Correction des TPs Manuel de référence

Généré par Doxygen 1.5.1

Thu Jan 10 16 :53 :20 2008

Table des matières

Chapitre 1

Correction des TPs Hiérarchie de répertoires

1.1 Correction des TPs Répertoires

Cette hiérarchie de répertoire est triée approximativement, mais pas complètement, par ordre alphabétique :

Chapitre 2

Correction des TPs Index des espaces de nommage

2.1 Correction des TPs Liste des espaces de nommage

Liste de tous les espaces de nommage avec une brève description :

Chapitre 3

Correction des TPs Index des structures de données

3.1 Correction des TPs Structures de données

Liste des structures de données avec une brève description :

Chapitre 4

Correction des TPs Index des fichiers

4.1 Correction des TPs Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 5

Correction des TPs Documentation des répertoires

5.1 Répertoire de référence de initialisation/

Fichiers

– fichier **initialisation.c**
code source des fonctions necessaires à l'initialisation du monopoly

– fichier **initialisation.h**
en tete des fonctions necessaires à l'initialisation du monopoly

5.2 Répertoire de référence de interaction/

Fichiers

– fichier **interaction.c**
code source des fonctions gÃ©rant l'interaction avec le joueur

– fichier **interaction.h**
en tete des fonctions gérant l'interaction avec le joueur

5.3 Répertoire de référence de outils/

Fichiers

– fichier **outils.c**
code source des outils

– fichier **outils.h**
en tete des outils

5.4 Répertoire de référence de sdl/

Fichiers

– fichier **sdl.c**
– fichier **sdl.h**

5.5 Répertoire de référence de traitement/

Fichiers

– fichier **traitement.c**
code source des fonctions de traitement.

– fichier **traitement.h**
en tete des fonctions de traitement.

Chapitre 6

Correction des TPs Documentation des espaces de nommage

6.1 Référence de l'espace de nommage std

Chapitre 7

Correction des TPs Documentation des structures de données

7.1 Référence de la structure cases

Champs de données

- int **int_type**
- union {
  struct {
    char **str_nom** [16]
    int **int_prix**
      > *nom de la case*
    int **int_prix_niveau**
      > *prix de la salle*
    joueur * **pjoueur_joueur**
      > *prix pour augnmenter d'un niveau*
    int **int_niveau**
      > *joueur possedant la case*
    int **int_groupe**
      > *indique le niveau de la salle*
    SDL_Surface * **surf_detail**
      > *indique*
    SDL_Surface * **surf_propriete**
  etaill > *surface de la case d*
    int **bool_etat**
    bool **bool_hypotheque**
      equ > *indique si la case est hypoth*
    int **int_valeur_hypotheque**
      >*indique si la case est hypothequee*
  } **case_salle**
  struct {
    char **str_nom** [16]
    SDL_Surface * **surf_detail**
      > *nom de la salle*
  } **case_administration**
  struct {
    char **str_nom** [16]

int **int_prix**
> *nom du lieu commun*
joueur * **pjoueur_joueur**
> *prix de la salle*
int **int_suivant**
> *joueur possedant la salle*
SDL_Surface * **surf_detail**
> *entier du lieu commun suivant sur le plateau*
SDL_Surface * **surf_propriete**
etaill > *surface de la case d*
bool **bool_hypotheque**
int **int_valeur_hypotheque**
>*indique si la case est hypothequee*
} **case_lieu_commun**
struct {
char **str_nom** [16]
int **int_prix**
> *nom de l'association*
joueur * **pjoueur_joueur**
> *prix de la salle*
int **int_autre_bureau**
> *joueur possedant la salle*
SDL_Surface * **surf_detail**
> *position de l'aurtre association sur le plateau*
SDL_Surface * **surf_propriete**
etaill > *surface de la case d*
bool **bool_hypotheque**
int **int_valeur_hypotheque**
>*indique si la case est hypothequee*
} **case_association**
struct {
char **str_nom** [16]
int **int_prix**
> *nom de l'association*
SDL_Surface * **surf_detail**
> *prix de la salle*
} **case_soiree**
struct {
char **str_nom** [16]
int **int_argent**
> *nom de la case sp*
} **machine_a_cafe**
struct {
char **str_nom** [16]
} **case_special**
};

> *indique le type de la structure*

- SDL_Surface * **surf_image**
- int **int_position**

> *surface de la case*

- SDL_Rect **rect_coordonnees**

> *indique la position de la case sur le plateau*

7.1.1 Description détaillée

Définition à la ligne 56 du fichier structure.h.

7.1.2 Documentation des champs

7.1.2.1 int int_type

Définition à la ligne 58 du fichier structure.h.

Référencé par action_possible(), affich_case(), affich_panneau_possessions(), attente_validation_propriete(), insertion_bonne_place_propriete(), possession_autres_cartes(), traitement_achat(), traitement_arrive_case(), traitement_bureau(), traitement_elimination_joueur(), traitement_loyer_association(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

7.1.2.2 char str_nom[16]

Définition à la ligne 65 du fichier structure.h.

7.1.2.3 int int_prix

> nom de l'association

Définition à la ligne 66 du fichier structure.h.

7.1.2.4 int int_prix_niveau

> prix de la salle

Définition à la ligne 67 du fichier structure.h.

7.1.2.5 joueur* pjoueur_joueur

> prix pour augnmenter d'un niveau

Définition à la ligne 68 du fichier structure.h.

7.1.2.6 int int_niveau

> joueur possedant la case

Définition à la ligne 69 du fichier structure.h.

Référencé par action_possible(), init_plateau_chargement(), et sauvegarde().

7.1.2.7 int int_groupe

> indique le niveau de la salle

Définition à la ligne 70 du fichier structure.h.

Référencé par possession_autres_cartes().

7.1.2.8 SDL_Surface* surf_detail

> indique à quel groupe appartient la salle

Définition à la ligne 71 du fichier structure.h.

7.1.2.9 SDL_Surface* surf_propriete

> surface de la case détaillée

Définition à la ligne 72 du fichier structure.h.

7.1.2.10 int bool_etat

Définition à la ligne 73 du fichier structure.h.

7.1.2.11 bool bool_hypotheque

> indique si la case est hypothéquée ou non

Définition à la ligne 74 du fichier structure.h.

7.1.2.12 int int_valeur_hypotheque

>indique si la case est hypothequee

Définition à la ligne 75 du fichier structure.h.

7.1.2.13 struct { ... } case_salle

Référencé par action_possible(), affich_case(), affich_case_detail(), affich_panneau_possessions(), attente_validation_propriete(), insertion_bonne_place_propriete(), possession_autres_cartes(), traitement_achat(), traitement_arrive_case(), traitement_augmentation_niveau(), traitement_bureau(), traitement_diminution_niveau(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

7.1.2.14 SDL_Surface* surf_detail

> nom de la salle

Définition à la ligne 81 du fichier structure.h.

7.1.2.15 struct { ... } case_administration

7.1.2.16 joueur* pjoueur_joueur

> prix de la salle

Définition à la ligne 88 du fichier structure.h.

7.1.2.17 int int_suivant

> joueur possedant la salle

Définition à la ligne 89 du fichier structure.h.

7.1.2.18 SDL_Surface* surf_detail

> entier du lieu commun suivant sur le plateau

Définition à la ligne 90 du fichier structure.h.

7.1.2.19 SDL_Surface* surf_propriete

> surface de la case détaillée

Définition à la ligne 91 du fichier structure.h.

7.1.2.20 struct { ... } case_lieu_commun

Référencé par action_possible(), affich_panneau_possessions(), attente_validation_propriete(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

7.1.2.21 joueur* pjoueur_joueur

> prix de la salle

Définition à la ligne 100 du fichier structure.h.

7.1.2.22 int int_autre_bureau

> joueur possedant la salle

Définition à la ligne 101 du fichier structure.h.

7.1.2.23 SDL_Surface* surf_detail

> position de l'aurtre association sur le plateau

Définition à la ligne 102 du fichier structure.h.

7.1.2.24 SDL_Surface* surf_propriete

> surface de la case détaillée

Définition à la ligne 103 du fichier structure.h.

7.1.2.25 struct { ... } case_association

Référencé par action_possible(), affich_panneau_possessions(), attente_validation_propriete(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

7.1.2.26 SDL_Surface* surf_detail

> prix de la salle

Définition à la ligne 112 du fichier structure.h.

7.1.2.27 struct { ... } case_soiree

Référencé par traitement_soiree().

7.1.2.28 int int_argent

> nom de la case spéciale

Définition à la ligne 118 du fichier structure.h.

Référencé par traitement_machine_a_cafe().

7.1.2.29 struct { ... } machine_a_cafe

Référencé par init_case_special(), init_plateau_chargement(), sauvegarde(), traitement_bureau(), traitement_bureau_laurence(), traitement_machine_a_cafe(), et traitement_soiree().

7.1.2.30 struct { ... } case_special

7.1.2.31 union { ... }

> indique le type de la structure

7.1.2.32 SDL_Surface* surf_image

Définition à la ligne 128 du fichier structure.h.

7.1.2.33 int int_position

> surface de la case

Définition à la ligne 129 du fichier structure.h.

Référencé par affich_case().

7.1.2.34 SDL_Rect rect_coordonnees

> indique la position de la case sur le plateau

Définition à la ligne 130 du fichier structure.h.

7.2 Référence de la structure information

Champs de données

– char **texte** [512]
– int **type**
> *contient le texte d'information*

– int **valeur**

7.2.1 Description détaillée

Définition à la ligne 27 du fichier structure.h.

7.2.2 Documentation des champs

7.2.2.1 char texte[512]

Définition à la ligne 28 du fichier structure.h.

7.2.2.2 int type

> contient le texte d'information

Définition à la ligne 29 du fichier structure.h.

Référencé par init_bureau_krystel(), et init_bureau_nadege().

7.2.2.3 int valeur

> indique le type d'information dont il s'agit > correspondant à l'information

Définition à la ligne 31 du fichier structure.h.

Référencé par init_bureau_krystel(), init_bureau_nadege(), et traitement_bureau().

7.3 Référence de la structure joueur

Champs de données

- int **int_argent**
- int **int_certificat**
  > *argent total du joueur*

- int **int_position**
  > *nb de certificats médicaux du joueurs*

- int **bool_debut**
  > *numéro de la case actuelle du joueur*

- char **str_nom** [64]
  > *indique si le joueur est à son premier tour*

- **joueur** * **pjoueur_suivant**
  > *le nom du joueur*

- SDL_Surface * **surf_image**
  > *pointe vers le joueur suivant*

- int **int_double_tire**
  > *image du jeton du joueur*

- bool **bool_laurence**
  > *nombre de doubles effectués par le joueur*

- int **int_laurence**
  > *permet de savoir si le joueur est en prison ou pas*

- **possession** * **propriete**
  > *nombre de tours passés dans le bureau de laurence*

7.3.1 Description détaillée

Définition à la ligne 41 du fichier structure.h.

7.3.2 Documentation des champs

7.3.2.1 int int_argent

Définition à la ligne 42 du fichier structure.h.

Référencé par action_possible(), affich_panneau_joueur(), init_joueur(), init_joueur_chargement(), joueur_possede_tout(), sauvegarde(), traitement_achat(), traitement_augmentation_niveau(), traitement_bureau(), traitement_bureau_laurence(), traitement_case_depart(), traitement_diminution_niveau(), traitement_machine_a_cafe(), traitement_mise_en_hypotheque(), traitement_payer_loyer(), traitement_rachat_hypotheque(), et traitement_soiree().

7.3.2.2 int int_certificat

> argent total du joueur

Définition à la ligne 43 du fichier structure.h.

Référencé par affich_panneau_joueur(), init_joueur(), init_joueur_chargement(), sauvegarde(), traitement_bureau(), et traitement_bureau_laurence().

7.3.2.3 int int_position

> nb de certificats médicaux du joueurs

Définition à la ligne 44 du fichier structure.h.

Référencé par action_possible(), affich_joueur_depart(), aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), init_joueur(), init_joueur_chargement(), nombre_joueur_case(), possession_autres_cartes(), reculer_jeton(), sauvegarde(), traitement_arrive_case(), et traitement_soiree().

7.3.2.4 int bool_debut

> numéro de la case actuelle du joueur

Définition à la ligne 45 du fichier structure.h.

Référencé par action_possible(), init_joueur(), init_joueur_chargement(), sauvegarde(), et traitement_case_depart().

7.3.2.5 char str_nom[64]

> indique si le joueur est à son premier tour

Définition à la ligne 46 du fichier structure.h.

Référencé par affich_panneau_joueur(), init_joueur(), init_joueur_chargement(), et sauvegarde().

7.3.2.6 joueur* pjoueur_suivant

> le nom du joueur

Définition à la ligne 47 du fichier structure.h.

Référencé par affich_joueur_depart(), aller_a_jeton(), aller_en_prison_jeton(), avancer_jeton(), init_anneau_joueur(), init_anneau_joueur_chargement(), init_joueur(), init_joueur_chargement(), jeu(), main(), nombre_joueur_case(), reculer_jeton(), sauvegarde(), traitement_bureau(), traitement_elimination_joueur(), et verification_victoire().

7.3.2.7 SDL_Surface* surf_image

> pointe vers le joueur suivant

Définition à la ligne 48 du fichier structure.h.

Référencé par affich_joueur(), affich_panneau_joueur(), init_joueur(), et init_joueur_chargement().

7.3.2.8 int int_double_tire

> image du jeton du joueur

Définition à la ligne 49 du fichier structure.h.

Référencé par init_joueur(), init_joueur_chargement(), jeu(), lancer_des(), sauvegarde(), et traitement_bureau_laurence().

7.3.2.9 bool bool_laurence

> nombre de doubles effectués par le joueur

Définition à la ligne 50 du fichier structure.h.

Référencé par aller_en_prison_jeton(), attente_clic(), init_joueur(), init_joueur_chargement(), sauvegarde(), et traitement_bureau_laurence().

7.3.2.10 int int_laurence

> permet de savoir si le joueur est en prison ou pas

Définition à la ligne 51 du fichier structure.h.

Référencé par init_joueur(), init_joueur_chargement(), sauvegarde(), et traitement_bureau_laurence().

7.3.2.11 possession* propriete

> nombre de tours passés dans le bureau de laurence

Définition à la ligne 52 du fichier structure.h.

Référencé par affich_panneau_possessions(), init_joueur(), init_joueur_chargement(), int_to_cases(), nombre_propriete(), sauvegarde(), traitement_achat(), traitement_bureau(), traitement_loyer_association(), et traitement_loyer_lieu_commun().

7.4 Référence de la structure possession

Champs de données

– **possession ∗ suivant**
– **cases ∗ propriete**

7.4.1 Description détaillée

Définition à la ligne 34 du fichier structure.h.

7.4.2 Documentation des champs

7.4.2.1 struct possession∗ suivant

Définition à la ligne 36 du fichier structure.h.

Référencé par affich_panneau_possessions(), creation_possession(), insertion_bonne_place_propriete(), int_to_cases(), nombre_propriete(), sauvegarde(), traitement_bureau(), traitement_elimination_joueur(), traitement_loyer_association(), et traitement_loyer_lieu_commun().

7.4.2.2 cases∗ propriete

Définition à la ligne 37 du fichier structure.h.

Référencé par affich_panneau_possessions(), creation_possession(), insertion_bonne_place_propriete(), int_to_cases(), sauvegarde(), traitement_bureau(), traitement_elimination_joueur(), et traitement_loyer_association().

7.5 Référence de la structure rvb_couleur

Champs de données

– int **rouge**
– int **vert**
– int **bleu**

7.5.1 Description détaillée

Définition à la ligne 20 du fichier structure.h.

7.5.2 Documentation des champs

7.5.2.1 int rouge

Définition à la ligne 21 du fichier structure.h.

Référencé par creation_case(), creation_case_detail(), creation_case_propriete(), init_case_association(), init_case_lieu_commun(), et init_rvb_couleur().

7.5.2.2 int vert

Définition à la ligne 22 du fichier structure.h.

Référencé par creation_case(), creation_case_detail(), creation_case_propriete(), init_case_association(), init_case_lieu_commun(), et init_rvb_couleur().

7.5.2.3 int bleu

Définition à la ligne 23 du fichier structure.h.

Référencé par creation_case(), creation_case_detail(), creation_case_propriete(), init_case_association(), init_case_lieu_commun(), et init_rvb_couleur().

Chapitre 8

Correction des TPs Documentation des fichiers

8.1 Référence du fichier constante.h

toutes les constantes necessaire au monopoly

Macros

- #define **CASE_EPAISSEUR** 3
- #define **CASE_LARGEUR** 85
- #define **CASE_HAUTEUR** 115
- #define **CASE_GROUPE_HAUTEUR** 20
- #define **DETAIL_EPAISSEUR** 3
- #define **DETAIL_LARGEUR** 300
- #define **DETAIL_HAUTEUR** 440
- #define **DETAIL_GROUPE_HAUTEUR** 70
- #define **DETAIL_MESSAGE_EPAISSEUR** 20
- #define **DETAIL_MESSAGE_POS_X** POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_LARGEUR)/2-DETAIL_MESSAGE_EPAISSEUR
- #define **DETAIL_MESSAGE_POS_Y** POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_HAUTEUR)/2-DETAIL_MESSAGE_EPAISSEUR-40
- #define **DETAIL_BOUTON_POS_X** DETAIL_MESSAGE_POS_X+(DETAIL_LARGEUR+2*DETAIL_MESSAGE_EPAISSEUR-165)/2
- #define **DETAIL_BOUTON_POS_Y** DETAIL_MESSAGE_POS_Y+(DETAIL_HAUTEUR+2*DETAIL_MESSAGE_EPAISSEUR)
- #define **POSITION_BAS** 0
- #define **POSITION_GAUCHE** 1
- #define **POSITION_HAUT** 2
- #define **POSITION_DROITE** 3
- #define **POSITION_COIN_1** 4
- #define **POSITION_COIN_2** 5
- #define **POSITION_COIN_3** 6
- #define **POSITION_COIN_4** 7
- #define **SALLE** 0
- #define **BUREAU_KRYSTEL** 1
- #define **BUREAU_NADEGE** 2
- #define **LC_WC** 3
- #define **LC_ASCENSEUR** 4
- #define **LC_RU** 5
- #define **LC_PARKING** 6
- #define **BDE** 7

- #define **BDS** 8
- #define **SOIREE_AREA** 9
- #define **SOIREE_GALA** 10
- #define **SP_APPARTEMENT** 11
- #define **SP_BUREAU_LAURENCE** 12
- #define **SP_MACHINE_CAFE** 13
- #define **SP_TABLEAU** 14
- #define **VIOLET** 0
- #define **BLEU_CIEL** 1
- #define **ROSE** 2
- #define **ORANGE** 3
- #define **ROUGE** 4
- #define **JAUNE** 5
- #define **VERT** 6
- #define **BLEU** 7
- #define **TAILLE_PLATEAU** 995
- #define **TAILLE_CENTRE** 765
- #define **ECRAN_LARGEUR** 1280
- #define **ECRAN_HAUTEUR** 1024
- #define **POS_Y_PLATEAU** ((ECRAN_HAUTEUR-TAILLE_PLATEAU)/2)
- #define **POS_X_PLATEAU** (256+POS_Y_PLATEAU)
- #define **TITRE_POS_X** 250
- #define **TITRE_POS_Y** 100
- #define **TEXTE_POS_X** 350
- #define **TEXTE_POS_Y** 400
- #define **POS_X_FLECHES** 500
- #define **POS_Y_FLECHES** 600
- #define **BTN_ACCUEIL_POS_X** 500
- #define **BTN_ACCUEIL_POS_Y** 750
- #define **BTN_ACCUEIL_LARGEUR** 162
- #define **BTN_ACCUEIL_HAUTEUR** 43
- #define **BTN_ACCUEIL_ESPACE** 60
- #define **NOM_CHAMP_POS_X** 400
- #define **NOM_CHAMP_POS_Y** 330
- #define **CHAMP_POS_X** 600
- #define **CHAMP_POS_Y** 350
- #define **CHAMP_LARGEUR** 300
- #define **CHAMP_HAUTEUR** 30
- #define **PANNEAU_MENU_POS_X** 10
- #define **PANNEAU_MENU_POS_Y** 20
- #define **PANNEAU_MENU_LARGEUR** 240
- #define **PANNEAU_MENU_HAUTEUR** 60
- #define **PANNEAU_OBJET_TAILLE** 40
- #define **PANNEAU_JOUEUR_POS_X** 10
- #define **PANNEAU_JOUEUR_POS_Y** (40+PANNEAU_MENU_HAUTEUR)
- #define **PANNEAU_JOUEUR_LARGEUR** 240
- #define **PANNEAU_JOUEUR_HAUTEUR** 150
- #define **PANNEAU_FDT_POS_X** 10
- #define **PANNEAU_FDT_POS_Y** (20+PANNEAU_JOUEUR_POS_Y+PANNEAU_JOUEUR_HAUTEUR)
- #define **PANNEAU_FDT_LARGEUR** 240
- #define **PANNEAU_FDT_HAUTEUR** 60
- #define **PANNEAU_DES_POS_X** 10
- #define **PANNEAU_DES_POS_Y** (20+PANNEAU_FDT_POS_Y+PANNEAU_FDT_HAUTEUR)
- #define **PANNEAU_DES_LARGEUR** 240
- #define **PANNEAU_DES_HAUTEUR** 60
- #define **PANNEAU_POSSESSION_POS_X** 10
- #define **PANNEAU_POSSESSION_POS_Y** (20+PANNEAU_DES_POS_Y+PANNEAU_DES_HAUTEUR)
- #define **PANNEAU_POSSESSION_LARGEUR** 240
- #define **PANNEAU_POSSESSION_HAUTEUR** 60
- #define **PROPRIETE_LARGEUR** 210
- #define **PROPRIETE_HAUTEUR** 30
- #define **POSITION** 0
- #define **ARGENT** 1
- #define **CERTIFICAT** 2
- #define **ANNIVERSAIRE** 3
- #define **ASTRID** 4
- #define **CTI** 5

- #define **CARTE_LAURENCE** 6
- #define **PRIX_NIVEAU_MIN** 500
- #define **DELAY** 200
- #define **ACTION_ACHAT** 0
- #define **ACTION_UNHYPOTHEQUE** 1
- #define **ACTION_HYPOTHEQUE** 2
- #define **ACTION_PLUS** 3
- #define **ACTION_MOINS** 4
- #define **ACTION_PLUS_MOINS** 5
- #define **ACTION_FINIR** 6
- #define **ACTION_PAYER** 7
- #define **ACTION_PERDRE** 8
- #define **ARRIVE_CASE** 0
- #define **CLICK_CASE** 1
- #define **MESSAGE_LARGEUR** 600
- #define **MESSAGE_HAUTEUR** 200
- #define **MESSAGE_MAX_LIGNE** 7
- #define **MESSAGE_MAX_CARACTERE** 64
- #define **MESSAGE_KRYSTEL** 0
- #define **MESSAGE_NADEGE** 1
- #define **MESSAGE_NORMAL** 2
- #define **MESSAGE_QUITTER** 3
- #define **MESSAGE_PRISON** 4
- #define **BTN_ACHAT** 0
- #define **BTN_UNHYPOTHEQUE** 1
- #define **BTN_HYPOTHEQUE** 2
- #define **BTN_PLUS** 3
- #define **BTN_MOINS** 4
- #define **BTN_FINIR** 5
- #define **ERREUR** -1
- #define **PRIX_NIVEAU_MIN** 500

8.1.1 Description détaillée

toutes les constantes necessaire au monopoly

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **constante.h**.

8.1.2 Documentation des macros

8.1.2.1 #define CASE_EPAISSEUR 3

Définition à la ligne 12 du fichier constante.h.

Référencé par affich_case(), creation_case(), creation_case_association(), creation_case_bureau(), creation_case_coin(), creation_case_lieu_commun(), et creation_case_soiree().

8.1.2.2 #define CASE_LARGEUR 85

Définition à la ligne 13 du fichier constante.h.

Référencé par affich_case(), creation_case(), creation_case_association(), creation_case_bureau(), creation_case_lieu_commun(), creation_case_soiree(), init_case_administration(), init_case_association(), init_case_lieu_commun(), init_case_salle(), et init_case_soiree().

8.1.2.3 #define CASE_HAUTEUR 115

Définition à la ligne 14 du fichier constante.h.

Référencé par affich_case(), affich_case_detail(), affich_centre(), affich_message(), attente_validation_message(), creation_case(), creation_case_association(), creation_case_bureau(), creation_case_coin(), creation_case_lieu_commun(), creation_case_soiree(), init_case_administration(), init_case_association(), init_case_lieu_commun(), init_case_salle(), init_case_soiree(), et init_case_special().

8.1.2.4 #define CASE_GROUPE_HAUTEUR 20

Définition à la ligne 15 du fichier constante.h.

Référencé par creation_case().

8.1.2.5 #define DETAIL_EPAISSEUR 3

Définition à la ligne 17 du fichier constante.h.

Référencé par creation_case_detail(), creation_case_detail_assoc(), et creation_case_detail_lc().

8.1.2.6 #define DETAIL_LARGEUR 300

Définition à la ligne 18 du fichier constante.h.

Référencé par affich_case_detail(), attente_validation_propriete(), creation_case_detail(), creation_case_detail_assoc(), et creation_case_detail_lc().

8.1.2.7 #define DETAIL_HAUTEUR 440

Définition à la ligne 19 du fichier constante.h.

Référencé par affich_case_detail(), attente_validation_propriete(), creation_case_detail(), creation_case_detail_assoc(), et creation_case_detail_lc().

8.1.2.8 #define DETAIL_GROUPE_HAUTEUR 70

Définition à la ligne 20 du fichier constante.h.

Référencé par creation_case_detail().

8.1.2.9 #define DETAIL_MESSAGE_EPAISSEUR 20

Définition à la ligne 21 du fichier constante.h.

Référencé par attente_validation_propriete().

8.1.2.10 #define DETAIL_MESSAGE_POS_X POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_LARGEUR)/2-DETAIL_MESSAGE_EPAISSEUR

Définition à la ligne 22 du fichier constante.h.

Référencé par affich_validation_propriete().

8.1.2.11 #define DETAIL_MESSAGE_POS_Y POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_HAUTEUR)/2-DETAIL_MESSAGE_EPAISSEUR-40

Définition à la ligne 23 du fichier constante.h.

Référencé par affich_validation_propriete().

8.1.2.12 #define DETAIL_BOUTON_POS_X DETAIL_MESSAGE_POS_X+(DETAIL_LARGEUR+2*DETAIL_MESSAGE_EPAISSEUR-165)/2

Définition à la ligne 24 du fichier constante.h.

Référencé par affich_validation_propriete(), et attente_validation_propriete().

8.1.2.13 #define DETAIL_BOUTON_POS_Y DETAIL_MESSAGE_POS_Y+(DETAIL_HAUTEUR+2*DETAIL_MESSAGE_EPAISSEUR)

Définition à la ligne 25 du fichier constante.h.

Référencé par affich_validation_propriete(), et attente_validation_propriete().

8.1.2.14 #define POSITION_BAS 0

Définition à la ligne 28 du fichier constante.h.

Référencé par init_case_administration(), init_case_association(), init_case_salle(), et init_case_soiree().

8.1.2.15 #define POSITION_GAUCHE 1

Définition à la ligne 29 du fichier constante.h.

Référencé par affich_case(), affich_joueur(), creation_case(), creation_case_bureau(), init_case_administration(), init_case_association(), init_case_salle(), et init_case_soiree().

8.1.2.16 #define POSITION_HAUT 2

Définition à la ligne 30 du fichier constante.h.

Référencé par affich_case(), affich_joueur(), creation_case(), creation_case_bureau(), init_case_administration(), init_case_association(), init_case_salle(), et init_case_soiree().

8.1.2.17 #define POSITION_DROITE 3

Définition à la ligne 31 du fichier constante.h.

Référencé par affich_case(), affich_joueur(), creation_case(), creation_case_bureau(), init_case_administration(), init_case_association(), init_case_salle(), et init_case_soiree().

8.1.2.18 #define POSITION_COIN_1 4

Définition à la ligne 32 du fichier constante.h.

8.1.2.19 #define POSITION_COIN_2 5

Définition à la ligne 33 du fichier constante.h.

8.1.2.20 #define POSITION_COIN_3 6

Définition à la ligne 34 du fichier constante.h.

8.1.2.21 #define POSITION_COIN_4 7

Définition à la ligne 35 du fichier constante.h.

8.1.2.22 #define SALLE 0

Définition à la ligne 37 du fichier constante.h.

Référencé par action_possible(), affich_case(), affich_panneau_possessions(), attente_validation_propriete(), init_plateau(), insertion_bonne_place_propriete(), possession_autres_cartes(), sauvegarde(), traitement_achat(), traitement_arrive_case(), traitement_bureau(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.23 #define BUREAU_KRYSTEL 1

Définition à la ligne 38 du fichier constante.h.

Référencé par creation_case_bureau(), init_plateau(), traitement_arrive_case(), et traitement_bureau().

8.1.2.24 #define BUREAU_NADEGE 2

Définition à la ligne 39 du fichier constante.h.

Référencé par init_plateau(), traitement_arrive_case(), et traitement_bureau().

8.1.2.25 #define LC_WC 3

Définition à la ligne 40 du fichier constante.h.

Référencé par action_possible(), attente_validation_propriete(), creation_case_detail_lc(), creation_case_lieu_commun(), init_case_lieu_commun(), init_plateau(), insertion_bonne_place_propriete(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.26 #define LC_ASCENSEUR 4

Définition à la ligne 41 du fichier constante.h.

Référencé par action_possible(), creation_case_detail_lc(), creation_case_lieu_commun(), init_case_lieu_commun(), init_plateau(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.27 #define LC_RU 5

Définition à la ligne 42 du fichier constante.h.

Référencé par action_possible(), creation_case_detail_lc(), creation_case_lieu_commun(), init_case_lieu_commun(), init_plateau(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.28 #define LC_PARKING 6

Définition à la ligne 43 du fichier constante.h.

Référencé par action_possible(), attente_validation_propriete(), creation_case_detail_lc(), creation_case_lieu_commun(), init_case_lieu_commun(), init_plateau(), insertion_bonne_place_propriete(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.29 #define BDE 7

Définition à la ligne 44 du fichier constante.h.

Référencé par action_possible(), affich_panneau_possessions(), attente_validation_propriete(), creation_case_association(), creation_case_detail_assoc(), init_case_association(), init_plateau(), insertion_bonne_place_propriete(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_loyer_association(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.30 #define BDS 8

Définition à la ligne 45 du fichier constante.h.

Référencé par action_possible(), affich_panneau_possessions(), attente_validation_propriete(), creation_case_detail_assoc(), init_plateau(), insertion_bonne_place_propriete(), traitement_achat(), traitement_arrive_case(), traitement_elimination_joueur(), traitement_loyer_association(), traitement_mise_en_hypotheque(), et traitement_rachat_hypotheque().

8.1.2.31 #define SOIREE_AREA 9

Définition à la ligne 46 du fichier constante.h.

Référencé par creation_case_soiree(), init_plateau(), et traitement_arrive_case().

8.1.2.32 #define SOIREE_GALA 10

Définition à la ligne 47 du fichier constante.h.

Référencé par creation_case_soiree(), init_plateau(), et traitement_arrive_case().

8.1.2.33 #define SP_APPARTEMENT 11

Définition à la ligne 48 du fichier constante.h.

Référencé par affich_case(), creation_case_coin(), init_case_special(), et init_plateau().

8.1.2.34 #define SP_BUREAU_LAURENCE 12

Définition à la ligne 49 du fichier constante.h.

Référencé par creation_case_coin(), init_case_special(), init_plateau(), et traitement_arrive_case().

8.1.2.35 #define SP_MACHINE_CAFE 13

Définition à la ligne 50 du fichier constante.h.

Référencé par creation_case_coin(), init_case_special(), init_plateau(), et traitement_arrive_case().

8.1.2.36 #define SP_TABLEAU 14

Définition à la ligne 51 du fichier constante.h.

Référencé par affich_case(), creation_case_coin(), init_case_special(), init_plateau(), et traitement_arrive_case().

8.1.2.37 #define VIOLET 0

Définition à la ligne 53 du fichier constante.h.

Référencé par init_plateau(), init_tab_rvb_couleur(), et possession_autres_cartes().

8.1.2.38 #define BLEU_CIEL 1

Définition à la ligne 54 du fichier constante.h.

Référencé par init_plateau(), et init_tab_rvb_couleur().

8.1.2.39 #define ROSE 2

Définition à la ligne 55 du fichier constante.h.

Référencé par init_plateau(), et init_tab_rvb_couleur().

8.1.2.40 #define ORANGE 3

Définition à la ligne 56 du fichier constante.h.

Référencé par init_plateau(), et init_tab_rvb_couleur().

8.1.2.41 #define ROUGE 4

Définition à la ligne 57 du fichier constante.h.

Référencé par init_plateau(), et init_tab_rvb_couleur().

8.1.2.42 #define JAUNE 5

Définition à la ligne 58 du fichier constante.h.

Référencé par init_plateau(), et init_tab_rvb_couleur().

8.1.2.43 #define VERT 6

Définition à la ligne 59 du fichier constante.h.

Référencé par init_plateau(), et init_tab_rvb_couleur().

8.1.2.44 #define BLEU 7

Définition à la ligne 60 du fichier constante.h.

Référencé par init_plateau(), init_tab_rvb_couleur(), et possession_autres_cartes().

8.1.2.45 #define TAILLE_PLATEAU 995

Définition à la ligne 62 du fichier constante.h.

Référencé par init_case_administration(), init_case_association(), init_case_lieu_commun(), init_case_salle(), init_case_soiree(), et init_case_special().

8.1.2.46 #define TAILLE_CENTRE 765

Définition à la ligne 63 du fichier constante.h.

Référencé par affich_case_detail(), affich_message(), et attente_validation_message().

8.1.2.47 #define ECRAN_LARGEUR 1280

Définition à la ligne 64 du fichier constante.h.

Référencé par affich_accueil(), affich_config(), attente_action_accueil(), et attente_action_-config().

8.1.2.48 #define ECRAN_HAUTEUR 1024

Définition à la ligne 65 du fichier constante.h.

Référencé par affich_possessions_cache().

8.1.2.49 #define POS_Y_PLATEAU ((ECRAN_HAUTEUR-TAILLE_-PLATEAU)/2)

Définition à la ligne 66 du fichier constante.h.

Référencé par affich_case_detail(), affich_centre(), affich_message(), attente_validation_-message(), init_case_administration(), init_case_association(), init_case_lieu_commun(), init_case_salle(), init_case_soiree(), et init_case_special().

8.1.2.50 #define POS_X_PLATEAU (256+POS_Y_PLATEAU)

Définition à la ligne 67 du fichier constante.h.

Référencé par affich_case_detail(), affich_centre(), affich_message(), attente_validation_-message(), init_case_administration(), init_case_association(), init_case_lieu_commun(), init_case_salle(), init_case_soiree(), et init_case_special().

8.1.2.51 #define TITRE_POS_X 250

Définition à la ligne 69 du fichier constante.h.

Référencé par affich_config().

8.1.2.52 #define TITRE_POS_Y 100

Définition à la ligne 70 du fichier constante.h.

Référencé par affich_accueil(), et affich_config().

8.1.2.53 #define TEXTE_POS_X 350

Définition à la ligne 71 du fichier constante.h.

8.1.2.54 #define TEXTE_POS_Y 400

Définition à la ligne 72 du fichier constante.h.

Référencé par affich_accueil().

8.1.2.55 #define POS_X_FLECHES 500

Définition à la ligne 74 du fichier constante.h.

Référencé par attente_action_accueil().

8.1.2.56 #define POS_Y_FLECHES 600

Définition à la ligne 75 du fichier constante.h.

Référencé par attente_action_accueil().

8.1.2.57 #define BTN_ACCUEIL_POS_X 500

Définition à la ligne 76 du fichier constante.h.

Référencé par affich_config(), et attente_action_config().

8.1.2.58 #define BTN_ACCUEIL_POS_Y 750

Définition à la ligne 77 du fichier constante.h.

Référencé par affich_accueil(), affich_config(), attente_action_accueil(), et attente_action_config().

8.1.2.59 #define BTN_ACCUEIL_LARGEUR 162

Définition à la ligne 78 du fichier constante.h.

Référencé par affich_accueil(), attente_action_accueil(), et attente_action_config().

8.1.2.60 #define BTN_ACCUEIL_HAUTEUR 43

Définition à la ligne 79 du fichier constante.h.

Référencé par attente_action_accueil(), et attente_action_config().

8.1.2.61 #define BTN_ACCUEIL_ESPACE 60

Définition à la ligne 80 du fichier constante.h.

Référencé par affich_accueil(), et attente_action_accueil().

8.1.2.62 #define NOM_CHAMP_POS_X 400

Définition à la ligne 82 du fichier constante.h.

Référencé par affich_config().

8.1.2.63 #define NOM_CHAMP_POS_Y 330

Définition à la ligne 83 du fichier constante.h.

Référencé par affich_config().

8.1.2.64 #define CHAMP_POS_X 600

Définition à la ligne 84 du fichier constante.h.

Référencé par affich_config(), et attente_action_config().

8.1.2.65 #define CHAMP_POS_Y 350

Définition à la ligne 85 du fichier constante.h.

Référencé par affich_config(), et attente_action_config().

8.1.2.66 #define CHAMP_LARGEUR 300

Définition à la ligne 86 du fichier constante.h.

Référencé par affich_config(), et attente_action_config().

8.1.2.67 #define CHAMP_HAUTEUR 30

Définition à la ligne 87 du fichier constante.h.

Référencé par affich_config(), et attente_action_config().

8.1.2.68 #define PANNEAU_MENU_POS_X 10

Définition à la ligne 89 du fichier constante.h.

Référencé par affich_panneau_menu(), et attente_clic().

8.1.2.69 #define PANNEAU_MENU_POS_Y 20

Définition à la ligne 90 du fichier constante.h.

Référencé par affich_panneau_menu(), et attente_clic().

8.1.2.70 #define PANNEAU_MENU_LARGEUR 240

Définition à la ligne 91 du fichier constante.h.

Référencé par affich_panneau_menu(), et attente_clic().

8.1.2.71 #define PANNEAU_MENU_HAUTEUR 60

Définition à la ligne 92 du fichier constante.h.

Référencé par affich_panneau_menu(), et attente_clic().

8.1.2.72 #define PANNEAU_OBJET_TAILLE 40

Définition à la ligne 93 du fichier constante.h.

Référencé par affich_panneau_menu(), et attente_clic().

8.1.2.73 #define PANNEAU_JOUEUR_POS_X 10

Définition à la ligne 95 du fichier constante.h.

Référencé par affich_panneau_joueur().

8.1.2.74 #define PANNEAU_JOUEUR_POS_Y (40+PANNEAU_MENU_HAUTEUR)

Définition à la ligne 96 du fichier constante.h.

Référencé par affich_panneau_joueur().

