

Durableverse - Les Trilobytes

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>File Index</b>	<b>1</b>
1.1	File List . . . . .	1
<b>2</b>	<b>File Documentation</b>	<b>3</b>
2.1	carte.h File Reference . . . . .	3
2.1.1	Detailed Description . . . . .	5
2.1.2	Function Documentation . . . . .	5
2.1.2.1	addCardSustainable() . . . . .	5
2.1.2.2	addDurability() . . . . .	5
2.1.2.3	copyCard() . . . . .	6
2.1.2.4	destroyCard() . . . . .	6
2.1.2.5	getCardId() . . . . .	6
2.1.2.6	getCardSustainable() . . . . .	7
2.1.2.7	getDurability() . . . . .	7
2.1.2.8	getEffects() . . . . .	7
2.1.2.9	getName() . . . . .	8
2.1.2.10	getPE() . . . . .	8
2.1.2.11	getSizeEffects() . . . . .	8
2.1.2.12	getSpecialisation() . . . . .	10
2.1.2.13	getType() . . . . .	10
2.1.2.14	isEndPhaseCard() . . . . .	11
2.1.2.15	removeCardSustainable() . . . . .	11
2.1.2.16	removeDurability() . . . . .	11
2.1.2.17	setCardSustainable() . . . . .	12

2.1.2.18	setDurability()	12
2.1.2.19	setSpecialisation()	12
2.2	interface.h File Reference	13
2.2.1	Detailed Description	13
2.2.2	Function Documentation	13
2.2.2.1	askCard()	13
2.2.2.2	printNewGame()	14
2.2.2.3	printNewPhase()	14
2.2.2.4	printNewRound()	14
2.2.2.5	printStateOfGame()	15
2.2.2.6	printStudentPool()	15
2.2.2.7	whichStudent()	15
2.3	jeu.h File Reference	16
2.3.1	Detailed Description	17
2.3.2	Function Documentation	17
2.3.2.1	createGame()	17
2.3.2.2	destroyGame()	17
2.3.2.3	endRound()	17
2.3.2.4	getCurrentPlayer()	18
2.3.2.5	getNotCurrentPlayer()	18
2.3.2.6	getOtherPlayer()	18
2.3.2.7	getPlayer()	19
2.3.2.8	getRound()	19
2.3.2.9	isGameFinished()	20
2.3.2.10	isRoundEven()	20
2.3.2.11	newRound()	20
2.3.2.12	playCard()	21
2.3.2.13	playPhase()	21
2.3.2.14	sendToGraveyard()	21
2.3.2.15	switchCurrentPlayer()	22

2.4	joueur.h File Reference	22
2.4.1	Detailed Description	24
2.4.2	Function Documentation	24
2.4.2.1	addAdditionalCard()	24
2.4.2.2	addAdditionalStudent()	25
2.4.2.3	addAdditionalSustainablePoints()	25
2.4.2.4	addEnergy()	25
2.4.2.5	addPlayerSustainablePoints()	26
2.4.2.6	changeSustOrDura()	26
2.4.2.7	countEnergy()	26
2.4.2.8	createEnsiie()	27
2.4.2.9	destroyEnsiie()	27
2.4.2.10	drawEnsiie()	27
2.4.2.11	enoughEnergy()	28
2.4.2.12	getAdditionalSustainablePoints()	28
2.4.2.13	getBoard()	28
2.4.2.14	getChangedSustOrDura()	29
2.4.2.15	getDeck()	29
2.4.2.16	getEnergy()	30
2.4.2.17	getFisaNumber()	30
2.4.2.18	getFiseNumber()	30
2.4.2.19	getGraveyard()	31
2.4.2.20	getHand()	31
2.4.2.21	getPlayerName()	31
2.4.2.22	getPlayerSustainablePoints()	32
2.4.2.23	getSumCardDurability()	32
2.4.2.24	getSumCardSustainable()	33
2.4.2.25	getUpgradedCardDurability()	33
2.4.2.26	getUpgradedCardSustainable()	33
2.4.2.27	initiateEnsiie()	34

2.4.2.28	isCurrentPlayer()	34
2.4.2.29	isInHand()	34
2.4.2.30	numberCard()	35
2.4.2.31	numberEleve()	35
2.4.2.32	playDistanciation()	35
2.4.2.33	removeAdditionalSustainablePoints()	36
2.4.2.34	removeEnergy()	36
2.4.2.35	setEnergy()	36
2.4.2.36	setIsCurrentPlayer()	37
2.4.2.37	setPlayerName()	37
2.4.2.38	updateSustAndDura()	37
2.4.2.39	upgradeCardDurability()	38
2.4.2.40	upgradeCardSustainable()	38
2.5	plateau.h File Reference	38
2.5.1	Detailed Description	39
2.5.2	Function Documentation	39
2.5.2.1	createBoard()	39
2.5.2.2	destroyBoard()	39
2.5.2.3	getAvailableSpotNumber()	40
2.5.2.4	getStaffPool()	40
2.5.2.5	getStudentPool()	40
2.5.2.6	getUnlockedSpotNumber()	41
2.5.2.7	isStaffSpecialised()	41
2.5.2.8	updateUnlockedSpots()	42
2.6	structure.h File Reference	42
2.6.1	Detailed Description	43
2.6.2	Function Documentation	43
2.6.2.1	addAndSort()	43
2.6.2.2	addElement()	43
2.6.2.3	createListCard()	43
2.6.2.4	destroyElements()	44
2.6.2.5	destroyListCard()	44
2.6.2.6	displaceElement()	44
2.6.2.7	getList()	45
2.6.2.8	getSize()	45
2.6.2.9	removeElement()	45
2.7	utils.h File Reference	46
2.7.1	Detailed Description	46
2.7.2	Function Documentation	46
2.7.2.1	max()	46

# Chapter 1

## File Index

### 1.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">carte.h</a>	Gestion du type carte . . . . .	3
<a href="#">interface.h</a>	Gestion de l'interface en console . . . . .	13
<a href="#">jeu.h</a>	Gestion du jeu . . . . .	16
<a href="#">joueur.h</a>	Gestion des joueurs . . . . .	22
<a href="#">plateau.h</a>	Gestion du plateau . . . . .	38
<a href="#">structure.h</a>	Gestion des listes de cartes . . . . .	42
<a href="#">utils.h</a>	Fonctions utilitaires . . . . .	46





## Chapter 2

# File Documentation

### 2.1 carte.h File Reference

Gestion du type carte.

```
#include "utils.h"
```

#### Typedefs

- typedef enum [type\\_card](#) **type\_card**
- typedef enum [card\\_effect](#) **card\_effect**
- typedef enum [id\\_cardname](#) **id\_cardname**
- typedef enum [card\\_specialise](#) **card\_specialise**
- typedef struct str\_card \* **Card**

*Définition abstraite de la structure correspondant à une carte.*

#### Enumerations

- enum [type\\_card](#) { **STUDENT**, **STAFF**, **ACTION** }

*Type d'une carte : Eleve, Personnel ou Action.*

- enum [card\\_effect](#) {  
**AE1**, **AE2**, **AA1**, **AA2**,  
**RE1**, **RE2**, **RA1**, **RA2**,  
**ADD**, **RDD**, **DR**, **E**,  
**CDD**, **REC**, **RFISE**, **RFISA**,  
**EV**, **DIP**, **DCH**, **RCY**,  
**ZPA**, **RVG**, **FER**, **DIS**,  
**BDE**, **PPL**, **PGL**, **PHPC**,  
**PMA**, **PIN** }

*Effets des cartes Personnel et Action.*

- enum `id_cardname` {  
**END\_PHASE**, **LIM**, **SZAFRANSKI**, **FAYE**,  
**MOUILLERON**, **DUMBRAVA**, **FOREST**, **BRUNEL**,  
**BOURARD**, **WATEL**, **Y**, **GOILARD**,  
**JEANNAS**, **MERABET**, **LIGOZAT**, **DUBOIS**,  
**LEJEUNE**, **MATHIAS**, **SALHAB**, **SAGNA**,  
**PREVEL**, **SANDRINE**, **DOSSANTOS**, **COURS\_DD**,  
**RECRUTEMENT**, **RENTREE\_FISE**, **RENTREE\_FISA**, **ENERGIE\_VERTE**,  
**DIPLOMATION**, **DECHARGE**, **RECYCLAGE**, **ZERO\_PAPIER**,  
**REPAS\_VEGETARIEN**, **FERMETURE\_ANNUELLE**, **DISTANCIATION\_SOCIALE**, **SOIREE\_BDE**,  
**PARCOURS\_PL**, **PARCOURS\_GL**, **PARCOURS\_HPC**, **PARCOURS\_MA**,  
**PARCOURS\_IN**, **FISE**, **FISA** }

