30](#)
- dumpPath
  - pathList.c, [72](#)
  - pathList.h, [33](#)
- elem, [7](#)
- element, [8](#)
- emptyPath

- pathList.c, 72
- pathList.h, 33
- endTurn
  - turn.c, 78
- engine.c
  - askCoord, 62
  - askUnit, 63
  - attackable, 63
  - gameInit, 63
  - grid, 66
  - isSurrounded, 63
  - launchAttack, 63
  - lineOfSight, 64
  - movable, 64
  - noPlayer, 66
  - pathFind, 64
  - playerAddUnit, 64
  - possiblePath, 64
  - selectUnit, 65
  - setTarget, 65
  - specialBoons, 65
  - tileWalkable, 65
  - tooMuchUnit, 66
  - updateLimits, 66
- engine.h
  - attackable, 25
  - gameInit, 25
  - grid, 28
  - isSurrounded, 25
  - launchAttack, 26
  - lineOfSight, 26
  - movable, 26
  - noPlayer, 28
  - pathFind, 26
  - possiblePath, 27
  - selectUnit, 27
  - setTarget, 27
  - tileWalkable, 27
- erase
  - unit.c, 85
  - unit.h, 43
- eraseTile
  - pathList.c, 73
  - pathList.h, 33
- fontColor
  - terminal.c, 57
  - terminal.h, 20
- freeAll
  - manageSignal.c, 47
  - manageSignal.h, 14
- gameInit
  - engine.c, 63
  - engine.h, 25
- gameMenu
  - menu.c, 60
  - menu.h, 22
- get2Char
  - manageString.c, 49
  - manageString.h, 15
- getCharKey
  - saveNLoad.c, 54
- getColor
  - terminal.c, 57
- getCoordS
  - manageString.c, 49
  - manageString.h, 15
- getCurrentNode
  - pathList.c, 73
  - pathList.h, 34
- getDirectionUnit
  - manageString.c, 50
  - manageString.h, 15
- getKey
  - saveNLoad.c, 54
- getNameEffect
  - manageString.c, 50
  - manageString.h, 16
- getNameUnit
  - manageString.c, 50
  - manageString.h, 16
- getSideAttacked
  - unit.c, 85
  - unit.h, 43
- getTile
  - pathList.c, 73
  - pathList.h, 34
- grid
  - engine.c, 66
  - engine.h, 28
- grid.c
  - borderRight, 59
  - gridDisp, 59
- grid.h
  - gridDisp, 21
- gridDisp
  - grid.c, 59
  - grid.h, 21
- hasPlay
  - turn.c, 79
  - turn.h, 38
- heal
  - unit.c, 85
  - unit.h, 43
- helpUnit
  - menu.c, 60
  - menu.h, 22
- include/controller/manageSignal.h, 13
- include/controller/manageString.h, 14
- include/controller/saveNLoad.h, 18
- include/controller/terminal.h, 19
- include/display/grid.h, 20
- include/display/menu.h, 21
- include/game/engine.h, 23
- include/game/listes.h, 28