8.1.2.75 #define PANNEAU_JOUEUR_LARGEUR 240

Définition à la ligne 97 du fichier constante.h.

Référencé par affich_panneau_joueur().

8.1.2.76 #define PANNEAU_JOUEUR_HAUTEUR 150

Définition à la ligne 98 du fichier constante.h.

Référencé par affich_panneau_joueur().

8.1.2.77 #define PANNEAU_FDT_POS_X 10

Définition à la ligne 100 du fichier constante.h.

Référencé par affich_panneau_fdt(), et attente_clic().

8.1.2.78 #define PANNEAU_FDT_POS_Y (20+PANNEAU_JOUEUR_POS_Y+PANNEAU_JOUEUR_HAUTEUR)

Définition à la ligne 101 du fichier constante.h.

Référencé par affich_panneau_fdt(), et attente_clic().

8.1.2.79 #define PANNEAU_FDT_LARGEUR 240

Définition à la ligne 102 du fichier constante.h.

Référencé par affich_panneau_fdt(), et attente_clic().

8.1.2.80 #define PANNEAU_FDT_HAUTEUR 60

Définition à la ligne 103 du fichier constante.h.

Référencé par affich_panneau_fdt(), et attente_clic().

8.1.2.81 #define PANNEAU_DES_POS_X 10

Définition à la ligne 105 du fichier constante.h.

Référencé par affich_panneau_des(), affich_panneau_des_bouton(), et attente_clic().

8.1.2.82 #define PANNEAU_DES_POS_Y (20+PANNEAU_FDT_POS_-Y+PANNEAU_FDT_HAUTEUR)

Définition à la ligne 106 du fichier constante.h.

Référencé par affich_panneau_des(), affich_panneau_des_bouton(), et attente_clic().

8.1.2.83 #define PANNEAU_DES_LARGEUR 240

Définition à la ligne 107 du fichier constante.h.

Référencé par affich_panneau_des(), affich_panneau_des_bouton(), et attente_clic().

8.1.2.84 #define PANNEAU_DES_HAUTEUR 60

Définition à la ligne 108 du fichier constante.h.

Référencé par affich_panneau_des(), affich_panneau_des_bouton(), et attente_clic().

8.1.2.85 #define PANNEAU_POSSESSION_POS_X 10

Définition à la ligne 110 du fichier constante.h.

Référencé par affich_panneau_possessions(), affich_possessions_cache(), et attente_clic().

8.1.2.86 #define PANNEAU_POSSESSION_POS_Y (20+PANNEAU_DES_-POS_Y+PANNEAU_DES_HAUTEUR)

Définition à la ligne 111 du fichier constante.h.

Référencé par affich_panneau_possessions(), affich_possessions_cache(), et attente_clic().

8.1.2.87 #define PANNEAU_POSSESSION_LARGEUR 240

Définition à la ligne 112 du fichier constante.h.

Référencé par affich_panneau_possessions(), affich_possessions_cache(), et attente_clic().

8.1.2.88 #define PANNEAU_POSSESSION_HAUTEUR 60

Définition à la ligne 113 du fichier constante.h.

Référencé par affich_panneau_possessions().

8.1.2.89 #define PROPRIETE_LARGEUR 210

Définition à la ligne 115 du fichier constante.h.

Référencé par creation_case_propriete().

8.1.2.90 #define PROPRIETE_HAUTEUR 30

Définition à la ligne 116 du fichier constante.h.

Référencé par affich_panneau_possessions(), attente_clic(), et creation_case_propriete().

8.1.2.91 #define POSITION 0

Définition à la ligne 118 du fichier constante.h.

Référencé par init_bureau_krystel(), init_bureau_nadege(), et traitement_bureau().

8.1.2.92 #define ARGENT 1

Définition à la ligne 119 du fichier constante.h.

Référencé par init_bureau_krystel(), init_bureau_nadege(), et traitement_bureau().

8.1.2.93 #define CERTIFICAT 2

Définition à la ligne 120 du fichier constante.h.

Référencé par init_bureau_krystel(), init_bureau_nadege(), et traitement_bureau().

8.1.2.94 #define ANNIVERSAIRE 3

Définition à la ligne 121 du fichier constante.h.

Référencé par init_bureau_nadege(), et traitement_bureau().

8.1.2.95 #define ASTRID 4

Définition à la ligne 122 du fichier constante.h.

Référencé par init_bureau_nadege(), et traitement_bureau().

8.1.2.96 #define CTI 5

Définition à la ligne 123 du fichier constante.h.

Référencé par init_bureau_nadege(), et traitement_bureau().

8.1.2.97 #define CARTE_LAURENCE 6

Définition à la ligne 124 du fichier constante.h.

Référencé par init_bureau_krystel(), init_bureau_nadege(), et traitement_bureau().

8.1.2.98 #define PRIX_NIVEAU_MIN 500

Définition à la ligne 162 du fichier constante.h.

Référencé par init_plateau().

8.1.2.99 #define DELAY 200

Définition à la ligne 128 du fichier constante.h.

Référencé par avancer_jeton(), lancer_des(), reculer_jeton(), et traitement_bureau().

8.1.2.100 #define ACTION_ACHAT 0

Définition à la ligne 130 du fichier constante.h.

Référencé par action_possible(), action_to_boutons(), et attente_validation_propriete().

8.1.2.101 #define ACTION_UNHYPOTHEQUE 1

Définition à la ligne 131 du fichier constante.h.

Référencé par action_to_boutons().

8.1.2.102 #define ACTION_HYPOTHEQUE 2

Définition à la ligne 132 du fichier constante.h.

Référencé par action_to_boutons().

8.1.2.103 #define ACTION_PLUS 3

Définition à la ligne 133 du fichier constante.h.

Référencé par action_possible(), action_to_boutons(), et attente_validation_propriete().

8.1.2.104 #define ACTION_MOINS 4

Définition à la ligne 134 du fichier constante.h.

Référencé par action_possible(), action_to_boutons(), et attente_validation_propriete().

8.1.2.105 #define ACTION_PLUS_MOINS 5

Définition à la ligne 135 du fichier constante.h.

Référencé par action_possible(), action_to_boutons(), et attente_validation_propriete().

8.1.2.106 #define ACTION_FINIR 6

Définition à la ligne 136 du fichier constante.h.

Référencé par action_possible(), action_to_boutons(), et attente_validation_propriete().

8.1.2.107 #define ACTION_PAYER 7

Définition à la ligne 137 du fichier constante.h.

Référencé par action_possible().

8.1.2.108 #define ACTION_PERDRE 8

Définition à la ligne 138 du fichier constante.h.

Référencé par action_possible().

8.1.2.109 #define ARRIVE_CASE 0

Définition à la ligne 140 du fichier constante.h.

Référencé par action_possible(), et traitement_arrive_case().

8.1.2.110 #define CLICK_CASE 1

Définition à la ligne 141 du fichier constante.h.

Référencé par action_possible(), et attente_clic().

8.1.2.111 #define MESSAGE_LARGEUR 600

Définition à la ligne 143 du fichier constante.h.

Référencé par affich_message(), attente_validation_message(), et creation_message().

8.1.2.112 #define MESSAGE_HAUTEUR 200

Définition à la ligne 144 du fichier constante.h.

Référencé par affich_message(), et attente_validation_message().

8.1.2.113 #define MESSAGE_MAX_LIGNE 7

Définition à la ligne 145 du fichier constante.h.

8.1.2.114 #define MESSAGE_MAX_CARACTERE 64

Définition à la ligne 146 du fichier constante.h.

Référencé par decoupage_string().

8.1.2.115 #define MESSAGE_KRYSTEL 0

Définition à la ligne 147 du fichier constante.h.

Référencé par creation_message(), et traitement_bureau().

8.1.2.116 #define MESSAGE_NADEGE 1

Définition à la ligne 148 du fichier constante.h.

Référencé par creation_message(), et traitement_bureau().

8.1.2.117 #define MESSAGE_NORMAL 2

Définition à la ligne 149 du fichier constante.h.

Référencé par attente_clic(), jeu(), lancer_des(), traitement_arrive_case(), traitement_machine_a_cafe(), et traitement_soiree().

8.1.2.118 #define MESSAGE_QUITTER 3

Définition à la ligne 150 du fichier constante.h.

Référencé par affich_message(), attente_validation_message(), et jeu().

8.1.2.119 #define MESSAGE_PRISON 4

Définition à la ligne 151 du fichier constante.h.

Référencé par affich_message(), attente_validation_message(), et traitement_bureau_laurence().

8.1.2.120 #define BTN_ACHAT 0

Définition à la ligne 153 du fichier constante.h.

Référencé par action_to_boutons(), et affich_validation_propriete().

8.1.2.121 #define BTN_UNHYPOTHEQUE 1

Définition à la ligne 154 du fichier constante.h.

Référencé par action_to_boutons(), et affich_validation_propriete().

8.1.2.122 #define BTN_HYPOTHEQUE 2

Définition à la ligne 155 du fichier constante.h.

Référencé par action_to_boutons(), et affich_validation_propriete().

8.1.2.123 #define BTN_PLUS 3

Définition à la ligne 156 du fichier constante.h.

Référencé par action_to_boutons(), et affich_validation_propriete().

8.1.2.124 #define BTN_MOINS 4

Définition à la ligne 157 du fichier constante.h.

Référencé par action_to_boutons(), et affich_validation_propriete().

8.1.2.125 #define BTN_FINIR 5

Définition à la ligne 158 du fichier constante.h.

Référencé par action_to_boutons(), et affich_validation_propriete().

8.1.2.126 #define ERREUR -1

Définition à la ligne 160 du fichier constante.h.

Référencé par action_possible().

8.1.2.127 #define PRIX_NIVEAU_MIN 500

Définition à la ligne 162 du fichier constante.h.

8.2 Référence du fichier header.h

en-tÃłte rassemblant toutes les autres du programme

Espaces de nommage

– namespace **std**

8.2.1 Description détaillée

en-tÃłte rassemblant toutes les autres du programme

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **header.h**.

8.3 Référence du fichier initialisation.c

code source des fonctions necessaires à l'initialisation du monopoly

Fonctions

- **rvb_couleur init_rvb_couleur** (int rouge, int vert, int bleu)
  initialise une structure joueur en fonction des paramètre

- void **init_tab_rvb_couleur** (**rvb_couleur** couleur[8])
  initialise le tableau de couleur

- void **init_case_salle** (SDL_Surface *surf_ecran, **rvb_couleur** couleurs[8], **cases** *pcase, int int_type, int int_numero, int int_prix, char *str_nom1, char *str_nom2, int int_groupe, int int_prix_niveau)
  initialise les cases salles

- void **init_case_association** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases associations

- void **init_case_lieu_commun** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases lieu commun

- void **init_case_administration** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases adimnistration

- void **init_case_soiree** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases soirée

- void **init_case_special** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases spéciales

- void **init_plateau** (SDL_Surface *surf_ecran, **rvb_couleur** couleurs[8], **cases** **plateau)
- **joueur** * **init_joueur** (char *nom, int numero_joueur)
  initialise le joueur

- void **init_bureau_krystel** (**information** *bureau_de_krystel[16])
  initialise les cartes bureau de krystel

- void **init_bureau_nadege** (**information** *bureau_de_nadege[16])
  initialise les cartes bureau de nadège

- **possession** * **init_anneau_possesions** ()
  initialise l'anneau des possesions du joueur

- **possession** * **creation_possession** (**cases** *propriete)
  ajoute de façon ordonnée une propriété dans la liste

- **joueur** * **init_anneau_joueur** (int nombre_joueur, char **str_nom_joueur)
  initialise le ring des joueurs

- **joueur** * **init_anneau_joueur_chargement** (int nombre_joueur, ifstream *file, **cases** **plateau)
  créer la liste des joueurs

– **joueur** ∗ **init_joueur_chargement** (int numero_joueur, ifstream ∗file, **cases** ∗∗plateau)
créer un joueur à partir de la sauvagarde

– **cases** ∗∗ **init_plateau_chargement** (ifstream ∗file, **cases** ∗∗plateau, **rvb_couleur** couleurs[8], SDL_Surface ∗surf_ecran)
charge les données du plateau

8.3.1 Description détaillée

code source des fonctions necessaires à l'initialisation du monopoly

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

0.0.1

Définition dans le fichier **initialisation.c**.

8.3.2 Documentation des fonctions

8.3.2.1 rvb_couleur init_rvb_couleur (int *rouge*, int *vert*, int *bleu*)

initialise une structure joueur en fonction des paramètre

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

rouge valeur pour la valeur rouge
vert valeur pour la valeur vert
bleu valeur pour la valeur bleu

Version :

0.0.1

Renvoie :

la couleur

Définition à la ligne 11 du fichier initialisation.c.

Référencé par init_tab_rvb_couleur().

```
12 {
13   rvb_couleur couleur;
14   couleur.rouge=rouge;
15   couleur.vert=vert;
16   couleur.bleu=bleu;
17   return(couleur);
18 }
```

8.3.2.2 void init_tab_rvb_couleur (rvb_couleur *couleur*[8])

initialise le tableau de couleur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

couleur tableau de couleur

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 20 du fichier initialisation.c.

Référencé par main().

```
21 {
22   couleur[VIOLET]=init_rvb_couleur(205,14,207);
23   couleur[BLEU_CIEL]=init_rvb_couleur(101,232,215);
24   couleur[ROSE]=init_rvb_couleur(251,138,244);
25   couleur[ORANGE]=init_rvb_couleur(255,130,8);
26   couleur[ROUGE]=init_rvb_couleur(255,0,0);
27   couleur[JAUNE]=init_rvb_couleur(255,255,0);
28   couleur[VERT]=init_rvb_couleur(0,255,0);
29   couleur[BLEU]=init_rvb_couleur(0,0,255);
30 }
```

8.3.2.3 void init_case_salle (SDL_Surface * *surf_ecran*, rvb_couleur *couleurs*[8], cases * *pcase*, int *int_type*, int *int_numero*, int *int_prix*, char * *str_nom1*, char * *str_nom2*, int *int_groupe*, int *int_prix_niveau*)

initialise les cases salles

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

couleurs tableau de couleur

pcase case en cours de création

int_type le type de case

int_numero numero de la case

int_prix prix de la case

str_nom1 nom de la case

str_nom2 nom de la case

int_groupe groupe de la case

int_prix_niveau prix pour augmenter d'un niveau

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 32 du fichier initialisation.c.

Référencé par init_plateau().

```
33 {
34   int position;
35   char* str_nom;
36   position=int_numero/10;
37   (*pcase).int_type=int_type;
38   (*pcase).surf_image=creation_case(surf_ecran, couleurs[int_groupe], str_nom1, str_nom2, int_prix, position);
39   (*pcase).int_position=position;
40   switch(position)
41   {
42     case POSITION_BAS:
43       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
44       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
45       break;
46     case POSITION_GAUCHE:
47       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
48       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
49       break;
50     case POSITION_HAUT:
51       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
52       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
53       break;
54     case POSITION_DROITE:
55       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
56       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
57       break;
58     default:
59       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
60       break;
61   }
62   ((*pcase).case_salle).bool_hypotheque=false;
63   ((*pcase).case_salle).int_valeur_hypotheque=0;
64   ((*pcase).case_salle).int_prix=int_prix;
65   ((*pcase).case_salle).int_prix_niveau=int_prix_niveau;
66   ((*pcase).case_salle).pjoueur_joueur=NULL;
67   ((*pcase).case_salle).int_niveau=0;
68   ((*pcase).case_salle).int_groupe=int_groupe;
69
70   str_nom = new char[strlen(str_nom1)+strlen(str_nom2)+2];
71   sprintf(str_nom, "%s %s",str_nom1, str_nom2);
72   ((*pcase).case_salle).surf_detail=creation_case_detail(surf_ecran, couleurs[int_groupe], str_nom, int_prix/10, int
73   ((*pcase).case_salle).surf_propriete=creation_case_propriete(surf_ecran, couleurs[int_groupe], str_nom);
74   delete[] str_nom;
75   ((*pcase).case_salle).bool_etat=1;
76 }
```

8.3.2.4 void init_case_association (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases associations

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'écran

pcase case en cours de création

int_ type type de la case

int_ numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 78 du fichier initialisation.c.

Référencé par init_plateau().

```
79 {
80   rvb_couleur couleur;
81   couleur.rouge=255;
82   couleur.bleu=100;
83   couleur.vert=255;
84   int position;
85   position=int_numero/10;
86   (*pcase).int_type=int_type;
87   (*pcase).surf_image=creation_case_association(surf_ecran, int_type);
88   (*pcase).int_position=position;
89   switch(position)
90   {
91     case POSITION_BAS:
92       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
93       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
94       break;
95     case POSITION_GAUCHE:
96       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
97       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
98       break;
99     case POSITION_HAUT:
100       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
101       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
102       break;
103     case POSITION_DROITE:
104       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
105       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
106       break;
107     default:
108       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
109       break;
110   }
111   ((*pcase).case_association).bool_hypotheque=false;
112   ((*pcase).case_association).int_valeur_hypotheque=750;
113   ((*pcase).case_association).int_prix=1500;
114   ((*pcase).case_association).pjoueur_joueur=NULL;
115   ((*pcase).case_association).int_autre_bureau=0;
116   ((*pcase).case_association).surf_detail=creation_case_detail_assoc(surf_ecran, int_type);
117   if(int_type==BDE) ((*pcase).case_association).surf_propriete=creation_case_propriete(surf_ecran, couleur, "B.D.E.'
118   else ((*pcase).case_association).surf_propriete=creation_case_propriete(surf_ecran, couleur, "B.D.S.");
119
120 }
```

8.3.2.5 void init_case_lieu_commun (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases lieu commun

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 122 du fichier initialisation.c.

Référencé par init_plateau().

```
123 {
124   rvb_couleur couleur;
125   couleur.rouge=255;
126   couleur.bleu=255;
127   couleur.vert=100;
128   int position;
129   position=int_numero/10;
130   (*pcase).int_type=int_type;
131   (*pcase).surf_image=creation_case_lieu_commun(surf_ecran, int_type);
132   (*pcase).int_position=position;
133   switch(int_type)
134   {
135     case LC_WC:
136       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-5*CASE_LARGEUR;
137       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
138       break;
139     case LC_ASCENSEUR:
140       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
141       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-5*CASE_LARGEUR;
142       break;
143     case LC_RU:
144       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+(4*CASE_LARGEUR);
145       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
146       break;
147     case LC_PARKING:
148       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
149       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+(4*CASE_LARGEUR);
150       break;
151     default:
152       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
153       break;
154   }
155   ((*pcase).case_lieu_commun).bool_hypotheque=false;
156   ((*pcase).case_lieu_commun).int_valeur_hypotheque=750;
157   ((*pcase).case_lieu_commun).int_prix=2000;
158   ((*pcase).case_lieu_commun).pjoueur_joueur=NULL;
```

```
159   ((*pcase).case_lieu_commun).int_suivant=0;
160   ((*pcase).case_lieu_commun).surf_detail=creation_case_detail_lc(surf_ecran, int_type);
161   switch(int_type)
162   {
163     case LC_WC:
164       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "W.C.");
165       break;
166     case LC_PARKING:
167       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "Parking");
168       break;
169     case LC_ASCENSEUR:
170       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "Ascenseur");
171       break;
172     case LC_RU:
173       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "R.U.");
174       break;
175     default:
176       break;
177   }
178 }
```

8.3.2.6 void init_case_administration (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases adimnistration

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 180 du fichier initialisation.c.

Référencé par init_plateau().

```
181 {
182   int position;
183   position=int_numero/10;
184   (*pcase).int_type=int_type;
185   (*pcase).surf_image=creation_case_bureau(surf_ecran, int_type, position);
186   (*pcase).int_position=position;
187   switch(position)
188   {
189     case POSITION_BAS:
190       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
191       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
192       break;
193     case POSITION_GAUCHE:
```

```
194       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
195       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
196       break;
197     case POSITION_HAUT:
198       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
199       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
200       break;
201     case POSITION_DROITE:
202       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
203       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
204       break;
205     default:
206       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
207       break;
208   }
209   ((*pcase).case_administration).surf_detail=NULL;
210 }
```

8.3.2.7 void init_case_soiree (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases soirée

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 212 du fichier initialisation.c.

Référencé par init_plateau().

```
213 {
214   int position;
215   position=int_numero/10;
216   (*pcase).int_type=int_type;
217   (*pcase).surf_image=creation_case_soiree(surf_ecran, int_type);
218   (*pcase).int_position=position;
219   switch(position)
220   {
221     case POSITION_BAS:
222       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
223       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
224       break;
225     case POSITION_GAUCHE:
226       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
227       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
228       break;
```

```
229     case POSITION_HAUT:
230       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
231       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
232       break;
233     case POSITION_DROITE:
234       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
235       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
236       break;
237     default:
238       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
239       break;
240   }
241
242   ((*pcase).case_soiree).int_prix=0;
243   ((*pcase).case_soiree).surf_detail=NULL;
244 }
```

8.3.2.8 void init_case_special (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases spéciales

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 246 du fichier initialisation.c.

Référencé par init_plateau().

```
247 {
248   int position;
249   position=int_numero/10;
250   (*pcase).int_type=int_type;
251   (*pcase).surf_image=creation_case_coin(surf_ecran, int_type);
252   (*pcase).int_position=position;
253   switch(int_type)
254   {
255     case SP_APPARTEMENT:
256       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
257       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
258       break;
259     case SP_BUREAU_LAURENCE:
260       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
261       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
262       break;
263     case SP_MACHINE_CAFE:
```

```
264       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
265       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
266
267       //il n'y a pas d'argent dans la machine à café pour le début
268       pcase->machine_a_cafe.int_argent=0;
269
270       break;
271     case SP_TABLEAU:
272       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
273       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
274       break;
275     default:
276       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
277       break;
278     }
279
280 }
```

8.3.2.9 void init_plateau (SDL_Surface * *surf_ecran*, rvb_couleur *couleurs*[8], cases ** *plateau*)

Définition à la ligne 282 du fichier initialisation.c.

Référencé par init_plateau_chargement(), et main().

```
283 {
284   init_case_special(surf_ecran, plateau[0], SP_APPARTEMENT, 0);
285   init_case_salle(surf_ecran, couleurs, plateau[1], SALLE, 1, 600, "Salle", "107", VIOLET, PRIX_NIVEAU_MIN);
286   init_case_administration(surf_ecran, plateau[2], BUREAU_NADEGE, 2);
287   init_case_salle(surf_ecran, couleurs, plateau[3], SALLE, 3, 600, "Salle", "109", VIOLET, PRIX_NIVEAU_MIN);
288   init_case_soiree(surf_ecran, plateau[4], SOIREE_AREA, 4);
289   init_case_lieu_commun(surf_ecran, plateau[5], LC_WC, 5);
290   init_case_salle(surf_ecran, couleurs, plateau[6], SALLE, 6, 1000, "Bureau de", "Arianne", BLEU_CIEL, PRIX_NIVEAU_M
291   init_case_administration(surf_ecran, plateau[7], BUREAU_KRYSTEL, 7);
292   init_case_salle(surf_ecran, couleurs, plateau[8], SALLE, 8, 1000, "Bureau de", "Muriel E.", BLEU_CIEL, PRIX_NIVEAU
293   init_case_salle(surf_ecran, couleurs, plateau[9], SALLE, 9, 1200, "Bureau de", "Anne-Marie", BLEU_CIEL, PRIX_NIVEA
294   init_case_special(surf_ecran, plateau[10], SP_BUREAU_LAURENCE, 10);
295   init_case_salle(surf_ecran, couleurs, plateau[11], SALLE, 11, 1400, "Bureau de", "Mathias", ROSE, PRIX_NIVEAU_MIN*
296   init_case_association(surf_ecran, plateau[12], BDS, 12);
297   init_case_salle(surf_ecran, couleurs, plateau[13], SALLE, 13, 1400, "Bureau de", "Muriel D.", ROSE, PRIX_NIVEAU_MI
298   init_case_salle(surf_ecran, couleurs, plateau[14], SALLE, 14, 1600, "Bureau de", "Alex", ROSE, PRIX_NIVEAU_MIN*2)
299   init_case_lieu_commun(surf_ecran, plateau[15], LC_ASCENSEUR, 15);
300   init_case_salle(surf_ecran, couleurs, plateau[16], SALLE, 16, 1800, "Salle", "304", ORANGE, PRIX_NIVEAU_MIN*2);
301   init_case_administration(surf_ecran, plateau[17], BUREAU_NADEGE, 17);
302   init_case_salle(surf_ecran, couleurs, plateau[18], SALLE, 18, 1800, "Salle", "305", ORANGE, PRIX_NIVEAU_MIN*2);
303   init_case_salle(surf_ecran, couleurs, plateau[19], SALLE, 19, 2000, "Salle", "308", ORANGE, PRIX_NIVEAU_MIN*2);
304   init_case_special(surf_ecran, plateau[20], SP_MACHINE_CAFE, 20);
305   init_case_salle(surf_ecran, couleurs, plateau[21], SALLE, 21, 2200, "Salle", "306", ROUGE, PRIX_NIVEAU_MIN*3);
306   init_case_administration(surf_ecran, plateau[22], BUREAU_KRYSTEL, 22);
307   init_case_salle(surf_ecran, couleurs, plateau[23], SALLE, 23, 2200, "Salle", "307", ROUGE, PRIX_NIVEAU_MIN*3);
308   init_case_salle(surf_ecran, couleurs, plateau[24], SALLE, 24, 2400, "Salle", "309", ROUGE, PRIX_NIVEAU_MIN*3);
309   init_case_lieu_commun(surf_ecran, plateau[25], LC_RU, 25);
310   init_case_salle(surf_ecran, couleurs, plateau[26], SALLE, 26, 2600, "Bureau de", "Yannick", JAUNE, PRIX_NIVEAU_MI
311   init_case_salle(surf_ecran, couleurs, plateau[27], SALLE, 27, 2600, "Bureau de", "Nisrine", JAUNE, PRIX_NIVEAU_MI
312   init_case_association(surf_ecran, plateau[28], BDE, 28);
313   init_case_salle(surf_ecran, couleurs, plateau[29], SALLE, 29, 2800, "Bureau de", "Astrid", JAUNE, PRIX_NIVEAU_MIN*
314   init_case_special(surf_ecran, plateau[30], SP_TABLEAU, 30);
315   init_case_salle(surf_ecran, couleurs, plateau[31], SALLE, 31, 3000, "Cafeteria", " ", VERT, PRIX_NIVEAU_MIN*4);
316   init_case_salle(surf_ecran, couleurs, plateau[32], SALLE, 32, 3000, "Amphi", " ", VERT, PRIX_NIVEAU_MIN*4);
317   init_case_administration(surf_ecran, plateau[33], BUREAU_NADEGE, 33);
318   init_case_salle(surf_ecran, couleurs, plateau[34], SALLE, 34, 3200, "Bureau de", "Florent", VERT, PRIX_NIVEAU_MIN*
319   init_case_lieu_commun(surf_ecran, plateau[35], LC_PARKING, 35);
320   init_case_administration(surf_ecran, plateau[36], BUREAU_KRYSTEL, 36);
321   init_case_salle(surf_ecran, couleurs, plateau[37], SALLE, 37, 3500, "Bureau de", "Pierre", BLEU, PRIX_NIVEAU_MIN*4
```

```
322   init_case_soiree(surf_ecran, plateau[38], SOIREE_GALA, 38);
323   init_case_salle(surf_ecran, couleurs, plateau[39], SALLE, 39, 4000, "Bureau de", "Nesim", BLEU, PRIX_NIVEAU_MIN*4)
324
325 }
```

8.3.2.10 joueur * init_joueur (char * *nom*, int *numero_joueur*)

initialise le joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

nom le nom du joueur

numero_joueur le numéro du joueur

Version :

0.0.1

Renvoie :

le joueur ainsi créé

Définition à la ligne 327 du fichier initialisation.c.

Référencé par init_anneau_joueur().

```
328 {
329   joueur* pjoueur;
330
331   pjoueur=new joueur;
332
333   strcpy(pjoueur->str_nom, nom);
334   pjoueur->int_laurence=0;
335   pjoueur->bool_laurence=false;
336   pjoueur->int_position=0;
337   pjoueur->int_argent=15000;
338   pjoueur->int_certificat=0;
339   pjoueur->int_double_tire=0;
340   pjoueur->bool_debut=false;
341   pjoueur->pjoueur_suivant=NULL;
342   pjoueur->surf_image=creation_joueur(numero_joueur+1);
343   pjoueur->propriete=init_anneau_possesions();
344
345   return(pjoueur);
346 }
```

8.3.2.11 void init_bureau_krystel (information * *bureau_de_krystel*[16])

initialise les cartes bureau de krystel

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

bureau_de_krystel tableau des cartes bureau de krystel

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 350 du fichier initialisation.c.

Référencé par main().

```
351 {
352   //pour chaque case du tableau
353   for(int i=0;i<16;i++)
354   {
355     //on allou en mémoire un espace pour une information
356     bureau_de_krystel[i]=new information;
357     //on test si l'allocation c'est bien déroulée
358     if (bureau_de_krystel[i]==NULL)
359     {
360       //cerr << "Erreur d'allocation" << endl;
361       exit(-1);
362     }
363   }
364   //on initialise le tableau d'information
365   strcpy(bureau_de_krystel[0]->texte,"Vous êtes convoqué au bureau de Nesim. Allez-y sans plus tarder.Si vous devez
366   bureau_de_krystel[0]->type=POSITION;
367   //bureau_de_krystel[0]->traitement=;
368   bureau_de_krystel[0]->valeur=39;
369
370   strcpy(bureau_de_krystel[1]->texte,"Vous avez été convoqué au bureau de Laurence. Ne passez pas par chez vous, ne
371   bureau_de_krystel[1]->type=CARTE_LAURENCE;
372   //bureau_de_krystel[1->->traitement=;
373   bureau_de_krystel[1]->valeur=10;
374
375   strcpy(bureau_de_krystel[2]->texte,"Vous avez raté votre DS d'info.Allez voir Florent! Si vous devez passer par ch
376   bureau_de_krystel[2]->type=POSITION;
377   //bureau_de_krystel[2]->traitement=;
378   bureau_de_krystel[2]->valeur=34;
379
380   strcpy(bureau_de_krystel[3]->texte,"3 en colle de physique! Allez voir Pierre dans son bureau! Si vous devez passe
381   bureau_de_krystel[3]->type=POSITION;
382   //bureau_de_krystel[3]->traitement=;
383   bureau_de_krystel[3]->valeur=37;
384
385   strcpy(bureau_de_krystel[4]->texte,"Vous devez payer vos frais de scolarité : 1500 Fintz.");
386   bureau_de_krystel[4]->type=ARGENT;
387   //bureau_de_krystel[4]->traitement=;
388   bureau_de_krystel[4]->valeur=-1500;
389
390   strcpy(bureau_de_krystel[5]->texte,"Vous avez laissé votre session ouverte : reculez de 3 cases.");
391   bureau_de_krystel[5]->type=POSITION;
392   //bureau_de_krystel[5]->traitement=;
393   bureau_de_krystel[5]->valeur=-3;
394
395   strcpy(bureau_de_krystel[6]->texte,"Il est 12 heures 14 minutes et 45 secondes, il vous reste 15 seconde pour plie
396   bureau_de_krystel[6]->type=POSITION;
397   //bureau_de_krystel[6]->traitement=;
398   bureau_de_krystel[6]->valeur=5;
399
400   strcpy(bureau_de_krystel[7]->texte,"Krystel a fait une erreur dans ses comptes : elle vous reverse 50 Fintz.");
401   bureau_de_krystel[7]->type=ARGENT;
402   //bureau_de_krystel[7]->traitement=;
403   bureau_de_krystel[7]->valeur=50;
404
405   strcpy(bureau_de_krystel[8]->texte,"Inspection de la CTI : versez 400 Fintz par niveeau pour chacune de vos salles
406   bureau_de_krystel[8]->type=ARGENT;
```

```
407   //bureau_de_krystel[8]->traitement=;
408   bureau_de_krystel[8]->valeur=-400;
409
410   strcpy(bureau_de_krystel[9]->texte,"Allez en salle 309. DS d'analyse ! Si vous devez passer par chez vous, touchez
411   bureau_de_krystel[9]->type=POSITION;
412   //bureau_de_krystel[9]->traitement=;
413   bureau_de_krystel[9]->valeur=24;
414
415
416   //=============================================================================================================
417   strcpy(bureau_de_krystel[10]->texte,"Dure journée de cours. Rentrez chez vous geeker un petit peu. N'oubliez pas v
418   bureau_de_krystel[10]->type=POSITION;
419   //bureau_de_krystel[10]->traitement=;
420   bureau_de_krystel[10]->valeur=0;
421
422   strcpy(bureau_de_krystel[11]->texte,"Certificat médical : ce certificat vous permet de sortir du bureau de Laurenc
423   bureau_de_krystel[11]->type=CERTIFICAT;
424   //bureau_de_krystel[11]->traitement=;
425   bureau_de_krystel[11]->valeur=1;
426
427   strcpy(bureau_de_krystel[12]->texte,"Vous avez reussi votre DS de Math! Vos parents vous donnent une petite récomp
428   bureau_de_krystel[12]->type=ARGENT;
429   //bureau_de_krystel[12]->traitement=;
430   bureau_de_krystel[12]->valeur=200;
431
432   strcpy(bureau_de_krystel[13]->texte,"Vous avez loupez 2H de math! Saleté de réveil ! Vous versez 100 Fintz");
433   bureau_de_krystel[13]->type=ARGENT;
434   //bureau_de_krystel[13]->traitement=;
435   bureau_de_krystel[13]->valeur=-100;
436
437   strcpy(bureau_de_krystel[14]->texte,"Felicitation votre prospection a Totoville vous rapporte 50 Fintz");
438   bureau_de_krystel[14]->type=ARGENT;
439   //bureau_de_krystel[14]->traitement=;
440   bureau_de_krystel[14]->valeur=50;
441
442   strcpy(bureau_de_krystel[15]->texte,"Aprés avoir pris une déroutée monumentale au ping-pong vous avez brisé la rac
443   bureau_de_krystel[15]->type=ARGENT;
444   //bureau_de_krystel[15]->traitement=;
445   bureau_de_krystel[15]->valeur=-50;
446
447   melanger_cartes(bureau_de_krystel);
448 }
```

8.3.2.12 void init_bureau_nadege (information * *bureau_de_nadege*[16])

initialise les cartes bureau de nadège

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

bureau_de_nadege tableau des cartes bureau de nadege

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 450 du fichier initialisation.c.

Référencé par main().

```
451 {
452   //pour chaque case du tableau
453   for(int i=0;i<16;i++)
454   {
455     //on allou en mémoire un espace pour une information
456     bureau_de_nadege[i]=new information;
457     //on test si l'allocation c'est bien déroulée
458     if (bureau_de_nadege[i]==NULL)
459     {
460       //cerr << "Erreur d'allocation" << endl;
461       exit(-1);
462     }
463   }
464   //on initialise le tableau d'information
465   strcpy(bureau_de_nadege[0]->texte,"Vous venez de planter le serveur de l'école. Allez voir alex! Si vous devez pas
466   bureau_de_nadege[0]->type=POSITION;
467   bureau_de_nadege[0]->valeur=14;
468
469   strcpy(bureau_de_nadege[1]->texte,"Fait avez fait sauté les plombs de l'école. Payez le montant des réparation : 1
470   bureau_de_nadege[1]->type=ARGENT;
471   bureau_de_nadege[1]->valeur=-100;
472
473   strcpy(bureau_de_nadege[2]->texte,"Votre assiduité en cours vous oblige é racheter des cahier pour prendre en note
474   bureau_de_nadege[2]->type=ARGENT;
475   bureau_de_nadege[2]->valeur=-50;
476
477   strcpy(bureau_de_nadege[3]->texte,"Vous venez de prendre -0.5 au dernier DS d'analyse. Astrid vous laisse le choi
478   bureau_de_nadege[3]->type=ASTRID;
479   bureau_de_nadege[3]->valeur=-50;
480
481   strcpy(bureau_de_nadege[4]->texte,"Votre taux d'absentéisme est inconcevable. Vous étes convoqué dans le bureau de
482   bureau_de_nadege[4]->type=CARTE_LAURENCE;
483   bureau_de_nadege[4]->valeur=10;
484
485   strcpy(bureau_de_nadege[5]->texte,"Certificat médical : ce certificat vous permet de sortir du bureau de Laurence
486   bureau_de_nadege[5]->type=CERTIFICAT;
487   bureau_de_nadege[5]->valeur=3;
488
489   strcpy(bureau_de_nadege[6]->texte,"Le dernier cours de la journée est annulé.Vous rentrez tranquilement chez vous"
490   bureau_de_nadege[6]->type=POSITION;
491   bureau_de_nadege[6]->valeur=0;
492
493   strcpy(bureau_de_nadege[7]->texte,"Inspection du comité des grandes écoles. Payez 500 Fintz par niveau pour nettoy
494   bureau_de_nadege[7]->type=CTI;
495   bureau_de_nadege[7]->valeur=50;
496
497   strcpy(bureau_de_nadege[8]->texte,"Inspection de la CTI : versez 400 Fintz par niveeau pour chacune de vos salles"
498   bureau_de_nadege[8]->type=CTI;
499   bureau_de_nadege[8]->valeur=-400;
500
501   strcpy(bureau_de_nadege[9]->texte,"La machine a café a encore eu des ratées. Vous gagnez 50 Fintz");
502   bureau_de_nadege[9]->type=ARGENT;
503   bureau_de_nadege[9]->valeur=50;
504
505   strcpy(bureau_de_nadege[10]->texte,"Un p*t*in de poteau de m*** a encore traversé la route en dehors des clous. Pa
506   bureau_de_nadege[10]->type=ARGENT;
507   bureau_de_nadege[10]->valeur=300;
508
509   strcpy(bureau_de_nadege[11]->texte,"C'est votre anniversaire! Chacun de vos adversaires vous donnes 100 Fintz");
510   bureau_de_nadege[11]->type=ANNIVERSAIRE;
511   bureau_de_nadege[11]->valeur=100;
512
513   strcpy(bureau_de_nadege[12]->texte,"Contrôle d'alcoolémie! Vous avez été contrôlé positif avec 3.5 g/L après avoi
514   bureau_de_nadege[12]->type=ARGENT;
515   bureau_de_nadege[12]->valeur=1000;
516
517   strcpy(bureau_de_nadege[13]->texte,"Vous organisé une visite guidée de la maison de Forky. Chacun des élèves doit
```

```
518   bureau_de_nadege[13]->type=ANNIVERSAIRE;
519   bureau_de_nadege[13]->valeur=-150;
520
521   strcpy(bureau_de_nadege[14]->texte,"Felicitation votre prospection a Totoland vous rapporte 150 Fintz");
522   bureau_de_nadege[14]->type=ARGENT;
523   bureau_de_nadege[14]->valeur=150;
524
525   strcpy(bureau_de_nadege[15]->texte,"Vous participez a une LAN et après 20h de jeu consécutives, dans un élan de co
526   bureau_de_nadege[15]->type=ARGENT;
527   bureau_de_nadege[15]->valeur=-200;
528
529   melanger_cartes(bureau_de_nadege);
530 }
```

8.3.2.13 possession * init_anneau_possesions ()

initialise l'anneau des possesions du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Version :

0.0.1

Renvoie :

NULL

Définition à la ligne 532 du fichier initialisation.c.