*Relation entre l'id d'une carte et son nom L'id d'une carte est compris entre 0 et 33, de 1 à 20 : cartes Personnels, de 21 à 31 : cartes Action, 32 : FISE, 33 : FISA et 0 : carte virtuelle de fin de phase.*

- enum `card_specialise` {  
**NOT\_SPECIALISED**, **PL**, **GL**, **HPC**,  
**MA**, **IN** }

*Spécialisation des élèves et profs.*

## Functions

- `Card createCard` (`id_cardname` id)
- void `destroyCard` (`Card` card)  
*Destructeur de carte.*
- `Card copyCard` (`Card` source)  
*Constructeur de carte par copie.*
- int `initiateQuantity` (`id_cardname` id)
- int `isEndPhaseCard` (`Card` card)  
*Fonction qui teste la fin de la phase en cours On décide d'ajouter une carte virtuelle fin de phase au jeu, toujours présente dans la main du joueur.*
- `type_card getType` (`Card` card)  
*Accesseur en lecture du type d'une carte.*
- `id_cardname getCardId` (`Card` card)  
*Accesseur en lecture de l'id d'une carte.*
- char \* `getName` (`Card` card)  
*Accesseur en lecture du nom d'une carte.*
- int `getPE` (`Card` card)  
*Accesseur en lecture du coût en PE d'une carte.*
- int `getCardSustainable` (`Card` card)  
*Accesseur en lecture des points de développement durable d'une carte.*
- int `getDurability` (`Card` card)  
*Accesseur en lecture des points de durabilité d'une carte.*
- `card_effect * getEffects` (`Card` card)  
*Accesseur en lecture des effets d'une carte.*
- int `getSizeEffects` (`Card` card)  
*Accesseur en lecture de la longueur de la liste des effets d'une carte.*
- `card_specialise getSpecialisation` (`Card` card)  
*Accesseur en lecture de la spécialisation d'une carte.*
- void `setDurability` (`Card` card, int value)  
*Accesseur en écriture du nombre de points de durabilité de la carte.*
- void `setCardSustainable` (`Card` card, int value)  
*Accesseur en écriture du nombre de points de sustainable\_points de la carte.*

- void `addDurability` (`Card` card, int value)  
*Modification en écriture du nombre de points de durabilité de la carte.*
- void `addCardSustainable` (`Card` card, int value)  
*Modification en écriture du nombre de points de sustainable\_points de la carte.*
- void `removeDurability` (`Card` card, int value)  
*Modification en écriture du nombre de points de durabilité de la carte (moins)*
- void `removeCardSustainable` (`Card` card, int value)  
*Modification en écriture du nombre de points de sustainable\_points de la carte (moins)*
- void `setSpecialisation` (`Card` card, `card_specialise` special)  
*Mutateur de la spécialisation d'une carte.*

### 2.1.1 Detailed Description

Gestion du type carte.

#### Author

Maureen Lachaize

### 2.1.2 Function Documentation

#### 2.1.2.1 addCardSustainable()

```
void addCardSustainable (  
    Card card,  
    int value )
```

Modification en écriture du nombre de points de sustainable\_points de la carte.

#### Parameters

<i>card</i>	une carte du jeu
<i>value</i>	l'entier correspondant à la valeur à ajouter à sustainable_points

#### 2.1.2.2 addDurability()

```
void addDurability (  
    Card card,  
    int value )
```

Modification en écriture du nombre de points de durabilité de la carte.

**Parameters**

<i>card</i>	une carte du jeu
<i>value</i>	l'entier correspondant à la valeur à ajouter à la durabilité

**2.1.2.3 copyCard()**

```
Card copyCard (
    Card source )
```

Constructeur de carte par copie.

**Parameters**

<i>source</i>	Une carte du jeu
---------------	------------------

**Returns**

Instance nouvellement allouée de type Card, identique à source

**2.1.2.4 destroyCard()**

```
void destroyCard (
    Card card )
```

Destructeur de carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**2.1.2.5 getCardId()**

```
id_cardname getCardId (
    Card card )
```

Accesseur en lecture de l'id d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

L'ID de la carte

**2.1.2.6 getCardSustainable()**

```
int getCardSustainable (
    Card card )
```

Accesseur en lecture des points de développement durable d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

Le nombre de points de développement durable de la carte

**2.1.2.7 getDurability()**

```
int getDurability (
    Card card )
```

Accesseur en lecture des points de durabilité d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

Le nombre de points de durabilité de la carte

**2.1.2.8 getEffects()**

```
card_effect * getEffects (
    Card card )
```

Accesseur en lecture des effets d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

La liste des effets de la carte

**2.1.2.9 getName()**

```
char * getName (
    Card card )
```

Accesseur en lecture du nom d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

Le nom de la carte (ex: Alain Faye, FISE, Recyclage)

**2.1.2.10 getPE()**

```
int getPE (
    Card card )
```

Accesseur en lecture du coût en PE d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

Le coût en PE de la carte

**2.1.2.11 getSizeEffects()**

```
int getSizeEffects (
    Card card )
```

Accesseur en lecture de la longueur de la liste des effets d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

La longueur de la liste des effets de la carte

**2.1.2.12 getSpecialisation()**

```
card_specialise getSpecialisation (
    Card card )
```

Accesseur en lecture de la spécialisation d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu valide
-------------	-------------------------

**Returns**

Renvoie la spécialisation d'une carte (avec l'énum correspondant)

**Author**

Alex Danduran–Lembezat

**2.1.2.13 getType()**

```
type_card getType (
    Card card )
```

Accesseur en lecture du type d'une carte.

**Parameters**

<i>card</i>	Une carte du jeu
-------------	------------------

**Returns**

Le type de la carte : Personnel, Etudiant ou Action



#### 2.1.2.14 isEndPhaseCard()

```
int isEndPhaseCard (
    Card card )
```

Fonction qui teste la fin de la phase en cours On décide d'ajouter une carte virtuelle fin de phase au jeu, toujours présente dans la main du joueur.

##### Parameters

<i>card</i>	Une carte du jeu
-------------	------------------

##### Returns

1 si card est la carte fin de phase et 0 sinon

#### 2.1.2.15 removeCardSustainable()

```
void removeCardSustainable (
    Card card,
    int value )
```

Modification en écriture du nombre de points de sustainable\_points de la carte (moins)

##### Parameters

<i>card</i>	une carte du jeu
<i>value</i>	l'entier correspondant à la valeur à retirer à sustainable_points

#### 2.1.2.16 removeDurability()

```
void removeDurability (
    Card card,
    int value )
```

Modification en écriture du nombre de points de durabilité de la carte (moins)

##### Parameters

<i>card</i>	une carte du jeu
<i>value</i>	l'entier correspondant à la valeur à retirer à la durabilité

### 2.1.2.17 setCardSustainable()

```
void setCardSustainable (
    Card card,
    int value )
```

Accesseur en écriture du nombre de points de sustainable\_points de la carte.

#### Parameters

<i>card</i>	une carte du jeu
<i>value</i>	l'entier correspondant à la valeur voulue de sustainable_points

### 2.1.2.18 setDurability()

```
void setDurability (
    Card card,
    int value )
```

Accesseur en écriture du nombre de points de durabilité de la carte.

#### Parameters

<i>card</i>	une carte du jeu
<i>value</i>	l'entier correspondant à la valeur voulue de durability

### 2.1.2.19 setSpecialisation()

```
void setSpecialisation (
    Card card,
    card_specialise special )
```

Mutateur de la spécialisation d'une carte.