- include/game/pathList.h, 31
- include/game/pawns.h, 36
- include/game/turn.h, 37
- include/units/unit.h, 39
- initPath
  - pathList.c, 73
  - pathList.h, 34
- initPawn
  - pawns.c, 77
- interrupt
  - manageSignal.c, 47
- isOutGrid
  - manageString.c, 50
- isSleeping
  - unit.c, 86
  - unit.h, 43
- isSurrounded
  - engine.c, 63
  - engine.h, 25
- launchAttack
  - engine.c, 63
  - engine.h, 26
- lineOfSight
  - engine.c, 64
  - engine.h, 26
- listes.c
  - addTarget, 68
  - addUnit, 68
  - countUnits, 69
  - destroyUnit, 69
  - dumpAllLists, 69
  - dumpList, 69
  - printList, 69
  - searchTarget, 69
  - targetList, 70
- listes.h
  - addTarget, 29
  - addUnit, 30
  - countUnits, 30
  - destroyUnit, 30
  - dumpAllLists, 30
  - dumpList, 30
  - printList, 30
  - searchTarget, 31
  - targetList, 31
- load
  - saveNLoad.c, 56
  - saveNLoad.h, 18
- mainMenu
  - menu.c, 60
  - menu.h, 22
- makePawns
  - pawns.c, 77
  - pawns.h, 37
- manageSignal.c
  - checkSignal, 47
  - freeAll, 47
- interrupt, 47
- terminator, 47
- timeDown, 47
- manageSignal.h
  - checkSignal, 14
  - freeAll, 14
- manageString.c
  - clearBuffer, 49
  - correctCoord, 49
  - get2Char, 49
  - getCoordS, 49
  - getDirectionUnit, 50
  - getNameEffect, 50
  - getNameUnit, 50
  - isOutGrid, 50
  - printNameUnit, 51
  - readDouble, 51
  - readLong, 51
  - readS, 51
  - rightSide, 51
- manageString.h
  - clearBuffer, 15
  - get2Char, 15
  - getCoordS, 15
  - getDirectionUnit, 15
  - getNameEffect, 16
  - getNameUnit, 16
  - printNameUnit, 16
  - readDouble, 16
  - readLong, 16
  - readS, 17
  - rightSide, 17
- menu.c
  - dispDirection, 60
  - gameMenu, 60
  - helpUnit, 60
  - mainMenu, 60
  - unitList, 60
  - unitMenu, 61
- menu.h
  - dispDirection, 22
  - gameMenu, 22
  - helpUnit, 22
  - mainMenu, 22
  - unitList, 22
  - unitMenu, 23
- minusEffect
  - unit.c, 86
  - unit.h, 43
- movable
  - engine.c, 64
  - engine.h, 26
- move
  - unit.c, 86
  - unit.h, 44
- next
  - pathList.c, 73
  - pathList.h, 34

- noPlayer
  - engine.c, 66
  - engine.h, 28
- outPath
  - pathList.c, 74
  - pathList.h, 34
- passTurn
  - turn.c, 79
  - turn.h, 38
- pathFind
  - engine.c, 64
  - engine.h, 26
- pathHead
  - pathList.c, 74
  - pathList.h, 35
- pathList.c
  - addCloseList, 72
  - addOpenList, 72
  - dumpAllPaths, 72
  - dumpPath, 72
  - emptyPath, 72
  - eraseTile, 73
  - getCurrentNode, 73
  - getTile, 73
  - initPath, 73
  - next, 73
  - outPath, 74
  - pathHead, 74
  - pathTail, 74
  - previous, 74
  - searchTile, 74
  - setTile, 75
  - toRightPath, 75
- pathList.h
  - addCloseList, 32
  - addOpenList, 33
  - dumpAllPaths, 33
  - dumpPath, 33
  - emptyPath, 33
  - eraseTile, 33
  - getCurrentNode, 34
  - getTile, 34
  - initPath, 34
  - next, 34
  - outPath, 34
  - pathHead, 35
  - pathTail, 35
  - previous, 35
  - searchTile, 35
  - setTile, 35
  - toRightPath, 36
- pathTail
  - pathList.c, 74
  - pathList.h, 35
- pawns
  - pawns.c, 77
  - pawns.h, 37
- pawns.c
  - checkPawn, 76
  - createPawn, 76
  - initPawn, 77
  - makePawns, 77
  - pawns, 77
- pawns.h
  - makePawns, 37
  - pawns, 37
- playAttack
  - turn.c, 79
  - turn.h, 39
- playMove
  - turn.c, 79
  - turn.h, 39
- playTurn
  - turn.c, 79
  - turn.h, 39
- playerAddUnit
  - engine.c, 64
- poison
  - unit.c, 86
  - unit.h, 44
- possiblePath
  - engine.c, 64
  - engine.h, 27
- powerBonus
  - unit.c, 86
  - unit.h, 44
- previous
  - pathList.c, 74
  - pathList.h, 35
- printList
  - listes.c, 69
  - listes.h, 30
- printNameUnit
  - manageString.c, 51
  - manageString.h, 16
- readDouble
  - manageString.c, 51
  - manageString.h, 16
- readLong
  - manageString.c, 51
  - manageString.h, 16
- readS
  - manageString.c, 51
  - manageString.h, 17
- recover
  - unit.c, 86
  - unit.h, 44
- reinitColor
  - terminal.c, 58
  - terminal.h, 20
- rightSide
  - manageString.c, 51
  - manageString.h, 17
- save