Référencé par init_joueur(), et init_joueur_chargement().

```
533 {
534     return(NULL);
535 }
```

8.3.2.14 possession * creation_possession (cases * *propriete*)

ajoute de façon ordonnée une propriété dans la liste

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

propriete propriété a ajouter

Version :

0.0.1

Renvoie :

la tète de la chaine

Définition à la ligne 538 du fichier initialisation.c.

Référencé par insertion_bonne_place_propriete().

```
539 {
540   possession* nv_possession;
541
542   nv_possession=new possession;
543
544   if (nv_possession==NULL)
545   {
546     cerr << "Erreur d'allocation memoire" << endl;
547     exit(-1);
548   }
549   else
550   {
551     nv_possession->propriete=propriete;
552     nv_possession->suivant=NULL;
553   }
554   return(nv_possession);
555
556 }
```

8.3.2.15 joueur * init_anneau_joueur (int *nombre_joueur*, char ** *str_nom_joueur*)

initialise le ring des joueurs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

nombre_joueur nombre de joueurs

str_nom_joueur tableau du nom des joueurs

Version :

0.0.1

Renvoie :

un des joueurs de l'anneau

Définition à la ligne 558 du fichier initialisation.c.

Référencé par main().

```
559 {
560   joueur* p_joueur_premier;
561
562   joueur* p_joueur_precedent;
563   joueur* p_joueur_encours;
564
565   //on crée le premier joueur
566   p_joueur_premier=init_joueur(str_nom_joueur[0],0);
567   p_joueur_precedent=p_joueur_premier;
568
569   for(int i=1;i<nombre_joueur;i++)
570   {
571     p_joueur_encours=init_joueur(str_nom_joueur[i], i);
572     p_joueur_precedent->pjoueur_suivant=p_joueur_encours;
573     p_joueur_precedent=p_joueur_encours;
574   }
575   p_joueur_precedent->pjoueur_suivant=p_joueur_premier;
576   return(p_joueur_premier);
577 }
```

8.3.2.16 cases **joueur * init_anneau_joueur_chargement (int *nombre_joueur*, ifstream * *file*, cases ** *plateau*)

créer la liste des joueurs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

file fichier de sauvegarde

plateau plateau de jeu

nombre_joueur nombre de joueur

Version :

0.0.1

Renvoie :

le premier joueur à joué

Définition à la ligne 579 du fichier initialisation.c.

Référencé par chargement().

```
580 {
581   joueur* p_joueur_premier;
582
583   joueur* p_joueur_precedent;
584   joueur* p_joueur_encours;
585
586   //on crée le premier joueur
587   p_joueur_premier=init_joueur_chargement(0, file,plateau);
588   p_joueur_precedent=p_joueur_premier;
589
590   for(int i=1;i<nombre_joueur;i++)
591   {
592     p_joueur_encours=init_joueur_chargement(i,file,plateau);
593     p_joueur_precedent->pjoueur_suivant=p_joueur_encours;
594     p_joueur_precedent=p_joueur_encours;
595   }
596   p_joueur_precedent->pjoueur_suivant=p_joueur_premier;
597
598   return(p_joueur_premier);
599 }
```

8.3.2.17 joueur * init_joueur_chargement (int *numero_joueur*, ifstream * *file*, cases ** *plateau*)

créer un joueur à partir de la sauvagarde

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

file fichier de sauvegarde

plateau plateau de jeu

numero_joueur numéro du joueur

Version :

0.0.1

Renvoie :

le premier joueur à joué

Définition à la ligne 601 du fichier initialisation.c.

Référencé par init_anneau_joueur_chargement().

```
602 {
603   joueur* pjoueur;
604   int int_propriete_tmp;
605 
606   pjoueur=new joueur;
607 
608   *file >> pjoueur->str_nom;
609   *file >> pjoueur->int_position;
610   *file >> pjoueur->int_argent;
611   *file >> pjoueur->int_certificat;
612   *file >> pjoueur->bool_debut;
613   *file >> pjoueur->int_double_tire;
614   *file >> pjoueur->bool_laurence;
615   *file >> pjoueur->int_laurence;
616 
617   pjoueur->pjoueur_suivant=NULL;
618   pjoueur->surf_image=creation_joueur(numero_joueur+1);
619   pjoueur->propriete=init_anneau_possesions();
620 
621   *file >> int_propriete_tmp;
622 
623   while (int_propriete_tmp != -1)
624   {
625     pjoueur->propriete=insertion_bonne_place_propriete(pjoueur->propriete,plateau[int_propriete_tmp]);
626     *file >> int_propriete_tmp;
627   }
628   return(pjoueur);
629 }
```

8.3.2.18 cases ** init_plateau_chargement (ifstream * *file*, cases ** *plateau*, rvb_couleur *couleurs*[8], SDL_Surface * *surf_ecran*)

charge les données du plateau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

file fichier de sauvegarde

plateau plateau de jeu

couleurs tableau de couleurs

surf_ecran surface de l'écran

Version :

0.0.1

Renvoie :

le plateau

Définition à la ligne 631 du fichier initialisation.c.

Référencé par chargement().

```
632 {
633   int tmp;
634   int i;
635 
636   i=0;
637 
638   //on allou d'abord le prix du parc gratuit
639   *file >> plateau[20]->machine_a_cafe.int_argent;
640 
641   init_plateau(surf_ecran, couleurs, plateau);
642   for (i = 0; i < 40; i++)
643   {
644     *file >> tmp;
645     if (tmp!=-1)
646     {
647       plateau[i]->case_salle.int_niveau=tmp;
648     }
649   }
650 
651   return(plateau);
652 
653 }
```

8.4 Référence du fichier initialisation.h

en tete des fonctions necessaires à l'initialisation du monopoly

Fonctions

- **rvb_couleur init_rvb_couleur** (int rouge, int vert, int bleu)
  initialise une structure joueur en fonction des paramètre

- void **init_tab_rvb_couleur** (**rvb_couleur** couleur[8])
  initialise le tableau de couleur

- void **init_case_salle** (SDL_Surface *surf_ecran, **rvb_couleur** couleurs[8], **cases** *pcase, int int_type, int int_numero, int int_prix, char *str_nom1, char *str_nom2, int int_groupe, int int_prix_niveau)
  initialise les cases salles

- void **init_case_association** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases associations

- void **init_case_lieu_commun** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases lieu commun

- void **init_case_soiree** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases soirée

- void **init_case_special** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases spéciales

- void **init_case_administration** (SDL_Surface *surf_ecran, **cases** *pcase, int int_type, int int_numero)
  initialise les cases adimnistration

- **possession** * **init_anneau_possesions** ()
  initialise l'anneau des possesions du joueur

- **possession** * **creation_possession** (**cases** *propriete)
  ajoute de façon ordonnée une propriété dans la liste

- **joueur** * **init_joueur** (char *nom, int numero_joueur)
  initialise le joueur

- void **init_bureau_krystel** (**information** *bureau_de_krystel[16])
  initialise les cartes bureau de krystel

- void **init_bureau_nadege** (**information** *bureau_de_nadege[16])
  initialise les cartes bureau de nadège

- void **init_plateau** (SDL_Surface *surf_ecran, **rvb_couleur** couleurs[8], **cases** **plateau)
- **joueur** * **init_anneau_joueur** (int nombre_joueur, char **str_nom_joueur)
  initialise le ring des joueurs

- **cases** ** **init_plateau_chargement** (ifstream *file, **cases** **plateau, **rvb_couleur** couleurs[8], SDL_Surface *surf_ecran)
  charge les données du plateau

– **joueur** * **init_anneau_joueur_chargement** (int nombre_joueur, ifstream *file, **cases** **plateau)
créer la liste des joueurs

– **joueur** * **init_joueur_chargement** (int numero_joueur, ifstream *file, **cases** **plateau)
créer un joueur à partir de la sauvagarde

8.4.1 Description détaillée

en tete des fonctions necessaires à l'initialisation du monopoly

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

0.0.1

Définition dans le fichier **initialisation.h**.

8.4.2 Documentation des fonctions

8.4.2.1 rvb_couleur init_rvb_couleur (int *rouge*, int *vert*, int *bleu*)

initialise une structure joueur en fonction des paramètre

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

rouge valeur pour la valeur rouge
vert valeur pour la valeur vert
bleu valeur pour la valeur bleu

Version :

0.0.1

Renvoie :

la couleur

Définition à la ligne 11 du fichier initialisation.c.

Référencé par init_tab_rvb_couleur().

```
12 {
13   rvb_couleur couleur;
14   couleur.rouge=rouge;
15   couleur.vert=vert;
16   couleur.bleu=bleu;
17   return(couleur);
18 }
```

8.4.2.2 void init_tab_rvb_couleur (rvb_couleur *couleur*[8])

initialise le tableau de couleur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

couleur tableau de couleur

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 20 du fichier initialisation.c.

Référencé par main().

```
21 {
22   couleur[VIOLET]=init_rvb_couleur(205,14,207);
23   couleur[BLEU_CIEL]=init_rvb_couleur(101,232,215);
24   couleur[ROSE]=init_rvb_couleur(251,138,244);
25   couleur[ORANGE]=init_rvb_couleur(255,130,8);
26   couleur[ROUGE]=init_rvb_couleur(255,0,0);
27   couleur[JAUNE]=init_rvb_couleur(255,255,0);
28   couleur[VERT]=init_rvb_couleur(0,255,0);
29   couleur[BLEU]=init_rvb_couleur(0,0,255);
30 }
```

8.4.2.3 void init_case_salle (SDL_Surface * *surf_ecran*, rvb_couleur *couleurs*[8], cases * *pcase*, int *int_type*, int *int_numero*, int *int_prix*, char * *str_nom1*, char * *str_nom2*, int *int_groupe*, int *int_prix_niveau*)

initialise les cases salles

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

couleurs tableau de couleur

pcase case en cours de création

int_type le type de case

int_numero numero de la case

int_prix prix de la case

str_nom1 nom de la case

str_nom2 nom de la case

int_groupe groupe de la case

int_prix_niveau prix pour augmenter d'un niveau

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 32 du fichier initialisation.c.

Référencé par init_plateau().

```
33 {
34   int position;
35   char* str_nom;
36   position=int_numero/10;
37   (*pcase).int_type=int_type;
38   (*pcase).surf_image=creation_case(surf_ecran, couleurs[int_groupe], str_nom1, str_nom2, int_prix, position);
39   (*pcase).int_position=position;
40   switch(position)
41   {
42     case POSITION_BAS:
43       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
44       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
45       break;
46     case POSITION_GAUCHE:
47       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
48       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
49       break;
50     case POSITION_HAUT:
51       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
52       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
53       break;
54     case POSITION_DROITE:
55       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
56       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
57       break;
58     default:
59       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
60       break;
61   }
62   ((*pcase).case_salle).bool_hypotheque=false;
63   ((*pcase).case_salle).int_valeur_hypotheque=0;
64   ((*pcase).case_salle).int_prix=int_prix;
65   ((*pcase).case_salle).int_prix_niveau=int_prix_niveau;
66   ((*pcase).case_salle).pjoueur_joueur=NULL;
67   ((*pcase).case_salle).int_niveau=0;
68   ((*pcase).case_salle).int_groupe=int_groupe;
69
70   str_nom = new char[strlen(str_nom1)+strlen(str_nom2)+2];
71   sprintf(str_nom, "%s %s",str_nom1, str_nom2);
72   ((*pcase).case_salle).surf_detail=creation_case_detail(surf_ecran, couleurs[int_groupe], str_nom, int_prix/10, int
73   ((*pcase).case_salle).surf_propriete=creation_case_propriete(surf_ecran, couleurs[int_groupe], str_nom);
74   delete[] str_nom;
75   ((*pcase).case_salle).bool_etat=1;
76 }
```

8.4.2.4 void init_case_association (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases associations

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 78 du fichier initialisation.c.

Référencé par init_plateau().

```
79 {
80   rvb_couleur couleur;
81   couleur.rouge=255;
82   couleur.bleu=100;
83   couleur.vert=255;
84   int position;
85   position=int_numero/10;
86   (*pcase).int_type=int_type;
87   (*pcase).surf_image=creation_case_association(surf_ecran, int_type);
88   (*pcase).int_position=position;
89   switch(position)
90   {
91     case POSITION_BAS:
92       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
93       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
94       break;
95     case POSITION_GAUCHE:
96       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
97       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
98       break;
99     case POSITION_HAUT:
100       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
101       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
102       break;
103     case POSITION_DROITE:
104       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
105       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
106       break;
107     default:
108       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
109       break;
110   }
111   ((*pcase).case_association).bool_hypotheque=false;
112   ((*pcase).case_association).int_valeur_hypotheque=750;
113   ((*pcase).case_association).int_prix=1500;
114   ((*pcase).case_association).pjoueur_joueur=NULL;
115   ((*pcase).case_association).int_autre_bureau=0;
116   ((*pcase).case_association).surf_detail=creation_case_detail_assoc(surf_ecran, int_type);
117   if(int_type==BDE) ((*pcase).case_association).surf_propriete=creation_case_propriete(surf_ecran, couleur, "B.D.E.'
118   else ((*pcase).case_association).surf_propriete=creation_case_propriete(surf_ecran, couleur, "B.D.S.");
119
120 }
```

8.4.2.5 void init_case_lieu_commun (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases lieu commun

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 122 du fichier initialisation.c.

Référencé par init_plateau().

```
123 {
124   rvb_couleur couleur;
125   couleur.rouge=255;
126   couleur.bleu=255;
127   couleur.vert=100;
128   int position;
129   position=int_numero/10;
130   (*pcase).int_type=int_type;
131   (*pcase).surf_image=creation_case_lieu_commun(surf_ecran, int_type);
132   (*pcase).int_position=position;
133   switch(int_type)
134   {
135     case LC_WC:
136       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-5*CASE_LARGEUR;
137       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
138       break;
139     case LC_ASCENSEUR:
140       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
141       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-5*CASE_LARGEUR;
142       break;
143     case LC_RU:
144       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+(4*CASE_LARGEUR);
145       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
146       break;
147     case LC_PARKING:
148       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
149       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+(4*CASE_LARGEUR);
150       break;
151     default:
152       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
153       break;
154   }
155   ((*pcase).case_lieu_commun).bool_hypotheque=false;
156   ((*pcase).case_lieu_commun).int_valeur_hypotheque=750;
157   ((*pcase).case_lieu_commun).int_prix=2000;
158   ((*pcase).case_lieu_commun).pjoueur_joueur=NULL;
```

```
159   ((*pcase).case_lieu_commun).int_suivant=0;
160   ((*pcase).case_lieu_commun).surf_detail=creation_case_detail_lc(surf_ecran, int_type);
161   switch(int_type)
162   {
163     case LC_WC:
164       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "W.C.");
165       break;
166     case LC_PARKING:
167       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "Parking");
168       break;
169     case LC_ASCENSEUR:
170       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "Ascenseur");
171       break;
172     case LC_RU:
173       ((*pcase).case_lieu_commun).surf_propriete=creation_case_propriete(surf_ecran, couleur, "R.U.");
174       break;
175     default:
176       break;
177   }
178 }
```

8.4.2.6 void init_case_soiree (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases soirée

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 212 du fichier initialisation.c.

Référencé par init_plateau().

```
213 {
214   int position;
215   position=int_numero/10;
216   (*pcase).int_type=int_type;
217   (*pcase).surf_image=creation_case_soiree(surf_ecran, int_type);
218   (*pcase).int_position=position;
219   switch(position)
220   {
221     case POSITION_BAS:
222       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
223       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
224       break;
225     case POSITION_GAUCHE:
```

```
226       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
227       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
228       break;
229     case POSITION_HAUT:
230       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
231       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
232       break;
233     case POSITION_DROITE:
234       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
235       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
236       break;
237     default:
238       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
239       break;
240   }
241
242   ((*pcase).case_soiree).int_prix=0;
243   ((*pcase).case_soiree).surf_detail=NULL;
244 }
```

8.4.2.7 void init_case_special (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases spéciales

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 246 du fichier initialisation.c.

Référencé par init_plateau().

```
247 {
248   int position;
249   position=int_numero/10;
250   (*pcase).int_type=int_type;
251   (*pcase).surf_image=creation_case_coin(surf_ecran, int_type);
252   (*pcase).int_position=position;
253   switch(int_type)
254   {
255     case SP_APPARTEMENT:
256       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
257       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
258       break;
259     case SP_BUREAU_LAURENCE:
260       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
```

```
261       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
262       break;
263     case SP_MACHINE_CAFE:
264       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
265       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
266
267       //il n'y a pas d'argent dans la machine à café pour le début
268       pcase->machine_a_cafe.int_argent=0;
269
270       break;
271     case SP_TABLEAU:
272       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
273       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
274       break;
275     default:
276       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
277       break;
278     }
279
280 }
```

8.4.2.8 void init_case_administration (SDL_Surface * *surf_ecran*, cases * *pcase*, int *int_type*, int *int_numero*)

initialise les cases adimnistration

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pcase case en cours de création

int_type type de la case

int_numero numéro de la case

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 180 du fichier initialisation.c.

Référencé par init_plateau().

```
181 {
182   int position;
183   position=int_numero/10;
184   (*pcase).int_type=int_type;
185   (*pcase).surf_image=creation_case_bureau(surf_ecran, int_type, position);
186   (*pcase).int_position=position;
187   switch(position)
188   {
189     case POSITION_BAS:
190       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
191       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
192       break;
193     case POSITION_GAUCHE:
```

```
194       (*pcase).rect_coordonnees.x=POS_X_PLATEAU;
195       (*pcase).rect_coordonnees.y=(POS_Y_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR-(int_numero%10*CASE_LARGEUR);
196       break;
197     case POSITION_HAUT:
198       (*pcase).rect_coordonnees.x=POS_X_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
199       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU;
200       break;
201     case POSITION_DROITE:
202       (*pcase).rect_coordonnees.x=(POS_X_PLATEAU+TAILLE_PLATEAU)-CASE_HAUTEUR;
203       (*pcase).rect_coordonnees.y=POS_Y_PLATEAU+CASE_HAUTEUR+((int_numero%10-1)*CASE_LARGEUR);
204       break;
205     default:
206       cerr << "Erreur de position pour la case numéro" << int_numero << endl;
207       break;
208   }
209   ((*pcase).case_administration).surf_detail=NULL;
210 }
```

8.4.2.9 possession* init_anneau_possesions ()

initialise l'anneau des possesions du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Version :

0.0.1

Renvoie :

NULL

Définition à la ligne 532 du fichier initialisation.c.

Référencé par init_joueur(), et init_joueur_chargement().

```
533 {
534     return(NULL);
535 }
```

8.4.2.10 possession* creation_possession (cases * *propriete*)

ajoute de façon ordonnée une propriété dans la liste

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

propriete propriété a ajouter

Version :

0.0.1

Renvoie :

la tète de la chaine

Définition à la ligne 538 du fichier initialisation.c.

Référencé par insertion_bonne_place_propriete().

```
539 {
540   possession* nv_possession;
541
542   nv_possession=new possession;
543
544   if (nv_possession==NULL)
545   {
546     cerr << "Erreur d'allocation memoire" << endl;
547     exit(-1);
548   }
549   else
550   {
551     nv_possession->propriete=propriete;
552     nv_possession->suivant=NULL;
553   }
554   return(nv_possession);
555
556 }
```

8.4.2.11 joueur* init_joueur (char * *nom*, int *numero_joueur*)

initialise le joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

nom le nom du joueur

numero_joueur le numéro du joueur

Version :

0.0.1

Renvoie :

le joueur ainsi créé

Définition à la ligne 327 du fichier initialisation.c.

Référencé par init_anneau_joueur().

```
328 {
329   joueur* pjoueur;
330
331   pjoueur=new joueur;
332
333   strcpy(pjoueur->str_nom, nom);
334   pjoueur->int_laurence=0;
335   pjoueur->bool_laurence=false;
336   pjoueur->int_position=0;
337   pjoueur->int_argent=15000;
338   pjoueur->int_certificat=0;
339   pjoueur->int_double_tire=0;
340   pjoueur->bool_debut=false;
341   pjoueur->pjoueur_suivant=NULL;
342   pjoueur->surf_image=creation_joueur(numero_joueur+1);
```

```
343   pjoueur->propriete=init_anneau_possesions();
344
345   return(pjoueur);
346 }
```

8.4.2.12 void init_bureau_krystel (information * *bureau_de_krystel*[16])

initialise les cartes bureau de krystel

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

bureau_de_krystel tableau des cartes bureau de krystel

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 350 du fichier initialisation.c.

Référencé par main().

```
351 {
352   //pour chaque case du tableau
353   for(int i=0;i<16;i++)
354   {
355     //on allou en mémoire un espace pour une information
356     bureau_de_krystel[i]=new information;
357     //on test si l'allocation c'est bien déroulée
358     if (bureau_de_krystel[i]==NULL)
359     {
360       //cerr << "Erreur d'allocation" << endl;
361       exit(-1);
362     }
363   }
364   //on initialise le tableau d'information
365   strcpy(bureau_de_krystel[0]->texte,"Vous êtes convoqué au bureau de Nesim. Allez-y sans plus tarder.Si vous devez
366   bureau_de_krystel[0]->type=POSITION;
367   //bureau_de_krystel[0]->traitement=;
368   bureau_de_krystel[0]->valeur=39;
369
370   strcpy(bureau_de_krystel[1]->texte,"Vous avez été convoqué au bureau de Laurence. Ne passez pas par chez vous, ne
371   bureau_de_krystel[1]->type=CARTE_LAURENCE;
372   //bureau_de_krystel[1->->traitement=;
373   bureau_de_krystel[1]->valeur=10;
374
375   strcpy(bureau_de_krystel[2]->texte,"Vous avez raté votre DS d'info.Allez voir Florent! Si vous devez passer par ch
376   bureau_de_krystel[2]->type=POSITION;
377   //bureau_de_krystel[2]->traitement=;
378   bureau_de_krystel[2]->valeur=34;
379
380   strcpy(bureau_de_krystel[3]->texte,"3 en colle de physique! Allez voir Pierre dans son bureau! Si vous devez passe
381   bureau_de_krystel[3]->type=POSITION;
382   //bureau_de_krystel[3]->traitement=;
383   bureau_de_krystel[3]->valeur=37;
384
385   strcpy(bureau_de_krystel[4]->texte,"Vous devez payer vos frais de scolarité : 1500 Fintz.");
```

```
386   bureau_de_krystel[4]->type=ARGENT;
387   //bureau_de_krystel[4]->traitement=;
388   bureau_de_krystel[4]->valeur=-1500;
389
390   strcpy(bureau_de_krystel[5]->texte,"Vous avez laissé votre session ouverte : reculez de 3 cases.");
391   bureau_de_krystel[5]->type=POSITION;
392   //bureau_de_krystel[5]->traitement=;
393   bureau_de_krystel[5]->valeur=-3;
394
395   strcpy(bureau_de_krystel[6]->texte,"Il est 12 heures 14 minutes et 45 secondes, il vous reste 15 seconde pour plie
396   bureau_de_krystel[6]->type=POSITION;
397   //bureau_de_krystel[6]->traitement=;
398   bureau_de_krystel[6]->valeur=5;
399
400   strcpy(bureau_de_krystel[7]->texte,"Krystel a fait une erreur dans ses comptes : elle vous reverse 50 Fintz.");
401   bureau_de_krystel[7]->type=ARGENT;
402   //bureau_de_krystel[7]->traitement=;
403   bureau_de_krystel[7]->valeur=50;
404
405   strcpy(bureau_de_krystel[8]->texte,"Inspection de la CTI : versez 400 Fintz par niveeau pour chacune de vos salles
406   bureau_de_krystel[8]->type=ARGENT;
407   //bureau_de_krystel[8]->traitement=;
408   bureau_de_krystel[8]->valeur=-400;
409
410   strcpy(bureau_de_krystel[9]->texte,"Allez en salle 309. DS d'analyse ! Si vous devez passer par chez vous, touchez
411   bureau_de_krystel[9]->type=POSITION;
412   //bureau_de_krystel[9]->traitement=;
413   bureau_de_krystel[9]->valeur=24;
414
415
416   //=============================================================================================================
417   strcpy(bureau_de_krystel[10]->texte,"Dure journée de cours. Rentrez chez vous geeker un petit peu. N'oubliez pas v
418   bureau_de_krystel[10]->type=POSITION;
419   //bureau_de_krystel[10]->traitement=;
420   bureau_de_krystel[10]->valeur=0;
421
422   strcpy(bureau_de_krystel[11]->texte,"Certificat médical : ce certificat vous permet de sortir du bureau de Lauren
423   bureau_de_krystel[11]->type=CERTIFICAT;
424   //bureau_de_krystel[11]->traitement=;
425   bureau_de_krystel[11]->valeur=1;
426
427   strcpy(bureau_de_krystel[12]->texte,"Vous avez reussi votre DS de Math! Vos parents vous donnent une petite récomp
428   bureau_de_krystel[12]->type=ARGENT;
429   //bureau_de_krystel[12]->traitement=;
430   bureau_de_krystel[12]->valeur=200;
431
432   strcpy(bureau_de_krystel[13]->texte,"Vous avez loupez 2H de math! Saleté de réveil ! Vous versez 100 Fintz");
433   bureau_de_krystel[13]->type=ARGENT;
434   //bureau_de_krystel[13]->traitement=;
435   bureau_de_krystel[13]->valeur=-100;
436
437   strcpy(bureau_de_krystel[14]->texte,"Felicitation votre prospection a Totoville vous rapporte 50 Fintz");
438   bureau_de_krystel[14]->type=ARGENT;
439   //bureau_de_krystel[14]->traitement=;
440   bureau_de_krystel[14]->valeur=50;
441
442   strcpy(bureau_de_krystel[15]->texte,"Aprés avoir pris une déroutée monumentale au ping-pong vous avez brisé la rac
443   bureau_de_krystel[15]->type=ARGENT;
444   //bureau_de_krystel[15]->traitement=;
445   bureau_de_krystel[15]->valeur=-50;
446
447   melanger_cartes(bureau_de_krystel);
448 }
```

8.4.2.13 void init_bureau_nadege (information * *bureau_de_nadege*[16])

initialise les cartes bureau de nadège

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

bureau_de_nadege tableau des cartes bureau de nadege

Version :

0.0.1

Renvoie :

rien

Définition à la ligne 450 du fichier initialisation.c.

Référencé par main().

```
451 {
452   //pour chaque case du tableau
453   for(int i=0;i<16;i++)
454   {
455     //on allou en mémoire un espace pour une information
456     bureau_de_nadege[i]=new information;
457     //on test si l'allocation c'est bien déroulée
458     if (bureau_de_nadege[i]==NULL)
459     {
460       //cerr << "Erreur d'allocation" << endl;
461       exit(-1);
462     }
463   }
464   //on initialise le tableau d'information
465   strcpy(bureau_de_nadege[0]->texte,"Vous venez de planter le serveur de l'école. Allez voir alex! Si vous devez pas
466   bureau_de_nadege[0]->type=POSITION;
467   bureau_de_nadege[0]->valeur=14;
468
469   strcpy(bureau_de_nadege[1]->texte,"Fait avez fait sauté les plombs de l'école. Payez le montant des réparation : 1
470   bureau_de_nadege[1]->type=ARGENT;
471   bureau_de_nadege[1]->valeur=-100;
472
473   strcpy(bureau_de_nadege[2]->texte,"Votre assiduité en cours vous oblige é racheter des cahier pour prendre en note
474   bureau_de_nadege[2]->type=ARGENT;
475   bureau_de_nadege[2]->valeur=-50;
476
477   strcpy(bureau_de_nadege[3]->texte,"Vous venez de prendre -0.5 au dernier DS d'analyse. Astrid vous laisse le choi
478   bureau_de_nadege[3]->type=ASTRID;
479   bureau_de_nadege[3]->valeur=-50;
480
481   strcpy(bureau_de_nadege[4]->texte,"Votre taux d'absentéisme est inconcevable. Vous étes convoqué dans le bureau de
482   bureau_de_nadege[4]->type=CARTE_LAURENCE;
483   bureau_de_nadege[4]->valeur=10;
484
485   strcpy(bureau_de_nadege[5]->texte,"Certificat médical : ce certificat vous permet de sortir du bureau de Laurence
486   bureau_de_nadege[5]->type=CERTIFICAT;
487   bureau_de_nadege[5]->valeur=3;
488
489   strcpy(bureau_de_nadege[6]->texte,"Le dernier cours de la journée est annulé.Vous rentrez tranquilement chez vous"
490   bureau_de_nadege[6]->type=POSITION;
491   bureau_de_nadege[6]->valeur=0;
492
```

```
493   strcpy(bureau_de_nadege[7]->texte,"Inspection du comité des grandes écoles. Payez 500 Fintz par niveau pour nettoy
494   bureau_de_nadege[7]->type=CTI;
495   bureau_de_nadege[7]->valeur=50;
496
497   strcpy(bureau_de_nadege[8]->texte,"Inspection de la CTI : versez 400 Fintz par niveeau pour chacune de vos salles'
498   bureau_de_nadege[8]->type=CTI;
499   bureau_de_nadege[8]->valeur=-400;
500
501   strcpy(bureau_de_nadege[9]->texte,"La machine a café a encore eu des ratées. Vous gagnez 50 Fintz");
502   bureau_de_nadege[9]->type=ARGENT;
503   bureau_de_nadege[9]->valeur=50;
504
505   strcpy(bureau_de_nadege[10]->texte,"Un p*t*in de poteau de m*** a encore traversé la route en dehors des clous. Pa
506   bureau_de_nadege[10]->type=ARGENT;
507   bureau_de_nadege[10]->valeur=300;
508
509   strcpy(bureau_de_nadege[11]->texte,"C'est votre anniversaire! Chacun de vos adversaires vous donnes 100 Fintz");
510   bureau_de_nadege[11]->type=ANNIVERSAIRE;
511   bureau_de_nadege[11]->valeur=100;
512
513   strcpy(bureau_de_nadege[12]->texte,"Contrôle d'alcoolémie! Vous avez été contrôlé positif avec 3.5 g/L après avoi
514   bureau_de_nadege[12]->type=ARGENT;
515   bureau_de_nadege[12]->valeur=1000;
516
517   strcpy(bureau_de_nadege[13]->texte,"Vous organisé une visite guidée de la maison de Forky. Chacun des élèves doit
518   bureau_de_nadege[13]->type=ANNIVERSAIRE;
519   bureau_de_nadege[13]->valeur=-150;
520
521   strcpy(bureau_de_nadege[14]->texte,"Felicitation votre prospection a Totoland vous rapporte 150 Fintz");
522   bureau_de_nadege[14]->type=ARGENT;
523   bureau_de_nadege[14]->valeur=150;
524
525   strcpy(bureau_de_nadege[15]->texte,"Vous participez a une LAN et après 20h de jeu consécutives, dans un élan de co
526   bureau_de_nadege[15]->type=ARGENT;
527   bureau_de_nadege[15]->valeur=-200;
528
529   melanger_cartes(bureau_de_nadege);
530 }
```

8.4.2.14 void init_plateau (SDL_Surface * *surf_ecran*, rvb_couleur *couleurs*[8], cases ** *plateau*)

Définition à la ligne 282 du fichier initialisation.c.

Référencé par init_plateau_chargement(), et main().

```
283 {
284   init_case_special(surf_ecran, plateau[0], SP_APPARTEMENT, 0);
285   init_case_salle(surf_ecran, couleurs, plateau[1], SALLE, 1, 600, "Salle", "107", VIOLET, PRIX_NIVEAU_MIN);
286   init_case_administration(surf_ecran, plateau[2], BUREAU_NADEGE, 2);
287   init_case_salle(surf_ecran, couleurs, plateau[3], SALLE, 3, 600, "Salle", "109", VIOLET, PRIX_NIVEAU_MIN);
288   init_case_soiree(surf_ecran, plateau[4], SOIREE_AREA, 4);
289   init_case_lieu_commun(surf_ecran, plateau[5], LC_WC, 5);
290   init_case_salle(surf_ecran, couleurs, plateau[6], SALLE, 6, 1000, "Bureau de", "Arianne", BLEU_CIEL, PRIX_NIVEAU_M
291   init_case_administration(surf_ecran, plateau[7], BUREAU_KRYSTEL, 7);
292   init_case_salle(surf_ecran, couleurs, plateau[8], SALLE, 8, 1000, "Bureau de", "Muriel E.", BLEU_CIEL, PRIX_NIVEAU
293   init_case_salle(surf_ecran, couleurs, plateau[9], SALLE, 9, 1200, "Bureau de", "Anne-Marie", BLEU_CIEL, PRIX_NIVEA
294   init_case_special(surf_ecran, plateau[10], SP_BUREAU_LAURENCE, 10);
295   init_case_salle(surf_ecran, couleurs, plateau[11], SALLE, 11, 1400, "Bureau de", "Mathias", ROSE, PRIX_NIVEAU_MIN*
296   init_case_association(surf_ecran, plateau[12], BDS, 12);
297   init_case_salle(surf_ecran, couleurs, plateau[13], SALLE, 13, 1400, "Bureau de", "Muriel D.", ROSE, PRIX_NIVEAU_MI
298   init_case_salle(surf_ecran, couleurs, plateau[14], SALLE, 14, 1600, "Bureau de", "Alex", ROSE, PRIX_NIVEAU_MIN*2)
299   init_case_lieu_commun(surf_ecran, plateau[15], LC_ASCENSEUR, 15);
300   init_case_salle(surf_ecran, couleurs, plateau[16], SALLE, 16, 1800, "Salle", "304", ORANGE, PRIX_NIVEAU_MIN*2);
```

```
301   init_case_administration(surf_ecran, plateau[17], BUREAU_NADEGE, 17);
302   init_case_salle(surf_ecran, couleurs, plateau[18], SALLE, 18, 1800, "Salle", "305", ORANGE, PRIX_NIVEAU_MIN*2);
303   init_case_salle(surf_ecran, couleurs, plateau[19], SALLE, 19, 2000, "Salle", "308", ORANGE, PRIX_NIVEAU_MIN*2);
304   init_case_special(surf_ecran, plateau[20], SP_MACHINE_CAFE, 20);
305   init_case_salle(surf_ecran, couleurs, plateau[21], SALLE, 21, 2200, "Salle", "306", ROUGE, PRIX_NIVEAU_MIN*3);
306   init_case_administration(surf_ecran, plateau[22], BUREAU_KRYSTEL, 22);
307   init_case_salle(surf_ecran, couleurs, plateau[23], SALLE, 23, 2200, "Salle", "307", ROUGE, PRIX_NIVEAU_MIN*3);
308   init_case_salle(surf_ecran, couleurs, plateau[24], SALLE, 24, 2400, "Salle", "309", ROUGE, PRIX_NIVEAU_MIN*3);
309   init_case_lieu_commun(surf_ecran, plateau[25], LC_RU, 25);
310   init_case_salle(surf_ecran, couleurs, plateau[26], SALLE, 26, 2600, "Bureau de", "Yannick", JAUNE, PRIX_NIVEAU_MI
311   init_case_salle(surf_ecran, couleurs, plateau[27], SALLE, 27, 2600, "Bureau de", "Nisrine", JAUNE, PRIX_NIVEAU_MI
312   init_case_association(surf_ecran, plateau[28], BDE, 28);
313   init_case_salle(surf_ecran, couleurs, plateau[29], SALLE, 29, 2800, "Bureau de", "Astrid", JAUNE, PRIX_NIVEAU_MIN*
314   init_case_special(surf_ecran, plateau[30], SP_TABLEAU, 30);
315   init_case_salle(surf_ecran, couleurs, plateau[31], SALLE, 31, 3000, "Cafeteria", " ", VERT, PRIX_NIVEAU_MIN*4);
316   init_case_salle(surf_ecran, couleurs, plateau[32], SALLE, 32, 3000, "Amphi", " ", VERT, PRIX_NIVEAU_MIN*4);
317   init_case_administration(surf_ecran, plateau[33], BUREAU_NADEGE, 33);
318   init_case_salle(surf_ecran, couleurs, plateau[34], SALLE, 34, 3200, "Bureau de", "Florent", VERT, PRIX_NIVEAU_MIN*
319   init_case_lieu_commun(surf_ecran, plateau[35], LC_PARKING, 35);
320   init_case_administration(surf_ecran, plateau[36], BUREAU_KRYSTEL, 36);
321   init_case_salle(surf_ecran, couleurs, plateau[37], SALLE, 37, 3500, "Bureau de", "Pierre", BLEU, PRIX_NIVEAU_MIN*4
322   init_case_soiree(surf_ecran, plateau[38], SOIREE_GALA, 38);
323   init_case_salle(surf_ecran, couleurs, plateau[39], SALLE, 39, 4000, "Bureau de", "Nesim", BLEU, PRIX_NIVEAU_MIN*4)
324
325 }
```

8.4.2.15 joueur* init_anneau_joueur (int *nombre_joueur*, char ** *str_nom_joueur*)

initialise le ring des joueurs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

nombre_joueur nombre de joueurs

str_nom_joueur tableau du nom des joueurs

Version :

0.0.1

Renvoie :

un des joueurs de l'anneau

Définition à la ligne 558 du fichier initialisation.c.

Référencé par main().

```
559 {
560   joueur* p_joueur_premier;
561
562   joueur* p_joueur_precedent;
563   joueur* p_joueur_encours;
564
565   //on crée le premier joueur
566   p_joueur_premier=init_joueur(str_nom_joueur[0],0);
567   p_joueur_precedent=p_joueur_premier;
568
569   for(int i=1;i<nombre_joueur;i++)
```

```
570   {
571     p_joueur_encours=init_joueur(str_nom_joueur[i], i);
572     p_joueur_precedent->pjoueur_suivant=p_joueur_encours;
573     p_joueur_precedent=p_joueur_encours;
574   }
575   p_joueur_precedent->pjoueur_suivant=p_joueur_premier;
576   return(p_joueur_premier);
577 }
```

8.4.2.16 cases** init_plateau_chargement (ifstream * *file*, cases ** *plateau*, rvb_couleur *couleurs*[8], SDL_Surface * *surf_ecran*)

charge les données du plateau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

file fichier de sauvegarde

plateau plateau de jeu

couleurs tableau de couleurs

surf_ecran surface de l'écran

Version :

0.0.1

Renvoie :

le plateau

Définition à la ligne 631 du fichier initialisation.c.