#### Parameters

<i>card</i>	Une carte du jeu valide
<i>special</i>	Une spécialisation valide (provenant de l'enum card_specialise)

#### Author

Alex Danduran–Lembezat

## 2.2 interface.h File Reference

Gestion de l'interface en console.

```
#include "joueur.h"
```

### Functions

- int [whichStudent](#) ()  
*Demande au joueur quel type de carte élève il souhaite prendre.*
- [Card askCard](#) ([Ensiie](#) player)  
*Réalise un appel à printHand pour les cartes dont le cout est inférieur au PE Demande si le joueur veut jouer une carte et, dans ce cas, quelle carte le joueur souhaite jouer et renvoie la carte (card) choisie par le joueur (renvoie une carte "default" (définie à l'avance avec un id=0) si le joueur n'en choisit aucune). Informe également le joueur s'il ne dispose pas d'assez de PE.*
- void [printNotEnoughPE](#) ()  
*Imprime sur le flux de sortie un message d'erreur. Utilisée lorsque le joueur souhaite jouer une carte dont le coût dépasse son nombre de PE.*
- void [printStatsOfGame](#) ([Ensiie](#) player1, [Ensiie](#) player2, int round)  
*Affiche l'état actuel du jeu : le plateau avec les cartes Eleve et Personnel.*
- void [printHand](#) ([List\\_Card](#) list\_hand, int energy)
- void [printListCard](#) ([List\\_Card](#) liste\_cartes)
- void [printGraveyard](#) ([List\\_Card](#) graveyard)
- void [printStudentPool](#) ([List\\_Card](#) pool, int round)  
*Imprime sur le flux de sortie le contenu de la liste de cartes élèves "pool" (particularité : affiche les sustainable\_points et durability\_points) affiche également les cartes FISA comme non jouables en tour pair.*
- void [printEndGame](#) (int status, [Ensiie](#) player1, [Ensiie](#) player2)
- void [printNewRound](#) (int round)  
*Imprime sur le flux de sortie qu'un nouveau tour vient de débiter et indique le numéro du tour ainsi que d'autres informations utiles (par exemple la présence ou non d'un nouvel élément "Personnel")*
- void [printNewPhase](#) ([Ensiie](#) player1, [Ensiie](#) player2)  
*Imprime sur le flux de sortie qu'une nouvelle phase vient de débiter et indique l'ENSIIE associée à cette phase.*
- void [printNewGame](#) ([Ensiie](#) player1, [Ensiie](#) player2)  
*Imprime sur le flux de sortie qu'une nouvelle partie commence et demande aux joueurs leurs noms.*

### 2.2.1 Detailed Description

Gestion de l'interface en console.

Author

Mathias DURAND

### 2.2.2 Function Documentation

#### 2.2.2.1 askCard()

```
Card askCard (  
    Ensiie player )
```

Réalise un appel à printHand pour les cartes dont le cout est inférieur au PE Demande si le joueur veut jouer une carte et, dans ce cas, quelle carte le joueur souhaite jouer et renvoie la carte (card) choisie par le joueur (renvoie une carte "default" (définie à l'avance avec un id=0) si le joueur n'en choisit aucune). Informe également le joueur s'il ne dispose pas d'assez de PE.

**Parameters**

<i>player</i>	un joueur valide (possédant une liste de carte à jouer (une main))
---------------	--

**Returns**

la carte que le joueur désire jouer

**2.2.2.2 printNewGame()**

```
void printNewGame (
    Ensie player1,
    Ensie player2 )
```

Imprime sur le flux de sortie qu'une nouvelle partie commence et demande aux joueurs leurs noms.

**Parameters**

<i>player1</i>	le premier joueur du jeu
<i>player2</i>	le second joueur du jeu

**2.2.2.3 printNewPhase()**

```
void printNewPhase (
    Ensie current_player,
    Ensie player1 )
```

Imprime sur le flux de sortie qu'une nouvelle phase vient de débiter et indique l'ENSIIE associée à cette phase.

**Parameters**

<i>player1</i>	le premier joueur du jeu
<i>player2</i>	le second joueur du jeu

**2.2.2.4 printNewRound()**

```
void printNewRound (
    int round )
```

Imprime sur le flux de sortie qu'un nouveau tour vient de débiter et indique le numéro du tour ainsi que d'autres informations utiles (par exemple la présence ou non d'un nouvel élément "Personnel")

## Parameters

<i>round</i>	un entier
--------------	-----------

## 2.2.2.5 printStateOfGame()

```
void printStateOfGame (
    Ensie player1,
    Ensie player2,
    int round )
```

Affiche l'état actuel du jeu : le plateau avec les cartes Eleve et Personnel.

## Parameters

<i>player1</i>	un joueur valide
<i>player2</i>	un joueur valide
<i>round</i>	le numéro du tour

## 2.2.2.6 printStudentPool()

```
void printStudentPool (
    List_Card pool,
    int round )
```

Imprime sur le flux de sortie le contenu de la liste de cartes élèves "pool" (particularité : affiche les sustainable\_↔ points et durability\_points) affiche également les cartes FISA comme non jouables en tour pair.

## Parameters

<i>pool</i>	une liste de cartes (card) : la liste de cartes élèves
<i>round</i>	entier désignant le numéro du tour

## 2.2.2.7 whichStudent()

```
int whichStudent ( )
```

Demande au joueur quel type de carte élève il souhaite prendre.

## Returns

renvoie 0 s'il choisit une FISE, 1 si il choisit une FISA

## 2.3 jeu.h File Reference

Gestion du jeu.

```
#include "joueur.h"
#include "interface.h"
```

### Typedefs

- typedef enum [game\\_status](#) **game\_status**
- typedef struct str\_game \* [Game](#)  
*Définition abstraite de la structure correspondant à une partie.*

### Enumerations

- enum [game\\_status](#) { **NOT\_FINISHED**, **ONE\_WINNER**, **TWO\_WINNER**, **DRAW** }  
*Différents états possibles du jeu : en cours, un gagnant, égalité ou match nul.*

### Functions

- [Game](#) [createGame](#) ()  
*Constructeur de partie.*
- void [destroyGame](#) ([Game](#) game)  
*Destructeur de partie : la partie est libérée.*
- void [newRound](#) ([Game](#) game)  
*Début d'un nouveau tour : mise à jour des espaces disponibles pour les cartes Personnel.*
- void [endRound](#) ([Game](#) game)  
*Fin du tour : Modifie les compteurs de DD de chaque joueur, incrémente le compteur de tours.*
- void [playPhase](#) ([Game](#) game)  
*Exécute une phase de jeu pour le joueur courant.*
- int [isGameFinished](#) ([Game](#) game)  
*Est-ce la fin de la partie ?*
- [Ensiie](#) [getCurrentPlayer](#) ([Game](#) game)  
*Retourne le joueur courant.*
- [Ensiie](#) [getNotCurrentPlayer](#) ([Game](#) game)  
*Retourne le joueur qui n'est pas le joueur courant.*
- void [switchCurrentPlayer](#) ([Game](#) game)  
*Échange le caractère de joueur courant entre les 2 joueurs Si joueur1 est le joueur courant, alors joueur2 devient le joueur courant et joueur1 n'est plus joueur courant Si joueur2 est le joueur courant, alors joueur1 devient le joueur courant et joueur2 n'est plus joueur courant.*
- int [isRoundEven](#) ([Game](#) game)  
*Permet de savoir si le tour de jeu est un tour pair ou impair.*
- [Ensiie](#) [getPlayer](#) ([Game](#) game, int player\_id)  
*getter de joueur*
- [Ensiie](#) [getOtherPlayer](#) ([Game](#) game, [Ensiie](#) player)  
*Renvoie le joueur qui n'est pas le joueur renseigné*
- int [getRound](#) ([Game](#) game)  
*getter de tour*
- int [playCard](#) ([Card](#) played\_card, [Game](#) game)  
*Joue une carte de la main du joueur courant : déplace la carte played\_card de la main du à son plateau ou à la défausse et en applique les effets, recalcul des PE.*
- void [sendToGraveyard](#) ([Game](#) game, [Ensiie](#) player, [List\\_Card](#) corresponding\_list, [Card](#) card)  
*Envoie dans la défausse d'un joueur une carte donnée provenant d'une liste donnée et retire ses effets si elle en a.*

### 2.3.1 Detailed Description

Gestion du jeu.

Author

Sirine HAMDANA

### 2.3.2 Function Documentation

#### 2.3.2.1 createGame()

```
Game createGame ( )
```

Constructeur de partie.

##### Returns

creee une partie

- Génère chaque ENSIIE (player1->is\_current\_player à 1 et player2->is\_current\_player à 0) donc par défaut player1 est le joueur courant
- Initialise nombre de tour (initialisé à 0)
- Initialise numéro de phase (0)

#### 2.3.2.2 destroyGame()

```
void destroyGame (  
    Game game )
```

Destructeur de partie : la partie est liberee.

##### Parameters

<i>game</i>	la partie a detruire
-------------	----------------------

#### 2.3.2.3 endRound()

```
void endRound (  
    Game game )
```

Fin du tour : Modifie les compteurs de DD de chaque joueur, incrémente le compteur de tours.

**Parameters**

<i>game</i>	une partie en cours
-------------	---------------------

**2.3.2.4 getCurrentPlayer()**

```
Ensiie getCurrentPlayer (
    Game game )
```

Retourne le joueur courant.

**Parameters**

<i>game</i>	une partie en cours
-------------	---------------------

**Returns**

renvoie le joueur actuellement en train de jouer

**2.3.2.5 getNotCurrentPlayer()**

```
Ensiie getNotCurrentPlayer (
    Game game )
```

Retourne le joueur qui n'est pas le joueur courant.

