Durableverse - Les Trilobytes

Generated by Doxygen 1.8.13

Contents

Chapter 1

File Index

1.1 File List

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

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

card	une carte du jeu
value	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

card	une carte du jeu
value	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

source	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

card	Une carte du jeu

2.1.2.5 getCardId()

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

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

Parameters

card	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

card	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

card	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

card	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

card	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

card	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

card	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

card	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

card	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

card	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

card	une carte du jeu
value	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

card	une carte du jeu
value	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

card	une carte du jeu
value	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

card	une carte du jeu
value	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

card	Une carte du jeu valide
special	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 printStateOfGame (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 egalement 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 débuter 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 débuter 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

player	un joueur valide (possèdant une liste de carte à jouer (une main))

Returns

la carte que le joueur désire jouer

2.2.2.2 printNewGame()

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

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

Parameters

player1	le premier joueur du jeu
player2	le second joueur du jeu

2.2.2.3 printNewPhase()

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

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

Parameters

player1	le premier joueur du jeu
player2	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 débuter et indique le numéro du tour ainsi que d'autres informations utiles (par exemple la présence ou non d'un nouvel élément "Personnel")

Parameters

round	un entier

2.2.2.5 printStateOfGame()

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

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

Parameters

player1	un joueur valide
player2	un joueur valide
round	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 egalement les cartes FISA comme non jouables en tour pair.

Parameters

pool	une liste de cartes (card) : la liste de cartes élèves
round	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 liberee.
- void newRound (Game game)

  Début d'un nouveau tour : mise a jour les 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 a 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

cree 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

game	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

game	une partie en cours

2.3.2.4 getCurrentPlayer()

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

Retourne le joueur courant.

Parameters

game	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

game	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

game	le jeu dont on veut le joueur
player	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

game	le jeu dont on veut le joueur
player_id	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

game	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

game	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

game	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

game	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

played_card	une carte de sa main
game	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

game	un game valide (bien initialisé)

2.3.2.14 sendToGraveyard()

```
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.

Parameters

game	le jeu bien initialisé
player	Joueur bien initialisé
corresponding_list	la liste dans laquelle se trouve la carte
card	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

game	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 update↩ SustAndDura(...)

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 dispobible 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 suffisemment 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 parmis 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 êtres 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 êtres 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

player	une ensiie valide
amount	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

player	une ensiie valide
amount	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

player	un joueur bien initialisé
points	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

player	un joueur bien initialisé
energy	la quantité d'énergie à ajouter

Author

Alex Danduran–Lembezat

2.4.2.5 addPlayerSustainablePoints()

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

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

Parameters

player	un joueur bien initialisé
points	le nombre de points à ajouter

2.4.2.6 changeSustOrDura()

```
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.

Parameters

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

2.4.2.7 countEnergy()

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

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

Parameters

player	un joueur bien initialisé
round	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

player	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

player	une ensiie valide

2.4.2.11 enoughEnergy()

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

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

Parameters

player	un joueur bien initialisé
card	la carte qu'il souhaite jouer

Author

Alex Danduran–Lembezat

2.4.2.12 getAdditionalSustainablePoints()

```
int getAdditionalSustainablePoints (
            Ensiie player )
```

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

Parameters

player	un joueur bien initialisé

Returns

Les points de développement durable supplémentaires du joueur

2.4.2.13 getBoard()

```
Board getBoard (
            Ensiie player )
```

Accesseur du plateau d'un joueur.

Parameters

player	Le joueur bien initialisé dont on veut le plateau

Returns

Renvoie le plateau du joueur

2.4.2.14 getChangedSustOrDura()

```
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.

Parameters

player	Joueur bien initialisé dont on veut récupérer les statistiques des cartes
dura_or_sust	DURABILITY si on veut récupérer la durabilité et SUSTAINABLE si on veut récupérer le développement
id	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 (
            Ensiie player )
```

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

Parameters

joueur	joueur dont on veut le deck

Returns

Le deck du joueur

2.4.2.16 getEnergy()

```
int getEnergy (
            Ensiie player )
```

Accesseur en lecture de l'énergie du joueur.