Référencé par chargement().

```
632 {
633   int tmp;
634   int i;
635 
636   i=0;
637 
638   //on allou d'abord le prix du parc gratuit
639   *file >> plateau[20]->machine_a_cafe.int_argent;
640 
641   init_plateau(surf_ecran, couleurs, plateau);
642   for (i = 0; i < 40; i++)
643   {
644     *file >> tmp;
645     if (tmp!=-1)
646     {
647       plateau[i]->case_salle.int_niveau=tmp;
648     }
649   }
650 
651   return(plateau);
652 
653 }
```

8.4.2.17 joueur* init_anneau_joueur_chargement (int *nombre_joueur*, ifstream * *file*, cases ** *plateau*)

créer la liste des joueurs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

file fichier de sauvegarde

plateau plateau de jeu

nombre_joueur nombre de joueur

Version :

0.0.1

Renvoie :

le premier joueur à joué

Définition à la ligne 579 du fichier initialisation.c.

Référencé par chargement().

```
580 {
581   joueur* p_joueur_premier;
582
583   joueur* p_joueur_precedent;
584   joueur* p_joueur_encours;
585
586   //on crée le premier joueur
587   p_joueur_premier=init_joueur_chargement(0, file,plateau);
588   p_joueur_precedent=p_joueur_premier;
589
590   for(int i=1;i<nombre_joueur;i++)
591   {
592     p_joueur_encours=init_joueur_chargement(i,file,plateau);
593     p_joueur_precedent->pjoueur_suivant=p_joueur_encours;
594     p_joueur_precedent=p_joueur_encours;
595   }
596   p_joueur_precedent->pjoueur_suivant=p_joueur_premier;
597
598   return(p_joueur_premier);
599 }
```

8.4.2.18 joueur* init_joueur_chargement (int *numero_joueur*, ifstream * *file*, cases ** *plateau*)

créer un joueur à partir de la sauvagarde

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

file fichier de sauvegarde

plateau plateau de jeu

numero_joueur numéro du joueur

Version :

0.0.1

Renvoie :

le premier joueur à joué

Définition à la ligne 601 du fichier initialisation.c.

Référencé par init_anneau_joueur_chargement().

```
602 {
603   joueur* pjoueur;
604   int int_propriete_tmp;
605 
606   pjoueur=new joueur;
607 
608   *file >> pjoueur->str_nom;
609   *file >> pjoueur->int_position;
610   *file >> pjoueur->int_argent;
611   *file >> pjoueur->int_certificat;
612   *file >> pjoueur->bool_debut;
613   *file >> pjoueur->int_double_tire;
614   *file >> pjoueur->bool_laurence;
615   *file >> pjoueur->int_laurence;
616 
617   pjoueur->pjoueur_suivant=NULL;
618   pjoueur->surf_image=creation_joueur(numero_joueur+1);
619   pjoueur->propriete=init_anneau_possesions();
620 
621   *file >> int_propriete_tmp;
622 
623   while (int_propriete_tmp != -1)
624   {
625     pjoueur->propriete=insertion_bonne_place_propriete(pjoueur->propriete,plateau[int_propriete_tmp]);
626     *file >> int_propriete_tmp;
627   }
628   return(pjoueur);
629 }
```

8.5 Référence du fichier interaction.c

code source des fonctions gÃl'rant l'interaction avec le joueur

Fonctions

- int **attente_clic** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu
- int **attente_action_accueil** (SDL_Surface *surf_ecran, SDL_Surface *surf_boutons[4], SDL_Surface *surf_nombre[6], SDL_Surface *surf_fleche_gauche, SDL_Surface *surf_fleche_droite)
  détermine l'action à entreprendre en fonction du clic du joueur sur la page d'accueil
- int **attente_action_config** (SDL_Surface *surf_ecran, char **str_nom_joueur, SDL_Surface *surf_boutons[6], SDL_Surface *surf_champ, int nombre_joueur)
  détermine l'action à entreprendre en fonction du clic du joueur sur la page des noms des joueurs
- int **attente_validation_propriete** (SDL_Surface *surf_ecran, **cases** **plateau, SDL_Surface *surf_centre, **cases** *pcase, **joueur** *pj_joueur, int int_type_action)
  détermine l'action à entreprendre en fonction du clic du joueur losqu'il est tombé sur une propriété
- int **attente_validation_message** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, char *titre, char *message, int int_type_message)
  détermine l'action à entreprendre en fonction du clic du joueur lors de l'affichage d'un message

8.5.1 Description détaillée

code source des fonctions gÃl'rant l'interaction avec le joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **interaction.c**.

8.5.2 Documentation des fonctions

8.5.2.1 int attente_clic (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'écran

surf_ centre surface centrale du plateau

pj_ joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_ de_ krystel tableau des cartes bureau de krystel

bureau_ de_ nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

-1 fonctionnement normal, 0 quitter le jeu, 1 tour du joueur terminé ce qui correspond à la dernière action du joueur

Définition à la ligne 11 du fichier interaction.c.

Référencé par jeu().

```
12 {
13   //Variable de boucle
14   int continuer;
15   //-1 fonctionnement normal, 0 quitter le jeu, 1 tour du joueur terminÃ
16   continuer=-1;
17   //Postion du clic
18   SDL_Rect position_clic;
19   //Etat des boutons (false = normal et true =  surbrillance)
20   bool bool_bouton_etat;
21   //Mode par dÃfaut
22   bool_bouton_etat=false;
23   //Variable d'Ãtat des dÃs
24   int int_lancer;
25   //1 si lancÃ, 0 autrement
26   int_lancer=0;
27   //EvÃnement sdl
28     SDL_Event event;
29     //Nombre tirÃ par les dÃs
30     int int_nb_tire;
31     //Nombre de propriÃtÃ du joueur
32     int int_nombre_propriete;
33     //Action possible
34     int int_action;
35
36     //Calcul du nombre de propriÃtÃ du joueur
37   int_nombre_propriete=nombre_propriete(pj_joueur);
38
39     //Tant que le joueur n'a pas quitter ou que le tour du joueur n'est pas terminÃ
40     while (continuer==-1)
41     {
42       //On attend un ÃvÃnement
43         SDL_WaitEvent(&event);
44         //Selon l'ÃvÃnement
45         switch(event.type)
46         {
47       //S'il s'agit d'un appui sur une touche
48       case SDL_KEYDOWN:
49         //Selon la touche
```

```
50          switch(event.key.keysym.sym)
51          {
52            //Echap
53            case SDLK_ESCAPE:
54              //On quitte le prgm
55              continuer = 0;
56              break;
57            //Pour toutes les autres touches
58            default:
59              //On fait rien...
60              break;
61          }
62          break;
63        //En cas de mouvement de la souris
64        case SDL_MOUSEMOTION:
65          //On sauvegarde les nouvelles coordonnÃ©es de la souris
66          position_clic.x=event.button.x;
67          position_clic.y=event.button.y;
68          //Si la souris passe sur le bouton de fin de tour
69          if(position_clic.x>PANNEAU_FDT_POS_X && position_clic.x<(PANNEAU_FDT_POS_X+PANNEAU_FDT_LARGEUR)
70          && position_clic.y>PANNEAU_FDT_POS_Y && position_clic.y<(PANNEAU_FDT_POS_Y+PANNEAU_FDT_HAUTEUR) && int_lancer
71          {
72            //On rÃ©affiche le panneau avec le bouton fin de tour en surbrillance
73            if(bool_bouton_etat==false) affich_panneau_fdt(surf_ecran, true);
74            //Le bouton est en surbrillance
75            bool_bouton_etat=true;
76          }
77          //Si le joueur passe la souris sur le bouton lancer dÃ©s
78          else if(position_clic.x>PANNEAU_DES_POS_X && position_clic.x<(PANNEAU_DES_POS_X+PANNEAU_DES_LARGEUR)
79          && position_clic.y>PANNEAU_DES_POS_Y && position_clic.y<(PANNEAU_DES_POS_Y+PANNEAU_DES_HAUTEUR) && int_lancer
80          {
81            //On rÃ©affiche le panneau avec le bouton lancer dÃ©s en surbrillance
82            if(bool_bouton_etat==false) affich_panneau_des_bouton(surf_ecran,1);
83            //L'un des boutons est en surbrillance
84            bool_bouton_etat=true;
85          }
86          //Si le joueur clic sur l'un des  boutons de commande (coin supÃ©rieur gauche)
87          else if(position_clic.x>PANNEAU_MENU_POS_X && position_clic.x<(PANNEAU_MENU_POS_X+PANNEAU_MENU_LARGEUR)
88          && position_clic.y>PANNEAU_MENU_POS_Y && position_clic.y<(PANNEAU_MENU_POS_Y+PANNEAU_MENU_HAUTEUR))
89          {
90            //S'il s'agit du bouton quitter
91            if(position_clic.x>PANNEAU_MENU_POS_X+30 && position_clic.x<(PANNEAU_MENU_POS_X+75)
92            && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
93            {
94
95              //Affichage du panneau du menu avec le bouton quitter en surbrillance
96              affich_panneau_menu(surf_ecran,1);
97              bool_bouton_etat=true;
98            }
99            //S'il s'agit du bouton sauvegarder
100           else if(position_clic.x>(PANNEAU_MENU_POS_X+60+PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_POS_
101           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
102           {
103             //Affichage du panneau du menu avec le bouton sauvegarder
104             affich_panneau_menu(surf_ecran,2);
105             bool_bouton_etat=true;
106           }
107           //S'il s'agit du bouton tourner plateau
108           else if(position_clic.x>(PANNEAU_MENU_POS_X+90+2*PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_PO
109           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
110           {
111             //Affichage du panneau du menu avec le bouton tourner en surbrillance
112             affich_panneau_menu(surf_ecran,3);
113             bool_bouton_etat=true;
114           }
115         }
116         //Autrement, si l'un des boutons est en surbrillance
```

```
117         else if(bool_bouton_etat==true)
118         {
119           //On rÃľaffiche le panneau avec le bouton fin de tour en surbrillance
120           affich_panneau_fdt(surf_ecran, false);
121           //On rÃľaffiche le panneau du menu
122           affich_panneau_menu(surf_ecran,0);
123           //On rÃľaffiche le panneau avec le bouton lancer dÃľs par dÃľfaut
124           if(int_lancer==0) affich_panneau_des_bouton(surf_ecran,0);
125           //Tout les boutons ne sont plus en surbrillance
126           bool_bouton_etat=false;
127         }
128 
129         break;
130       //En cas de clic
131       case SDL_MOUSEBUTTONUP:
132         //Sauvegarde des coordonnÃľes du clic
133         position_clic.x=event.button.x;
134         position_clic.y=event.button.y;
135         //Si le joueur a cliquÃľ sur le bouton lancer dÃľs
136         if(position_clic.x>PANNEAU_DES_POS_X && position_clic.x<(PANNEAU_DES_POS_X+PANNEAU_DES_LARGEUR)
137         && position_clic.y>PANNEAU_DES_POS_Y && position_clic.y<(PANNEAU_DES_POS_Y+PANNEAU_DES_HAUTEUR) && int_lance
138         {
139           //on verifie que le joueur ne soit pas en prison
140           if (pj_joueur->bool_laurence)
141           {
142             traitement_bureau_laurence(surf_ecran, surf_centre, plateau, pj_joueur, bureau_de_krystel, bureau_de_na
143             //Le joueur a lancÃľ les dÃľs
144             int_lancer=1;
145           }
146           else
147           {
148             //On lance les dÃľs
149             int_nb_tire=lancer_des(surf_ecran,surf_centre,pj_joueur, plateau);
150             //On appelle la fonction traitement s'il n'a pas Ãľtait envoyÃľ en prison
151             if (!pj_joueur->bool_laurence)
152             {
153               avancer_jeton (surf_ecran, surf_centre, int_nb_tire,pj_joueur,plateau,bureau_de_krystel,bureau_de_nade
154             }
155 
156             //Affichage du panneau contenant les informations joueurs
157             affich_panneau_possessions(surf_ecran,pj_joueur);
158 
159             //Affichage du panneau contenant les informations joueurs
160             affich_panneau_joueur(surf_ecran,pj_joueur);
161 
162             //Le joueur a lancÃľ les dÃľs
163             int_lancer=1;
164           }
165 
166         }
167         //Si le joueur clic sur l'un des  boutons de commande (coin supÃľrieur gauche)
168         else if(position_clic.x>PANNEAU_MENU_POS_X && position_clic.x<(PANNEAU_MENU_POS_X+PANNEAU_MENU_LARGEUR)
169         && position_clic.y>PANNEAU_MENU_POS_Y && position_clic.y<(PANNEAU_MENU_POS_Y+PANNEAU_MENU_HAUTEUR))
170         {
171           //S'il s'agit du bouton quitter
172           if(position_clic.x>PANNEAU_MENU_POS_X+30 && position_clic.x<(PANNEAU_MENU_POS_X+75)
173           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
174           {
175             //On quitte le prgm
176             continuer = 0;
177           }
178           //S'il s'agit du bouton sauvegarder
179           else if(position_clic.x>(PANNEAU_MENU_POS_X+60+PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_POS
180           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
181           {
182             sauvegarde(plateau, pj_joueur);
183             attente_validation_message(surf_ecran, surf_centre, "Sauvegarde", "La sauvegarde Ãă ÃľtÃľ effectuÃľ ave
```

```
184           }
185           //S'il s'agit du bouton tourner plateau
186           else if(position_clic.x>(PANNEAU_MENU_POS_X+90+2*PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_PO
187           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
188           {
189
190           }
191         }
192         //Si le joueur a cliquÃĭ sur le bouton de fin de tour
193         else if(position_clic.x>PANNEAU_FDT_POS_X && position_clic.x<(PANNEAU_FDT_POS_X+PANNEAU_FDT_LARGEUR)
194         && position_clic.y>PANNEAU_FDT_POS_Y && position_clic.y<(PANNEAU_FDT_POS_Y+PANNEAU_FDT_HAUTEUR) && int_lance
195         {
196           //On arrete la boucle et on passe au joueur suivant
197           continuer=1;
198         }
199         //Si le joueur a cliquÃĭ sur le panneau d'affichage des propriÃĭtÃĭs
200         else if(position_clic.x>PANNEAU_POSSESSION_POS_X && position_clic.x<(PANNEAU_POSSESSION_POS_X+PANNEAU_POSSES
201         && position_clic.y>PANNEAU_POSSESSION_POS_Y && position_clic.y<(PANNEAU_POSSESSION_POS_Y+(40+((PROPRIETE_HAU
202         {
203           int_action=action_possible(plateau,(position_clic.y-PANNEAU_POSSESSION_POS_Y-40)/(PROPRIETE_HAUTEUR+5), p
204
205
206
207           //On affiche la propriÃĭtÃĭ sur laquelle il a cliquÃĭ
208           attente_validation_propriete(surf_ecran, plateau, surf_centre, int_to_cases((position_clic.y-PANNEAU_POSSI
209         }
210         break;
211       //Pour tout les autres ÃĭvÃĭnements
212       default:
213         //On ne fait rien...
214         break;
215         }
216     }
217
218     //Renvoit de la derniÃĭre action du joueur
219   return(continuer);
220 }
```

8.5.2.2 int attente_action_accueil (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_boutons*[4], SDL_Surface * *surf_nombre*[6], SDL_Surface * *surf_fleche_gauche*, SDL_Surface * *surf_fleche_droite*)

détermine l'action à entreprendre en fonction du clic du joueur sur la page d'accueil

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran
surf_boutons tableau des surfaces des boutons de l'accueil
surf_nombre tableau des surfaces des chiffres
surf_fleche_gauche surface de la flèche de gauche
surf_fleche_droite surface de la flèche de droite

Version :

1.0

Renvoie :

le nombre de joueur si ça c'est bien passé sinon -1

Définition à la ligne 222 du fichier interaction.c.

Référencé par affich_accueil().

```
223 {
224   //Joueur actuel
225   int i;
226   i=0;
227   int continuer;
228   continuer=1;
229   //Variable d'Ã©tat des boutons (0 aucun survolÃ©, 1 un des boutons survolÃ©)
230   int int_bouton_etat;
231   int_bouton_etat=0;
232 
233   //Etat des fichiers de sauvegarde (existence ou non)
234   int etat_fichier;
235 
236   //Evenement sdl
237   SDL_Event event;
238 
239   //CoordonnÃ©es du clic
240   SDL_Rect position_clic;
241   //Position temporaire
242   SDL_Rect position_temp;
243   //Position actuel du pointeur
244   SDL_Rect position;
245 
246   position_clic.x=0;
247   position_clic.y=0;
248 
249   //Surface de cache
250   SDL_Surface* surf_cache;
251 
252   etat_fichier=0;
253   if(is_readable("sauvegarde_joueur.txt") && is_readable("sauvegarde_plateau.txt")) etat_fichier=1;
254 
255   //Modification de la position du premier champ
256   position.x=POS_X_FLECHES;
257   position.y=POS_Y_FLECHES;
258   //On colle le premier champ sur le fond
259   SDL_BlitSurface(surf_fleche_gauche, NULL, surf_ecran, &position);
260 
261   //Modification la position du second champ
262   position.x=POS_X_FLECHES+200;
263   position.y=POS_Y_FLECHES;
264   //On colle le second champ sur le fond
265   SDL_BlitSurface(surf_fleche_droite, NULL, surf_ecran, &position);
266 
267   //CrÃ©ation de la surface du cache
268   surf_cache=SDL_CreateRGBSurface(SDL_HWSURFACE, 90, (surf_nombre[5])->h, 32, 0, 0, 0, 0);
269   //Remplissage de noir du cache
270   SDL_FillRect(surf_cache, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
271 
272   //Position du cache
273   position.x=POS_X_FLECHES+100;
274   position.y=POS_Y_FLECHES-100;
275   //Collage du cache sur l'Ã©cran
276   SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
277 
278   //Collage du nombre
279   SDL_BlitSurface(surf_nombre[i%5+1], NULL, surf_ecran, &position);
280 
281   //Mise Ãã jour de l'Ã©cran
282   SDL_Flip(surf_ecran);
283 
284   //Tant qu'on a pas quitter ou valider
285   while (continuer==1)
```

```
286   {
287     //On attend un Ãľvÿnement
288     SDL_WaitEvent(&event);
289     //Selon l'Ãľvÿnement
290     switch(event.type)
291     {
292       case SDL_KEYDOWN:
293         switch(event.key.keysym.sym)
294         {
295           case SDLK_RETURN:
296             continuer=0;
297             break;
298           case SDLK_RIGHT:
299             if(i<4)
300             {
301               //Calcul du nouveau nombre du joueur
302               i=(i+1)%5;
303               //On colle le cache sur l'Ãľcran
304               SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
305               //On colle l'image du joueur sur l'Ãľcran
306               SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
307               //Mise Ãă jour de l'Ãľcran
308               SDL_Flip(surf_ecran);
309             }
310             break;
311           case SDLK_LEFT:
312             if(i>0)
313             {
314               //Calcul du nouveau nombre du joueur
315               i=(i-1)%5;
316               //On colle le cache sur l'Ãľcran
317               SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
318               //On colle l'image du joueur sur l'Ãľcran
319               SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
320               //Mise Ãă jour de l'Ãľcran
321               SDL_Flip(surf_ecran);
322             }
323             break;
324           default:
325             break;
326         }
327       //En cas de dÃľplacement de la souris
328       case SDL_MOUSEMOTION:
329         //On rÃľcupÃŋre les coordonnÃľes du clic
330         position_clic.x=event.button.x;
331         position_clic.y=event.button.y;
332         //S'il s'agit du bouton valider
333         if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCUEIL_LA
334         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
335         {
336           //On modifie la position du bouton valider
337           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
338           position_temp.y=BTN_ACCUEIL_POS_Y;
339           //On colle le bouton valider sur le fond
340           SDL_BlitSurface(surf_boutons[1], NULL, surf_ecran, &position_temp);
341           //on met Ãă jour l'Ãľcran
342           SDL_Flip(surf_ecran);
343           int_bouton_etat=1;
344         }
345         //S'il s'agit du bouton quitter
346         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[2]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
347         && position_clic.y>10 && position_clic.y<(surf_boutons[2]->h+10))
348         {
349           //On modifie la position du bouton quitter
350           position_temp.x=ECRAN_LARGEUR-(surf_boutons[2]->w+10);
351           position_temp.y=10;
352           //On colle le bouton quitter sur le fond
```

```
353           SDL_BlitSurface(surf_boutons[3], NULL, surf_ecran, &position_temp);
354           //On met Ãă jour l'ÃĽcran
355           SDL_Flip(surf_ecran);
356           int_bouton_etat=1;
357         }
358         //S'il s'agit du bouton chargement et que les fichiers existent
359         else if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCUI
360         && position_clic.y>BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HA
361         {
362           //On modifie la position du bouton chargement
363           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
364           position_temp.y=BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE;
365           //On colle le bouton charger sur le fond
366           SDL_BlitSurface(surf_boutons[5], NULL, surf_ecran, &position_temp);
367           //on met Ãă jour l'ÃĽcran
368           SDL_Flip(surf_ecran);
369           int_bouton_etat=1;
370         }
371         //Si un des boutons est en surbrillance
372         else if (int_bouton_etat==1)
373         {
374           //On modifie la position du bouton valider
375           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
376           position_temp.y=BTN_ACCUEIL_POS_Y;
377           //On colle le bouton valider sur le fond
378           SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position_temp);
379
380           //On modifie la position du bouton quitter
381           position_temp.x=ECRAN_LARGEUR-(surf_boutons[2]->w+10);
382           position_temp.y=10;
383           //On colle le bouton quitter sur le fond
384           SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position_temp);
385
386           //On modifie la position du bouton chargement
387           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
388           position_temp.y=BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE;
389           //On colle le bouton charger sur le fond
390           SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position_temp);
391
392           //On met Ãă jour l'ÃĽcran
393           SDL_Flip(surf_ecran);
394           int_bouton_etat=0;
395         }
396
397         break;
398       //En cas de clic sur un bouton
399       case SDL_MOUSEBUTTONUP:
400         //On sauvegarde les coordonnÃĽes du clic
401         position_clic.x=event.button.x;
402         position_clic.y=event.button.y;
403         //Si on a cliquer sur la fleche gauche
404         if(position_clic.x>POS_X_FLECHES && position_clic.x<(POS_X_FLECHES+surf_fleche_gauche->w)
405         && position_clic.y>POS_Y_FLECHES && position_clic.y<(POS_Y_FLECHES+surf_fleche_gauche->h) && i>0)
406         {
407           //Calcul le nouveau nombre de joueur
408           i=(i-1)%5;
409           //On colle le cache sur l'ÃĽcran
410           SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
411           //On colle l'image du nombre sur l'ÃĽcran
412           SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
413           //Mise Ãă jour de l'ÃĽcran
414           SDL_Flip(surf_ecran);
415         }
416         //S'il s'agit du bouton quitter
417         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[2]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
418         && position_clic.y>10 && position_clic.y<(surf_boutons[2]->h+10))
419         {
```

```
420           continuer=-1;
421         }
422         //Fleche droite
423         else if(position_clic.x>(POS_X_FLECHES+200) && position_clic.x<(POS_X_FLECHES+200+surf_fleche_droite->w)
424         && position_clic.y>POS_Y_FLECHES && position_clic.y<(POS_Y_FLECHES+surf_fleche_droite->h) && i<5)
425         {
426           //Calcul du nouveau nombre du joueur
427           i=(i+1)%5;
428           //On colle le cache sur l'Ã¯cran
429           SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
430           //On colle l'image du joueur sur l'Ã¯cran
431           SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
432           //Mise Ãa jour de l'Ã¯cran
433           SDL_Flip(surf_ecran);
434         }
435         //Bouton valider
436         else if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCU
437         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
438         {
439           continuer=0;
440         }
441         //Bouton chargement
442         else if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCU
443         && position_clic.y>BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_H
444         {
445           continuer=-2;
446         }
447
448         break;
449       //En cas d'autre Ã¯vÃ¯nement
450       default:
451         //On ne fait rien...
452         break;
453     }
454     }
455     //Si le joueur Ãa voulu quitter, on retourne sa demande
456     if(continuer==-1) return(continuer);
457     //Si le joueur Ãa voulu charger une nouvelle partie, on retourne sa demande
458     if(continuer==-2) return(continuer);
459     //Sinon on retourne le nombre de joueur
460   return(i+1);
461
462 }
```

8.5.2.3 int attente_action_config (SDL_Surface * *surf_ecran*, char ** *str_nom_joueur*, SDL_Surface * *surf_boutons*[6], SDL_Surface * *surf_champ*, int *nombre_joueur*)

détermine l'action à entreprendre en fonction du clic du joueur sur la page des noms des joueurs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

str_nom_joueur tableau des noms des joueurs

surf_boutons surface des 6 boutons

surf_champ surfaces des champs d'écriture

nombre_joueur nombre de joueurs qui jouent

Version :

1.0

Renvoie :

action du joueur

Définition à la ligne 464 du fichier interaction.c.

Référencé par affich_config().

```
465 {
466   int i;
467   i=0;
468   //Etat des boutons (surbrillance ou non)
469   int int_bouton_etat;
470   int_bouton_etat=0;
471 
472   int continuer;
473   continuer=2;
474 
475   //Champ en cours de modification
476   int int_champ;
477   int_champ=0;
478 
479   //Surface du pion
480   SDL_Surface* surf_pion;
481   surf_pion=NULL;
482 
483   //Surface du cache du pion;
484   SDL_Surface* surf_cache;
485   surf_cache=NULL;
486 
487   //Surface du nom
488   SDL_Surface* surf_nom;
489   surf_nom=NULL;
490 
491   //EvÃ¨nement sdl
492   SDL_Event event;
493 
494   //CoordonnÃ¨es du clic
495   SDL_Rect position_clic;
496   position_clic.x=0;
497   position_clic.y=0;
498 
499   //Ancienne position
500   SDL_Rect position;
501 
502   //Position du champ
503   SDL_Rect position_champ;
504 
505   //Police d'Ã¨criture
506   TTF_Font* police;
507   police=NULL;
508 
509   //Couleur du texte
510   SDL_Color couleur_texte = {0, 0, 0};
511 
512   //Ouverture de la police d'ecriture
513   police = TTF_OpenFont("sdl/police/police.ttf", 17);
514 
515   //CrÃ¨ation du pion
516   surf_pion=SDL_CreateRGBSurface(SDL_HWSURFACE, 20, 20, 32, 0, 0, 0, 0);
517   //Remplissage de l'image du pion
518   SDL_FillRect(surf_pion, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 255));
519 
520   //Ecriture du numÃ¨ro sur la surface
```

```
521   surf_cache=SDL_CreateRGBSurface(SDL_HWSURFACE, 20, 20, 32, 0, 0, 0, 0);
522   //Remplissage du bouton pion
523   SDL_FillRect(surf_cache, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
524
525   //Modification de la position du pion
526   position.x=CHAMP_POS_X-30;
527   position.y=CHAMP_POS_Y+5;
528   //Affichage du pion
529   SDL_BlitSurface(surf_pion, NULL, surf_ecran, &position);
530
531   //Mise Ãă jour de l'ÃĽcran
532   SDL_Flip(surf_ecran);
533
534   //tant que le joueur n'a pas fait prÃĽcÃĽdent, quitter ou jouer
535   while (continuer==2)
536   {
537     //On attend un ÃĽvÃĽnement sdl
538     SDL_WaitEvent(&event);
539     //Selon le type de l'ÃĽvÃĽnement
540     switch(event.type)
541     {
542       //Si on appui sur une touche
543       case SDL_KEYDOWN:
544         //Selon l'ÃĽvÃĽnement
545         switch(event.key.keysym.sym)
546         {
547           //En cas de tabulation
548           case SDLK_TAB:
549             //Affichage du cache
550             SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
551             //Passage au champ suivant
552             int_champ=(int_champ+1)%nombre_joueur;
553             //Calcul de la position du champ en fonction du numÃĽro du champ
554             position.y=int_champ*CHAMP_HAUTEUR*2+CHAMP_POS_Y+5;
555             //Affichage du pion
556             SDL_BlitSurface(surf_pion, NULL, surf_ecran, &position);
557             //Mise Ãă jour de l'ÃĽcran
558             SDL_Flip(surf_ecran);
559             break;
560           //En cas de validation
561           case SDLK_RETURN:
562             continuer=1;
563             for (int i=0;i<nombre_joueur;i++)
564             {
565               //on vÃĽrifie que le nom n'est pas vide
566               if (str_nom_joueur[i][0]=='\0')
567               {
568                 continuer=2;
569               }
570               else
571               {
572                 //on verifie que les noms ne soient pas identiques
573                 for (int j=i+1;j<nombre_joueur;j++)
574                 {
575                   if (strcmp(str_nom_joueur[i],str_nom_joueur[j])==0)
576                   {
577                     continuer=2;
578                   }
579                 }
580               }
581             }
582             break;
583           //Pour toutes les autres touches
584           default:
585             //On positionne le champ
586             position_champ.x=CHAMP_POS_X;
587             position_champ.y=CHAMP_POS_Y+int_champ*2*CHAMP_HAUTEUR;
```

```
588             //On met un cache sur le champ actuel
589             SDL_BlitSurface(surf_champ, NULL, surf_ecran, &position_champ);
590             //On positionne le texte
591             position_champ.x=CHAMP_POS_X+5;
592             position_champ.y=CHAMP_POS_Y+int_champ*2*CHAMP_HAUTEUR-10;
593             //Ecriture du titre sur la surface de titre
594             strcpy(str_nom_joueur[int_champ],ajout_caractere(str_nom_joueur[int_champ],event.key.keysym.sym));
595             //Ecriture du nom du joueur modifiÃĽ
596             surf_nom = TTF_RenderText_Blended(police, str_nom_joueur[int_champ], couleur_texte);
597             //On colle le nom du joueur sur le champ
598             SDL_BlitSurface(surf_nom, NULL, surf_ecran, &position_champ);
599             //Mise Ãă jour de l'ÃĽcran
600             SDL_Flip(surf_ecran);
601             break;
602         }
603       //En cas de dÃĽplacement de la souris
604       case SDL_MOUSEMOTION:
605         //Sauvegarde de la position du clic
606         position_clic.x=event.button.x;
607         position_clic.y=event.button.y;
608         //S'il s'agit du bouton retour
609         if(position_clic.x>(BTN_ACCUEIL_POS_X) && position_clic.x<(BTN_ACCUEIL_POS_X+BTN_ACCUEIL_LARGEUR)
610         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
611         {
612           //On modifie la position du bouton retour
613           position_champ.x=BTN_ACCUEIL_POS_X;
614           position_champ.y=BTN_ACCUEIL_POS_Y;
615           //On colle le bouton retour sur le fond
616           SDL_BlitSurface(surf_boutons[1], NULL, surf_ecran, &position_champ);
617           //On met l'ÃĽcran Ãă jour
618           SDL_Flip(surf_ecran);
619           int_bouton_etat=1;
620         }
621         //S'il s'agit du bouton valider
622         else if(position_clic.x>(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w)
623         && position_clic.x<((BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w)+BTN_ACCUEIL_LARGEUR)
624         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
625         {
626           //On modifie la position du bouton valider
627           position_champ.x=(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w);
628           position_champ.y=BTN_ACCUEIL_POS_Y;
629           //On colle le bouton valider sur le fond
630           SDL_BlitSurface(surf_boutons[3], NULL, surf_ecran, &position_champ);
631           //On met l'ÃĽcran Ãă jour
632           SDL_Flip(surf_ecran);
633           int_bouton_etat=1;
634         }
635         //S'il s'agit du bouton quitter
636         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[4]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
637         && position_clic.y>10 && position_clic.y<(surf_boutons[4]->h+10))
638         {
639           //On modifie la position du bouton quitter
640           position_champ.x=ECRAN_LARGEUR-(surf_boutons[4]->w+10);
641           position_champ.y=10;
642           //On colle le bouton quitter sur le fond
643           SDL_BlitSurface(surf_boutons[5], NULL, surf_ecran, &position_champ);
644           //On met Ãă jour l'ÃĽcran
645           SDL_Flip(surf_ecran);
646           int_bouton_etat=1;
647         }
648         //Si un des boutons est en surbrillance
649         else if (int_bouton_etat==1)
650         {
651           //On modifie la position du bouton valider
652           position_champ.x=(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w);
653           position_champ.y=BTN_ACCUEIL_POS_Y;
654           //On colle le bouton valider sur le fond
```

```
655           SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position_champ);
656
657           //On modifie la position du bouton retour
658           position_champ.x=BTN_ACCUEIL_POS_X;
659           position_champ.y=BTN_ACCUEIL_POS_Y;
660           //On colle le bouton retour sur le fond
661           SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position_champ);
662
663           //On modifie la position du bouton quitter
664           position_champ.x=ECRAN_LARGEUR-(surf_boutons[4]->w+10);
665           position_champ.y=10;
666           //On colle le bouton quitter sur le fond
667           SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position_champ);
668
669           //On met Ãă jour l'ÃĪcran
670           SDL_Flip(surf_ecran);
671           int_bouton_etat=0;
672         }
673
674         break;
675       //En cas de clic
676       case SDL_MOUSEBUTTONUP:
677         //On sauvegarde les coordonnÃĪes du clic
678         position_clic.x=event.button.x;
679         position_clic.y=event.button.y;
680         //S'il s'agit du bouton valider
681         if(position_clic.x>(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w) && position_clic.x<((BTN_ACCUEIL_POS_X+20+(su
682         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
683         {
684           continuer=1;
685
686           for (int i=0;i<nombre_joueur;i++)
687           {
688             //on vÃĪrifie que le nom n'est pas vide
689             if (str_nom_joueur[i][0]=='\0')
690             {
691               continuer=2;
692             }
693             else
694             {
695               //on verifie que les noms ne soient pas identiques
696               for (int j=i+1;j<nombre_joueur;j++)
697               {
698                 if (strcmp(str_nom_joueur[i],str_nom_joueur[j])==0)
699                 {
700                   continuer=2;
701                 }
702               }
703             }
704           }
705         }
706         //S'il s'agit du bouton retour
707         else if(position_clic.x>(BTN_ACCUEIL_POS_X) && position_clic.x<(BTN_ACCUEIL_POS_X+BTN_ACCUEIL_LARGEUR)
708         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
709         {
710           continuer=-1;
711         }
712         //S'il s'agit du bouton quitter
713         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[4]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
714         && position_clic.y>10 && position_clic.y<(surf_boutons[4]->h+10))
715         {
716           continuer=0;
717         }
718         //Si on a cliquer sur un des champs
719         else if(position_clic.x>(CHAMP_POS_X) && position_clic.x<(CHAMP_POS_X+CHAMP_LARGEUR)
720         && position_clic.y>CHAMP_POS_Y && position_clic.y<(CHAMP_POS_Y+CHAMP_HAUTEUR*2*nombre_joueur))
721         {
```

```
722           //Affichage du cache
723           SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
724           //On calcul sur quel champ le joueur Ãă cliquÃĭ
725           int_champ=((position_clic.y-CHAMP_POS_Y)/(CHAMP_HAUTEUR*2));
726           //Calcul de la position du champ en fonction du numÃĭro du champ
727           position.y=int_champ*CHAMP_HAUTEUR*2+CHAMP_POS_Y+5;
728           //Affichage du pion
729           SDL_BlitSurface(surf_pion, NULL, surf_ecran, &position);
730           //Mise Ãă jour de l'Ãĭcran
731           SDL_Flip(surf_ecran);
732         }
733         break;
734       //S'il s'agit d'un autre ÃĭvÃĭnement
735       default:
736         //On ne fait rien.
737         break;
738     }
739   }
740
741   //Fermeture de la police d'ecriture
742   TTF_CloseFont(police);
743   //LibÃĭration des surfaces
744   SDL_FreeSurface(surf_pion);
745   SDL_FreeSurface(surf_cache);
746   SDL_FreeSurface(surf_nom);
747   //On retourne la valeur de continuer
748   return(continuer);
749
750 }
```

8.5.2.4 int attente_validation_propriete (SDL_Surface * *surf_ecran*, cases ** *plateau*, SDL_Surface * *surf_centre*, cases * *pcase*, joueur * *pj_joueur*, int *int_type_action*)

détermine l'action à entreprendre en fonction du clic du joueur losqu'il est tombé sur une propriété

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

plateau plateau de jeu

pcase case en cours de traitement

pj_joueur joueur qui joue actuellement

int_type_action indique le type de l'action à entreprendre

Version :

1.0

Renvoie :

action du joueur

Définition à la ligne 752 du fichier interaction.c.