**Parameters**

<i>game</i>	une partie en cours
-------------	---------------------

**Returns**

renvoie le joueur actuellement pas en train de jouer

**2.3.2.6 getOtherPlayer()**

```
Ensiie getOtherPlayer (
    Game game,
    Ensiie player )
```

Renvoie le joueur qui n'est pas le joueur renseigné



## Parameters

<i>game</i>	le jeu dont on veut le joueur
<i>player</i>	Le joueur tel qu'on veut l'autre joueur

## Returns

L'autre joueur qui n'est pas player

## 2.3.2.7 getPlayer()

```
Ensiie getPlayer (
    Game game,
    int player_id )
```

getter de joueur

## Parameters

<i>game</i>	le jeu dont on veut le joueur
<i>player</i> ↔ <i>_id</i>	le numero du joueur a recuperer

## Returns

le joueur correspondant au numero player\_number

## 2.3.2.8 getRound()

```
int getRound (
    Game game )
```

getter de tour

## Parameters

<i>game</i>	le jeu dont on veut le tour
-------------	-----------------------------

## Returns

le tour correspondant au jeu game

### 2.3.2.9 isGameFinished()

```
int isGameFinished (
    Game game )
```

Est-ce la fin de la partie ?

#### Parameters

<i>game</i>	une partie en cours
-------------	---------------------

#### Returns

0 si la partie n'est pas finie, 1 si le joueur 1 a gagné, 2 si le joueur 2 a gagné et 3 s'il y a égalité

### 2.3.2.10 isRoundEven()

```
int isRoundEven (
    Game game )
```

Permet de savoir si le tour de jeu est un tour pair ou impair.

#### Parameters

<i>game</i>	jeu valide en cours
-------------	---------------------

#### Returns

1 si le tour de jeu est pair 0 si le tour de jeu est impair

#### Author

Alex Danduran-Lembezat

### 2.3.2.11 newRound()

```
Board newRound (
    Game game )
```

Début d'un nouveau tour : mise a jour les espaces disponibles pour les cartes Personnel.

#### Parameters

<i>game</i>	qui a été créé et initialisé
-------------	------------------------------

### 2.3.2.12 playCard()

```
void playCard (
    Card played_card,
    Game game )
```

Joue une carte de la main du joueur courant : déplace la carte `played_card` de la main du à son plateau ou a la défausse et en applique les effets, recalcul des PE.

#### Parameters

<i>played_card</i>	une carte de sa main
<i>game</i>	le jeu

#### Returns

1 si la carte de fin de phase est jouée, 0 sinon

### 2.3.2.13 playPhase()

```
void playPhase (
    Game game )
```

Exécute une phase de jeu pour le joueur courant.

#### Parameters

<i>game</i>	un game valide (bien initialisé)
-------------	----------------------------------

### 2.3.2.14 sendToGraveyard()

```
void sendToGraveyard (
    Game game,
    Ensie player,
    List_Card corresponding_list,
    Card card )
```

Envoie dans la défausse d'un joueur une carte donnée provenant d'une liste donnée et retire ses effets si elle en a.

#### Parameters

<i>game</i>	le jeu bien initialisé
<i>player</i>	Joueur bien initialisé
<i>corresponding_list</i>	la liste dans laquelle se trouve la carte
<i>card</i>	la carte que l'on cherche à envoyer à la défausse

**Author**

Alex Danduran–Lembezat

**2.3.2.15 switchCurrentPlayer()**

```
void switchCurrentPlayer (
    Game game )
```

Échange le caractère de joueur courant entre les 2 joueurs Si joueur1 est le joueur courant, alors joueur2 devient le joueur courant et joueur1 n'est plus joueur courant Si joueur2 est le joueur courant, alors joueur1 devient le joueur courant et joueur2 n'est plus joueur courant.

**Parameters**

<i>game</i>	jeu valide (contient 2 joueurs valides)
-------------	---

**2.4 joueur.h File Reference**

Gestion des joueurs.

```
#include "plateau.h"
```

**Typedefs**

- typedef enum [sust\\_or\\_dura](#) **sust\_or\_dura**
- typedef struct str\_ensiie \* [Ensiie](#)  
*Définition abstraite de la structure correspondant à un joueur.*

**Enumerations**

- enum [sust\\_or\\_dura](#) { **SUSTAINABLE**, **DURABILITY** }  
*Permet de rendre plus lisible l'appel aux fonctions `changeSustOrDura(...)`, `getChangedSustOrDura(...)` et `updateSustAndDura(...)`*

**Functions**

- [Ensiie](#) [createEnsiie](#) ()  
*Constructeur de ensiie.*
- void [destroyEnsiie](#) ([Ensiie](#) player)  
*Destructeur de ensiie.*
- void [initiateEnsiie](#) ([Ensiie](#) player)  
*Initialisation de ensiie : initialise la pioche.*
- void [drawEnsiie](#) ([Ensiie](#) player)

- Piocher une carte : le joueur pioche une carte.*
- void **addEleve** (id\_cardname id, [Ensiie](#) player)
- int **numberEleve** ([Ensiie](#) player)
- Calcul du nombre de cartes Eleve reçue en début de phase.*
- void **addAdditionalStudent** ([Ensiie](#) player, int amount)
- Augmente d'une quantité donnée le nombre de carte Eleve à piocher au début du tour suivant.*
- int **numberCard** ([Ensiie](#) player)
- Renvoie combien de cartes player doit piocher en debut de phase et réinitialise le compteur de cartes additionnelles à ajouter.*
- void **addAdditionalCard** ([Ensiie](#) player, int amount)
- Augmente d'une quantité donnée le nombre de carte à piocher au début du tour suivant.*
- int **isInHand** ([Ensiie](#) player, [Card](#) card)
- Teste si une carte est dans la main d'un joueur.*
- void **addEnergy** ([Ensiie](#) player, int energy)
- Ajoute des points d'énergie à un joueur.*
- void **removeEnergy** ([Ensiie](#) player, int energy)
- Retire des points d'énergie à un joueur.*
- int **getEnergy** ([Ensiie](#) player)
- Accesseur en lecture de l'énergie du joueur.*
- void **setEnergy** ([Ensiie](#) player, int energy)
- Mutateur du nombre de point d'énergie d'un joueur.*
- void **countEnergy** ([Ensiie](#) player, int round)
- Doit être appelé après la fonction addEleve(). Permet de compter le nombre de points d'énergie disponible au joueur au début de sa phase en fonction du tour courant et des cartes élève qu'il possède.*
- int **enoughEnergy** ([Ensiie](#) player, [Card](#) card)
- Renseigne si oui ou non un joueur a suffisamment d'énergie pour jouer une carte donnée.*
- char \* **getPlayerName** ([Ensiie](#) player)
- Récupère le nom du joueur spécifié*
- [List\\_Card](#) **getDeck** ([Ensiie](#) player)
- Récupère la liste des cartes du deck d'un joueur spécifié*
- [List\\_Card](#) **getHand** ([Ensiie](#) player)
- Récupère la liste des cartes en main d'un joueur spécifié*
- [List\\_Card](#) **getGraveyard** ([Ensiie](#) player)
- Récupère la liste des cartes correspondant à la défausse d'un joueur spécifié*
- int **getFiseNumber** ([Ensiie](#) player)
- Récupère le nombre de carte FISE placées sur le plateau du joueur spécifié*
- int **getFisaNumber** ([Ensiie](#) player)
- Récupère le nombre de carte FISA placées sur le plateau du joueur spécifié*
- int **getPlayerSustainablePoints** ([Ensiie](#) player)
- Accesseur en lecture des points de développement durable d'un joueur.*
- void **addPlayerSustainablePoints** ([Ensiie](#) player, int points)
- Permet d'ajouter des points de DD à un joueur.*
- int **getAdditionalSustainablePoints** ([Ensiie](#) player)
- Accesseur en lecture des points supplémentaires de développement durable d'un joueur.*
- void **addAdditionalSustainablePoints** ([Ensiie](#) player, int points)
- Permet d'ajouter des points de DD supplémentaires à la fin du tour à un joueur.*
- void **removeAdditionalSustainablePoints** ([Ensiie](#) player, int points)
- Permet de retirer des points de DD parmi les points supplémentaires à la fin du tour à un joueur.*
- [Board](#) **getBoard** ([Ensiie](#) player)
- Accesseur du plateau d'un joueur.*
- int **isCurrentPlayer** ([Ensiie](#) player)

- Permet de savoir si oui ou non c'est au tour du joueur en question de jouer.*

  - void `setIsCurrentPlayer` (`Ensiie` player, int valeur)