Parameters

player	un joueur bien initialisé

Returns

L'énergie du joueur en question

Author

Alex Danduran–Lembezat

2.4.2.17 getFisaNumber()

```
int getFisaNumber (
            Ensiie player )
```

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

Parameters

player	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 (
            Ensiie player )
```

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

Parameters

player	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 (
            Ensiie player )
```

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

Parameters

player	joueur dont on veut la défausse

Returns

La défausse du joueur

2.4.2.20 getHand()

```
List_Card getHand (
            Ensiie player )
```

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

Parameters

player	joueur dont on veut la main

Returns

La main du joueur

2.4.2.21 getPlayerName()

```
char* getPlayerName (
            Ensiie player )
```

Récupère le nom du joueur spécifié

Parameters

joueur	joueur dont on veut le nom

Returns

Le nom du joueur

2.4.2.22 getPlayerSustainablePoints()

```
int getPlayerSustainablePoints (
            Ensiie player )
```

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

Parameters

player	un joueur bien initialisé

Returns

Les points de développement durable du joueur

2.4.2.23 getSumCardDurability()

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

Renvoie la somme des points de durabilité des cartes Elève.

Parameters

player	Joueur bien initialisé
is_round_even	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 (
            Ensiie player,
            int is_round_even )
```

Renvoie la somme des points de développement des cartes Elève.

Parameters

player	Joueur bien initialisé
is_round_even	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 (
            Ensiie player )
```

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

Parameters

player	Joueur bien initialisé

Returns

1 si la carte doit être améliorée, 0 sinon

2.4.2.26 getUpgradedCardSustainable()

```
int getUpgradedCardSustainable (
            Ensiie player )
```

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

Parameters

player	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

player	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

player	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

player	le joueur courant

Returns

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

2.4.2.30 numberCard()

```
int numberCard (
            Ensiie player )
```

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

Parameters

player	une ensiie valide

Returns

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

2.4.2.31 numberEleve()

```
int numberEleve (
            Ensiie player )
```

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

Parameters

player	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 (
            Ensiie player )
```

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

Parameters

player	Joueur bien initialisé

2.4.2.33 removeAdditionalSustainablePoints()

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

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

Parameters

player	un joueur bien initialisé
points	le nombre de points à retirer

2.4.2.34 removeEnergy()

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

Retire des points d'énergie à un joueur.

Parameters

player	un joueur bien initialisé
energy	la quantité d'énergie à retirer

Author

Alex Danduran–Lembezat

2.4.2.35 setEnergy()

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

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

Parameters

player	un joueur bien initialisé
energy	la quantité d'énergie à mettre

Author

Alex Danduran–Lembezat

2.4.2.36 setIsCurrentPlayer()

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

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

Parameters

player	Joueur bien initialisé

2.4.2.37 setPlayerName()

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

Change le nom du joueur spécifié

Parameters

joueur	joueur dont on veut changer le nom
name	le nom à donner au joueur

2.4.2.38 updateSustAndDura()

```
void updateSustAndDura (
            Ensiie player )
```

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

Parameters

player	Joueur bien initialisé dont on veut mettre à jour les statistiques de carte

2.4.2.39 upgradeCardDurability()

```
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.

Parameters

player	Joueur bien initialisé

2.4.2.40 upgradeCardSustainable()

```
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.

Parameters

player	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

b	Un plateau valide

2.5.2.3 getAvailableSpotNumber()

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

Renvoie le nombre d'emplacement Personnels disponible.

Parameters

board	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

board	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

board	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 débloqué (varie en 1 et 3)

Parameters

board	Plateau de jeu d'un joueur bien initialisé

Returns

Nombre d'emplacement Personnels débloqué

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

board	Plateau de jeu d'un joueur bien initialisé
specialisation	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

board	Plateau de jeu d'un joueur bien initialisé
round	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

card	Une carte valide
list_card	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

card	Une carte valide
list_card	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

list	le pointeur vers le premier élément de la liste de carte
size	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

list_card	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

card	Une carte valide
original_list	Une liste de cartes valide dans laquelle se trouve la carte
destination_list	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

list_card	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

list_card	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

list_card	Une liste de cartes valide
index	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

a	un entier quelconque
b	un entier quelconque

Returns

le plus grand des deux entiers passés en argument

Index