Référencé par attente_clic(), et traitement_arrive_case().

```
753 {
754   //Surface du fond
755   SDL_Surface* surf_fond;
756   surf_fond=NULL;
757 
758   //EvÃ¨nement sdl
759   SDL_Event event;
760 
761   //CoordonnÃ¨es
762   SDL_Rect position;
763   //CoordonnÃ¨es du clic
764   SDL_Rect position_clic;
765 
766   int int_position;
767   int int_type;
768   int_type=pcase->int_type;
769   int int_etat;
770   int_etat=0;
771   int continuer;
772   continuer=0;
773 
774   //CrÃ¨ation du fond
775   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR+DETAIL_MESSAGE_EPAISSEUR*2, DETAIL_HAUTEUR+DETAIL_MES
776   //Remplissage de l'image de fond
777   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 125, 0));
778 
779   //Position de la carte sur le message
780   position.x=DETAIL_MESSAGE_EPAISSEUR;
781   position.y=DETAIL_MESSAGE_EPAISSEUR;
782 
783   //Collage de la carte sur le message
784   if(int_type==SALLE) SDL_BlitSurface(pcase->case_salle.surf_detail, NULL, surf_fond, &position);
785   else if(int_type==BDE || int_type==BDS) SDL_BlitSurface(pcase->case_association.surf_detail, NULL, surf_fond, &pos
786   else if(int_type>=LC_WC && int_type<=LC_PARKING) SDL_BlitSurface(pcase->case_lieu_commun.surf_detail, NULL, surf_
787   else cerr << "ERREUR !!" << endl;
788 
789   //Affichage de la propriÃ¨tÃ¨
790   action_to_boutons(surf_ecran, surf_fond, int_type_action, 0);
791 
792   //Tant que le joueur n'a pas fait d'action
793   while(!continuer)
794   {
795       //On attend un Ã¨vÃ¨nement
796         SDL_WaitEvent(&event);
797         //Selon l'Ã¨vÃ¨nement
798         switch(event.type)
799         {
800       //S'il s'agit d'un appui sur une touche
801       case SDL_KEYDOWN:
802         //Selon la touche
803         switch(event.key.keysym.sym)
804         {
805           //Echap
806           case SDLK_ESCAPE:
807             //On quitte le prgm
808             continuer = 1;
809             break;
810           //Pour toutes les autres touches
811           default:
812             //On fait rien...
813             break;
814         }
815         break;
816       //En cas de mouvement de la souris
817       case SDL_MOUSEMOTION:
818         //On sauvegarde les nouvelles coordonnÃ¨es de la souris
819         position_clic.x=event.button.x;
```

```
820         position_clic.y=event.button.y;
821
822         //S'il s'agit du premier bouton
823         if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
824         && position_clic.y>DETAIL_BOUTON_POS_Y && position_clic.y<(DETAIL_BOUTON_POS_Y+40))
825         {
826           action_to_boutons(surf_ecran, surf_fond, int_type_action, 1);
827           int_etat=1;
828         }
829         //Bouton 2 sauf pour le cas oÃź on finit
830         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
831         && position_clic.y>DETAIL_BOUTON_POS_Y+50 && position_clic.y<(DETAIL_BOUTON_POS_Y+90) && int_type_action !=
832         {
833           action_to_boutons(surf_ecran, surf_fond, int_type_action, 2);
834           int_etat=2;
835         }
836         //Bouton 3 dans le cas d'un plus moins
837         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
838         && position_clic.y>DETAIL_BOUTON_POS_Y+100 && position_clic.y<(DETAIL_BOUTON_POS_Y+140) && int_type_action =
839         {
840           action_to_boutons(surf_ecran, surf_fond, int_type_action, 3);
841           int_etat=2;
842         }
843         else if(int_etat!=0)
844         {
845           action_to_boutons(surf_ecran, surf_fond, int_type_action, 0);
846           int_etat=0;
847         }
848
849         break;
850       case SDL_MOUSEBUTTONUP:
851         //On sauvegarde les nouvelles coordonnÃľes de la souris
852         position_clic.x=event.button.x;
853         position_clic.y=event.button.y;
854         //Premier bouton
855         if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
856         && position_clic.y>DETAIL_BOUTON_POS_Y && position_clic.y<(DETAIL_BOUTON_POS_Y+50))
857         {
858           //Selon l'action Ãă faire
859           switch(int_type_action)
860           {
861             //Dans le cas d'un achat
862             case ACTION_ACHAT:
863               traitement_achat(pcase,pj_joueur);
864               break;
865             //Augmentation de niveau
866             case ACTION_PLUS:
867               traitement_augmentation_niveau(pcase,pj_joueur);
868               break;
869             //Diminution de niveau
870             case ACTION_MOINS:
871               traitement_diminution_niveau(pcase,pj_joueur);
872               break;
873             //Dans le cas d'une dÃľshypothÃÎque
874             case ACTION_PLUS_MOINS:
875               traitement_augmentation_niveau(pcase,pj_joueur);
876               break;
877             //Dans tout les autres cas
878             default:
879               break;
880           }
881           continuer=1;
882         }
883         //Bouton 2 sauf pour le cas oÃź on finit
884         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
885         && position_clic.y>DETAIL_BOUTON_POS_Y+50 && position_clic.y<(DETAIL_BOUTON_POS_Y+90) && int_type_action !=
886         {
```

```
887           switch(int_type_action)
888           {
889             case ACTION_PLUS_MOINS:
890               traitement_diminution_niveau(pcase,pj_joueur);
891               break;
892             //Dans tout les autres cas
893             default:
894               break;
895           }
896           continuer=1;
897         }
898         //Bouton 3 dans le cas d'un plus moins
899         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
900         && position_clic.y>DETAIL_BOUTON_POS_Y+100 && position_clic.y<(DETAIL_BOUTON_POS_Y+140) && int_type_action =
901         {
902           continuer=1;
903         }
904         break;
905       default:
906         break;
907         }
908   }
909
910   //on rÃľactualise la case ou se trouve le joueur afin qu'il efface le jeton
911   affich_case(surf_ecran,pcase);
912   //on compte le nombre de personne sur la case ou doit Ãłtre afficher le joueur actuel
913   int_position=nombre_joueur_case(pj_joueur);
914   //on affiche le joueur sur la case s'il se trouve dessus
915   if(plateau[pj_joueur->int_position]==pcase) affich_joueur(surf_ecran, pj_joueur,int_position+1, pcase);
916
917   //Mise Ãă jour de l'Ãľcran
918   SDL_Flip(surf_ecran);
919
920   //RÃľaffichage du centre
921   affich_centre(surf_ecran,surf_centre);
922   //Mise Ãă jour de l'Ãľcran
923   SDL_Flip(surf_ecran);
924   return(0);
925
926 }
```

8.5.2.5 int attente_validation_message (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, char * *titre*, char * *message*, int *int_type_message*)

détermine l'action à entreprendre en fonction du clic du joueur lors de l'affichage d'un message

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

titre le titre du message

message le message à afficher

int_type_message type du message à afficher

Version :

1.0

Renvoie :

action du joueur

Définition à la ligne 928 du fichier interaction.c.

Référencé par attente_clic(), jeu(), lancer_des(), traitement_arrive_case(), traitement_bureau(), traitement_bureau_laurence(), traitement_machine_a_cafe(), et traitement_soiree().

```
929 {
930   //Surface du fond
931   SDL_Surface* surf_fond;
932   surf_fond=NULL;
933 
934   //EvÃ¨nement sdl
935   SDL_Event event;
936 
937   //Surface du message
938   SDL_Surface* surf_message;
939 
940   //Position du bouton
941   SDL_Rect position_bouton;
942 
943   //Position du message
944   SDL_Rect position_message;
945 
946   //CoordonnÃ¨es du clic
947   SDL_Rect position_clic;
948 
949   //Nombre de ligne du texte compilÃ¨
950   int int_nombre_ligne;
951   int_nombre_ligne=0;
952 
953   //Texte dÃ¨coupÃ¨
954   char*** texte;
955   texte=new char**[1];
956 
957   //Etat des boutons
958   int int_etat;
959   int_etat=0;
960 
961   //Variable de boucle
962   int continuer;
963   continuer=2;
964 
965   int i;
966 
967   //Fonction de dÃ¨coupe du texte
968   int_nombre_ligne=decoupage_string(texte, message);
969   //CrÃ¨ation du message
970   surf_message=creation_message(surf_ecran, titre, *texte, int_type_message, int_nombre_ligne);
971 
972   //Affichage du message Ã  l'Ã¨cran
973   affich_message(surf_ecran,surf_message,int_type_message, 0);
974 
975   //Calcul de la position du message
976   position_message.x=POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_LARGEUR)/2;
977   position_message.y=POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_HAUTEUR)/2;
978 
979   //Calcul de la postion du premier pour Ã¨viter de le recalculer Ã  chaque test
980   position_bouton.x=position_message.x+(MESSAGE_LARGEUR-160)/2;
981   position_bouton.y=position_message.y+MESSAGE_HAUTEUR-60;
982 
983   //cas particulier s'il s'agit du message quitter
984   if(int_type_message==MESSAGE_QUITTER)
985   {
986     position_bouton.x=position_message.x+(MESSAGE_LARGEUR-320)/3;
```

```
987     position_bouton.y=position_message.y+MESSAGE_HAUTEUR-60;
988   }
989
990   //Tant que le joueur n'a pas fait d'action
991   while(continuer==2)
992   {
993       //On attend un Ã'vÃ'nement
994         SDL_WaitEvent(&event);
995         //Selon l'Ã'vÃ'nement
996         switch(event.type)
997         {
998       //En cas de mouvement de la souris
999       case SDL_MOUSEMOTION:
1000         //On sauvegarde les nouvelles coordonnÃ'es de la souris
1001         position_clic.x=event.button.x;
1002         position_clic.y=event.button.y;
1003         //S'il s'agit du premier bouton
1004         if(position_clic.x>(position_bouton.x) && position_clic.x<(position_bouton.x+160)
1005         && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message!=MESSA
1006         {
1007           //Bouton nÂř1 en surbrillance
1008           affich_message(surf_ecran,surf_message,int_type_message, 1);
1009           int_etat=1;
1010         }
1011         //Dans le cas d'un message quitter, on a un second bouton
1012         else if(position_clic.x>(position_bouton.x+(MESSAGE_LARGEUR-160)/3) && position_clic.x<(position_bouton.x+
1013         && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message==MESSA
1014         {
1015           //Mise en surbrillance du second bouton
1016           affich_message(surf_ecran,surf_message,int_type_message, 2);
1017           int_etat=2;
1018         }
1019         //Si jamais c'est un message prison, on a trois boutons
1020         else if(int_type_message==MESSAGE_PRISON)
1021         {
1022           //Bouton 1 : Attendre
1023           if(position_clic.x>(position_message.x+30) && position_clic.x<(position_message.x+190)
1024           && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1025           {
1026             //Mise en surbrillance
1027             affich_message(surf_ecran,surf_message,int_type_message, 1);
1028             int_etat=1;
1029           }
1030           //Bouton 2 : Payer
1031           else if(position_clic.x>(position_message.x+(MESSAGE_LARGEUR-160)/2) && position_clic.x<(position_message
1032           && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1033           {
1034             //Mise en surbrillance
1035             affich_message(surf_ecran,surf_message,int_type_message, 2);
1036             int_etat=2;
1037           }
1038           //Bouton 3 : Certificat
1039           else if(position_clic.x>(position_message.x+MESSAGE_LARGEUR-190) && position_clic.x<(position_message.x+M
1040           && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1041           {
1042             //Mise en surbrillance
1043             affich_message(surf_ecran,surf_message,int_type_message, 3);
1044             int_etat=3;
1045           }
1046           //Autement, si jamais l'un des boutonsdu message prison est en surbrillance
1047           else if(int_etat!=0)
1048           {
1049             //On rÃ'affiche le message avec les boutons normaux
1050             affich_message(surf_ecran,surf_message,int_type_message, 0);
1051             int_etat=0;
1052           }
1053         }
```

```
1054        //Autement, si jamais l'un des boutons est en surbrillance
1055        else if(int_etat!=0)
1056        {
1057          //On rÃľaffiche le message avec les boutons normaux
1058          affich_message(surf_ecran,surf_message,int_type_message, 0);
1059          int_etat=0;
1060        }
1061
1062        break;
1063      case SDL_MOUSEBUTTONUP:
1064        //On sauvegarde les nouvelles coordonnÃľes de la souris
1065        position_clic.x=event.button.x;
1066        position_clic.y=event.button.y;
1067        //Si on appui sur le premier bouton
1068        if(position_clic.x>(position_bouton.x) && position_clic.x<(position_bouton.x+160)
1069        && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message!=MESSA
1070        {
1071          //On sort de la boucle
1072          continuer=0;
1073        }
1074        else if(int_type_message==MESSAGE_PRISON)
1075        {
1076          //Bouton 1 : Attendre
1077          if(position_clic.x>(position_message.x+30) && position_clic.x<(position_message.x+190)
1078          && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1079          {
1080            continuer=4;
1081          }
1082          //Bouton 2 : Payer
1083          else if(position_clic.x>(position_message.x+(MESSAGE_LARGEUR-160)/2) && position_clic.x<(position_message
1084          && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1085          {
1086            continuer=5;
1087          }
1088          //Bouton 3 : Certificat
1089          else if(position_clic.x>(position_message.x+MESSAGE_LARGEUR-190) && position_clic.x<(position_message.x+M
1090          && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1091          {
1092            continuer=6;
1093          }
1094        }
1095        //S'il s'agit d'un message quitter et que l'on clique sur le bouton sauvegarder
1096        else if(position_clic.x>(position_bouton.x+(MESSAGE_LARGEUR-160)/3) && position_clic.x<(position_bouton.x+(
1097        && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message==MESSA
1098        {
1099          //Normalement ici il y Ãă la fonction de sauvegarde
1100          continuer=3;
1101        }
1102        break;
1103      default:
1104        break;
1105        }
1106  }
1107
1108  //RÃľaffichage du centre
1109  affich_centre(surf_ecran,surf_centre);
1110  //Mise Ãă jour de l'Ãľcran
1111  SDL_Flip(surf_ecran);
1112
1113  //LibÃľration du message compilÃľ
1114  for (i = 0; i <int_nombre_ligne; i++)
1115  {
1116    delete texte[0][i];
1117  }
1118
1119  delete texte;
1120
```

```
1121   //On retourne l'action du joueur
1122   return(continuer);
1123
1124 }
```

8.6 Référence du fichier interaction.h

en tete des fonctions gérant l'interaction avec le joueur

Fonctions

- int **attente_clic** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu
- int **attente_action_accueil** (SDL_Surface *surf_ecran, SDL_Surface *surf_boutons[4], SDL_Surface *surf_nombre[6], SDL_Surface *surf_fleche_gauche, SDL_Surface *surf_fleche_droite)
  détermine l'action à entreprendre en fonction du clic du joueur sur la page d'accueil
- int **attente_action_config** (SDL_Surface *surf_ecran, char **str_nom_joueur, SDL_Surface *surf_boutons[6], SDL_Surface *surf_champ, int nombre_joueur)
  détermine l'action à entreprendre en fonction du clic du joueur sur la page des noms des joueurs
- int **attente_validation_propriete** (SDL_Surface *surf_ecran, **cases** **plateau, SDL_Surface *surf_centre, **cases** *pcase, **joueur** *pj_joueur, int int_type_action)
  détermine l'action à entreprendre en fonction du clic du joueur losqu'il est tombé sur une propriété
- int **attente_validation_message** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, char *titre, char *message, int int_type_message)
  détermine l'action à entreprendre en fonction du clic du joueur lors de l'affichage d'un message

8.6.1 Description détaillée

en tete des fonctions gérant l'interaction avec le joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **interaction.h**.

8.6.2 Documentation des fonctions

8.6.2.1 int attente_clic (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'écran

surf_ centre surface centrale du plateau

pj_ joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_ de_ krystel tableau des cartes bureau de krystel

bureau_ de_ nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

-1 fonctionnement normal, 0 quitter le jeu, 1 tour du joueur terminé ce qui correspond à la dernière action du joueur

Définition à la ligne 11 du fichier interaction.c.

Référencé par jeu().

```
12 {
13   //Variable de boucle
14   int continuer;
15   //-1 fonctionnement normal, 0 quitter le jeu, 1 tour du joueur terminÃĺ
16   continuer=-1;
17   //Postion du clic
18   SDL_Rect position_clic;
19   //Etat des boutons (false = normal et true =  surbrillance)
20   bool bool_bouton_etat;
21   //Mode par dÃĺfaut
22   bool_bouton_etat=false;
23   //Variable d'Ãĺtat des dÃ©s
24   int int_lancer;
25   //1 si lancÃĺ, 0 autrement
26   int_lancer=0;
27   //EvÃĺnement sdl
28     SDL_Event event;
29     //Nombre tirÃĺ par les dÃ©s
30     int int_nb_tire;
31     //Nombre de propriÃĺtÃĺ du joueur
32     int int_nombre_propriete;
33     //Action possible
34     int int_action;
35
36     //Calcul du nombre de propriÃĺtÃĺ du joueur
37   int_nombre_propriete=nombre_propriete(pj_joueur);
38
39     //Tant que le joueur n'a pas quitter ou que le tour du joueur n'est pas terminÃĺ
40     while (continuer==-1)
41     {
42       //On attend un Ãĺvÿnement
43         SDL_WaitEvent(&event);
44         //Selon l'Ãĺvÿnement
45         switch(event.type)
46         {
47       //S'il s'agit d'un appui sur une touche
48       case SDL_KEYDOWN:
49         //Selon la touche
```

```
50          switch(event.key.keysym.sym)
51          {
52            //Echap
53            case SDLK_ESCAPE:
54              //On quitte le prgm
55              continuer = 0;
56              break;
57            //Pour toutes les autres touches
58            default:
59              //On fait rien...
60              break;
61          }
62          break;
63        //En cas de mouvement de la souris
64        case SDL_MOUSEMOTION:
65          //On sauvegarde les nouvelles coordonnÃ©es de la souris
66          position_clic.x=event.button.x;
67          position_clic.y=event.button.y;
68          //Si la souris passe sur le bouton de fin de tour
69          if(position_clic.x>PANNEAU_FDT_POS_X && position_clic.x<(PANNEAU_FDT_POS_X+PANNEAU_FDT_LARGEUR)
70          && position_clic.y>PANNEAU_FDT_POS_Y && position_clic.y<(PANNEAU_FDT_POS_Y+PANNEAU_FDT_HAUTEUR) && int_lancer
71          {
72            //On rÃ©affiche le panneau avec le bouton fin de tour en surbrillance
73            if(bool_bouton_etat==false) affich_panneau_fdt(surf_ecran, true);
74            //Le bouton est en surbrillance
75            bool_bouton_etat=true;
76          }
77          //Si le joueur passe la souris sur le bouton lancer dÃ©s
78          else if(position_clic.x>PANNEAU_DES_POS_X && position_clic.x<(PANNEAU_DES_POS_X+PANNEAU_DES_LARGEUR)
79          && position_clic.y>PANNEAU_DES_POS_Y && position_clic.y<(PANNEAU_DES_POS_Y+PANNEAU_DES_HAUTEUR) && int_lancer
80          {
81            //On rÃ©affiche le panneau avec le bouton lancer dÃ©s en surbrillance
82            if(bool_bouton_etat==false) affich_panneau_des_bouton(surf_ecran,1);
83            //L'un des boutons est en surbrillance
84            bool_bouton_etat=true;
85          }
86          //Si le joueur clic sur l'un des  boutons de commande (coin supÃ©rieur gauche)
87          else if(position_clic.x>PANNEAU_MENU_POS_X && position_clic.x<(PANNEAU_MENU_POS_X+PANNEAU_MENU_LARGEUR)
88          && position_clic.y>PANNEAU_MENU_POS_Y && position_clic.y<(PANNEAU_MENU_POS_Y+PANNEAU_MENU_HAUTEUR))
89          {
90            //S'il s'agit du bouton quitter
91            if(position_clic.x>PANNEAU_MENU_POS_X+30 && position_clic.x<(PANNEAU_MENU_POS_X+75)
92            && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
93            {
94
95              //Affichage du panneau du menu avec le bouton quitter en surbrillance
96              affich_panneau_menu(surf_ecran,1);
97              bool_bouton_etat=true;
98            }
99            //S'il s'agit du bouton sauvegarder
100           else if(position_clic.x>(PANNEAU_MENU_POS_X+60+PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_POS_
101           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
102           {
103             //Affichage du panneau du menu avec le bouton sauvegarder
104             affich_panneau_menu(surf_ecran,2);
105             bool_bouton_etat=true;
106           }
107           //S'il s'agit du bouton tourner plateau
108           else if(position_clic.x>(PANNEAU_MENU_POS_X+90+2*PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_PO
109           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
110           {
111             //Affichage du panneau du menu avec le bouton tourner en surbrillance
112             affich_panneau_menu(surf_ecran,3);
113             bool_bouton_etat=true;
114           }
115         }
116         //Autrement, si l'un des boutons est en surbrillance
```

```
117         else if(bool_bouton_etat==true)
118         {
119           //On rÃľaffiche le panneau avec le bouton fin de tour en surbrillance
120           affich_panneau_fdt(surf_ecran, false);
121           //On rÃľaffiche le panneau du menu
122           affich_panneau_menu(surf_ecran,0);
123           //On rÃľaffiche le panneau avec le bouton lancer dÃľs par dÃľfaut
124           if(int_lancer==0) affich_panneau_des_bouton(surf_ecran,0);
125           //Tout les boutons ne sont plus en surbrillance
126           bool_bouton_etat=false;
127         }
128 
129         break;
130       //En cas de clic
131       case SDL_MOUSEBUTTONUP:
132         //Sauvegarde des coordonnÃľes du clic
133         position_clic.x=event.button.x;
134         position_clic.y=event.button.y;
135         //Si le joueur a cliquÃľ sur le bouton lancer dÃľs
136         if(position_clic.x>PANNEAU_DES_POS_X && position_clic.x<(PANNEAU_DES_POS_X+PANNEAU_DES_LARGEUR)
137         && position_clic.y>PANNEAU_DES_POS_Y && position_clic.y<(PANNEAU_DES_POS_Y+PANNEAU_DES_HAUTEUR) && int_lance
138         {
139           //on verifie que le joueur ne soit pas en prison
140           if (pj_joueur->bool_laurence)
141           {
142             traitement_bureau_laurence(surf_ecran, surf_centre, plateau, pj_joueur, bureau_de_krystel, bureau_de_na
143             //Le joueur a lancÃľ les dÃľs
144             int_lancer=1;
145           }
146           else
147           {
148             //On lance les dÃľs
149             int_nb_tire=lancer_des(surf_ecran,surf_centre,pj_joueur, plateau);
150             //On appelle la fonction traitement s'il n'a pas Ãľtait envoyÃľ en prison
151             if (!pj_joueur->bool_laurence)
152             {
153               avancer_jeton (surf_ecran, surf_centre, int_nb_tire,pj_joueur,plateau,bureau_de_krystel,bureau_de_nade
154             }
155 
156             //Affichage du panneau contenant les informations joueurs
157             affich_panneau_possessions(surf_ecran,pj_joueur);
158 
159             //Affichage du panneau contenant les informations joueurs
160             affich_panneau_joueur(surf_ecran,pj_joueur);
161 
162             //Le joueur a lancÃľ les dÃľs
163             int_lancer=1;
164           }
165 
166         }
167         //Si le joueur clic sur l'un des  boutons de commande (coin supÃľrieur gauche)
168         else if(position_clic.x>PANNEAU_MENU_POS_X && position_clic.x<(PANNEAU_MENU_POS_X+PANNEAU_MENU_LARGEUR)
169         && position_clic.y>PANNEAU_MENU_POS_Y && position_clic.y<(PANNEAU_MENU_POS_Y+PANNEAU_MENU_HAUTEUR))
170         {
171           //S'il s'agit du bouton quitter
172           if(position_clic.x>PANNEAU_MENU_POS_X+30 && position_clic.x<(PANNEAU_MENU_POS_X+75)
173           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
174           {
175             //On quitte le prgm
176             continuer = 0;
177           }
178           //S'il s'agit du bouton sauvegarder
179           else if(position_clic.x>(PANNEAU_MENU_POS_X+60+PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_POS_
180           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
181           {
182             sauvegarde(plateau, pj_joueur);
183             attente_validation_message(surf_ecran, surf_centre, "Sauvegarde", "La sauvegarde Ãă ÃľtÃľ effectuÃľ ave
```

```
184           }
185           //S'il s'agit du bouton tourner plateau
186           else if(position_clic.x>(PANNEAU_MENU_POS_X+90+2*PANNEAU_OBJET_TAILLE) && position_clic.x<(PANNEAU_MENU_PO
187           && position_clic.y>PANNEAU_MENU_POS_Y+10 && position_clic.y<(PANNEAU_MENU_POS_Y+55))
188           {
189
190           }
191         }
192         //Si le joueur a cliquÃ© sur le bouton de fin de tour
193         else if(position_clic.x>PANNEAU_FDT_POS_X && position_clic.x<(PANNEAU_FDT_POS_X+PANNEAU_FDT_LARGEUR)
194         && position_clic.y>PANNEAU_FDT_POS_Y && position_clic.y<(PANNEAU_FDT_POS_Y+PANNEAU_FDT_HAUTEUR) && int_lance
195         {
196           //On arrete la boucle et on passe au joueur suivant
197           continuer=1;
198         }
199         //Si le joueur a cliquÃ© sur le panneau d'affichage des propriÃ©tÃ©s
200         else if(position_clic.x>PANNEAU_POSSESSION_POS_X && position_clic.x<(PANNEAU_POSSESSION_POS_X+PANNEAU_POSSES
201         && position_clic.y>PANNEAU_POSSESSION_POS_Y && position_clic.y<(PANNEAU_POSSESSION_POS_Y+(40+((PROPRIETE_HAU
202         {
203           int_action=action_possible(plateau,(position_clic.y-PANNEAU_POSSESSION_POS_Y-40)/(PROPRIETE_HAUTEUR+5), p
204
205
206
207           //On affiche la propriÃ©tÃ© sur laquelle il a cliquÃ©
208           attente_validation_propriete(surf_ecran, plateau, surf_centre, int_to_cases((position_clic.y-PANNEAU_POSSI
209         }
210         break;
211       //Pour tout les autres Ã©vÃ©nements
212       default:
213         //On ne fait rien...
214         break;
215         }
216     }
217
218     //Renvoit de la derniÃ¨re action du joueur
219   return(continuer);
220 }
```

8.6.2.2 int attente_action_accueil (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_boutons*[4], SDL_Surface * *surf_nombre*[6], SDL_Surface * *surf_fleche_gauche*, SDL_Surface * *surf_fleche_droite*)

détermine l'action à entreprendre en fonction du clic du joueur sur la page d'accueil

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran
surf_boutons tableau des surfaces des boutons de l'accueil
surf_nombre tableau des surfaces des chiffres
surf_fleche_gauche surface de la flèche de gauche
surf_fleche_droite surface de la flèche de droite

Version :

1.0

Renvoie :

le nombre de joueur si ça c'est bien passé sinon -1

Définition à la ligne 222 du fichier interaction.c.

Référencé par affich_accueil().

```
223 {
224   //Joueur actuel
225   int i;
226   i=0;
227   int continuer;
228   continuer=1;
229   //Variable d'Ã©tat des boutons (0 aucun survolÃ©, 1 un des boutons survolÃ©)
230   int int_bouton_etat;
231   int_bouton_etat=0;
232 
233   //Etat des fichiers de sauvegarde (existence ou non)
234   int etat_fichier;
235 
236   //Evenement sdl
237   SDL_Event event;
238 
239   //CoordonnÃ©es du clic
240   SDL_Rect position_clic;
241   //Position temporaire
242   SDL_Rect position_temp;
243   //Position actuel du pointeur
244   SDL_Rect position;
245 
246   position_clic.x=0;
247   position_clic.y=0;
248 
249   //Surface de cache
250   SDL_Surface* surf_cache;
251 
252   etat_fichier=0;
253   if(is_readable("sauvegarde_joueur.txt") && is_readable("sauvegarde_plateau.txt")) etat_fichier=1;
254 
255   //Modification de la position du premier champ
256   position.x=POS_X_FLECHES;
257   position.y=POS_Y_FLECHES;
258   //On colle le premier champ sur le fond
259   SDL_BlitSurface(surf_fleche_gauche, NULL, surf_ecran, &position);
260 
261   //Modification la position du second champ
262   position.x=POS_X_FLECHES+200;
263   position.y=POS_Y_FLECHES;
264   //On colle le second champ sur le fond
265   SDL_BlitSurface(surf_fleche_droite, NULL, surf_ecran, &position);
266 
267   //CrÃ©ation de la surface du cache
268   surf_cache=SDL_CreateRGBSurface(SDL_HWSURFACE, 90, (surf_nombre[5])->h, 32, 0, 0, 0, 0);
269   //Remplissage de noir du cache
270   SDL_FillRect(surf_cache, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
271 
272   //Position du cache
273   position.x=POS_X_FLECHES+100;
274   position.y=POS_Y_FLECHES-100;
275   //Collage du cache sur l'Ã©cran
276   SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
277 
278   //Collage du nombre
279   SDL_BlitSurface(surf_nombre[i%5+1], NULL, surf_ecran, &position);
280 
281   //Mise Ãã jour de l'Ã©cran
282   SDL_Flip(surf_ecran);
283 
284   //Tant qu'on a pas quitter ou valider
285   while (continuer==1)
```

```
286   {
287     //On attend un ÃÏvÃÏnement
288     SDL_WaitEvent(&event);
289     //Selon l'ÃÏvÃÏnement
290     switch(event.type)
291     {
292       case SDL_KEYDOWN:
293         switch(event.key.keysym.sym)
294         {
295           case SDLK_RETURN:
296             continuer=0;
297             break;
298           case SDLK_RIGHT:
299             if(i<4)
300             {
301               //Calcul du nouveau nombre du joueur
302               i=(i+1)%5;
303               //On colle le cache sur l'ÃÏcran
304               SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
305               //On colle l'image du joueur sur l'ÃÏcran
306               SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
307               //Mise Ãà jour de l'ÃÏcran
308               SDL_Flip(surf_ecran);
309             }
310             break;
311           case SDLK_LEFT:
312             if(i>0)
313             {
314               //Calcul du nouveau nombre du joueur
315               i=(i-1)%5;
316               //On colle le cache sur l'ÃÏcran
317               SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
318               //On colle l'image du joueur sur l'ÃÏcran
319               SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
320               //Mise Ãà jour de l'ÃÏcran
321               SDL_Flip(surf_ecran);
322             }
323             break;
324           default:
325             break;
326         }
327       //En cas de dÃÏplacement de la souris
328       case SDL_MOUSEMOTION:
329         //On rÃÏcupÃÍre les coordonnÃÏes du clic
330         position_clic.x=event.button.x;
331         position_clic.y=event.button.y;
332         //S'il s'agit du bouton valider
333         if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCUEIL_LA
334         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
335         {
336           //On modifie la position du bouton valider
337           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
338           position_temp.y=BTN_ACCUEIL_POS_Y;
339           //On colle le bouton valider sur le fond
340           SDL_BlitSurface(surf_boutons[1], NULL, surf_ecran, &position_temp);
341           //on met Ãà jour l'ÃÏcran
342           SDL_Flip(surf_ecran);
343           int_bouton_etat=1;
344         }
345         //S'il s'agit du bouton quitter
346         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[2]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
347         && position_clic.y>10 && position_clic.y<(surf_boutons[2]->h+10))
348         {
349           //On modifie la position du bouton quitter
350           position_temp.x=ECRAN_LARGEUR-(surf_boutons[2]->w+10);
351           position_temp.y=10;
352           //On colle le bouton quitter sur le fond
```

```
353           SDL_BlitSurface(surf_boutons[3], NULL, surf_ecran, &position_temp);
354           //On met Ãă jour l'ÃĽcran
355           SDL_Flip(surf_ecran);
356           int_bouton_etat=1;
357         }
358         //S'il s'agit du bouton chargement et que les fichiers existent
359         else if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCUI
360         && position_clic.y>BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HA
361         {
362           //On modifie la position du bouton chargement
363           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
364           position_temp.y=BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE;
365           //On colle le bouton charger sur le fond
366           SDL_BlitSurface(surf_boutons[5], NULL, surf_ecran, &position_temp);
367           //on met Ãă jour l'ÃĽcran
368           SDL_Flip(surf_ecran);
369           int_bouton_etat=1;
370         }
371         //Si un des boutons est en surbrillance
372         else if (int_bouton_etat==1)
373         {
374           //On modifie la position du bouton valider
375           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
376           position_temp.y=BTN_ACCUEIL_POS_Y;
377           //On colle le bouton valider sur le fond
378           SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position_temp);
379 
380           //On modifie la position du bouton quitter
381           position_temp.x=ECRAN_LARGEUR-(surf_boutons[2]->w+10);
382           position_temp.y=10;
383           //On colle le bouton quitter sur le fond
384           SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position_temp);
385 
386           //On modifie la position du bouton chargement
387           position_temp.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
388           position_temp.y=BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE;
389           //On colle le bouton charger sur le fond
390           SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position_temp);
391 
392           //On met Ãă jour l'ÃĽcran
393           SDL_Flip(surf_ecran);
394           int_bouton_etat=0;
395         }
396 
397         break;
398       //En cas de clic sur un bouton
399       case SDL_MOUSEBUTTONUP:
400         //On sauvegarde les coordonnÃĽes du clic
401         position_clic.x=event.button.x;
402         position_clic.y=event.button.y;
403         //Si on a cliquer sur la fleche gauche
404         if(position_clic.x>POS_X_FLECHES && position_clic.x<(POS_X_FLECHES+surf_fleche_gauche->w)
405         && position_clic.y>POS_Y_FLECHES && position_clic.y<(POS_Y_FLECHES+surf_fleche_gauche->h) && i>0)
406         {
407           //Calcul le nouveau nombre de joueur
408           i=(i-1)%5;
409           //On colle le cache sur l'ÃĽcran
410           SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
411           //On colle l'image du nombre sur l'ÃĽcran
412           SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
413           //Mise Ãă jour de l'ÃĽcran
414           SDL_Flip(surf_ecran);
415         }
416         //S'il s'agit du bouton quitter
417         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[2]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
418         && position_clic.y>10 && position_clic.y<(surf_boutons[2]->h+10))
419         {
```

```
420           continuer=-1;
421         }
422         //Fleche droite
423         else if(position_clic.x>(POS_X_FLECHES+200) && position_clic.x<(POS_X_FLECHES+200+surf_fleche_droite->w)
424         && position_clic.y>POS_Y_FLECHES && position_clic.y<(POS_Y_FLECHES+surf_fleche_droite->h) && i<5)
425         {
426           //Calcul du nouveau nombre du joueur
427           i=(i+1)%5;
428           //On colle le cache sur l'Ã¯cran
429           SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
430           //On colle l'image du joueur sur l'Ã¯cran
431           SDL_BlitSurface(surf_nombre[i+1], NULL, surf_ecran, &position);
432           //Mise Ãa jour de l'Ã¯cran
433           SDL_Flip(surf_ecran);
434         }
435         //Bouton valider
436         else if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCU
437         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
438         {
439           continuer=0;
440         }
441         //Bouton chargement
442         else if(position_clic.x>((ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2) && position_clic.x<((ECRAN_LARGEUR-BTN_ACCU
443         && position_clic.y>BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HA
444         {
445           continuer=-2;
446         }
447
448         break;
449       //En cas d'autre Ã¯vÃ¯nement
450       default:
451         //On ne fait rien...
452         break;
453     }
454     }
455     //Si le joueur Ãa voulu quitter, on retourne sa demande
456     if(continuer==-1) return(continuer);
457     //Si le joueur Ãa voulu charger une nouvelle partie, on retourne sa demande
458     if(continuer==-2) return(continuer);
459     //Sinon on retourne le nombre de joueur
460   return(i+1);
461
462 }
```

8.6.2.3 int attente_action_config (SDL_Surface * *surf_ecran*, char ** *str_nom_joueur*, SDL_Surface * *surf_boutons*[6], SDL_Surface * *surf_champ*, int *nombre_joueur*)

détermine l'action à entreprendre en fonction du clic du joueur sur la page des noms des joueurs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

str_nom_joueur tableau des noms des joueurs

surf_boutons surface des 6 boutons

surf_champ surfaces des champs d'écriture

nombre_joueur nombre de joueurs qui jouent

Version :

1.0

Renvoie :

action du joueur

Définition à la ligne 464 du fichier interaction.c.