*Mutateur de la variable `is_current_player` qui renseigne si le joueur en question est le joueur courant.*
- void `changeSustOrDura` (`Ensiie` player, `sust_or_dura` dura\_or\_sust, `id_cardname` id, int value)

*Permet de changer de manière persistante la durabilité ou le développement des cartes FISE ou FISA pour un joueur.*
- int `getChangedSustOrDura` (`Ensiie` player, `sust_or_dura` dura\_or\_sust, `id_cardname` id)

*Permet de récupérer la quantité de modification des statistiques des cartes en fonction de si on veut la durabilité ou le développement pour un FISE ou un FISA.*
- void `updateSustAndDura` (`Ensiie` player)

*Permet de mettre à jour la durabilité et le développement de toutes les cartes Eleve du joueur donné*
- void `upgradeCardSustainable` (`Ensiie` player)

*Permet de signifier l'amélioration des cartes Elèves pour qu'elles aient 1 point de développement en plus dès leur arrivée sur le plateau.*
- int `getUpgradedCardSustainable` (`Ensiie` player)

*Accesseur de la variable `upgraded_cards_durability` qui renseigne si les nouvelles cartes élèves doivent étre améliorées.*
- void `upgradeCardDurability` (`Ensiie` player)

*Permet de signifier l'amélioration des cartes Elèves pour qu'elles aient 1 point de durabilité en plus dès leur arrivée sur le plateau.*
- int `getUpgradedCardDurability` (`Ensiie` player)

*Accesseur de la variable `upgraded_cards_sustainable` qui renseigne si les nouvelles cartes élèves doivent étre améliorées.*
- int `getSumCardSustainable` (`Ensiie` player, int is\_round\_even)

*Renvoie la somme des points de développement des cartes Elève.*
- int `getSumCardDurability` (`Ensiie` player, int is\_round\_even)

*Renvoie la somme des points de durabilité des cartes Elève.*
- void `playDistanciation` (`Ensiie` player)

*Retire la moitié des cartes FISE et des cartes FISA du plateau.*
- void `setPlayerName` (`Ensiie` player, char \*name)

*Change le nom du joueur spécifié*

## 2.4.1 Detailed Description

Gestion des joueurs.

Author

Alex Danduran–Lembezat

## 2.4.2 Function Documentation

### 2.4.2.1 addAdditionalCard()

```
void addAdditionalCard (
    Ensiie player,
    int amount )
```

Augmente d'une quantité donnée le nombre de carte à piocher au début du tour suivant.

## Parameters

<i>player</i>	une ensiie valide
<i>amount</i>	un entier positif

## 2.4.2.2 addAdditionalStudent()

```
void addAdditionalStudent (  
    Ensiie player,  
    int amount )
```

Augmente d'une quantité donnée le nombre de carte Eleve à piocher au début du tour suivant.

## Parameters

<i>player</i>	une ensiie valide
<i>amount</i>	un entier positif

## 2.4.2.3 addAdditionalSustainablePoints()

```
void addAdditionalSustainablePoints (  
    Ensiie player,  
    int points )
```

Permet d'ajouter des points de DD supplémentaires à la fin du tour à un joueur.

## Parameters

<i>player</i>	un joueur bien initialisé
<i>points</i>	le nombre de points à ajouter

## 2.4.2.4 addEnergy()

```
void addEnergy (  
    Ensiie player,  
    int energy )
```

Ajoute des points d'énergie à un joueur.

## Parameters

<i>player</i>	un joueur bien initialisé
<i>energy</i>	la quantité d'énergie à ajouter

**Author**

Alex Danduran–Lembezat

**2.4.2.5 addPlayerSustainablePoints()**

```
void addPlayerSustainablePoints (
    Ensie player,
    int points )
```

Permet d'ajouter des points de DD à un joueur.

**Parameters**

<i>player</i>	un joueur bien initialisé
<i>points</i>	le nombre de points à ajouter

**2.4.2.6 changeSustOrDura()**

```
void changeSustOrDura (
    Ensie player,
    sust_or_dura dura_or_sust,
    id_cardname id,
    int value )
```

Permet de changer de manière persistante la durabilité ou le développement des cartes FISE ou FISA pour un joueur.

**Parameters**

<i>player</i>	Joueur bien initialisé dont on veut altérer les statistiques des cartes
<i>dura_or_sust</i>	DURABILITY si on veut changer la durabilité et SUSTAINABLE si on veut changer le développement
<i>id</i>	FISE si on veut modifier les statistiques des cartes FISE et FISA pour celles des cartes FISA
<i>value</i>	(Peut être négatif). La valeur donnant de combien on veut modifier les statistiques

**2.4.2.7 countEnergy()**

```
void countEnergy (
    Ensie player,
    int round )
```

Doit être appelé après la fonction addEleve(). Permet de compter le nombre de points d'énergie disponible au joueur au début de sa phase en fonction du tour courant et des cartes élève qu'il possède.



## Parameters

<i>player</i>	un joueur bien initialisé
<i>round</i>	le numéro du tour courant

## Author

Alex Danduran–Lembezat

## 2.4.2.8 createEnsiie()

```
Ensiie createEnsiie ( )
```

Constructeur de ensiie.

## Returns

Une ensiie en situation initiale : 0 points de developpement durable, un deck vide, une main vide, un plateau vide, 0 PE, une defausse vide.

## 2.4.2.9 destroyEnsiie()

```
void destroyEnsiie (
    Ensiie player )
```

Destructeur de ensiie.

## Parameters

<i>player</i>	une ensiie a ete creee
---------------	------------------------

## Author

Sirine HAMDANA

## 2.4.2.10 drawEnsiie()

```
void drawEnsiie (
    Ensiie player )
```

Piocher une carte : le joueur pioche une carte.

**Parameters**

<i>player</i>	une ensiie valide
---------------	-------------------

**2.4.2.11 enoughEnergy()**

```
int enoughEnergy (
    Ensie player,
    Card card )
```

Renseigne si oui ou non un joueur a suffisamment d'énergie pour jouer une carte donnée.

**Parameters**

<i>player</i>	un joueur bien initialisé
<i>card</i>	la carte qu'il souhaite jouer

**Author**

Alex Danduran–Lembezat

**2.4.2.12 getAdditionalSustainablePoints()**

```
int getAdditionalSustainablePoints (
    Ensie player )
```

Accesseur en lecture des points supplémentaires de développement durable d'un joueur.

**Parameters**

<i>player</i>	un joueur bien initialisé
---------------	---------------------------

**Returns**

Les points de développement durable supplémentaires du joueur

**2.4.2.13 getBoard()**

```
Board getBoard (
    Ensie player )
```

Accesseur du plateau d'un joueur.

## Parameters

<i>player</i>	Le joueur bien initialisé dont on veut le plateau
---------------	---

## Returns

Renvoie le plateau du joueur

## 2.4.2.14 getChangedSustOrDura()

```
int getChangedSustOrDura (
    Ensie player,
    sust_or_dura dura_or_sust,
    id_cardname id )
```

Permet de récupérer la quantité de modification des statistiques des cartes en fonction de si on veut la durabilité ou le développement pour un FISE ou un FISA.

## Parameters

<i>player</i>	Joueur bien initialisé dont on veut récupérer les statistiques des cartes
<i>dura_or_sust</i>	DURABILITY si on veut récupérer la durabilité et SUSTAINABLE si on veut récupérer le développement
<i>id</i>	FISE si on veut récupérer les statistiques des cartes FISE et FISA pour celles des cartes FISA

## Returns

(Peut être négatif). Retourne la quantité de modification des statistiques

## 2.4.2.15 getDeck()

```
List_Card getDeck (
    Ensie player )
```

Récupère la liste des cartes du deck d'un joueur spécifié

## Parameters

<i>joueur</i>	joueur dont on veut le deck
---------------	-----------------------------

## Returns

Le deck du joueur

#### 2.4.2.16 getEnergy()

```
int getEnergy (
    Ensie player )
```

Accesseur en lecture de l'énergie du joueur.