- saveNLoad.c, 56
- saveNLoad.h, 18
- saveNLoad.c
  - checkDecrypt, 54
  - crypt, 54
  - decrypt, 54
  - getCharKey, 54
  - getKey, 54
  - load, 56
  - save, 56
- saveNLoad.h
  - load, 18
  - save, 18
- searchTarget
  - listes.c, 69
  - listes.h, 31
- searchTile
  - pathList.c, 74
  - pathList.h, 35
- selectUnit
  - engine.c, 65
  - engine.h, 27
- setAction
  - turn.c, 79
- setDirection
  - unit.c, 87
  - unit.h, 44
- setTarget
  - engine.c, 65
  - engine.h, 27
- setTile
  - pathList.c, 75
  - pathList.h, 35
- sleep
  - unit.c, 87
  - unit.h, 44
- specialBoons
  - engine.c, 65
- src/controller/manageSignal.c, 46
- src/controller/manageString.c, 48
- src/controller/saveNLoad.c, 53
- src/controller/terminal.c, 56
- src/display/grid.c, 58
- src/display/menu.c, 59
- src/game/engine.c, 61
- src/game/listes.c, 66
- src/game/pathList.c, 70
- src/game/pawns.c, 75
- src/game/turn.c, 77
- src/main.c, 80
- src/units/unit.c, 80
- targetList
  - listes.c, 70
  - listes.h, 31
- targetStat, 8
- terminal.c
  - color, 57
  - fontColor, 57
- getColor, 57
- reinitColor, 58
- terminal.h
  - color, 20
  - fontColor, 20
  - reinitColor, 20
- terminator
  - manageSignal.c, 47
- tileWalkable
  - engine.c, 65
  - engine.h, 27
- timeDown
  - manageSignal.c, 47
- toRightPath
  - pathList.c, 75
  - pathList.h, 36
- tooMuchUnit
  - engine.c, 66
- turn.c
  - changeDirection, 78
  - endTurn, 78
  - hasPlay, 79
  - passTurn, 79
  - playAttack, 79
  - playMove, 79
  - playTurn, 79
  - setAction, 79
- turn.h
  - changeDirection, 38
  - hasPlay, 38
  - passTurn, 38
  - playAttack, 39
  - playMove, 39
  - playTurn, 39
- unit, 9
- unit.c
  - addEffect, 82
  - allStatic, 82
  - attack, 82
  - canAttack, 82
  - canBlock, 84
  - canGetPassed, 84
  - canMove, 84
  - canTeleport, 84
  - copy, 85
  - erase, 85
  - getSideAttacked, 85
  - heal, 85
  - isSleeping, 86
  - minusEffect, 86
  - move, 86
  - poison, 86
  - powerBonus, 86
  - recover, 86
  - setDirection, 87
  - sleep, 87
  - unitInit, 87
- unit.h

- addEffect, [40](#)
- allStatic, [41](#)
- attack, [41](#)
- canAttack, [41](#)
- canBlock, [41](#)
- canGetPassed, [42](#)
- canMove, [42](#)
- canTeleport, [42](#)
- copy, [42](#)
- erase, [43](#)
- getSideAttacked, [43](#)
- heal, [43](#)
- isSleeping, [43](#)
- minusEffect, [43](#)
- move, [44](#)
- poison, [44](#)
- powerBonus, [44](#)
- recover, [44](#)
- setDirection, [44](#)
- sleep, [44](#)
- unitInit, [46](#)
- unitInit
  - unit.c, [87](#)
  - unit.h, [46](#)
- unitList
  - menu.c, [60](#)
  - menu.h, [22](#)
- unitMenu
  - menu.c, [61](#)
  - menu.h, [23](#)
- unitStat, [10](#)
- updateLimits
  - engine.c, [66](#)
- vector, [10](#)