Référencé par affich_config().

```
465 {
466   int i;
467   i=0;
468   //Etat des boutons (surbrillance ou non)
469   int int_bouton_etat;
470   int_bouton_etat=0;
471 
472   int continuer;
473   continuer=2;
474 
475   //Champ en cours de modification
476   int int_champ;
477   int_champ=0;
478 
479   //Surface du pion
480   SDL_Surface* surf_pion;
481   surf_pion=NULL;
482 
483   //Surface du cache du pion;
484   SDL_Surface* surf_cache;
485   surf_cache=NULL;
486 
487   //Surface du nom
488   SDL_Surface* surf_nom;
489   surf_nom=NULL;
490 
491   //EvÃ¨nement sdl
492   SDL_Event event;
493 
494   //CoordonnÃ¨es du clic
495   SDL_Rect position_clic;
496   position_clic.x=0;
497   position_clic.y=0;
498 
499   //Ancienne position
500   SDL_Rect position;
501 
502   //Position du champ
503   SDL_Rect position_champ;
504 
505   //Police d'Ã¨criture
506   TTF_Font* police;
507   police=NULL;
508 
509   //Couleur du texte
510   SDL_Color couleur_texte = {0, 0, 0};
511 
512   //Ouverture de la police d'ecriture
513   police = TTF_OpenFont("sdl/police/police.ttf", 17);
514 
515   //CrÃ¨ation du pion
516   surf_pion=SDL_CreateRGBSurface(SDL_HWSURFACE, 20, 20, 32, 0, 0, 0, 0);
517   //Remplissage de l'image du pion
518   SDL_FillRect(surf_pion, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 255));
519 
520   //Ecriture du numÃ¨ro sur la surface
```

```
521   surf_cache=SDL_CreateRGBSurface(SDL_HWSURFACE, 20, 20, 32, 0, 0, 0, 0);
522   //Remplissage du bouton pion
523   SDL_FillRect(surf_cache, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
524
525   //Modification de la position du pion
526   position.x=CHAMP_POS_X-30;
527   position.y=CHAMP_POS_Y+5;
528   //Affichage du pion
529   SDL_BlitSurface(surf_pion, NULL, surf_ecran, &position);
530
531   //Mise Ãă jour de l'ÃĽcran
532   SDL_Flip(surf_ecran);
533
534   //tant que le joueur n'a pas fait prÃĽcÃĽdent, quitter ou jouer
535   while (continuer==2)
536   {
537     //On attend un ÃĽvÃĽnement sdl
538     SDL_WaitEvent(&event);
539     //Selon le type de l'ÃĽvÃĽnement
540     switch(event.type)
541     {
542       //Si on appui sur une touche
543       case SDL_KEYDOWN:
544         //Selon l'ÃĽvÃĽnement
545         switch(event.key.keysym.sym)
546         {
547           //En cas de tabulation
548           case SDLK_TAB:
549             //Affichage du cache
550             SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
551             //Passage au champ suivant
552             int_champ=(int_champ+1)%nombre_joueur;
553             //Calcul de la position du champ en fonction du numÃĽro du champ
554             position.y=int_champ*CHAMP_HAUTEUR*2+CHAMP_POS_Y+5;
555             //Affichage du pion
556             SDL_BlitSurface(surf_pion, NULL, surf_ecran, &position);
557             //Mise Ãă jour de l'ÃĽcran
558             SDL_Flip(surf_ecran);
559             break;
560           //En cas de validation
561           case SDLK_RETURN:
562             continuer=1;
563             for (int i=0;i<nombre_joueur;i++)
564             {
565               //on vÃĽrifie que le nom n'est pas vide
566               if (str_nom_joueur[i][0]=='\0')
567               {
568                 continuer=2;
569               }
570               else
571               {
572                 //on verifie que les noms ne soient pas identiques
573                 for (int j=i+1;j<nombre_joueur;j++)
574                 {
575                   if (strcmp(str_nom_joueur[i],str_nom_joueur[j])==0)
576                   {
577                     continuer=2;
578                   }
579                 }
580               }
581             }
582             break;
583           //Pour toutes les autres touches
584           default:
585             //On positionne le champ
586             position_champ.x=CHAMP_POS_X;
587             position_champ.y=CHAMP_POS_Y+int_champ*2*CHAMP_HAUTEUR;
```

```
588             //On met un cache sur le champ actuel
589             SDL_BlitSurface(surf_champ, NULL, surf_ecran, &position_champ);
590             //On positionne le texte
591             position_champ.x=CHAMP_POS_X+5;
592             position_champ.y=CHAMP_POS_Y+int_champ*2*CHAMP_HAUTEUR-10;
593             //Ecriture du titre sur la surface de titre
594             strcpy(str_nom_joueur[int_champ],ajout_caractere(str_nom_joueur[int_champ],event.key.keysym.sym));
595             //Ecriture du nom du joueur modifiÃľ
596             surf_nom = TTF_RenderText_Blended(police, str_nom_joueur[int_champ], couleur_texte);
597             //On colle le nom du joueur sur le champ
598             SDL_BlitSurface(surf_nom, NULL, surf_ecran, &position_champ);
599             //Mise Ãă jour de l'Ãľcran
600             SDL_Flip(surf_ecran);
601             break;
602         }
603       //En cas de dÃľplacement de la souris
604       case SDL_MOUSEMOTION:
605         //Sauvegarde de la position du clic
606         position_clic.x=event.button.x;
607         position_clic.y=event.button.y;
608         //S'il s'agit du bouton retour
609         if(position_clic.x>(BTN_ACCUEIL_POS_X) && position_clic.x<(BTN_ACCUEIL_POS_X+BTN_ACCUEIL_LARGEUR)
610         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
611         {
612           //On modifie la position du bouton retour
613           position_champ.x=BTN_ACCUEIL_POS_X;
614           position_champ.y=BTN_ACCUEIL_POS_Y;
615           //On colle le bouton retour sur le fond
616           SDL_BlitSurface(surf_boutons[1], NULL, surf_ecran, &position_champ);
617           //On met l'Ãľcran Ãă jour
618           SDL_Flip(surf_ecran);
619           int_bouton_etat=1;
620         }
621         //S'il s'agit du bouton valider
622         else if(position_clic.x>(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w)
623         && position_clic.x<((BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w)+BTN_ACCUEIL_LARGEUR)
624         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
625         {
626           //On modifie la position du bouton valider
627           position_champ.x=(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w);
628           position_champ.y=BTN_ACCUEIL_POS_Y;
629           //On colle le bouton valider sur le fond
630           SDL_BlitSurface(surf_boutons[3], NULL, surf_ecran, &position_champ);
631           //On met l'Ãľcran Ãă jour
632           SDL_Flip(surf_ecran);
633           int_bouton_etat=1;
634         }
635         //S'il s'agit du bouton quitter
636         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[4]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
637         && position_clic.y>10 && position_clic.y<(surf_boutons[4]->h+10))
638         {
639           //On modifie la position du bouton quitter
640           position_champ.x=ECRAN_LARGEUR-(surf_boutons[4]->w+10);
641           position_champ.y=10;
642           //On colle le bouton quitter sur le fond
643           SDL_BlitSurface(surf_boutons[5], NULL, surf_ecran, &position_champ);
644           //On met Ãă jour l'Ãľcran
645           SDL_Flip(surf_ecran);
646           int_bouton_etat=1;
647         }
648         //Si un des boutons est en surbrillance
649         else if (int_bouton_etat==1)
650         {
651           //On modifie la position du bouton valider
652           position_champ.x=(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w);
653           position_champ.y=BTN_ACCUEIL_POS_Y;
654           //On colle le bouton valider sur le fond
```

```
655           SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position_champ);
656
657           //On modifie la position du bouton retour
658           position_champ.x=BTN_ACCUEIL_POS_X;
659           position_champ.y=BTN_ACCUEIL_POS_Y;
660           //On colle le bouton retour sur le fond
661           SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position_champ);
662
663           //On modifie la position du bouton quitter
664           position_champ.x=ECRAN_LARGEUR-(surf_boutons[4]->w+10);
665           position_champ.y=10;
666           //On colle le bouton quitter sur le fond
667           SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position_champ);
668
669           //On met Ãă jour l'ÃĽcran
670           SDL_Flip(surf_ecran);
671           int_bouton_etat=0;
672         }
673
674         break;
675       //En cas de clic
676       case SDL_MOUSEBUTTONUP:
677         //On sauvegarde les coordonnÃĽes du clic
678         position_clic.x=event.button.x;
679         position_clic.y=event.button.y;
680         //S'il s'agit du bouton valider
681         if(position_clic.x>(BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w) && position_clic.x<((BTN_ACCUEIL_POS_X+20+(su
682         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
683         {
684           continuer=1;
685
686           for (int i=0;i<nombre_joueur;i++)
687           {
688             //on vÃĽrifie que le nom n'est pas vide
689             if (str_nom_joueur[i][0]=='\0')
690             {
691               continuer=2;
692             }
693             else
694             {
695               //on verifie que les noms ne soient pas identiques
696               for (int j=i+1;j<nombre_joueur;j++)
697               {
698                 if (strcmp(str_nom_joueur[i],str_nom_joueur[j])==0)
699                 {
700                   continuer=2;
701                 }
702               }
703             }
704           }
705         }
706         //S'il s'agit du bouton retour
707         else if(position_clic.x>(BTN_ACCUEIL_POS_X) && position_clic.x<(BTN_ACCUEIL_POS_X+BTN_ACCUEIL_LARGEUR)
708         && position_clic.y>BTN_ACCUEIL_POS_Y && position_clic.y<(BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_HAUTEUR))
709         {
710           continuer=-1;
711         }
712         //S'il s'agit du bouton quitter
713         else if(position_clic.x>(ECRAN_LARGEUR-(surf_boutons[4]->w+10)) && position_clic.x<(ECRAN_LARGEUR-10)
714         && position_clic.y>10 && position_clic.y<(surf_boutons[4]->h+10))
715         {
716           continuer=0;
717         }
718         //Si on a cliquer sur un des champs
719         else if(position_clic.x>(CHAMP_POS_X) && position_clic.x<(CHAMP_POS_X+CHAMP_LARGEUR)
720         && position_clic.y>CHAMP_POS_Y && position_clic.y<(CHAMP_POS_Y+CHAMP_HAUTEUR*2*nombre_joueur))
721         {
```

```
722           //Affichage du cache
723           SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
724           //On calcul sur quel champ le joueur Ãă cliquÃĭ
725           int_champ=((position_clic.y-CHAMP_POS_Y)/(CHAMP_HAUTEUR*2));
726           //Calcul de la position du champ en fonction du numÃĭro du champ
727           position.y=int_champ*CHAMP_HAUTEUR*2+CHAMP_POS_Y+5;
728           //Affichage du pion
729           SDL_BlitSurface(surf_pion, NULL, surf_ecran, &position);
730           //Mise Ãă jour de l'Ãĭcran
731           SDL_Flip(surf_ecran);
732         }
733         break;
734       //S'il s'agit d'un autre ÃĭvÃĭnement
735       default:
736         //On ne fait rien.
737         break;
738     }
739   }
740
741   //Fermeture de la police d'ecriture
742   TTF_CloseFont(police);
743   //LibÃĭration des surfaces
744   SDL_FreeSurface(surf_pion);
745   SDL_FreeSurface(surf_cache);
746   SDL_FreeSurface(surf_nom);
747   //On retourne la valeur de continuer
748   return(continuer);
749
750 }
```

8.6.2.4 int attente_validation_propriete (SDL_Surface * *surf_ecran*, cases ** *plateau*, SDL_Surface * *surf_centre*, cases * *pcase*, joueur * *pj_joueur*, int *int_type_action*)

détermine l'action à entreprendre en fonction du clic du joueur losqu'il est tombé sur une propriété

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

plateau plateau de jeu

pcase case en cours de traitement

pj_joueur joueur qui joue actuellement

int_type_action indique le type de l'action à entreprendre

Version :

1.0

Renvoie :

action du joueur

Définition à la ligne 752 du fichier interaction.c.

Référencé par attente_clic(), et traitement_arrive_case().

```
753 {
754   //Surface du fond
755   SDL_Surface* surf_fond;
756   surf_fond=NULL;
757 
758   //EvÃ¨nement sdl
759   SDL_Event event;
760 
761   //CoordonnÃ¨es
762   SDL_Rect position;
763   //CoordonnÃ¨es du clic
764   SDL_Rect position_clic;
765 
766   int int_position;
767   int int_type;
768   int_type=pcase->int_type;
769   int int_etat;
770   int_etat=0;
771   int continuer;
772   continuer=0;
773 
774   //CrÃ¨ation du fond
775   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR+DETAIL_MESSAGE_EPAISSEUR*2, DETAIL_HAUTEUR+DETAIL_MES
776   //Remplissage de l'image de fond
777   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 125, 0));
778 
779   //Position de la carte sur le message
780   position.x=DETAIL_MESSAGE_EPAISSEUR;
781   position.y=DETAIL_MESSAGE_EPAISSEUR;
782 
783   //Collage de la carte sur le message
784   if(int_type==SALLE) SDL_BlitSurface(pcase->case_salle.surf_detail, NULL, surf_fond, &position);
785   else if(int_type==BDE || int_type==BDS) SDL_BlitSurface(pcase->case_association.surf_detail, NULL, surf_fond, &pos
786   else if(int_type>=LC_WC && int_type<=LC_PARKING) SDL_BlitSurface(pcase->case_lieu_commun.surf_detail, NULL, surf_
787   else cerr << "ERREUR !!" << endl;
788 
789   //Affichage de la propriÃ¨tÃ¨
790   action_to_boutons(surf_ecran, surf_fond, int_type_action, 0);
791 
792   //Tant que le joueur n'a pas fait d'action
793   while(!continuer)
794   {
795       //On attend un Ã¨vÃ¨nement
796         SDL_WaitEvent(&event);
797         //Selon l'Ã¨vÃ¨nement
798         switch(event.type)
799         {
800       //S'il s'agit d'un appui sur une touche
801       case SDL_KEYDOWN:
802         //Selon la touche
803         switch(event.key.keysym.sym)
804         {
805           //Echap
806           case SDLK_ESCAPE:
807             //On quitte le prgm
808             continuer = 1;
809             break;
810           //Pour toutes les autres touches
811           default:
812             //On fait rien...
813             break;
814         }
815         break;
816       //En cas de mouvement de la souris
817       case SDL_MOUSEMOTION:
818         //On sauvegarde les nouvelles coordonnÃ¨es de la souris
819         position_clic.x=event.button.x;
```

```
820         position_clic.y=event.button.y;
821
822         //S'il s'agit du premier bouton
823         if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
824         && position_clic.y>DETAIL_BOUTON_POS_Y && position_clic.y<(DETAIL_BOUTON_POS_Y+40))
825         {
826           action_to_boutons(surf_ecran, surf_fond, int_type_action, 1);
827           int_etat=1;
828         }
829         //Bouton 2 sauf pour le cas oÃź on finit
830         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
831         && position_clic.y>DETAIL_BOUTON_POS_Y+50 && position_clic.y<(DETAIL_BOUTON_POS_Y+90) && int_type_action !=
832         {
833           action_to_boutons(surf_ecran, surf_fond, int_type_action, 2);
834           int_etat=2;
835         }
836         //Bouton 3 dans le cas d'un plus moins
837         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
838         && position_clic.y>DETAIL_BOUTON_POS_Y+100 && position_clic.y<(DETAIL_BOUTON_POS_Y+140) && int_type_action =
839         {
840           action_to_boutons(surf_ecran, surf_fond, int_type_action, 3);
841           int_etat=2;
842         }
843         else if(int_etat!=0)
844         {
845           action_to_boutons(surf_ecran, surf_fond, int_type_action, 0);
846           int_etat=0;
847         }
848
849         break;
850       case SDL_MOUSEBUTTONUP:
851         //On sauvegarde les nouvelles coordonnÃľes de la souris
852         position_clic.x=event.button.x;
853         position_clic.y=event.button.y;
854         //Premier bouton
855         if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
856         && position_clic.y>DETAIL_BOUTON_POS_Y && position_clic.y<(DETAIL_BOUTON_POS_Y+50))
857         {
858           //Selon l'action Ãă faire
859           switch(int_type_action)
860           {
861             //Dans le cas d'un achat
862             case ACTION_ACHAT:
863               traitement_achat(pcase,pj_joueur);
864               break;
865             //Augmentation de niveau
866             case ACTION_PLUS:
867               traitement_augmentation_niveau(pcase,pj_joueur);
868               break;
869             //Diminution de niveau
870             case ACTION_MOINS:
871               traitement_diminution_niveau(pcase,pj_joueur);
872               break;
873             //Dans le cas d'une dÃľshypothÃÎque
874             case ACTION_PLUS_MOINS:
875               traitement_augmentation_niveau(pcase,pj_joueur);
876               break;
877             //Dans tout les autres cas
878             default:
879               break;
880           }
881           continuer=1;
882         }
883         //Bouton 2 sauf pour le cas oÃź on finit
884         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
885         && position_clic.y>DETAIL_BOUTON_POS_Y+50 && position_clic.y<(DETAIL_BOUTON_POS_Y+90) && int_type_action !=
886         {
```

```
887           switch(int_type_action)
888           {
889             case ACTION_PLUS_MOINS:
890               traitement_diminution_niveau(pcase,pj_joueur);
891               break;
892             //Dans tout les autres cas
893             default:
894               break;
895           }
896           continuer=1;
897         }
898         //Bouton 3 dans le cas d'un plus moins
899         else if(position_clic.x>DETAIL_BOUTON_POS_X && position_clic.x<(DETAIL_BOUTON_POS_X+165)
900         && position_clic.y>DETAIL_BOUTON_POS_Y+100 && position_clic.y<(DETAIL_BOUTON_POS_Y+140) && int_type_action =
901         {
902           continuer=1;
903         }
904         break;
905       default:
906         break;
907         }
908   }
909
910   //on rÃľactualise la case ou se trouve le joueur afin qu'il efface le jeton
911   affich_case(surf_ecran,pcase);
912   //on compte le nombre de personne sur la case ou doit Ãłtre afficher le joueur actuel
913   int_position=nombre_joueur_case(pj_joueur);
914   //on affiche le joueur sur la case s'il se trouve dessus
915   if(plateau[pj_joueur->int_position]==pcase) affich_joueur(surf_ecran, pj_joueur,int_position+1, pcase);
916
917   //Mise Ãă jour de l'Ãľcran
918   SDL_Flip(surf_ecran);
919
920   //RÃľaffichage du centre
921   affich_centre(surf_ecran,surf_centre);
922   //Mise Ãă jour de l'Ãľcran
923   SDL_Flip(surf_ecran);
924   return(0);
925
926 }
```

8.6.2.5 int attente_validation_message (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, char * *titre*, char * *message*, int *int_type_message*)

détermine l'action à entreprendre en fonction du clic du joueur lors de l'affichage d'un message

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

titre le titre du message

message le message à afficher

int_type_message type du message à afficher

Version :

1.0

Renvoie :

action du joueur

Définition à la ligne 928 du fichier interaction.c.

Référencé par attente_clic(), jeu(), lancer_des(), traitement_arrive_case(), traitement_bureau(), traitement_bureau_laurence(), traitement_machine_a_cafe(), et traitement_soiree().

```
929 {
930   //Surface du fond
931   SDL_Surface* surf_fond;
932   surf_fond=NULL;
933 
934   //EvÃ¨nement sdl
935   SDL_Event event;
936 
937   //Surface du message
938   SDL_Surface* surf_message;
939 
940   //Position du bouton
941   SDL_Rect position_bouton;
942 
943   //Position du message
944   SDL_Rect position_message;
945 
946   //CoordonnÃ¨es du clic
947   SDL_Rect position_clic;
948 
949   //Nombre de ligne du texte compilÃ¨
950   int int_nombre_ligne;
951   int_nombre_ligne=0;
952 
953   //Texte dÃ¨coupÃ¨
954   char*** texte;
955   texte=new char**[1];
956 
957   //Etat des boutons
958   int int_etat;
959   int_etat=0;
960 
961   //Variable de boucle
962   int continuer;
963   continuer=2;
964 
965   int i;
966 
967   //Fonction de dÃ¨coupe du texte
968   int_nombre_ligne=decoupage_string(texte, message);
969   //CrÃ¨ation du message
970   surf_message=creation_message(surf_ecran, titre, *texte, int_type_message, int_nombre_ligne);
971 
972   //Affichage du message Ãă l'Ã¨cran
973   affich_message(surf_ecran,surf_message,int_type_message, 0);
974 
975   //Calcul de la position du message
976   position_message.x=POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_LARGEUR)/2;
977   position_message.y=POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_HAUTEUR)/2;
978 
979   //Calcul de la postion du premier pour Ã¨viter de le recalculer Ãă chaque test
980   position_bouton.x=position_message.x+(MESSAGE_LARGEUR-160)/2;
981   position_bouton.y=position_message.y+MESSAGE_HAUTEUR-60;
982 
983   //cas particulier s'il s'agit du message quitter
984   if(int_type_message==MESSAGE_QUITTER)
985   {
986     position_bouton.x=position_message.x+(MESSAGE_LARGEUR-320)/3;
```

```
987     position_bouton.y=position_message.y+MESSAGE_HAUTEUR-60;
988   }
989
990   //Tant que le joueur n'a pas fait d'action
991   while(continuer==2)
992   {
993       //On attend un Ã'vÃ'nement
994         SDL_WaitEvent(&event);
995         //Selon l'Ã'vÃ'nement
996         switch(event.type)
997         {
998       //En cas de mouvement de la souris
999       case SDL_MOUSEMOTION:
1000         //On sauvegarde les nouvelles coordonnÃ'es de la souris
1001         position_clic.x=event.button.x;
1002         position_clic.y=event.button.y;
1003         //S'il s'agit du premier bouton
1004         if(position_clic.x>(position_bouton.x) && position_clic.x<(position_bouton.x+160)
1005         && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message!=MESSA
1006         {
1007           //Bouton nÂř1 en surbrillance
1008           affich_message(surf_ecran,surf_message,int_type_message, 1);
1009           int_etat=1;
1010         }
1011         //Dans le cas d'un message quitter, on a un second bouton
1012         else if(position_clic.x>(position_bouton.x+(MESSAGE_LARGEUR-160)/3) && position_clic.x<(position_bouton.x+
1013         && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message==MESSA
1014         {
1015           //Mise en surbrillance du second bouton
1016           affich_message(surf_ecran,surf_message,int_type_message, 2);
1017           int_etat=2;
1018         }
1019         //Si jamais c'est un message prison, on a trois boutons
1020         else if(int_type_message==MESSAGE_PRISON)
1021         {
1022           //Bouton 1 : Attendre
1023           if(position_clic.x>(position_message.x+30) && position_clic.x<(position_message.x+190)
1024           && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1025           {
1026             //Mise en surbrillance
1027             affich_message(surf_ecran,surf_message,int_type_message, 1);
1028             int_etat=1;
1029           }
1030           //Bouton 2 : Payer
1031           else if(position_clic.x>(position_message.x+(MESSAGE_LARGEUR-160)/2) && position_clic.x<(position_message
1032           && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1033           {
1034             //Mise en surbrillance
1035             affich_message(surf_ecran,surf_message,int_type_message, 2);
1036             int_etat=2;
1037           }
1038           //Bouton 3 : Certificat
1039           else if(position_clic.x>(position_message.x+MESSAGE_LARGEUR-190) && position_clic.x<(position_message.x+M
1040           && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1041           {
1042             //Mise en surbrillance
1043             affich_message(surf_ecran,surf_message,int_type_message, 3);
1044             int_etat=3;
1045           }
1046           //Autement, si jamais l'un des boutonsdu message prison est en surbrillance
1047           else if(int_etat!=0)
1048           {
1049             //On rÃ'affiche le message avec les boutons normaux
1050             affich_message(surf_ecran,surf_message,int_type_message, 0);
1051             int_etat=0;
1052           }
1053         }
```

```
1054        //Autement, si jamais l'un des boutons est en surbrillance
1055        else if(int_etat!=0)
1056        {
1057          //On rÃľaffiche le message avec les boutons normaux
1058          affich_message(surf_ecran,surf_message,int_type_message, 0);
1059          int_etat=0;
1060        }
1061
1062        break;
1063      case SDL_MOUSEBUTTONUP:
1064        //On sauvegarde les nouvelles coordonnÃľes de la souris
1065        position_clic.x=event.button.x;
1066        position_clic.y=event.button.y;
1067        //Si on appui sur le premier bouton
1068        if(position_clic.x>(position_bouton.x) && position_clic.x<(position_bouton.x+160)
1069        && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message!=MESSA
1070        {
1071          //On sort de la boucle
1072          continuer=0;
1073        }
1074        else if(int_type_message==MESSAGE_PRISON)
1075        {
1076          //Bouton 1 : Attendre
1077          if(position_clic.x>(position_message.x+30) && position_clic.x<(position_message.x+190)
1078          && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1079          {
1080            continuer=4;
1081          }
1082          //Bouton 2 : Payer
1083          else if(position_clic.x>(position_message.x+(MESSAGE_LARGEUR-160)/2) && position_clic.x<(position_message
1084          && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1085          {
1086            continuer=5;
1087          }
1088          //Bouton 3 : Certificat
1089          else if(position_clic.x>(position_message.x+MESSAGE_LARGEUR-190) && position_clic.x<(position_message.x+M
1090          && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+40))
1091          {
1092            continuer=6;
1093          }
1094        }
1095        //S'il s'agit d'un message quitter et que l'on clique sur le bouton sauvegarder
1096        else if(position_clic.x>(position_bouton.x+(MESSAGE_LARGEUR-160)/3) && position_clic.x<(position_bouton.x+(
1097        && position_clic.y>(position_bouton.y) && position_clic.y<(position_bouton.y+50) && int_type_message==MESSA
1098        {
1099          //Normalement ici il y Ãă la fonction de sauvegarde
1100          continuer=3;
1101        }
1102        break;
1103      default:
1104        break;
1105        }
1106  }
1107
1108  //RÃľaffichage du centre
1109  affich_centre(surf_ecran,surf_centre);
1110  //Mise Ãă jour de l'Ãľcran
1111  SDL_Flip(surf_ecran);
1112
1113  //LibÃľration du message compilÃľ
1114  for (i = 0; i <int_nombre_ligne; i++)
1115  {
1116    delete texte[0][i];
1117  }
1118
1119  delete texte;
1120
```

```
1121   //On retourne l'action du joueur
1122   return(continuer);
1123
1124 }
```

8.7 Référence du fichier monopoly.c

code source du main

Fonctions

– int **main** (int argc, char ∗argv[])
détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu

8.7.1 Description détaillée

code source du main

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **monopoly.c**.

8.7.2 Documentation des fonctions

8.7.2.1 int main (int *argc*, char ∗ *argv*[ ])

détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

argc nombre d'argument lors du lancement du programme

argv tableau des arguments lors du lancement du programme

Version :

1.0

Renvoie :

0

Définition à la ligne 13 du fichier monopoly.c.

```
14 {
15   information* bureau_de_krystel[16];
16   information* bureau_de_nadege[16];
17
18
```

```
19   //initialisation du random
20   srand(time(NULL));
21
22   //ring de joueurs
23   joueur* j_anneau_joueurs;
24   j_anneau_joueurs=NULL;
25
26   //ring de joueurs temporaire
27   joueur* pj_joueurs;
28   pj_joueurs=NULL;
29
30   //Surface de l'Ãl'cran
31   SDL_Surface* surf_ecran = NULL;
32
33   int i;
34
35   //Nombre de joueur
36   int nombre_joueur;
37   int int_nombre_joueur;
38
39   int int_retour_config;
40   int int_retour_accueil;
41   int_retour_accueil=1;
42
43   //Nom des joueurs
44   char** str_nom_joueur;
45   str_nom_joueur=NULL;
46
47   //Tableau de couleur
48   rvb_couleur couleurs[8];
49
50   //Plateau de jeu
51   cases** plateau;
52
53   //Le plateau de jeu est composÃl' de 40 pointeurs vers des instances de la structure cases
54   plateau=new cases*[40];
55
56   //Chaque cases est un pointeur vers une instance de la structure case
57   for(i=0;i<40;i++)
58   {
59     plateau[i]=new cases;
60   }
61
62   //Initialisation des couleurs
63   init_tab_rvb_couleur(couleurs);
64
65   //Initialisation de sdl
66   surf_ecran=init_sdl();
67
68   //Affichage de l'Ãl'cran d'accueil
69   nombre_joueur=affich_accueil(surf_ecran);
70   if(nombre_joueur!=-1)
71   {
72     //S'il ne s'agit pas du chargement d'une partie
73     if(nombre_joueur!=-2)
74     {
75       nombre_joueur++;
76       //Allocation des joueurs
77       str_nom_joueur=new char*[nombre_joueur];
78
79       //Initialisation du tableau
80       for(i=0;i<nombre_joueur;i++)
81       {
82         str_nom_joueur[i]=new char[16];
83         //Remplissage de la ligne avec des fins de ligne
84         memset(str_nom_joueur[i],(int)'\0',1);
85       }
```

```
86      //Affichage de l'Ãľcran de configuration
87      while((int_retour_config=affich_config(surf_ecran,str_nom_joueur,nombre_joueur))==-1 && int_retour_accueil)
88      {
89        //Libÿration des noms des joueurs
90        for(i=0;i<nombre_joueur;i++)
91        {
92          delete(str_nom_joueur[i]);
93        }
94        delete(str_nom_joueur);
95        //Affichage de l'Ãľcran d'accueil
96        nombre_joueur=affich_accueil(surf_ecran);
97        if(nombre_joueur!=-1)
98        {
99          nombre_joueur++;
100          //Allocation des joueurs
101          str_nom_joueur=new char*[nombre_joueur];
102          //Initialisation du tableau
103          for(i=0;i<nombre_joueur;i++)
104          {
105            str_nom_joueur[i]=new char[16];;
106            //Remplissage du tableau avec des fins de ligne
107            memset(str_nom_joueur[i],(int)'\0',1);
108          }
109        }
110        else
111        {
112          int_retour_accueil=0;
113        }
114      }
115    }
116    //Si le jeu peut commencer
117    if((int_retour_config*int_retour_accueil)!=0 || nombre_joueur==-2)
118    {
119      //S'il ne s'agit pas d'un chargement
120      if(nombre_joueur!=-2)
121      {
122        //Initialisation des joueurs
123        j_anneau_joueurs=init_anneau_joueur(nombre_joueur, str_nom_joueur);
124
125        //Initialisation du plateau
126        init_plateau(surf_ecran, couleurs, plateau);
127      }
128      //S'il s'agit d'un chargement
129      else
130      {
131        j_anneau_joueurs=chargement(j_anneau_joueurs, plateau, couleurs, surf_ecran);
132      }
133      //Initialisation du bureau de krystel et du bureau de NadÃĺge
134      init_bureau_krystel(bureau_de_krystel);
135      init_bureau_nadege(bureau_de_nadege);
136
137      pj_joueurs=j_anneau_joueurs;
138      int_nombre_joueur=1;
139      while((pj_joueurs=pj_joueurs->pjoueur_suivant)!=j_anneau_joueurs) int_nombre_joueur++;;
140
141      //Lancement du jeu
142      jeu(surf_ecran, j_anneau_joueurs,plateau,int_nombre_joueur, bureau_de_krystel, bureau_de_nadege);
143    }
144  }
145  //On quitte le module SDL
146  SDL_Quit();
147  //On quitte le module TTF
148  TTF_Quit();
149  //Libÿration des cases du tableau
150  for(i=0;i<40;i++) delete(plateau[i]);
151  //Libÿration des noms des joueurs
152  if(nombre_joueur!=-2)
```

```
153   {
154     for(i=0;i<nombre_joueur;i++) delete(str_nom_joueur[i]);
155     delete(str_nom_joueur);
156   }
157   //LibÃ¯ration du plateau
158   delete(plateau);
159 
160   return (0);
161 }
```

8.8 Référence du fichier monopoly.h

header du main

Fonctions

– int **main** (int argc, char ∗argv[])
détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu

8.8.1 Description détaillée

header du main

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **monopoly.h**.

8.8.2 Documentation des fonctions

8.8.2.1 int main (int *argc*, char ∗ *argv*[ ])

détermine l'action à entreprendre en fonction du clic du joueur pendant le jeu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

argc nombre d'argument lors du lancement du programme

argv tableau des arguments lors du lancement du programme

Version :

1.0

Renvoie :

0

Définition à la ligne 13 du fichier monopoly.c.

```
14 {
15   information* bureau_de_krystel[16];
16   information* bureau_de_nadege[16];
17
18
```

```
19   //initialisation du random
20   srand(time(NULL));
21
22   //ring de joueurs
23   joueur* j_anneau_joueurs;
24   j_anneau_joueurs=NULL;
25
26   //ring de joueurs temporaire
27   joueur* pj_joueurs;
28   pj_joueurs=NULL;
29
30   //Surface de l'Ãl'cran
31   SDL_Surface* surf_ecran = NULL;
32
33   int i;
34
35   //Nombre de joueur
36   int nombre_joueur;
37   int int_nombre_joueur;
38
39   int int_retour_config;
40   int int_retour_accueil;
41   int_retour_accueil=1;
42
43   //Nom des joueurs
44   char** str_nom_joueur;
45   str_nom_joueur=NULL;
46
47   //Tableau de couleur
48   rvb_couleur couleurs[8];
49
50   //Plateau de jeu
51   cases** plateau;
52
53   //Le plateau de jeu est composÃl' de 40 pointeurs vers des instances de la structure cases
54   plateau=new cases*[40];
55
56   //Chaque cases est un pointeur vers une instance de la structure case
57   for(i=0;i<40;i++)
58   {
59     plateau[i]=new cases;
60   }
61
62   //Initialisation des couleurs
63   init_tab_rvb_couleur(couleurs);
64
65   //Initialisation de sdl
66   surf_ecran=init_sdl();
67
68   //Affichage de l'Ãl'cran d'accueil
69   nombre_joueur=affich_accueil(surf_ecran);
70   if(nombre_joueur!=-1)
71   {
72     //S'il ne s'agit pas du chargement d'une partie
73     if(nombre_joueur!=-2)
74     {
75       nombre_joueur++;
76       //Allocation des joueurs
77       str_nom_joueur=new char*[nombre_joueur];
78
79       //Initialisation du tableau
80       for(i=0;i<nombre_joueur;i++)
81       {
82         str_nom_joueur[i]=new char[16];
83         //Remplissage de la ligne avec des fins de ligne
84         memset(str_nom_joueur[i],(int)'\0',1);
85       }
```

```
86      //Affichage de l'Ãľcran de configuration
87      while((int_retour_config=affich_config(surf_ecran,str_nom_joueur,nombre_joueur))==-1 && int_retour_accueil)
88      {
89        //Libÿration des noms des joueurs
90        for(i=0;i<nombre_joueur;i++)
91        {
92          delete(str_nom_joueur[i]);
93        }
94        delete(str_nom_joueur);
95        //Affichage de l'Ãľcran d'accueil
96        nombre_joueur=affich_accueil(surf_ecran);
97        if(nombre_joueur!=-1)
98        {
99          nombre_joueur++;
100          //Allocation des joueurs
101          str_nom_joueur=new char*[nombre_joueur];
102          //Initialisation du tableau
103          for(i=0;i<nombre_joueur;i++)
104          {
105            str_nom_joueur[i]=new char[16];;
106            //Remplissage du tableau avec des fins de ligne
107            memset(str_nom_joueur[i],(int)'\0',1);
108          }
109        }
110        else
111        {
112          int_retour_accueil=0;
113        }
114      }
115    }
116    //Si le jeu peut commencer
117    if((int_retour_config*int_retour_accueil)!=0 || nombre_joueur==-2)
118    {
119      //S'il ne s'agit pas d'un chargement
120      if(nombre_joueur!=-2)
121      {
122        //Initialisation des joueurs
123        j_anneau_joueurs=init_anneau_joueur(nombre_joueur, str_nom_joueur);
124
125        //Initialisation du plateau
126        init_plateau(surf_ecran, couleurs, plateau);
127      }
128      //S'il s'agit d'un chargement
129      else
130      {
131        j_anneau_joueurs=chargement(j_anneau_joueurs, plateau, couleurs, surf_ecran);
132      }
133      //Initialisation du bureau de krystel et du bureau de NadÃĺge
134      init_bureau_krystel(bureau_de_krystel);
135      init_bureau_nadege(bureau_de_nadege);
136
137      pj_joueurs=j_anneau_joueurs;
138      int_nombre_joueur=1;
139      while((pj_joueurs=pj_joueurs->pjoueur_suivant)!=j_anneau_joueurs) int_nombre_joueur++;;
140
141      //Lancement du jeu
142      jeu(surf_ecran, j_anneau_joueurs,plateau,int_nombre_joueur, bureau_de_krystel, bureau_de_nadege);
143    }
144  }
145  //On quitte le module SDL
146  SDL_Quit();
147  //On quitte le module TTF
148  TTF_Quit();
149  //Libÿration des cases du tableau
150  for(i=0;i<40;i++) delete(plateau[i]);
151  //Libÿration des noms des joueurs
152  if(nombre_joueur!=-2)
```

```
153   {
154     for(i=0;i<nombre_joueur;i++) delete(str_nom_joueur[i]);
155     delete(str_nom_joueur);
156   }
157   //LibÃ¯ration du plateau
158   delete(plateau);
159
160   return (0);
161 }
```

8.9 Référence du fichier outils.c

code source des outils

Fonctions

- char * **ajout_caractere** (char *texte, int int_caractere)
  ajoute un caractere (en code UTF-8) en fin de chaine

- int **nombre_joueur_case** (**joueur** *pj_joueur)
  calcule le nombre de joueur sur une case

- void **melanger_cartes** (**information** *cartes_a_melanger[16])
  mélange les cartes du bureau en effectuant une suite aléatoire d'échanges

- void **swap_carte** (**information** *cartes_a_melanger[16], int position_case1, int position_case2)
  échange de cartes dans un tableau d'un bureau

- void **joueur_possede_tout** (**cases** **plateau, **joueur** *joueur_encours)
- **possession** * **insertion_bonne_place_propriete** (**possession** *premier_maillon, **cases** *case_achetee)
- bool **possession_autres_cartes** (**cases** **plateau, **joueur** *pj_joueur)
  verifie si le joueur possède les autres cartes de la même couleur que celle ou il se trouve

- int **decoupage_string** (char ***str_tab, char *str_a_diviser)
  divise une chaine en caractère en tableau de chaîne caractère

- int **action_possible** (**cases** **plateau, int int_position_cases_etudiee, **joueur** *pj_joueur, int int_etat, int int_nb_tire)
  analyse toutes les actions possibles pour un joueur pour une case donnée

- int **nombre_propriete** (**joueur** *pj_joueur)
  compte le nombre de propriétés du joueur

- **cases** * **int_to_cases** (int int_indice_case, **joueur** *pj_joueur)
  à partir de l'indice de chaine des propriété elle trouve la case correspondante

- bool **verification_victoire** (**joueur** *pj_joueur)
  compte le nombre de propriétés du joueur

- void **action_to_boutons** (SDL_Surface *surf_ecran, SDL_Surface *surf_fond, int int_action, int int_etat)
  définit les boutons à afficher en fonction des actions possibles sur une propriété

- void **sauvegarde** (**cases** **plateau, **joueur** *pj_joueur_tete)
  sauvegarde de la partie

- **joueur** * **chargement** (**joueur** *pj_joueur, **cases** **plateau, **rvb_couleur** couleurs[8], SDL_Surface *surf_ecran)
  chargement de la partie

- bool **is_readable** (const std : :string &file)
  test si un fichier existe

8.9.1 Description détaillée

code source des outils

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **outils.c**.

8.9.2 Documentation des fonctions

8.9.2.1 char * ajout_caractere (char * *texte*, int *int_caractere*)

ajoute un caractere (en code UTF-8) en fin de chaine

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

texte texte initial

int_caractere entier correspondant à une lettre

Version :

1.0

Renvoie :

la chaine initiale avec le caractère à la fin

Définition à la ligne 11 du fichier outils.c.

Référencé par attente_action_config().

```
12 {
13   char c_caractere[2];
14   char texte_temp[16];
15   c_caractere[0]=(char)int_caractere;
16   c_caractere[1]='\0';
17 
18   strcpy(texte_temp,texte);
19 
20 
21   //on vÃ©rifie si la chaine est non vide
22   if ((texte_temp!=NULL)) //&& (texte_temp[0]!='\0'))
23   {
24     //si le caractÃ¨re rentrÃ© est le backspace
25     if(int_caractere==8)
26     {
27       //la derniÃ¨re case pleine de la chaine prend la valeur \0
28       texte_temp[strlen(texte_temp)-1]='\0';
```

```
29     }
30     //s'il ne s'agit pas du backspace
31     else
32     {
33       //la dernier case de la chaine de caractÃÍre prend pour valeur le caractÃÍre de la table ascii coorespondant
34       //si la chaine n'est pas pleine et si le caractÃÍre est admissible
35       if ((strlen(texte_temp)<14)
36         && (
37         ((int_caractere>=97) && (int_caractere<=122)) //lettres minuscules
38         || ((int_caractere>=30) && (int_caractere<=39)) //chiffres
39         || (int_caractere==95) //underscore
40         || (int_caractere==45) //tirÃ¯
41         || (int_caractere==32) // espace
42         || (int_caractere==46))) //point
43       {
44       strcpy(texte_temp,strcat(texte_temp,c_caractere));
45       }
46     }
47   }
48   strcpy(texte,texte_temp);
49   return(texte);
50 }
```

8.9.2.2 int nombre_joueur_case (joueur * *pj_joueur*)

calcule le nombre de joueur sur une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur le joueur actuellement en train de jouer son tour

Version :

1.0

Renvoie :

le nombre de joueur sur la case

Définition à la ligne 52 du fichier outils.c.

Référencé par aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), et reculer_jeton().

```
53 {
54   int int_mm_case;
55
56   int_mm_case=0;
57
58   joueur* pj_joueur_encours;
59
60   pj_joueur_encours=pj_joueur->pjoueur_suivant;
61
62   while (pj_joueur_encours != pj_joueur)
63   {
64     if (pj_joueur_encours->int_position==pj_joueur->int_position)
65     {
66       int_mm_case++;
67     }
68     pj_joueur_encours=pj_joueur_encours->pjoueur_suivant;
```

```
69   }
70   return(int_mm_case);
71 }
```

8.9.2.3 void melanger_cartes (information * *cartes_a_melanger*[16])

mélange les cartes du bureau en effectuant une suite aléatoire d'échanges

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

cartes_a_melanger tableau des cartes à mélanger

Version :

1.0

Renvoie :

rien

Définition à la ligne 73 du fichier outils.c.

Référencé par init_bureau_krystel(), et init_bureau_nadege().

```
74 {
75   for (int i=0;i<1000;i++)
76   {
77     swap_carte(cartes_a_melanger, rand()%16, rand()%16);
78   }
79 }
```

8.9.2.4 void swap_carte (information * *cartes_a_melanger*[16], int *position_case1*, int *position_case2*)

échange de cartes dans un tableau d'un bureau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

cartes_a_melanger tableau des cartes à mélanger
position_case1 position de la première carte à échanger dans le tableau
position_case2 position de la seconde carte à échanger dans le tableau

Version :

1.0

Renvoie :

rien

Définition à la ligne 81 du fichier outils.c.

Référencé par melanger_cartes().

```
82 {
83   information* pi_tmp;
84
85   pi_tmp=cartes_a_melanger[position_case1];
86   cartes_a_melanger[position_case1]=cartes_a_melanger[position_case2];
87   cartes_a_melanger[position_case2]=pi_tmp;
88 }
```

8.9.2.5 void joueur_possede_tout (cases ** *plateau*, joueur * *joueur_encours*)

Définition à la ligne 90 du fichier outils.c.

```
91 {
92   int i;
93   bool retour;
94
95   joueur_encours->int_argent=9999999;
96
97   for(i=1;i<30;i++)
98   {
99     //tente d'acheter toutes les cases (la fonction traitement_achat se charge de verifier si elle sont achetable ou
100     retour=traitement_achat(plateau[i],joueur_encours);
101   }
102 }
```

8.9.2.6 possession* insertion_bonne_place_propriete (possession * *premier_maillon*, cases * *case_achetee*)

Définition à la ligne 104 du fichier outils.c.