##### Parameters

<i>player</i>	un joueur bien initialisé
---------------	---------------------------

##### Returns

L'énergie du joueur en question

##### Author

Alex Danduran-Lembezat

#### 2.4.2.17 getFisaNumber()

```
int getFisaNumber (
    Ensie player )
```

Récupère le nombre de carte FISA placées sur le plateau du joueur spécifié

##### Parameters

<i>player</i>	joueur dont on veut le nombre de carte FISA
---------------	---

##### Returns

Le nombre de carte FISA placées sur le plateau d'un joueur

#### 2.4.2.18 getFiseNumber()

```
int getFiseNumber (
    Ensie player )
```

Récupère le nombre de carte FISE placées sur le plateau du joueur spécifié

##### Parameters

<i>player</i>	joueur dont on veut le nombre de carte FISE
---------------	---

**Returns**

Le nombre de carte FISE placées sur le plateau d'un joueur

**2.4.2.19 getGraveyard()**

```
List_Card getGraveyard (
    Ensiee player )
```

Récupère la liste des cartes correspondant à la défausse d'un joueur spécifié

**Parameters**

<i>player</i>	joueur dont on veut la défausse
---------------	---------------------------------

**Returns**

La défausse du joueur

**2.4.2.20 getHand()**

```
List_Card getHand (
    Ensiee player )
```

Récupère la liste des cartes en main d'un joueur spécifié

**Parameters**

<i>player</i>	joueur dont on veut la main
---------------	-----------------------------

**Returns**

La main du joueur

**2.4.2.21 getPlayerName()**

```
char* getPlayerName (
    Ensiee player )
```

Récupère le nom du joueur spécifié

**Parameters**

<i>joueur</i>	joueur dont on veut le nom
---------------	----------------------------

**Returns**

Le nom du joueur

**2.4.2.22 getPlayerSustainablePoints()**

```
int getPlayerSustainablePoints (
    Ensie player )
```

Accesseur en lecture des points de développement durable d'un joueur.

**Parameters**

<i>player</i>	un joueur bien initialisé
---------------	---------------------------

**Returns**

Les points de développement durable du joueur

**2.4.2.23 getSumCardDurability()**

```
int getSumCardDurability (
    Ensie player,
    int is_round_even )
```

Renvoie la somme des points de durabilité des cartes Elève.

**Parameters**

<i>player</i>	Joueur bien initialisé
<i>is_round_even</i>	1 si le tour est pair, 0 s'il est impair

**Returns**

Somme des points de durabilité des cartes Elèves FISE si tour impair et FISE + FISA si tour pair

#### 2.4.2.24 getSumCardSustainable()

```
int getSumCardSustainable (
    Ensie player,
    int is_round_even )
```

Renvoie la somme des points de développement des cartes Elève.

##### Parameters

<i>player</i>	Joueur bien initialisé
<i>is_round_even</i>	1 si le tour est pair, 0 s'il est impair

##### Returns

Somme des points de développement des cartes Elèves FISE si tour impair et FISE + FISA si tour pair

#### 2.4.2.25 getUpgradedCardDurability()

```
int getUpgradedCardDurability (
    Ensie player )
```

Accesseur de la variable `upgraded_cards_sustainable` qui renseigne si les nouvelles cartes élèves doivent être améliorées.

##### Parameters

<i>player</i>	Joueur bien initialisé
---------------	------------------------

##### Returns

1 si la carte doit être améliorée, 0 sinon

#### 2.4.2.26 getUpgradedCardSustainable()

```
int getUpgradedCardSustainable (
    Ensie player )
```

Accesseur de la variable `upgraded_cards_durability` qui renseigne si les nouvelles cartes élèves doivent être améliorées.

##### Parameters

<i>player</i>	Joueur bien initialisé
---------------	------------------------

**Returns**

1 si la carte doit être améliorée, 0 sinon

**2.4.2.27 initiateEnsiie()**

```
void initiateEnsiie (
    Ensiie player )
```

Initialisation de ensiie : initialise la pioche.

**Parameters**

<i>player</i>	une ensiie a ete creee
---------------	------------------------

**2.4.2.28 isCurrentPlayer()**

```
int isCurrentPlayer (
    Ensiie player )
```

Permet de savoir si oui ou non c'est au tour du joueur en question de jouer.

**Parameters**

<i>player</i>	Joueur bien initialisé
---------------	------------------------

**Returns**

Renvoie 1 si le joueur est le joueur en cours et 0 sinon

**2.4.2.29 isInHand()**

```
int isInHand (
    Ensiie player,
    Card card )
```

Teste si une carte est dans la main d'un joueur.

**Parameters**

<i>player</i>	le joueur courant
---------------	-------------------



**Returns**

1 si card est dans la main de player et 0 sinon

**2.4.2.30 numberCard()**

```
int numberCard (
    Ensie player )
```

Renvoie combien de cartes player doit piocher en debut de phase et réinitialise le compteur de cartes additionnelles à ajouter.

**Parameters**

<i>player</i>	une ensie valide
---------------	------------------

**Returns**

le nombre de cartes que doit piocher player au debut de sa phase

**2.4.2.31 numberEleve()**

```
int numberEleve (
    Ensie player )
```

Calcul du nombre de cartes Eleve reçue en début de phase.

**Parameters**

<i>player</i>	le joueur courant
---------------	-------------------

**Returns**

le nombre de cartes Eleve que player doit recevoir au debut de sa phase

**2.4.2.32 playDistanciation()**

```
void playDistanciation (
    Ensie player )
```

Retire la moitié des cartes FISE et des cartes FISA du plateau.

**Parameters**

<i>player</i>	Joueur bien initialisé
---------------	------------------------

**2.4.2.33 removeAdditionalSustainablePoints()**

```
void removeAdditionalSustainablePoints (
    Ensie player,
    int points )
```

Permet de retirer des points de DD parmi les points supplémentaires à la fin du tour à un joueur.

**Parameters**

<i>player</i>	un joueur bien initialisé
<i>points</i>	le nombre de points à retirer

**2.4.2.34 removeEnergy()**

```
void removeEnergy (
    Ensie player,
    int energy )
```

Retire des points d'énergie à un joueur.

**Parameters**

<i>player</i>	un joueur bien initialisé
<i>energy</i>	la quantité d'énergie à retirer

**Author**

Alex Danduran–Lembezat

**2.4.2.35 setEnergy()**

```
void setEnergy (
    Ensie player,
    int energy )
```

Mutateur du nombre de point d'énergie d'un joueur.

## Parameters

<i>player</i>	un joueur bien initialisé
<i>energy</i>	la quantité d'énergie à mettre

## Author

Alex Danduran–Lembezat

## 2.4.2.36 setIsCurrentPlayer()

```
void setIsCurrentPlayer (
    Ensie player,
    int valeur )
```

Mutateur de la variable `is_current_player` qui renseigne si le joueur en question est le joueur courant.

## Parameters

<i>player</i>	Joueur bien initialisé
---------------	------------------------

## 2.4.2.37 setPlayerName()

```
void setPlayerName (
    Ensie player,
    char * name )
```

Change le nom du joueur spécifié

## Parameters

<i>joueur</i>	joueur dont on veut changer le nom
<i>name</i>	le nom à donner au joueur

## 2.4.2.38 updateSustAndDura()

```
void updateSustAndDura (
    Ensie player )
```

Permet de mettre à jour la durabilité et le développement de toutes les cartes Eleve du joueur donné

## Parameters

<i>player</i>	Joueur bien initialisé dont on veut mettre à jour les statistiques de carte
---------------	---

## 2.4.2.39 upgradeCardDurability()

```
void upgradeCardDurability (
    Ensie player )
```

Permet de signifier l'amélioration des cartes Elèves pour qu'elles aient 1 point de durabilité en plus dès leur arrivée sur le plateau.

## Parameters

<i>player</i>	Joueur bien initialisé
---------------	------------------------

## 2.4.2.40 upgradeCardSustainable()

```
void upgradeCardSustainable (
    Ensie player )
```

Permet de signifier l'amélioration des cartes Elèves pour qu'elles aient 1 point de développement en plus dès leur arrivée sur le plateau.

## Parameters

<i>player</i>	Joueur bien initialisé
---------------	------------------------

## 2.5 plateau.h File Reference

Gestion du plateau.

```
#include "structure.h"
```

## Typedefs

- typedef struct str\_board \* [Board](#)

*Type plateau, chaque joueur a son propre plateau Qui est composé des cartes Eleve et des cartes Personnel en jeu  
Définition abstraite de la structure correspondant à un plateau.*

## Functions

- [Board](#) [createBoard](#) ( )  
*Constructeur de plateau : cree un plateau et l'initialise comme vide avec 1 emplacement pour une carte Personnel.*
- void [destroyBoard](#) ([Board](#) b)  
*Destructeur de plateau.*
- void [updateUnlockedSpots](#) ([Board](#) board, int round)  
*Met à jour le nombre d'emplacement de cartes Personnels disponibles en fonction du nombre de tour écoulé*
- [List\\_Card](#) [getStudentPool](#) ([Board](#) board)  
*Renseigne l'ensemble des cartes Elève en place sur le plateau d'un joueur donné*
- [List\\_Card](#) [getStaffPool](#) ([Board](#) board)  
*Renseigne l'ensemble des cartes Personnels en place sur le plateau.*
- unsigned int [getUnlockedSpotNumber](#) ([Board](#) board)  
*Renvoie le nombre d'emplacement Personnels débloqué (varie en 1 et 3)*
- int [getAvailableSpotNumber](#) ([Board](#) board)  
*Renvoie le nombre d'emplacement Personnels disponible.*
- int [isStaffSpecialised](#) ([Board](#) board, [card\\_specialise](#) specialisation)  
*Renvoie si oui ou non il existe au moins une carte Personnel sur le plateau avec une spécialisation donnée.*

### 2.5.1 Detailed Description

Gestion du plateau.