Référencé par init_joueur_chargement(), et traitement_achat().

```
105 {
106   possession* nv_maillon;
107   possession* maillon_encours;
108   possession* maillon_precedent;
109
110   nv_maillon=creation_possession(case_achetee);
111
112   //si le premier maillon n'existe pas on retourne le nouveau maillon qui est alors la tete
113   if (premier_maillon==NULL)
114   {
115     return(nv_maillon);
116   }
117   else
118   {
119     //sinon on va chercher Ãă bien le positionner
120     //si c'est une case de propriete
121     if (nv_maillon->propriete->int_type==SALLE)
122     {
123         //le premier maillon a comparer est le premier
124
125         maillon_encours=premier_maillon;
126
127         //on avance dans la chaine jusqu'a trouver une case de la meme couleur ou Ãă arriver au bout
128         while ((maillon_encours->suivant != NULL) &&  (nv_maillon->propriete->case_salle.int_groupe != maillon_encou
129         {
130             maillon_precedent=maillon_encours;
131             maillon_encours=maillon_encours->suivant;
132         }
133         //on rÃľcupÃÍre ainsi le premier maillon ayant la meme couleur que celui qu'on veut insÃľrer
```

```
134         //on insÃÍre alors le nv_maillon juste aprÃÍs celui ainsi trouvÃľ
135         nv_maillon->suivant=maillon_encours->suivant;
136         maillon_encours->suivant=nv_maillon;
137         return(premier_maillon);
138     }
139     else
140     {
141       //sinon il peut s'agir soit d'un lieu commun soit du bds/bde
142       //on cherche alors de la meme maniÃÍre que pour les groupe pour les cases normales les cartes de mÃłme type
143 
144       //si c'est le bde ou le bds
145       if (nv_maillon->propriete->int_type==BDE)
146       {
147           //le maillon a comparer est le premier
148           maillon_encours=premier_maillon;
149 
150           while ((maillon_encours->suivant != NULL) &&  (maillon_encours->propriete->int_type!=BDS))
151           {
152               maillon_precedent=maillon_encours;
153               maillon_encours=maillon_encours->suivant;
154           }
155           //on rÃľcupÃÍre ainsi le bds ou la fin de chaine
156           //si on est en bout de chaine on l'insÃÍre au dÃľbut
157           if(maillon_encours->suivant==NULL)
158           {
159             nv_maillon->suivant=premier_maillon;
160             return(nv_maillon);
161           }
162           //sinon on l'insere aprÃÍs le bds
163           else
164           {
165           nv_maillon->suivant=maillon_encours->suivant;
166           maillon_encours->suivant=nv_maillon;
167           return(premier_maillon);
168           }
169       }
170       else if (nv_maillon->propriete->int_type==BDS)
171       {
172 
173           //le premier maillon a comparer est le premier
174           maillon_encours=premier_maillon;
175 
176           while ((maillon_encours->suivant != NULL) &&  (maillon_encours->propriete->int_type!=BDS))
177           {
178             maillon_precedent=maillon_encours;
179             maillon_encours=maillon_encours->suivant;
180           }
181           //on rÃľcupÃÍre ainsi le bds ou la fin de chaine
182           //si on est en bout de chaine on l'insÃÍre au dÃľbut
183           if(maillon_encours->suivant==NULL)
184           {
185             nv_maillon->suivant=premier_maillon;
186             return(nv_maillon);
187           }
188           //sinon on l'insere aprÃÍs le bds
189           else
190           {
191           nv_maillon->suivant=maillon_encours->suivant;
192           maillon_encours->suivant=nv_maillon;
193           return(premier_maillon);
194           }
195       }
196 
197       //sinon ce sont les lieux communs
198       else if ((nv_maillon->propriete->int_type >= LC_WC) && (nv_maillon->propriete->int_type<=LC_PARKING))
199       {
200           //le premier maillon a comparer est le second
```

```
201           maillon_encours=premier_maillon;
202
203           while ((maillon_encours->suivant != NULL) &&  ((maillon_encours->propriete->int_type < LC_WC) || (maillon_
204           {
205             maillon_precedent=maillon_encours;
206             maillon_encours=maillon_encours->suivant;
207           }
208           //on rÃľcupÃÎre ainsi le premier lieu commun ou la fin de chaine
209           //si on est en bout de chaine on l'insÃÎre au dÃľbut
210
211
212           if(maillon_encours->suivant==NULL)
213           {
214             nv_maillon->suivant=premier_maillon;
215             return(nv_maillon);
216           }
217           //sinon on l'insere aprÃÍs le premier lieu commun trouvÃľ
218           else
219           {
220             nv_maillon->suivant=maillon_encours->suivant;
221             maillon_encours->suivant=nv_maillon;
222             return(premier_maillon);
223           }
224       }
225     }
226   }
227   return(false);
228 }
```

8.9.2.7 bool possession_autres_cartes (cases ** *plateau*, joueur * *pj_joueur*)

verifie si le joueur possède les autres cartes de la même couleur que celle ou il se trouve

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau tableau contenant les cases du plateau

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

vrai s'il les possède sinon faux

Définition à la ligne 230 du fichier outils.c.

Référencé par action_possible().

```
231 {
232   int i;
233   int k;
234
235   //on dÃľfinit le groupe de la case oÃź se trouve le joueur
236   if (plateau[pj_joueur->int_position]->int_type == SALLE)
237   {
238     //on dÃľfinit le groupe de la case oÃź se trouve le joueur
239     //pour les groupe vert ou bleu ils faut trouver une seule autre carte du groupe
```

```
240     if ((plateau[pj_joueur->int_position]->case_salle.int_groupe ==VIOLET) || (plateau[pj_joueur->int_position]->cas
241     {
242       //on cherche deux cases avant et deux cases aprÃÍs (les cases de ces couleurs ne sont jamais Ãľloignãľs de plu
243       if (   (plateau[((pj_joueur->int_position - 2)+40)%40]->int_type == SALLE)
244         && (plateau[((pj_joueur->int_position - 2)+40)%40]->case_salle.int_groupe == plateau[pj_joueur->int_positio
245         && (plateau[((pj_joueur->int_position - 2)+40)%40]->case_salle.pjoueur_joueur == pj_joueur))
246       {
247         return (true);
248       }
249       else if (  (plateau[(pj_joueur->int_position + 2)%40]->int_type == SALLE)
250           && (plateau[(pj_joueur->int_position + 2)%40]->case_salle.int_groupe == plateau[pj_joueur->int_position]-
251           && (plateau[(pj_joueur->int_position + 2)%40]->case_salle.pjoueur_joueur == pj_joueur))
252       {
253         return(true);
254       }
255     }
256     //pour les autres il faut trouver forcement trouver deux autres carte du mm groupe
257     else
258     {
259       k=0;
260       //on cherche trois cases avant et trois cases aprÃÍs (les cases de ces couleurs ne sont jamais Ãľloignãľs de p
261       for (i =((pj_joueur->int_position - 3)+40)%40; i < (pj_joueur->int_position + 3)%40 ; i++)
262       {
263         //on ne recompte la case ou est actuellement le joueur
264         if (i != pj_joueur->int_position)
265         {
266           if (   (plateau[i]->int_type == SALLE)
267             && (plateau[i]->case_salle.int_groupe == plateau[pj_joueur->int_position]->case_salle.int_groupe)
268             && (plateau[i]->case_salle.pjoueur_joueur == pj_joueur))
269           {
270             k++;
271           }
272
273         }
274
275       }
276       //on verifie si on a trouver deux cases
277       if (k==2)
278       {
279         return(true);
280       }
281     }
282   }
283   return(false);
284 }
```

8.9.2.8 int decoupage_string (char *** *str_tab*, char * *str_a_diviser*)

divise une chaine en caractère en tableau de chaîne caractère

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

str_tab tableau de chaine de caractère pour le retour de la chaine une fois divisé
str_a_diviser la chaine de caractère à diviser

Version :

1.0

Renvoie :

le nombre de lignes

Définition à la ligne 286 du fichier outils.c.

Référencé par attente_validation_message().

```
287 {
288   int int_ligne_max;
289
290   int i;
291   int k; //curseur indiquant la ligne en cours de crÃľation
292   int j; //curseur de dÃľplacement dans str_a_diviser
293
294   i=0;
295   k=0;
296   j=0;
297
298
299   int_ligne_max=((int)(strlen(str_a_diviser)/MESSAGE_MAX_CARACTERE)+1)*2;
300
301   //on alloue en mÃľmoire un tableau de chaine de caractÃÎre de int_ligne_max lignes contenant chacune des chaine de
302   str_tab[0]=new char*[int_ligne_max];
303
304   for (i = 0 ; i < int_ligne_max ; i++)
305   {
306     str_tab[0][i]=new char[MESSAGE_MAX_CARACTERE];
307   }
308
309   //tant qu'on est pas au bout de la chaine principale
310   while (strlen(str_a_diviser) >= MESSAGE_MAX_CARACTERE)
311   {
312     //on met le curseur Ãă 63
313     j=MESSAGE_MAX_CARACTERE - 1;
314
315     //puis on cherche avant, le premier espace
316     while (str_a_diviser[j] != ' ')
317     {
318       j--;
319     }
320
321     //on copie les j premiers caractÃÎres
322     strncpy(str_tab[0][k],str_a_diviser,j);
323     //on marque la fin de chaine
324     str_tab[0][k][j]='\0';
325
326     //on deplace le pointeur
327     str_a_diviser=str_a_diviser+j;
328     //on passe a la ligne Ãă crÃľer suivante
329     k++;
330   }
331   //aprÃÍs il ne nous reste plus que le morceau de chaine d'une taille infÃľrieur Ãă MESSAGE_MAX_CARACTERE
332   strcpy(str_tab[0][k],str_a_diviser);
333
334   return(k+1);
335 }
```

8.9.2.9 int action_possible (cases ** *plateau*, int *int_position_cases_etudiee*, joueur * *pj_joueur*, int *int_etat*, int *int_nb_tire*)

analyse toutes les actions possibles pour un joueur pour une case donnée

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau tableau contenant les cases du plateau

int_position_cases_etudiee cases à partir de laquelle on doit faire le traitement

pj_joueur joueur qui joue actuellement

int_etat indique si le joueur demande a modifier le niveau d'une salle ou s'il demande une hypothèque ou s'il est seulement arriver sur une salle

int_nb_tire nombre indiqué par les dés

Version :

1.0

Renvoie :

ACTION_ACHAT s'il peut acheter, ACTION_HUNHYPOTHEQUE s'il peut déshypotéquer, ACTION_PLUS s'il peut augmenter le niveau de la salle, ACTION_MOINS s'il peut diminuer le niveau de la salle, ACTION_PLUS_MOINS s'il peut augmenter ou diminuer le niveau de la salle, ACTION_FINIR s'il ne peut que finir le tour, ACTION_PAYER s'il doit payer le loyer

Définition à la ligne 337 du fichier outils.c.

Référencé par attente_clic(), et traitement_arrive_case().

```
338 {
339   int i;
340   //on vÃĽrifie s'il s'agit l'arrivÃĽ sur une case
341   if (int_etat == ARRIVE_CASE)
342   {
343   //on dÃĽtermine le type de la case ou il se trouve
344
345     //s'il se trouve sur une salle
346     if (plateau[pj_joueur->int_position]->int_type==SALLE)
347     {
348       //on vÃĽrifie si elle n'appartient Ãă personne et que le joueur Ãă l'argent et qu'elle n'est pas au premier to
349       if (!(pj_joueur->bool_debut) && (plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur==NULL) && (pj_jou
350       {
351         //il peut alors l'acheter ou finir ce tour ou abandonner
352         return(ACTION_ACHAT);
353       }
354       //sinon si elle appartient Ãă un adversaire
355       else if (plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur != pj_joueur)
356       {
357         //si il a assez d'argent
358         if (pj_joueur->int_argent >= (plateau[pj_joueur->int_position]->case_salle.int_prix/100)*plateau[pj_joueur-
359         {
360           //il peut alors payer
361           return(ACTION_PAYER);
362         }
363         else
364         {
365           return(ACTION_PERDRE);
366         }
367       }
368       //sinon si la case lui appartient
369       else if (plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur == pj_joueur)
370       {
371         //s'il possÃĺde toutes les autres cartes de mÃŞme couleur
372         if (possession_autres_cartes(plateau,pj_joueur))
373         {
374           //si son niveau est au maximum il ne peut que diminuer
375           if (plateau[pj_joueur->int_position]->case_salle.int_niveau == 5)
376           {
```

```
377             return(ACTION_MOINS);
378           }
379
380           //si son niveau est nul il ne peut qu'augmenter s'il a de l'argent
381           else if ((plateau[pj_joueur->int_position]->case_salle.int_niveau == 0) && (pj_joueur->int_argent >= plate
382           {
383             return(ACTION_PLUS);
384           }
385
386           //dans aucun des deux cas prÃ¯cedent s'il a assez d'argent il peut faire les deux
387           else if (pj_joueur->int_argent >= plateau[pj_joueur->int_position]->case_salle.int_prix_niveau)
388           {
389             return(ACTION_PLUS_MOINS);
390           }
391           else
392           {
393             return(ACTION_FINIR);
394           }
395
396         }
397         else
398         {
399           //il ne peut rien faire si ce n'est finir le tour
400           return(ACTION_FINIR);
401         }
402       }
403     }
404     //s'il se trouve sur une case association ou sur un lieu commun
405     else if (  (plateau[pj_joueur->int_position]->int_type == LC_PARKING)
406         || (plateau[pj_joueur->int_position]->int_type == LC_RU)
407         || (plateau[pj_joueur->int_position]->int_type == LC_ASCENSEUR)
408         || (plateau[pj_joueur->int_position]->int_type == LC_WC))
409     {
410       //on vÃ¯rifie si elle n'appartient Ãã personne et qu'il Ãã assez d'argent
411       if (!(pj_joueur->bool_debut) && (plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur==NULL) &&
412       {
413         //il peut alors l'acheter ou finir ce tour
414         return(ACTION_ACHAT);
415       }
416       //sinon si elle appartient Ãã un adversaire
417       else if ((plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur != pj_joueur) && (plateau[pj_joueu
418       {
419         //si il a assez d'argent
420         if (pj_joueur->int_argent >= traitement_loyer_lieu_commun(plateau[pj_joueur->int_position]->case_lieu_commun
421         {
422           //il peut alors payer
423           return(ACTION_PAYER);
424         }
425         else
426         {
427           //sinon il perd
428           return(ACTION_PERDRE);
429         }
430       }
431       //sinon si la case lui appartient
432       else if (plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur == pj_joueur)
433       {
434         //il le peut que finir ce tour
435         return(ACTION_FINIR);
436       }
437     }
438     else if (  (plateau[pj_joueur->int_position]->int_type == BDE)
439         || (plateau[pj_joueur->int_position]->int_type == BDS))
440     {
441       //on vÃ¯rifie si elle n'appartient Ãã personne et qu'il Ãã assez d'argent
442       if (!(pj_joueur->bool_debut) && (plateau[pj_joueur->int_position]->case_association.pjoueur_joueur==NULL) &&
443       {
```

```
444         //il peut alors l'acheter ou finir ce tour
445         return(ACTION_ACHAT);
446       }
447       else if ((plateau[pj_joueur->int_position]->case_association.pjoueur_joueur != pj_joueur) && (plateau[pj_joue
448       {
449         //si il a assez d'argent
450         if (pj_joueur->int_argent >= traitement_loyer_association(plateau[pj_joueur->int_position]->case_lieu_commu
451         {
452           //il peut alors payer
453           return(ACTION_PAYER);
454         }
455         else
456         {
457           //sinon il perd
458           return(ACTION_PERDRE);
459         }
460       }
461       //sinon si la case lui appartient
462       else if (plateau[pj_joueur->int_position]->case_association.pjoueur_joueur == pj_joueur)
463       {
464         //il le peut que finir ce tour
465         return(ACTION_FINIR);
466       }
467     }
468   }
469   else if (int_etat == CLICK_CASE)
470   {
471     i=0;
472     while (plateau[i]!=int_to_cases(int_position_cases_etudiee,pj_joueur)) i++;
473     if (plateau[i]->int_type == SALLE)
474     {
475       //si le joueur est sur une case du mÃłme groupe
476       if (plateau[pj_joueur->int_position]->case_salle.int_groupe == plateau[i]->case_salle.int_groupe)
477       {
478         //si sont niveau est nul il ne peut qu'augmenter
479         if (plateau[i]->case_salle.int_niveau == 0)
480         {
481           return(ACTION_PLUS);
482         }
483         //si son niveau est au maximum il ne peut que diminuer
484         else if (plateau[i]->case_salle.int_niveau == 5)
485         {
486           return(ACTION_MOINS);
487         }
488         else
489         {
490           return(ACTION_PLUS_MOINS);
491         }
492       }
493     }
494     else
495     {
496       //il ne peut que suivre les indications spÃľcifique Ãă chaque case ou chaque carte
497       return(ACTION_FINIR);
498     }
499   }
500
501   return(ERREUR);
502 }
```

8.9.2.10 int nombre_propriete (joueur * *pj_joueur*)

compte le nombre de propriétés du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

le nombre de propriété du joueur

Définition à la ligne 504 du fichier outils.c.

Référencé par affich_panneau_possessions(), et attente_clic().

```
505 {
506   possession* propriete_temp;
507   int i;
508 
509   i=0;
510   propriete_temp=pj_joueur->propriete;
511   while(propriete_temp!=NULL)
512   {
513     i++;
514     propriete_temp=propriete_temp->suivant;
515   }
516 
517   return(i);
518 
519 }
```

8.9.2.11 cases * int_to_cases (int *int_indice_case*, joueur * *pj_joueur*)

à partir de l'indice de chaine des propriété elle trouve la case correspondante

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

int_indice_case indice de la case recherché fourni a partir de la liste de propriété du panneau gauche

Version :

1.0

Renvoie :

la case voulue

Définition à la ligne 521 du fichier outils.c.

Référencé par action_possible(), et attente_clic().

```
522 {
523   possession* propriete_en_cours;
524   int i;
525 
526   //on assigne la propriÃŀtÃŀ encours avec la premiÃíre propriÃŀtÃŀ du joueur pour pouvoir parcourir la chaÃõne
527   propriete_en_cours=pj_joueur->propriete;
528 
529   //on se dÃŀplace dans la chaÃõne jusqu'Ãă l'indice voulue
530   for (i = 0; i < int_indice_case; i++)
531   {
532     propriete_en_cours=propriete_en_cours->suivant;
533   }
534   //on retourne ainsi la propriete trouvÃŀe
535   return(propriete_en_cours->propriete);
536 }
```

8.9.2.12 bool verification_victoire (joueur * *pj_joueur*)

compte le nombre de propriétés du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

true si le joueur à gagnersinon false

Définition à la ligne 538 du fichier outils.c.

Référencé par jeu().

```
539 {
540   joueur* joueur_encours;
541   int i;
542 
543   i=0;
544 
545   //le joueur gagne s'il est seul
546   joueur_encours=pj_joueur->pjoueur_suivant;
547 
548   while (joueur_encours != pj_joueur)
549   {
550     i++;
551     joueur_encours=joueur_encours->pjoueur_suivant;
552   }
553 
554   if (i > 0)
555   {
556     return(false);
557   }
558   else
559   {
560     return(true);
561   }
562 }
```

8.9.2.13 void action_to_boutons (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_fond*, int *int_action*, int *int_etat*)

définit les boutons à afficher en fonction des actions possibles sur une propriété

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_fond surface de la propriété

int_action actions possibles

int_etat état des boutons

Version :

1.0

Renvoie :

nothing

Définition à la ligne 564 du fichier outils.c.

Référencé par attente_validation_propriete().

```
565 {
566
567   //Selon l'action Ãã faire
568   switch(int_action)
569   {
570     //Dans la cas d'un achat
571     case ACTION_ACHAT:
572       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_ACHAT, BTN_FINIR);
573       break;
574     //Dans le cas d'une deshypothÃÍque
575     case  ACTION_UNHYPOTHEQUE:
576       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_UNHYPOTHEQUE, BTN_FINIR);
577       break;
578     //Dans le cas d'une hypothÃÍque
579     case  ACTION_HYPOTHEQUE:
580       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_HYPOTHEQUE, BTN_FINIR);
581       break;
582     //Pour augmenter le niveau d'une salle
583     case  ACTION_PLUS:
584       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_PLUS, BTN_FINIR);
585       break;
586     //Pour diminuer le niveau d'une salle
587     case  ACTION_MOINS:
588       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_MOINS, BTN_FINIR);
589       break;
590     //Pour augmenter ou diminuer le niveau d'une salle
591     case  ACTION_PLUS_MOINS:
592       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 3, BTN_PLUS, BTN_MOINS, BTN_FINIR);
593       break;
594     //Pour finir l'action
595     case  ACTION_FINIR:
596       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 1, BTN_FINIR);
597       break;
598     //Dans tout les autres cas
599     default:
600       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 1, BTN_FINIR);
```

```
601       break;
602   }
603
604 }
```

8.9.2.14 void sauvegarde (cases ** *plateau*, joueur * *pj_joueur_tete*)

sauvegarde de la partie

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau plateau du jeu

pj_joueur_tete joueur en cours

Version :

1.0

Renvoie :

nothing

Définition à la ligne 606 du fichier outils.c.

Référencé par attente_clic(), et jeu().

```
607 {
608   //Joueur temporaire
609   joueur* pj_joueur;
610
611   //PropriÃŀtÃŀ temporaire
612   possession* possession_temp;
613
614   //Nombre de joueur
615   int int_nombre_joueur;
616   int_nombre_joueur=1;
617
618   //Nominateur de fin de joueur
619   int int_fin;
620   int_fin=-1;
621
622   //Compteur de propriÃŀtÃŀ
623   int i;
624
625   //On commence au premier joueur
626   pj_joueur=pj_joueur_tete;
627
628   //On compte le nombre de joueur
629   while((pj_joueur=pj_joueur->pjoueur_suivant)!=pj_joueur_tete) int_nombre_joueur++;
630
631   //On commence au premier joueur
632   pj_joueur=pj_joueur_tete;
633
634   ofstream file("sauvegarde_joueur.txt");
635
636   file << int_nombre_joueur<<endl;
637
638   //Tant qu'on est pas revenu au joueur de tete
639   while(pj_joueur->pjoueur_suivant!=pj_joueur_tete)
```

```
640   {
641     file << pj_joueur->str_nom << endl;
642     file << pj_joueur->int_position << endl;
643     file << pj_joueur->int_argent << endl;
644     file << pj_joueur->int_certificat << endl;
645     file << pj_joueur->bool_debut << endl;
646     file << pj_joueur->int_double_tire << endl;
647     file << pj_joueur->bool_laurence << endl;
648     file << pj_joueur->int_laurence << endl;
649 
650     possession_temp=pj_joueur->propriete;
651     while(possession_temp!=NULL)
652     {
653       i=0;
654       while (plateau[i]!=possession_temp->propriete) i++;
655       file << i << endl;
656       possession_temp=possession_temp->suivant;
657     }
658 
659     file << int_fin << endl << endl;
660     //On passe au joueur suivant
661     pj_joueur=pj_joueur->pjoueur_suivant;
662   }
663 
664     //Sauvegarde du dernier joueur
665     file << pj_joueur->str_nom << endl;
666     file << pj_joueur->int_position << endl;
667     file << pj_joueur->int_argent << endl;
668     file << pj_joueur->int_certificat << endl;
669     file << pj_joueur->bool_debut << endl;
670     file << pj_joueur->int_double_tire << endl;
671     file << pj_joueur->bool_laurence << endl;
672     file << pj_joueur->int_laurence << endl;
673 
674     possession_temp=pj_joueur->propriete;
675     while(possession_temp!=NULL)
676     {
677       i=0;
678       while (plateau[i]!=possession_temp->propriete) i++;
679       file << i << endl;
680       possession_temp=possession_temp->suivant;
681     }
682 
683     file << int_fin << endl << endl;
684     //On passe au joueur suivant
685     pj_joueur=pj_joueur->pjoueur_suivant;
686 
687   //Sauvegarde du plateau
688   ofstream file2("sauvegarde_plateau.txt");
689 
690   //On Ã¯crit le total de la cagnotte
691   file2 << plateau[20]->machine_a_cafe.int_argent << endl;
692   //Pour chacune des cases
693   for(i=0;i<40;i++)
694   {
695     //S'il sagit d'une salle, on Ã¯crit son niveau
696     if(plateau[i]->int_type==SALLE) file2 << plateau[i]->case_salle.int_niveau << endl;
697     //Sinon on met -1
698     else file2 << int_fin << endl;
699   }
700 
701 
702 }
```

8.9.2.15 joueur * chargement (joueur * *pj_joueur*, cases ** *plateau*, rvb_couleur *couleurs*[8], SDL_Surface * *surf_ecran*)

chargement de la partie

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau plateau du jeu

pj_joueur joueur en cours

couleurs tableau de couleur

surf_ecran surface de l'écran

Version :

1.0

Renvoie :

un joueur de la liste de joueur

Définition à la ligne 704 du fichier outils.c.

Référencé par main().

```
705 {
706
707   ifstream file1("sauvegarde_joueur.txt", ios::in);
708   if (!file1)
709   {
710     cerr << "Allocation inaboutie de sauvegarde_joueur.txt" << endl;
711   }
712
713   ifstream file2("sauvegarde_plateau.txt",ios::in);
714   if (!file2)
715   {
716     cerr << "Allocation inaboutie de sauvegarde_plateau.txt" << endl;
717   }
718
719
720   int nb_joueurs;
721
722   file1 >> nb_joueurs;
723
724   pj_joueur=init_anneau_joueur_chargement(nb_joueurs, &file1,plateau);
725   plateau=init_plateau_chargement(&file2, plateau, couleurs,surf_ecran);
726
727   file1.close();
728   file2.close();
729
730   return(pj_joueur);
731
732 }
```

8.9.2.16 joueur *bool is_readable (const std : :string & *file*)

test si un fichier existe

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

file le fichier à tester

Version :

1.0

Renvoie :

si le fichier existe ou non

Définition à la ligne 734 du fichier outils.c.

Référencé par attente_action_accueil().

```
735 {
736     std::ifstream fichier( file.c_str() );
737     return !fichier.fail();
738 }
```

8.10 Référence du fichier outils.h

en tete des outils

Fonctions

- void **joueur_possede_tout** (**cases** **plateau, **joueur** *joueur_encours)
- char * **ajout_caractere** (char *texte, int int_caractere)
  ajoute un caractere (en code UTF-8) en fin de chaine

- int **nombre_joueur_case** (**joueur** *pj_joueur)
  calcule le nombre de joueur sur une case

- void **swap_carte** (**information** *cartes_a_melanger[16], int position_case1, int position_case2)
  échange de cartes dans un tableau d'un bureau

- void **melanger_cartes** (**information** *cartes_a_melanger[16])
  mélange les cartes du bureau en effectuant une suite aléatoire d'échanges

- **possession** * **insertion_bonne_place_propriete** (**possession** *premier_maillon, **cases** *case_achetee)
- bool **possession_autres_cartes** (**cases** **plateau, **joueur** *pj_joueur)
  verifie si le joueur possède les autres cartes de la même couleur que celle ou il se trouve

- int **action_possible** (**cases** **plateau, int int_position_cases_etudiee, **joueur** *pj_joueur, int int_etat, int int_nb_tire)
  analyse toutes les actions possibles pour un joueur pour une case donnée

- int **decoupage_string** (char ***str_tab, char *str_a_diviser)
  divise une chaine en caractère en tableau de chaîne caractère

- **cases** * **int_to_cases** (int int_indice_case, **joueur** *pj_joueur)
  à partir de l'indice de chaine des propriété elle trouve la case correspondante

- int **nombre_propriete** (**joueur** *pj_joueur)
  compte le nombre de propriétés du joueur

- bool **verification_victoire** (**joueur** *pj_joueur)
  compte le nombre de propriétés du joueur

- void **action_to_boutons** (SDL_Surface *surf_ecran, SDL_Surface *surf_fond, int int_action, int int_etat)
  définit les boutons à afficher en fonction des actions possibles sur une propriété

- void **sauvegarde** (**cases** **plateau, **joueur** *pj_joueur_tete)
  sauvegarde de la partie

- **joueur** * **chargement** (**joueur** *pj_joueur, **cases** **plateau, **rvb_couleur** couleurs[8], SDL_Surface *surf_ecran)
  chargement de la partie

- bool **is_readable** (const std : :string &file)
  test si un fichier existe

8.10.1 Description détaillée

en tete des outils

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **outils.h**.

8.10.2 Documentation des fonctions

8.10.2.1 void joueur_possede_tout (cases ** *plateau*, joueur * *joueur_encours*)

Définition à la ligne 90 du fichier outils.c.

```
91 {
92   int i;
93   bool retour;
94
95   joueur_encours->int_argent=9999999;
96
97   for(i=1;i<30;i++)
98   {
99     //tente d'acheter toutes les cases (la fonction traitement_achat se charge de verifier si elle sont achetable ou
100     retour=traitement_achat(plateau[i],joueur_encours);
101   }
102 }
```

8.10.2.2 char* ajout_caractere (char * *texte*, int *int_caractere*)

ajoute un caractere (en code UTF-8) en fin de chaine

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

texte texte initial

int_caractere entier correspondant à une lettre

Version :

1.0

Renvoie :

la chaine initiale avec le caractère à la fin

Définition à la ligne 11 du fichier outils.c.

Référencé par attente_action_config().

```
12 {
13   char c_caractere[2];
14   char texte_temp[16];
15   c_caractere[0]=(char)int_caractere;
16   c_caractere[1]='\0';
17
18   strcpy(texte_temp,texte);
19
20
21   //on vÃ¯rifie si la chaine est non vide
22   if ((texte_temp!=NULL)) //&& (texte_temp[0]!='\0'))
23   {
24     //si le caractÃÍre rentrÃ¯ est le backspace
25     if(int_caractere==8)
26     {
27       //la derniÃÍre case pleine de la chaine prend la valeur \0
28       texte_temp[strlen(texte_temp)-1]='\0';
29     }
30     //s'il ne s'agit pas du backspace
31     else
32     {
33       //la dernier case de la chaine de caractÃÍre prend pour valeur le caractÃÍre de la table ascii coorespondant
34       //si la chaine n'est pas pleine et si le caractÃÍre est admissible
35       if ((strlen(texte_temp)<14)
36         && (
37         ((int_caractere>=97) && (int_caractere<=122)) //lettres minuscules
38         || ((int_caractere>=30) && (int_caractere<=39)) //chiffres
39         || (int_caractere==95) //underscore
40         || (int_caractere==45) //tirÃ¯
41         || (int_caractere==32) // espace
42         || (int_caractere==46))) //point
43       {
44       strcpy(texte_temp,strcat(texte_temp,c_caractere));
45       }
46     }
47   }
48   strcpy(texte,texte_temp);
49   return(texte);
50 }
```

8.10.2.3 int nombre_joueur_case (joueur * *pj_joueur*)

calcule le nombre de joueur sur une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur le joueur actuellement en train de jouer son tour

Version :

1.0

Renvoie :

le nombre de joueur sur la case

Définition à la ligne 52 du fichier outils.c.

Référencé par aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), et reculer_jeton().

```
53 {
54   int int_mm_case;
55
56   int_mm_case=0;
57
58   joueur* pj_joueur_encours;
59
60   pj_joueur_encours=pj_joueur->pjoueur_suivant;
61
62   while (pj_joueur_encours != pj_joueur)
63   {
64     if (pj_joueur_encours->int_position==pj_joueur->int_position)
65     {
66       int_mm_case++;
67     }
68     pj_joueur_encours=pj_joueur_encours->pjoueur_suivant;
69   }
70   return(int_mm_case);
71 }
```

8.10.2.4 void swap_carte (information * *cartes_a_melanger*[16], int *position_case1*, int *position_case2*)

échange de cartes dans un tableau d'un bureau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

cartes_a_melanger tableau des cartes à mélanger

position_case1 position de la première carte à échanger dans le tableau

position_case2 position de la seconde carte à échanger dans le tableau

Version :

1.0

Renvoie :

rien

Définition à la ligne 81 du fichier outils.c.

Référencé par melanger_cartes().

```
82 {
83   information* pi_tmp;
84
85   pi_tmp=cartes_a_melanger[position_case1];
86   cartes_a_melanger[position_case1]=cartes_a_melanger[position_case2];
87   cartes_a_melanger[position_case2]=pi_tmp;
88 }
```

8.10.2.5 void melanger_cartes (information * *cartes_a_melanger*[16])

mélange les cartes du bureau en effectuant une suite aléatoire d'échanges

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

cartes_a_melanger tableau des cartes à mélanger

Version :

1.0

Renvoie :

rien

Définition à la ligne 73 du fichier outils.c.

Référencé par init_bureau_krystel(), et init_bureau_nadege().

```
74 {
75   for (int i=0;i<1000;i++)
76   {
77     swap_carte(cartes_a_melanger, rand()%16, rand()%16);
78   }
79 }
```

8.10.2.6 possession* insertion_bonne_place_propriete (possession * *premier_maillon*, cases * *case_achetee*)

Définition à la ligne 104 du fichier outils.c.

Référencé par init_joueur_chargement(), et traitement_achat().

```
105 {
106   possession* nv_maillon;
107   possession* maillon_encours;
108   possession* maillon_precedent;
109
110   nv_maillon=creation_possession(case_achetee);
111
112   //si le premier maillon n'existe pas on retourne le nouveau maillon qui est alors la tete
113   if (premier_maillon==NULL)
114   {
115     return(nv_maillon);
116   }
117   else
118   {
119     //sinon on va chercher Ãă bien le positionner
120     //si c'est une case de propriete
121     if (nv_maillon->propriete->int_type==SALLE)
122     {
123         //le premier maillon a comparer est le premier
124
125         maillon_encours=premier_maillon;
126
127         //on avance dans la chaine jusqu'a trouver une case de la meme couleur ou Ãă arriver au bout
128         while ((maillon_encours->suivant != NULL) &&  (nv_maillon->propriete->case_salle.int_groupe != maillon_encou
129         {
130             maillon_precedent=maillon_encours;
131             maillon_encours=maillon_encours->suivant;
132         }
133         //on rÃľcupÃÎre ainsi le premier maillon ayant la meme couleur que celui qu'on veut insÃľrer
134         //on insÃÎre alors le nv_maillon juste aprÃÍs celui ainsi trouvÃľ
```

```
135         nv_maillon->suivant=maillon_encours->suivant;
136         maillon_encours->suivant=nv_maillon;
137         return(premier_maillon);
138     }
139     else
140     {
141       //sinon il peut s'agir soit d'un lieu commun soit du bds/bde
142       //on cherche alors de la meme maniÃÍre que pour les groupe pour les cases normales les cartes de mÃŧme type
143 
144       //si c'est le bde ou le bds
145       if (nv_maillon->propriete->int_type==BDE)
146       {
147           //le maillon a comparer est le premier
148           maillon_encours=premier_maillon;
149 
150           while ((maillon_encours->suivant != NULL) &&  (maillon_encours->propriete->int_type!=BDS))
151           {
152               maillon_precedent=maillon_encours;
153               maillon_encours=maillon_encours->suivant;
154           }
155           //on rÃŧcupÃÍre ainsi le bds ou la fin de chaine
156           //si on est en bout de chaine on l'insÃÍre au dÃŧbut
157           if(maillon_encours->suivant==NULL)
158           {
159             nv_maillon->suivant=premier_maillon;
160             return(nv_maillon);
161           }
162           //sinon on l'insere aprÃÍs le bds
163           else
164           {
165           nv_maillon->suivant=maillon_encours->suivant;
166           maillon_encours->suivant=nv_maillon;
167           return(premier_maillon);
168           }
169       }
170       else if (nv_maillon->propriete->int_type==BDS)
171       {
172 
173           //le premier maillon a comparer est le premier
174           maillon_encours=premier_maillon;
175 
176           while ((maillon_encours->suivant != NULL) &&  (maillon_encours->propriete->int_type!=BDS))
177           {
178             maillon_precedent=maillon_encours;
179             maillon_encours=maillon_encours->suivant;
180           }
181           //on rÃŧcupÃÍre ainsi le bds ou la fin de chaine
182           //si on est en bout de chaine on l'insÃÍre au dÃŧbut
183           if(maillon_encours->suivant==NULL)
184           {
185             nv_maillon->suivant=premier_maillon;
186             return(nv_maillon);
187           }
188           //sinon on l'insere aprÃÍs le bds
189           else
190           {
191           nv_maillon->suivant=maillon_encours->suivant;
192           maillon_encours->suivant=nv_maillon;
193           return(premier_maillon);
194           }
195       }
196 
197       //sinon ce sont les lieux communs
198       else if ((nv_maillon->propriete->int_type >= LC_WC) && (nv_maillon->propriete->int_type<=LC_PARKING))
199       {
200           //le premier maillon a comparer est le second
201           maillon_encours=premier_maillon;
```

```
202
203           while ((maillon_encours->suivant != NULL) &&  ((maillon_encours->propriete->int_type < LC_WC) || (maillon_
204           {
205             maillon_precedent=maillon_encours;
206             maillon_encours=maillon_encours->suivant;
207           }
208           //on rÃľcupÃíre ainsi le premier lieu commun ou la fin de chaine
209           //si on est en bout de chaine on l'insÃíre au dÃľbut
210
211
212           if(maillon_encours->suivant==NULL)
213           {
214             nv_maillon->suivant=premier_maillon;
215             return(nv_maillon);
216           }
217           //sinon on l'insere aprÃís le premier lieu commun trouvÃľ
218           else
219           {
220             nv_maillon->suivant=maillon_encours->suivant;
221             maillon_encours->suivant=nv_maillon;
222             return(premier_maillon);
223           }
224       }
225     }
226   }
227   return(false);
228 }
```

8.10.2.7 bool possession_autres_cartes (cases ** *plateau*, joueur * *pj_joueur*)

verifie si le joueur possède les autres cartes de la même couleur que celle ou il se trouve

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau tableau contenant les cases du plateau

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

vrai s'il les possède sinon faux

Définition à la ligne 230 du fichier outils.c.