#### Author

Alex Danduran–Lembezat

### 2.5.2 Function Documentation

#### 2.5.2.1 [createBoard](#)()

```
Board createBoard ( )
```

Constructeur de plateau : cree un plateau et l'initialise comme vide avec 1 emplacement pour une carte Personnel.

#### Returns

Le plateau cree

#### 2.5.2.2 [destroyBoard](#)()

```
void destroyBoard (
    Board b )
```

Destructeur de plateau.

**Parameters**

<i>b</i>	Un plateau valide
----------	-------------------

**2.5.2.3 getAvailableSpotNumber()**

```
int getAvailableSpotNumber (
    Board board )
```

Renvoie le nombre d'emplacement Personnels disponible.

**Parameters**

<i>board</i>	Plateau de jeu d'un joueur bien initialisé
--------------	--

**Returns**

Nombre d'emplacement Personnels disponible

**2.5.2.4 getStaffPool()**

```
List_Card getStaffPool (
    Board board )
```

Renseigne l'ensemble des cartes Personnels en place sur le plateau.

**Parameters**

<i>board</i>	Le plateau d'un joueur bien initialisé
--------------	--

**Returns**

La liste des cartes de type Personnels d'un joueur posées sur son plateau

**2.5.2.5 getStudentPool()**

```
List_Card getStudentPool (
    Board board )
```

Renseigne l'ensemble des cartes Elève en place sur le plateau d'un joueur donné

#### Parameters

<i>board</i>	Le plateau d'un joueur bien initialisé
--------------	--

#### Returns

La liste des cartes de type Elève d'un joueur posées sur son plateau

#### 2.5.2.6 getUnlockedSpotNumber()

```
unsigned int getUnlockedSpotNumber (  
    Board board )
```

Renvoie le nombre d'emplacement Personnels débloquent (varie en 1 et 3)

#### Parameters

<i>board</i>	Plateau de jeu d'un joueur bien initialisé
--------------	--

#### Returns

Nombre d'emplacement Personnels débloquent

#### 2.5.2.7 isStaffSpecialised()

```
int isStaffSpecialised (  
    Board board,  
    card_specialise specialisation )
```

Renvoie si oui ou non il existe au moins une carte Personnel sur le plateau avec une spécialisation donnée.

#### Parameters

<i>board</i>	Plateau de jeu d'un joueur bien initialisé
<i>specialisation</i>	Un spécialisation valide

#### Returns

1 si il existe une carte Personnel posée avec la spécialisation donnée, 0 sinon

### 2.5.2.8 updateUnlockedSpots()

```
void updateUnlockedSpots (
    Board board,
    int round )
```

Met à jour le nombre d'emplacement de cartes Personnels disponibles en fonction du nombre de tour écoulé

#### Parameters

<i>board</i>	Plateau de jeu d'un joueur bien initialisé
<i>round</i>	Numéro du actuelle du tour de jeu

## 2.6 structure.h File Reference

Gestion des listes de cartes.

```
#include "carte.h"
```

### Typedefs

- typedef struct str\_list\_card \* [List\\_Card](#)  
*Type liste de cartes, permettant d'implémenter la main, la pioche, la défausse, les cartes élèves et les cartes personnels en jeu Définition abstraite de la structure correspondant à une liste de cartes.*

### Functions

- [List\\_Card](#) [createListCard](#) ()  
*Constructeur de liste de cartes.*
- void [destroyListCard](#) ([List\\_Card](#) list\_card)  
*Destructeur de liste de cartes.*
- void [destroyElements](#) ([Card](#) \*list, int size)  
*Permet de détruire tous les éléments d'une liste itérable.*
- [Card](#) [removeElement](#) ([List\\_Card](#) list\_card, int index)  
*Supprime une carte d'une List\_Card.*
- void [addElement](#) ([List\\_Card](#) list\_card, [Card](#) card)  
*Ajoute une carte à la fin d'une liste de cartes.*
- void [addAndSort](#) ([List\\_Card](#) list\_card, [Card](#) card)  
*Ajoute une carte à une liste et trie la liste.*
- void [shuffle](#) ([List\\_Card](#) list\_card)
- int [exists](#) ([List\\_Card](#) list\_card, [Card](#) card)
- void [displaceElement](#) ([Card](#) card, [List\\_Card](#) original\_list, [List\\_Card](#) destination\_list)  
*Déplace une carte donnée depuis sa liste d'origine vers une liste de destination.*
- [Card](#) \* [getList](#) ([List\\_Card](#) list\_card)  
*Accesseur en lecture de la liste de cartes.*
- int [getSize](#) ([List\\_Card](#) list\_card)  
*Accesseur en lecture de la taille de la liste de cartes.*



## 2.6.1 Detailed Description

Gestion des listes de cartes.

Author

Sirine Hamdana

## 2.6.2 Function Documentation

### 2.6.2.1 addAndSort()

```
void addAndSort (
    List_Card list_card,
    Card card )
```

Ajoute une carte à une liste et trie la liste.

Parameters

<i>card</i>	Une carte valide
<i>list_card</i>	Une liste de cartes valide

### 2.6.2.2 addElement()

```
void addElement (
    List_Card list_card,
    Card card )
```

Ajoute une carte à la fin d'une liste de cartes.

Parameters

<i>card</i>	Une carte valide
<i>list_card</i>	Une liste de cartes valide

### 2.6.2.3 createListCard()

```
List_Card createListCard ( )
```

Constructeur de liste de cartes.

## Returns

Cree une liste de cartes et la renvoie

### 2.6.2.4 destroyElements()

```
void destroyElements (
    Card * list,
    int size )
```

Permet de détruire tous les éléments d'une liste itérable.

#### Parameters

<i>list</i>	le pointeur vers le premier élément de la liste de carte
<i>size</i>	l'entier correspondant à la taille de cette liste

### 2.6.2.5 destroyListCard()

```
void destroyListCard (
    List_Card list_card )
```

Destructeur de liste de cartes.

#### Parameters

<i>list_card</i>	une liste de cartes valide à détruire
------------------	---------------------------------------

### 2.6.2.6 displaceElement()

```
void displaceElement (
    Card card,
    List_Card original_list,
    List_Card destination_list )
```

Déplace une carte donnée depuis sa liste d'origine vers une liste de destination.

#### Parameters

<i>card</i>	Une carte valide
<i>original_list</i>	Une liste de cartes valide dans laquelle se trouve la carte
<i>destination_list</i>	Une liste de cartes valide dans laquelle on veut déplacer la carte

### 2.6.2.7 getList()

```
List_Card getList (
    List_Card list_card )
```

Accesseur en lecture de la liste de cartes.

#### Parameters

<i>list_card</i>	Une liste de cartes valide
------------------	----------------------------

#### Returns

La liste de cartes

### 2.6.2.8 getSize()

```
int getSize (
    List_Card list_card )
```

Accesseur en lecture de la taille de la liste de cartes.

#### Parameters

<i>list_card</i>	Une liste de cartes valide
------------------	----------------------------

#### Returns

La taille de la liste de cartes

### 2.6.2.9 removeElement()

```
Card removeElement (
    List_Card list_card,
    int index )
```

Supprime une carte d'une List\_Card.

#### Parameters

<i>list_card</i>	Une liste de cartes valide
<i>index</i>	Un entier

**Returns**

La carte situee a l'indice index de list\_card, Si index est négatif ou plus grand que la taille de la liste, renvoie NULL

## 2.7 utils.h File Reference

Fonctions utilitaires.

**Functions**

- int `max` (int a, int b)  
*Renvoie le maximum de 2 entiers.*

### 2.7.1 Detailed Description

Fonctions utilitaires.

**Author**

Alex Danduran–Lembezat

### 2.7.2 Function Documentation

#### 2.7.2.1 max()

```
int max (  
    int a,  
    int b )
```

Renvoie le maximum de 2 entiers.

**Parameters**

<i>a</i>	un entier quelconque
<i>b</i>	un entier quelconque

**Returns**

le plus grand des deux entiers passés en argument

# Index

- addAdditionalCard
  - [joueur.h, 24](#)
- addAdditionalStudent
  - [joueur.h, 25](#)
- addAdditionalSustainablePoints
  - [joueur.h, 25](#)
- addAndSort
  - [structure.h, 43](#)
- addCardSustainable
  - [carte.h, 5](#)
- addDurability
  - [carte.h, 5](#)
- addElement
  - [structure.h, 43](#)
- addEnergy
  - [joueur.h, 25](#)
- addPlayerSustainablePoints
  - [joueur.h, 26](#)
- askCard
  - [interface.h, 13](#)
- carte.h, [3](#)
  - [addCardSustainable, 5](#)
  - [addDurability, 5](#)
  - [copyCard, 6](#)
  - [destroyCard, 6](#)
  - [getCardId, 6](#)
  - [getCardSustainable, 7](#)
  - [getDurability, 7](#)
  - [getEffects, 7](#)
  - [getName, 8](#)
  - [getPE, 8](#)
  - [getSizeEffects, 8](#)
  - [getSpecialisation, 10](#)
  - [getType, 10](#)
  - [isEndPhaseCard, 10](#)
  - [removeCardSustainable, 11](#)
  - [removeDurability, 11](#)
  - [setCardSustainable, 11](#)
  - [setDurability, 12](#)
  - [setSpecialisation, 12](#)
- changeSustOrDura
  - [joueur.h, 26](#)
- copyCard
  - [carte.h, 6](#)
- countEnergy
  - [joueur.h, 26](#)
- createBoard
  - [plateau.h, 39](#)
- createEnsiie
  - [joueur.h, 27](#)
- createGame
  - [jeu.h, 17](#)
- createListCard
  - [structure.h, 43](#)
- destroyBoard
  - [plateau.h, 39](#)
- destroyCard
  - [carte.h, 6](#)
- destroyElements
  - [structure.h, 44](#)
- destroyEnsiie
  - [joueur.h, 27](#)
- destroyGame
  - [jeu.h, 17](#)
- destroyListCard
  - [structure.h, 44](#)
- displaceElement
  - [structure.h, 44](#)
- drawEnsiie
  - [joueur.h, 27](#)
- endRound
  - [jeu.h, 17](#)
- enoughEnergy
  - [joueur.h, 28](#)
- getAdditionalSustainablePoints
  - [joueur.h, 28](#)
- getAvailableSpotNumber
  - [plateau.h, 40](#)
- getBoard
  - [joueur.h, 28](#)
- getCardId
  - [carte.h, 6](#)
- getCardSustainable
  - [carte.h, 7](#)
- getChangedSustOrDura
  - [joueur.h, 29](#)
- getCurrentPlayer
  - [jeu.h, 18](#)
- getDeck
  - [joueur.h, 29](#)
- getDurability
  - [carte.h, 7](#)
- getEffects
  - [carte.h, 7](#)
- getEnergy
  - [joueur.h, 29](#)

- getFisaNumber
  - joueur.h, [30](#)
- getFiseNumber
  - joueur.h, [30](#)
- getGraveyard
  - joueur.h, [31](#)
- getHand
  - joueur.h, [31](#)
- getList
  - structure.h, [45](#)
- getName
  - carte.h, [8](#)
- getNotCurrentPlayer
  - jeu.h, [18](#)
- getOtherPlayer
  - jeu.h, [18](#)
- getPE
  - carte.h, [8](#)
- getPlayer
  - jeu.h, [19](#)
- getPlayerName
  - joueur.h, [31](#)
- getPlayerSustainablePoints
  - joueur.h, [32](#)
- getRound
  - jeu.h, [19](#)
- getSize
  - structure.h, [45](#)
- getSizeEffects
  - carte.h, [8](#)
- getSpecialisation
  - carte.h, [10](#)
- getStaffPool
  - plateau.h, [40](#)
- getStudentPool
  - plateau.h, [40](#)
- getSumCardDurability
  - joueur.h, [32](#)
- getSumCardSustainable
  - joueur.h, [32](#)
- getType
  - carte.h, [10](#)
- getUnlockedSpotNumber
  - plateau.h, [41](#)
- getUpgradedCardDurability
  - joueur.h, [33](#)
- getUpgradedCardSustainable
  - joueur.h, [33](#)
- initiateEnsiie
  - joueur.h, [34](#)
- interface.h, [13](#)
  - askCard, [14](#)
  - printNewGame, [14](#)
  - printNewPhase, [14](#)
  - printNewRound, [14](#)
  - printStateOfGame, [15](#)
  - printStudentPool, [15](#)
  - whichStudent, [15](#)
- isCurrentPlayer
  - joueur.h, [34](#)
- isEndPhaseCard
  - carte.h, [10](#)
- isGameFinished
  - jeu.h, [19](#)
- isInHand
  - joueur.h, [34](#)
- isRoundEven
  - jeu.h, [20](#)
- isStaffSpecialised
  - plateau.h, [41](#)
- jeu.h, [16](#)
  - createGame, [17](#)
  - destroyGame, [17](#)
  - endRound, [17](#)
  - getCurrentPlayer, [18](#)
  - getNotCurrentPlayer, [18](#)
  - getOtherPlayer, [18](#)
  - getPlayer, [19](#)
  - getRound, [19](#)
  - isGameFinished, [19](#)
  - isRoundEven, [20](#)
  - newRound, [20](#)
  - playCard, [21](#)
  - playPhase, [21](#)
  - sendToGraveyard, [21](#)
  - switchCurrentPlayer, [22](#)
- joueur.h, [22](#)
  - addAdditionalCard, [24](#)
  - addAdditionalStudent, [25](#)
  - addAdditionalSustainablePoints, [25](#)
  - addEnergy, [25](#)
  - addPlayerSustainablePoints, [26](#)
  - changeSustOrDura, [26](#)
  - countEnergy, [26](#)
  - createEnsiie, [27](#)
  - destroyEnsiie, [27](#)
  - drawEnsiie, [27](#)
  - enoughEnergy, [28](#)
  - getAdditionalSustainablePoints, [28](#)
  - getBoard, [28](#)
  - getChangedSustOrDura, [29](#)
  - getDeck, [29](#)
  - getEnergy, [29](#)
  - getFisaNumber, [30](#)
  - getFiseNumber, [30](#)
  - getGraveyard, [31](#)
  - getHand, [31](#)
  - getPlayerName, [31](#)
  - getPlayerSustainablePoints, [32](#)
  - getSumCardDurability, [32](#)
  - getSumCardSustainable, [32](#)
  - getUpgradedCardDurability, [33](#)
  - getUpgradedCardSustainable, [33](#)
  - initiateEnsiie, [34](#)
  - isCurrentPlayer, [34](#)
  - isInHand, [34](#)

- numberCard, 35
- numberEleve, 35
- playDistanciation, 35
- removeAdditionalSustainablePoints, 36
- removeEnergy, 36
- setEnergy, 36
- setIsCurrentPlayer, 37
- setPlayerName, 37
- updateSustAndDura, 37
- upgradeCardDurability, 38
- upgradeCardSustainable, 38

max

- utils.h, 46

newRound

- jeu.h, 20

numberCard

- joueur.h, 35

numberEleve

- joueur.h, 35

plateau.h, 38

- createBoard, 39
- destroyBoard, 39
- getAvailableSpotNumber, 40
- getStaffPool, 40
- getStudentPool, 40
- getUnlockedSpotNumber, 41
- isStaffSpecialised, 41
- updateUnlockedSpots, 41

playCard

- jeu.h, 21

playDistanciation

- joueur.h, 35

playPhase

- jeu.h, 21

printNewGame

- interface.h, 14

printNewPhase

- interface.h, 14

printNewRound

- interface.h, 14

printStateOfGame

- interface.h, 15

printStudentPool

- interface.h, 15

removeAdditionalSustainablePoints

- joueur.h, 36

removeCardSustainable

- carte.h, 11

removeDurability

- carte.h, 11

removeElement

- structure.h, 45

removeEnergy

- joueur.h, 36

sendToGraveyard

- jeu.h, 21

setCardSustainable

- carte.h, 11

setDurability

- carte.h, 12

setEnergy

- joueur.h, 36

setIsCurrentPlayer

- joueur.h, 37

setPlayerName

- joueur.h, 37

setSpecialisation

- carte.h, 12

structure.h, 42

- addAndSort, 43
- addElement, 43
- createListCard, 43
- destroyElements, 44
- destroyListCard, 44
- displaceElement, 44
- getList, 45
- getSize, 45
- removeElement, 45

switchCurrentPlayer

- jeu.h, 22

updateSustAndDura

- joueur.h, 37

updateUnlockedSpots

- plateau.h, 41

upgradeCardDurability

- joueur.h, 38

upgradeCardSustainable

- joueur.h, 38

utils.h, 46

- max, 46

whichStudent

- interface.h, 15