Référencé par action_possible().

```
231 {
232   int i;
233   int k;
234
235   //on dÃľfinit le groupe de la case oÃź se trouve le joueur
236   if (plateau[pj_joueur->int_position]->int_type == SALLE)
237   {
238     //on dÃľfinit le groupe de la case oÃź se trouve le joueur
239     //pour les groupe vert ou bleu ils faut trouver une seule autre carte du groupe
240     if ((plateau[pj_joueur->int_position]->case_salle.int_groupe ==VIOLET) || (plateau[pj_joueur->int_position]->cas
```

```
241     {
242       //on cherche deux cases avant et deux cases aprÃÍs (les cases de ces couleurs ne sont jamais Ã'loignÃ's de plu
243       if (   (plateau[((pj_joueur->int_position - 2)+40)%40]->int_type == SALLE)
244         && (plateau[((pj_joueur->int_position - 2)+40)%40]->case_salle.int_groupe == plateau[pj_joueur->int_positior
245         && (plateau[((pj_joueur->int_position - 2)+40)%40]->case_salle.pjoueur_joueur == pj_joueur))
246       {
247         return (true);
248       }
249       else if (  (plateau[(pj_joueur->int_position + 2)%40]->int_type == SALLE)
250           && (plateau[(pj_joueur->int_position + 2)%40]->case_salle.int_groupe == plateau[pj_joueur->int_position]->
251           && (plateau[(pj_joueur->int_position + 2)%40]->case_salle.pjoueur_joueur == pj_joueur))
252       {
253         return(true);
254       }
255     }
256     //pour les autres il faut trouver forcement trouver deux autres carte du mm groupe
257     else
258     {
259       k=0;
260       //on cherche trois cases avant et trois cases aprÃÍs (les cases de ces couleurs ne sont jamais Ã'loignÃ's de p
261       for (i =((pj_joueur->int_position - 3)+40)%40; i < (pj_joueur->int_position + 3)%40 ; i++)
262       {
263         //on ne recompte la case ou est actuellement le joueur
264         if (i != pj_joueur->int_position)
265         {
266           if (   (plateau[i]->int_type == SALLE)
267             && (plateau[i]->case_salle.int_groupe == plateau[pj_joueur->int_position]->case_salle.int_groupe)
268             && (plateau[i]->case_salle.pjoueur_joueur == pj_joueur))
269           {
270             k++;
271           }
272
273         }
274
275       }
276       //on verifie si on a trouver deux cases
277       if (k==2)
278       {
279         return(true);
280       }
281     }
282   }
283   return(false);
284 }
```

8.10.2.8 int action_possible (cases ** *plateau*, int *int_position_cases_etudiee*, joueur * *pj_joueur*, int *int_etat*, int *int_nb_tire*)

analyse toutes les actions possibles pour un joueur pour une case donnée

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau tableau contenant les cases du plateau

int_position_cases_etudiee cases à partir de laquelle on doit faire le traitement

pj_joueur joueur qui joue actuellement

int_etat indique si le joueur demande a modifier le niveau d'une salle ou s'il demande une hypothèque ou s'il est seulement arriver sur une salle

int_nb_tire nombre indiqué par les dés

Version :

1.0

Renvoie :

ACTION_ACHAT s'il peut acheter, ACTION_HUNHYPOTHEQUE s'il peut déshypotéquer, ACTION_PLUS s'il peut augmenter le niveau de la salle, ACTION_MOINS s'il peut diminuer le niveau de la salle, ACTION_PLUS_MOINS s'il peut augmenter ou diminuer le niveau de la salle, ACTION_FINIR s'il ne peut que finir le tour, ACTION_PAYER s'il doit payer le loyer

Définition à la ligne 337 du fichier outils.c.

Référencé par attente_clic(), et traitement_arrive_case().

```
338 {
339   int i;
340   //on vÃĽrifie s'il s'agit l'arrivÃĽ sur une case
341   if (int_etat == ARRIVE_CASE)
342   {
343   //on dÃĽtermine le type de la case ou il se trouve
344
345     //s'il se trouve sur une salle
346     if (plateau[pj_joueur->int_position]->int_type==SALLE)
347     {
348       //on vÃĽrifie si elle n'appartient Ãă personne et que le joueur Ãă l'argent et qu'elle n'est pas au premier to
349       if (!(pj_joueur->bool_debut) && (plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur==NULL) && (pj_jou
350       {
351         //il peut alors l'acheter ou finir ce tour ou abandonner
352         return(ACTION_ACHAT);
353       }
354       //sinon si elle appartient Ãă un adversaire
355       else if (plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur != pj_joueur)
356       {
357         //si il a assez d'argent
358         if (pj_joueur->int_argent >= (plateau[pj_joueur->int_position]->case_salle.int_prix/100)*plateau[pj_joueur-
359         {
360           //il peut alors payer
361           return(ACTION_PAYER);
362         }
363         else
364         {
365           return(ACTION_PERDRE);
366         }
367       }
368       //sinon si la case lui appartient
369       else if (plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur == pj_joueur)
370       {
371         //s'il possÃÍde toutes les autres cartes de mÃŽme couleur
372         if (possession_autres_cartes(plateau,pj_joueur))
373         {
374           //si son niveau est au maximum il ne peut que diminuer
375           if (plateau[pj_joueur->int_position]->case_salle.int_niveau == 5)
376           {
377             return(ACTION_MOINS);
378           }
379
380           //si son niveau est nul il ne peut qu'augmenter s'il a de l'argent
381           else if ((plateau[pj_joueur->int_position]->case_salle.int_niveau == 0) && (pj_joueur->int_argent >= plate
382           {
383             return(ACTION_PLUS);
384           }
385
386           //dans aucun des deux cas prÃĽcedent s'il a assez d'argent il peut faire les deux
387           else if (pj_joueur->int_argent >= plateau[pj_joueur->int_position]->case_salle.int_prix_niveau)
388           {
```

```
389             return(ACTION_PLUS_MOINS);
390           }
391           else
392           {
393             return(ACTION_FINIR);
394           }
395
396         }
397         else
398         {
399           //il ne peut rien faire si ce n'est finir le tour
400           return(ACTION_FINIR);
401         }
402       }
403     }
404     //s'il se trouve sur une case association ou sur un lieu commun
405     else if (  (plateau[pj_joueur->int_position]->int_type == LC_PARKING)
406         || (plateau[pj_joueur->int_position]->int_type == LC_RU)
407         || (plateau[pj_joueur->int_position]->int_type == LC_ASCENSEUR)
408         || (plateau[pj_joueur->int_position]->int_type == LC_WC))
409     {
410       //on vÃ©rifie si elle n'appartient Ãă personne et qu'il Ãă assez d'argent
411       if (!(pj_joueur->bool_debut) && (plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur==NULL) &&
412       {
413         //il peut alors l'acheter ou finir ce tour
414         return(ACTION_ACHAT);
415       }
416       //sinon si elle appartient Ãă un adversaire
417       else if ((plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur != pj_joueur) && (plateau[pj_joueu
418       {
419         //si il a assez d'argent
420         if (pj_joueur->int_argent >= traitement_loyer_lieu_commun(plateau[pj_joueur->int_position]->case_lieu_commun
421         {
422           //il peut alors payer
423           return(ACTION_PAYER);
424         }
425         else
426         {
427           //sinon il perd
428           return(ACTION_PERDRE);
429         }
430       }
431       //sinon si la case lui appartient
432       else if (plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur == pj_joueur)
433       {
434         //il le peut que finir ce tour
435         return(ACTION_FINIR);
436       }
437     }
438     else if (  (plateau[pj_joueur->int_position]->int_type == BDE)
439         || (plateau[pj_joueur->int_position]->int_type == BDS))
440     {
441       //on vÃ©rifie si elle n'appartient Ãă personne et qu'il Ãă assez d'argent
442       if (!(pj_joueur->bool_debut) && (plateau[pj_joueur->int_position]->case_association.pjoueur_joueur==NULL) &&
443       {
444         //il peut alors l'acheter ou finir ce tour
445         return(ACTION_ACHAT);
446       }
447       else if ((plateau[pj_joueur->int_position]->case_association.pjoueur_joueur != pj_joueur) && (plateau[pj_joueu
448       {
449         //si il a assez d'argent
450         if (pj_joueur->int_argent >= traitement_loyer_association(plateau[pj_joueur->int_position]->case_lieu_commun
451         {
452           //il peut alors payer
453           return(ACTION_PAYER);
454         }
455         else
```

```
456         {
457           //sinon il perd
458           return(ACTION_PERDRE);
459         }
460       }
461       //sinon si la case lui appartient
462       else if (plateau[pj_joueur->int_position]->case_association.pjoueur_joueur == pj_joueur)
463       {
464         //il le peut que finir ce tour
465         return(ACTION_FINIR);
466       }
467     }
468   }
469   else if (int_etat == CLICK_CASE)
470   {
471     i=0;
472     while (plateau[i]!=int_to_cases(int_position_cases_etudiee,pj_joueur)) i++;
473     if (plateau[i]->int_type == SALLE)
474     {
475       //si le joueur est sur une case du mÃ¤me groupe
476       if (plateau[pj_joueur->int_position]->case_salle.int_groupe == plateau[i]->case_salle.int_groupe)
477       {
478         //si sont niveau est nul il ne peut qu'augmenter
479         if (plateau[i]->case_salle.int_niveau == 0)
480         {
481           return(ACTION_PLUS);
482         }
483         //si son niveau est au maximum il ne peut que diminuer
484         else if (plateau[i]->case_salle.int_niveau == 5)
485         {
486           return(ACTION_MOINS);
487         }
488         else
489         {
490           return(ACTION_PLUS_MOINS);
491         }
492       }
493     }
494     else
495     {
496       //il ne peut que suivre les indications spÃ©cifique Ãà chaque case ou chaque carte
497       return(ACTION_FINIR);
498     }
499   }
500
501   return(ERREUR);
502 }
```

8.10.2.9 int decoupage_string (char *** *str_tab*, char * *str_a_diviser*)

divise une chaine en caractère en tableau de chaîne caractère

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

str_tab tableau de chaine de caractère pour le retour de la chaine une fois divisé

str_a_diviser la chaine de caractère à diviser

Version :

1.0

Renvoie :

le nombre de lignes

Définition à la ligne 286 du fichier outils.c.

Référencé par attente_validation_message().

```
287 {
288   int int_ligne_max;
289
290   int i;
291   int k; //curseur indiquant la ligne en cours de crÃľation
292   int j; //curseur de dÃľplacement dans str_a_diviser
293
294   i=0;
295   k=0;
296   j=0;
297
298
299   int_ligne_max=((int)(strlen(str_a_diviser)/MESSAGE_MAX_CARACTERE)+1)*2;
300
301   //on alloue en mÃľmoire un tableau de chaine de caractÃÍre de int_ligne_max lignes contenant chacune des chaine de
302   str_tab[0]=new char*[int_ligne_max];
303
304   for (i = 0 ; i < int_ligne_max ; i++)
305   {
306     str_tab[0][i]=new char[MESSAGE_MAX_CARACTERE];
307   }
308
309   //tant qu'on est pas au bout de la chaine principale
310   while (strlen(str_a_diviser) >= MESSAGE_MAX_CARACTERE)
311   {
312     //on met le curseur Ãă 63
313     j=MESSAGE_MAX_CARACTERE - 1;
314
315     //puis on cherche avant, le premier espace
316     while (str_a_diviser[j] != ' ')
317     {
318       j--;
319     }
320
321     //on copie les j premiers caractÃÍres
322     strncpy(str_tab[0][k],str_a_diviser,j);
323     //on marque la fin de chaine
324     str_tab[0][k][j]='\0';
325
326     //on deplace le pointeur
327     str_a_diviser=str_a_diviser+j;
328     //on passe a la ligne Ãă crÃľer suivante
329     k++;
330   }
331   //aprÃÍs il ne nous reste plus que le morceau de chaine d'une taille infÃľrieur Ãă MESSAGE_MAX_CARACTERE
332   strcpy(str_tab[0][k],str_a_diviser);
333
334   return(k+1);
335 }
```

8.10.2.10 cases* int_to_cases (int *int_indice_case*, joueur * *pj_joueur*)

à partir de l'indice de chaine des propriété elle trouve la case correspondante

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

int_indice_case indice de la case recherché fourni a partir de la liste de propriété du panneau gauche

Version :

1.0

Renvoie :

la case voulue

Définition à la ligne 521 du fichier outils.c.

Référencé par action_possible(), et attente_clic().

```
522 {
523   possession* propriete_en_cours;
524   int i;
525
526   //on assigne la propriÃŀtÃŀ encours avec la premiÃŀre propriÃŀtÃŀ du joueur pour pouvoir parcourir la chaÃõne
527   propriete_en_cours=pj_joueur->propriete;
528
529   //on se dÃŀplace dans la chaÃõne jusqu'Ãă l'indice voulue
530   for (i = 0; i < int_indice_case; i++)
531   {
532     propriete_en_cours=propriete_en_cours->suivant;
533   }
534   //on retourne ainsi la propriete trouvÃŀe
535   return(propriete_en_cours->propriete);
536 }
```

8.10.2.11 int nombre_propriete (joueur * *pj_joueur*)

compte le nombre de propriétés du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

le nombre de propriété du joueur

Définition à la ligne 504 du fichier outils.c.

Référencé par affich_panneau_possessions(), et attente_clic().

```
505 {
506   possession* propriete_temp;
507   int i;
508
509   i=0;
```

```
510   propriete_temp=pj_joueur->propriete;
511   while(propriete_temp!=NULL)
512   {
513     i++;
514     propriete_temp=propriete_temp->suivant;
515   }
516
517   return(i);
518
519 }
```

8.10.2.12 bool verification_victoire (joueur * *pj_joueur*)

compte le nombre de propriétés du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

true si le joueur à gagnersinon false

Définition à la ligne 538 du fichier outils.c.

Référencé par jeu().

```
539 {
540   joueur* joueur_encours;
541   int i;
542
543   i=0;
544
545   //le joueur gagne s'il est seul
546   joueur_encours=pj_joueur->pjoueur_suivant;
547
548   while (joueur_encours != pj_joueur)
549   {
550     i++;
551     joueur_encours=joueur_encours->pjoueur_suivant;
552   }
553
554   if (i > 0)
555   {
556     return(false);
557   }
558   else
559   {
560     return(true);
561   }
562 }
```

8.10.2.13 void action_to_boutons (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_fond*, int *int_action*, int *int_etat*)

définit les boutons à afficher en fonction des actions possibles sur une propriété

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_fond surface de la propriété

int_action actions possibles

int_etat état des boutons

Version :

1.0

Renvoie :

nothing

Définition à la ligne 564 du fichier outils.c.

Référencé par attente_validation_propriete().

```
565 {
566
567   //Selon l'action Ãã faire
568   switch(int_action)
569   {
570     //Dans la cas d'un achat
571     case ACTION_ACHAT:
572       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_ACHAT, BTN_FINIR);
573       break;
574     //Dans le cas d'une deshypothÃÍque
575     case  ACTION_UNHYPOTHEQUE:
576       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_UNHYPOTHEQUE, BTN_FINIR);
577       break;
578     //Dans le cas d'une hypothÃÍque
579     case  ACTION_HYPOTHEQUE:
580       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_HYPOTHEQUE, BTN_FINIR);
581       break;
582     //Pour augmenter le niveau d'une salle
583     case  ACTION_PLUS:
584       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_PLUS, BTN_FINIR);
585       break;
586     //Pour diminuer le niveau d'une salle
587     case  ACTION_MOINS:
588       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 2, BTN_MOINS, BTN_FINIR);
589       break;
590     //Pour augmenter ou diminuer le niveau d'une salle
591     case  ACTION_PLUS_MOINS:
592       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 3, BTN_PLUS, BTN_MOINS, BTN_FINIR);
593       break;
594     //Pour finir l'action
595     case  ACTION_FINIR:
596       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 1, BTN_FINIR);
597       break;
598     //Dans tout les autres cas
599     default:
600       affich_validation_propriete(surf_ecran, surf_fond, int_etat, 1, BTN_FINIR);
```

```
601       break;
602   }
603
604 }
```

8.10.2.14 void sauvegarde (cases ** *plateau*, joueur * *pj_joueur_tete*)

sauvegarde de la partie

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau plateau du jeu

pj_joueur_tete joueur en cours

Version :

1.0

Renvoie :

nothing

Définition à la ligne 606 du fichier outils.c.

Référencé par attente_clic(), et jeu().

```
607 {
608   //Joueur temporaire
609   joueur* pj_joueur;
610
611   //PropriÃŀtÃŀ temporaire
612   possession* possession_temp;
613
614   //Nombre de joueur
615   int int_nombre_joueur;
616   int_nombre_joueur=1;
617
618   //Nominateur de fin de joueur
619   int int_fin;
620   int_fin=-1;
621
622   //Compteur de propriÃŀtÃŀ
623   int i;
624
625   //On commence au premier joueur
626   pj_joueur=pj_joueur_tete;
627
628   //On compte le nombre de joueur
629   while((pj_joueur=pj_joueur->pjoueur_suivant)!=pj_joueur_tete) int_nombre_joueur++;
630
631   //On commence au premier joueur
632   pj_joueur=pj_joueur_tete;
633
634   ofstream file("sauvegarde_joueur.txt");
635
636   file << int_nombre_joueur<<endl;
637
638   //Tant qu'on est pas revenu au joueur de tete
639   while(pj_joueur->pjoueur_suivant!=pj_joueur_tete)
```

```
640   {
641     file << pj_joueur->str_nom << endl;
642     file << pj_joueur->int_position << endl;
643     file << pj_joueur->int_argent << endl;
644     file << pj_joueur->int_certificat << endl;
645     file << pj_joueur->bool_debut << endl;
646     file << pj_joueur->int_double_tire << endl;
647     file << pj_joueur->bool_laurence << endl;
648     file << pj_joueur->int_laurence << endl;
649 
650     possession_temp=pj_joueur->propriete;
651     while(possession_temp!=NULL)
652     {
653       i=0;
654       while (plateau[i]!=possession_temp->propriete) i++;
655       file << i << endl;
656       possession_temp=possession_temp->suivant;
657     }
658 
659     file << int_fin << endl << endl;
660     //On passe au joueur suivant
661     pj_joueur=pj_joueur->pjoueur_suivant;
662   }
663 
664     //Sauvegarde du dernier joueur
665     file << pj_joueur->str_nom << endl;
666     file << pj_joueur->int_position << endl;
667     file << pj_joueur->int_argent << endl;
668     file << pj_joueur->int_certificat << endl;
669     file << pj_joueur->bool_debut << endl;
670     file << pj_joueur->int_double_tire << endl;
671     file << pj_joueur->bool_laurence << endl;
672     file << pj_joueur->int_laurence << endl;
673 
674     possession_temp=pj_joueur->propriete;
675     while(possession_temp!=NULL)
676     {
677       i=0;
678       while (plateau[i]!=possession_temp->propriete) i++;
679       file << i << endl;
680       possession_temp=possession_temp->suivant;
681     }
682 
683     file << int_fin << endl << endl;
684     //On passe au joueur suivant
685     pj_joueur=pj_joueur->pjoueur_suivant;
686 
687   //Sauvegarde du plateau
688   ofstream file2("sauvegarde_plateau.txt");
689 
690   //On Ã¯crit le total de la cagnotte
691   file2 << plateau[20]->machine_a_cafe.int_argent << endl;
692   //Pour chacune des cases
693   for(i=0;i<40;i++)
694   {
695     //S'il sagit d'une salle, on Ã¯crit son niveau
696     if(plateau[i]->int_type==SALLE) file2 << plateau[i]->case_salle.int_niveau << endl;
697     //Sinon on met -1
698     else file2 << int_fin << endl;
699   }
700 
701 
702 }
```

8.10.2.15 joueur* chargement (joueur * *pj_joueur*, cases ** *plateau*, rvb_couleur *couleurs*[8], SDL_Surface * *surf_ecran*)

chargement de la partie

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

plateau plateau du jeu

pj_joueur joueur en cours

couleurs tableau de couleur

surf_ecran surface de l'écran

Version :

1.0

Renvoie :

un joueur de la liste de joueur

Définition à la ligne 704 du fichier outils.c.

Référencé par main().

```
705 {
706
707   ifstream file1("sauvegarde_joueur.txt", ios::in);
708   if (!file1)
709   {
710     cerr << "Allocation inaboutie de sauvegarde_joueur.txt" << endl;
711   }
712
713   ifstream file2("sauvegarde_plateau.txt",ios::in);
714   if (!file2)
715   {
716     cerr << "Allocation inaboutie de sauvegarde_plateau.txt" << endl;
717   }
718
719
720   int nb_joueurs;
721
722   file1 >> nb_joueurs;
723
724   pj_joueur=init_anneau_joueur_chargement(nb_joueurs, &file1,plateau);
725   plateau=init_plateau_chargement(&file2, plateau, couleurs,surf_ecran);
726
727   file1.close();
728   file2.close();
729
730   return(pj_joueur);
731
732 }
```

8.10.2.16 bool is_readable (const std : :string & *file*)

test si un fichier existe

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

file le fichier à tester

Version :

1.0

Renvoie :

si le fichier existe ou non

Définition à la ligne 734 du fichier outils.c.

Référencé par attente_action_accueil().

```
735 {
736     std::ifstream fichier( file.c_str() );
737     return !fichier.fail();
738 }
```

8.11 Référence du fichier sdl.c

Fonctions

- void **mapause** (void)
- SDL_Surface * **rotation_90** (SDL_Surface *Depart)
  permet de tourner une image de 90 degrÃľ
- SDL_Surface * **rotation_180** (SDL_Surface *Depart)
  permet de tourner une image de 180 degrÃľ
- SDL_Surface * **rotation_270** (SDL_Surface *Depart)
  permet de tourner une image de 270 degrÃľ
- SDL_Surface * **init_sdl** (void)
  fonction d'initialisation de sdl
- SDL_Surface * **creation_joueur** (int int_joueur)
  crÃľer la surface du joueur
- void **affich_joueur** (SDL_Surface *surf_ecran, **joueur** *pj_joueur, int int_position, **cases** *pcase)
  affiche un joueur Ãă l'Ãlcran
- SDL_Surface * **creation_case** (SDL_Surface *surf_ecran, **rvb_couleur** couleur, char *texte1, char *texte2, int int_prix, int int_position)
  crÃľer une case de type salle
- SDL_Surface * **creation_case_bureau** (SDL_Surface *ecran, int int_type, int int_position)
  crÃľer une case de type bureau
- SDL_Surface * **creation_case_lieu_commun** (SDL_Surface *ecran, int int_type)
  crÃľer une case de type lieu commun
- SDL_Surface * **creation_case_association** (SDL_Surface *ecran, int int_type)
  crÃľer une case de type association
- SDL_Surface * **creation_case_soiree** (SDL_Surface *ecran, int int_type)
  crÃľer une case de type soirÃľe
- SDL_Surface * **creation_case_coin** (SDL_Surface *ecran, int int_type)
  crÃľer une case spÃľcial (chacun des coins)
- SDL_Surface * **creation_case_detail** (SDL_Surface *surf_ecran, **rvb_couleur** couleur, char *texte, int int_prix, int int_prix_niveau)
  crÃľer la surface de la case dÃľtaillÃľ
- SDL_Surface * **creation_case_detail_lc** (SDL_Surface *surf_ecran, int int_type)
  crÃľer la surface de la case dÃľtaille d'une case lieu commun
- SDL_Surface * **creation_case_detail_assoc** (SDL_Surface *surf_ecran, int int_type)
  crÃľer la surface de la case dÃľtaille d'une case association
- SDL_Surface * **creation_case_propriete** (SDL_Surface *surf_ecran, **rvb_couleur** couleur, char *texte)
  crÃľer la surface de la propriÃľtÃľ qui s'affichera dans le panneau des propriÃľtÃľs
- int **affich_accueil** (SDL_Surface *surf_ecran)
  affiche la page d'accueil

- int **affich_config** (SDL_Surface *surf_ecran, char **str_nom_joueur, int nombre_joueur)
  affiche la page de configuration de la partie

- void **affich_case** (SDL_Surface *surf_ecran, **cases** *pcase)
  affiche une case Ãă l'Ãľcran

- void **affich_case_detail** (SDL_Surface *surf_ecran, **cases** *pcase)
  affiche le dÃľtail d'une case

- void **affich_centre** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre)
  affiche le centre du plateau

- void **affich_panneau_menu** (SDL_Surface *surf_ecran, int int_etat_bouton)
  affiche le panneau du menu

- void **affich_panneau_joueur** (SDL_Surface *surf_ecran, **joueur** *j_anneau_joueurs)
  affiche le panneau d'information sur le joueur

- void **affich_panneau_possessions** (SDL_Surface *surf_ecran, **joueur** *j_anneau_joueurs)
  affiche le panneau des possessions du joueur

- void **affich_possessions_cache** (SDL_Surface *surf_ecran)
  affiche le cache du panneau de possesion du joueur

- void **affich_panneau_des_bouton** (SDL_Surface *surf_ecran, int int_image)
  affiche le panneau contenant le bouton lancer dÃľ

- SDL_Surface ** **creation_des** (void)
  renvoit un tableau de surface contenant chacune des faces du dÃľ

- void **destruction_des** (SDL_Surface **surf_des)
  dÃľtruit les surfaces des dÃľs

- void **affich_panneau_fdt** (SDL_Surface *surf_ecran, bool bool_etat)
  affiche le panneau du menu

- void **affich_panneau_des** (SDL_Surface *surf_ecran, SDL_Surface **surf_des, int int_de1, int int_de2)
  affiche le panneau du menu

- void **affich_joueur_depart** (SDL_Surface *surf_ecran, **cases** **plateau, **joueur** *j_anneau_joueurs, int nombre_joueur)
  affiche tout les joueurs sur la case de dÃľpart

- int **affich_validation_propriete** (SDL_Surface *surf_ecran, SDL_Surface *surf_fond, int int_etat, int int_nombre_boutons,...)
  affiche le message des propriÃľtÃľ ainsi que les boutons d'actions possibles

- SDL_Surface * **creation_message** (SDL_Surface *surf_ecran, char *titre, char **message, int int_type_message, int int_nbre_ligne)
  crÃľer la surface d'un message Ãă afficher

- void **affich_message** (SDL_Surface *surf_ecran, SDL_Surface *surf_message, int int_type_message, int int_etat)
  affiche un message Ãă l'Ãľcran

8.11.1 Documentation des fonctions

8.11.1.1 void mapause (void)

Définition à la ligne 4 du fichier sdl.c.

```
5 {
6     int continuer = 1;
7     SDL_Event event;
8     while (continuer)
9     {
10         SDL_WaitEvent(&event);
11         switch(event.type)
12         {
13             case SDL_QUIT:
14                 continuer = 0;
15         break;
16       case SDL_KEYDOWN:
17         switch(event.key.keysym.sym)
18         {
19           case SDLK_ESCAPE:
20             continuer = 0;
21             break;
22           default:
23           break;
24         }
25         break;
26       default:
27       break;
28         }
29     }
30 }
```

8.11.1.2 SDL_Surface * rotation_90 (SDL_Surface * *Depart*)

permet de tourner une image de 90 degrÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

Depart surface de dÃľpart

Version :

1.0

Renvoie :

la surface tournÃľe

Définition à la ligne 32 du fichier sdl.c.

Référencé par affich_case(), creation_case(), creation_case_bureau(), creation_case_lieu_commun(), et creation_case_soiree().

```
33 {
34   //La surface d'arrivee
35   SDL_Surface* arrivee;
36   arrivee = SDL_CreateRGBSurface(SDL_HWSURFACE,Depart->h,Depart->w,Depart->format->BitsPerPixel,0,0,0,0);
```

```
37
38   int x;
39   int y;
40
41   for(y=0;y<Depart->h;y+=1)
42   {
43     for(x=0;x<Depart->w;x+=1)
44     {
45       ((Uint32*)arrivee->pixels)[y+((arrivee->h-1)-x)*arrivee->w]=((Uint32*)Depart->pixels)[x+y*Depart->w];
46     }
47   }
48   return(arrivee);
49 }
```

8.11.1.3 SDL_Surface * rotation_180 (SDL_Surface * *Depart*)

permet de tourner une image de 180 degrÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

Depart surface de dÃľpart

Version :

1.0

Renvoie :

la surface tournÃľe

Définition à la ligne 51 du fichier sdl.c.

Référencé par affich_case(), creation_case(), creation_case_association(), creation_case_bureau(), et creation_case_lieu_commun().

```
52 {
53   // La surface d'arrivee
54   SDL_Surface* arrivee;
55   arrivee = SDL_CreateRGBSurface(SDL_HWSURFACE,Depart->w,Depart->h,Depart->format->BitsPerPixel,0,0,0,0);
56
57   int x;
58   int y;
59   for(y=0;y<Depart->h;y+=1)
60   {
61     for(x=0;x<Depart->w;x+=1)
62     {
63       ((Uint32*)arrivee->pixels)[((arrivee->w-1)-x)+((arrivee->h-1)-y)*arrivee->w]=((Uint32*)Depart->pixels)[x+y*Depa
64     }
65   }
66   return(arrivee);
67 }
```

8.11.1.4 SDL_Surface * rotation_270 (SDL_Surface * *Depart*)

permet de tourner une image de 270 degrÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

Depart surface de dÃľpart

Version :

1.0

Renvoie :

la surface tournÃľe

Définition à la ligne 69 du fichier sdl.c.

Référencé par affich_case(), creation_case(), creation_case_association(), creation_case_bureau(), et creation_case_lieu_commun().

```
70 {
71   // La surface d'arrivee
72   SDL_Surface* arrivee;
73   arrivee = SDL_CreateRGBSurface(SDL_HWSURFACE,Depart->h,Depart->w,Depart->format->BitsPerPixel,0,0,0,0);
74
75   int x,y;
76   for(y=0;y<Depart->h;y+=1)
77   {
78     for(x=0;x<Depart->w;x+=1)
79     {
80       ((Uint32*)arrivee->pixels)[((arrivee->w-1)-y)+x*arrivee->w]=((Uint32*)Depart->pixels)[x+y*Depart->w];
81     }
82   }
83   return(arrivee);
84 }
```

8.11.1.5 SDL_Surface * init_sdl (void)

fonction d'initialisation de sdl

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Version :

1.0

Renvoie :

la surface de l'Ãľcran

Définition à la ligne 86 du fichier sdl.c.

Référencé par main().

```
87 {
88   //Surface de l'ecran
89   SDL_Surface *ecran=NULL;
90
91   //Initialisation en mode video de sdl
92   SDL_Init(SDL_INIT_VIDEO);
93
```

```
94   //Répétition des touches
95   SDL_EnableKeyRepeat(80,80);
96
97   //Initialisation du mode TTF (Pour l'ecriture)
98   TTF_Init();
99
100   //Chargement du mode video en resolution 800*600 et 32 couleur, plein ecran.
101   ecran = SDL_SetVideoMode(1280, 1024, 32, SDL_HWSURFACE|SDL_FULLSCREEN);
102
103   //Chargement du titre de la fenetre
104   SDL_WM_SetCaption("Monopoly", NULL);
105
106   //On remplit le fond de noir
107   SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 25, 25, 75));
108
109   //On retourne la surface de l'écran
110   return(ecran);
111 }
```

8.11.1.6 SDL_Surface * creation_joueur (int *int_joueur*)

crÃľer la surface du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

int_joueur numÃľro du joueur

Version :

1.0

Renvoie :

la surface du joueur

Définition à la ligne 113 du fichier sdl.c.

Référencé par init_joueur(), et init_joueur_chargement().

```
114 {
115   //Surface du joueur
116   SDL_Surface* surf_joueur;
117   surf_joueur=NULL;
118   //Chemin d'accès à l'image
119   char chemin[256];
120
121   //Stockage du chemin vers l'image du joueur en fonction de son numéro
122   sprintf(chemin,"sdl/images/joueur%d.png",int_joueur);
123
124   //Chargement de l'image
125   surf_joueur=IMG_Load(chemin);
126
127   //On retourne l'image du joueur
128   return(surf_joueur);
129
130 }
```

8.11.1.7 void affich_joueur (SDL_Surface * *surf_ecran*, joueur * *pj_joueur*, int *int_position*, cases * *pcase*)

affiche un joueur Ãă l'Ãľcran

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

pj_joueur joueur en cours

int_position position du joueur sur la case

pcase case sur laquelle se trouve le joueur

Version :

1.0

Renvoie :

++

Définition à la ligne 132 du fichier sdl.c.

Référencé par affich_joueur_depart(), aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), et reculer_jeton().

```
133 {
134   //Position du joueur
135   SDL_Rect position;
136 
137   //Selon la position de la case
138   switch((*pcase).int_position)
139   {
140     //On calcul les coordonnées de l'image du joueur
141     case POSITION_GAUCHE:
142       position.x=(*pcase).rect_coordonnees.x+25+30*(int_position%2);
143       position.y=(*pcase).rect_coordonnees.y+5+25*(int_position%3);
144       break;
145     case POSITION_DROITE:
146       position.x=(*pcase).rect_coordonnees.x+35+30*(int_position%2);
147       position.y=(*pcase).rect_coordonnees.y+5+25*(int_position%3);
148       break;
149     case POSITION_HAUT:
150       position.x=(*pcase).rect_coordonnees.x+5+25*(int_position%3);
151       position.y=(*pcase).rect_coordonnees.y+25+30*(int_position%2);
152       break;
153     default:
154       position.x=(*pcase).rect_coordonnees.x+5+25*(int_position%3);
155       position.y=(*pcase).rect_coordonnees.y+35+30*(int_position%2);
156       break;
157   }
158   //On colle l'image du joueur sur le fond
159   SDL_BlitSurface(pj_joueur->surf_image, NULL, surf_ecran, &position);
160   //On met à jour l'écran
161   SDL_Flip(surf_ecran);
162 }
```

8.11.1.8 SDL_Surface * creation_case (SDL_Surface * *surf_ecran*, rvb_couleur *couleur*, char * *texte1*, char * *texte2*, int *int_prix*, int *int_position*)

crÃľer une case de type salle

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

couleur couleur du groupe

texte1 premiÃÍre partie du nom de la case

texte2 seconde partie du nom de la case

int_prix prix de la case

int_position cotÃľ sur lequel se trouve la case

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 164 du fichier sdl.c.

Référencé par init_case_salle().

```
165 {
166   //Chaine de caractere du prix
167   char texte_prix[255];
168 
169   //Surface de la case
170   SDL_Surface* surf_case;
171   surf_case=NULL;
172 
173   //Surface du fond de la case
174   SDL_Surface* surf_fond;
175   surf_fond=NULL;
176 
177   //Surface de la couleur du groupe
178   SDL_Surface* surf_groupe;
179   surf_groupe=NULL;
180 
181   //Surface du fond du groupe
182   SDL_Surface* surf_fond_groupe;
183   surf_fond_groupe=NULL;
184 
185   //Surface du texte partie 1
186   SDL_Surface* surf_texte1;
187   surf_texte1=NULL;
188 
189   //Surface du texte partie 2
190   SDL_Surface* surf_texte2;
191   surf_texte2=NULL;
192 
193   //Surface du texte partie 2
194   SDL_Surface* surf_texte_prix;
195   surf_texte_prix=NULL;
196 
197   //Police d'écriture
```

```
198   TTF_Font* police;
199   police=NULL;
200
201   //Couleur du texte
202   SDL_Color couleur_texte = {0, 0, 0};
203   //Couleur du fond du texte
204   SDL_Color couleur_fond_texte = {16, 246, 128};
205
206   //Position
207   SDL_Rect position;
208
209   //Ouverture de la police d'ecriture
210   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
211
212   //Ecriture de la première partie du texte
213   surf_texte1 = TTF_RenderText_Shaded(police, texte1, couleur_texte, couleur_fond_texte);
214   //Ecriture de la seconde partie du texte
215   surf_texte2 = TTF_RenderText_Shaded(police, texte2, couleur_texte, couleur_fond_texte);
216   //Enregistrement du prix
217   sprintf(texte_prix,"%d Fintz",int_prix);
218   //Ecriture du prix
219   surf_texte_prix = TTF_RenderText_Shaded(police, texte_prix, couleur_texte, couleur_fond_texte);
220
221   //Fermeture de la police d'ecriture
222   TTF_CloseFont(police);
223
224   //Création de la surface de la case
225   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
226   //Remplissage de noir pour faire le cadre
227   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
228
229   //Création de la surface de fond
230   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
231   //Remplissage de vert tapis
232   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 16, 246, 128));
233
234   //Création de la surface du groupe
235   surf_groupe=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_GROUPE_HAUTEUR+2*CASE_EPAISSEUR, 32, 0, 0, 0, 0
236   //Remplissage de noir pour faire le cadre
237   SDL_FillRect(surf_groupe, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
238
239   //Création de la surface de fond du groupe
240   surf_fond_groupe=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_GROUPE_HAUTEUR, 32, 0, 0
241   //Remplissage de la couleur de fond du groupe
242   SDL_FillRect(surf_fond_groupe, NULL, SDL_MapRGB(surf_ecran->format, couleur.rouge, couleur.vert, couleur.bleu));
243
244   //On modifie la position du fond du groupe
245   position.x=CASE_EPAISSEUR;
246   position.y=CASE_EPAISSEUR;
247   //On colle le fond du groupe sur ce dernier
248   SDL_BlitSurface(surf_fond_groupe, NULL, surf_groupe, &position);
249
250   //On modifie la position du fond
251   position.x=CASE_EPAISSEUR;
252   position.y=CASE_EPAISSEUR;
253   //On colle le fond sur la case
254   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
255
256   //On modifie la position du groupe
257   position.x=0;
258   position.y=0;
259   //On colle le groupe sur la case
260   SDL_BlitSurface(surf_groupe, NULL, surf_case, &position);
261
262   //On centre la position de la premiere partie texte
263   position.x=(CASE_LARGEUR-(*surf_texte1).w)/2;
264   position.y=30;
```

```
265   //On colle la première partie du texte
266   SDL_BlitSurface(surf_texte1, NULL, surf_case, &position);
267
268   //On centre la position de la deuxieme partie du texte
269   position.x=(CASE_LARGEUR-(*surf_texte2).w)/2;
270   position.y=50;
271   //On colle la deuxième partie du texte
272   SDL_BlitSurface(surf_texte2, NULL, surf_case, &position);
273
274   //On centre la position du prix
275   position.x=(CASE_LARGEUR-(*surf_texte_prix).w)/2;
276   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
277   //On colle le prix
278   SDL_BlitSurface(surf_texte_prix, NULL, surf_case, &position);
279
280   //Selon la position
281   switch(int_position)
282   {
283     //Si la case est à droite
284     case POSITION_DROITE:
285       //Rotation de l'image de 90ř
286       surf_case = rotation_90(surf_case);
287       break;
288     //Si la case est en haut
289     case POSITION_HAUT:
290       //Rotation de l'image de 180ř
291       surf_case = rotation_180(surf_case);
292       break;
293     //Si la case est à gauche
294     case POSITION_GAUCHE:
295       //Rotation de l'image de 270ř
296       surf_case = rotation_270(surf_case);
297       break;
298     //Par défaut, la case est en bas
299     default:
300     break;
301   }
302
303   //Libération des surfaces temporaires
304   SDL_FreeSurface(surf_fond);
305   SDL_FreeSurface(surf_fond_groupe);
306   SDL_FreeSurface(surf_groupe);
307   SDL_FreeSurface(surf_texte1);
308   SDL_FreeSurface(surf_texte2);
309   SDL_FreeSurface(surf_texte_prix);
310
311   //On retourne la surface de la case
312   return(surf_case);
313 }
```

8.11.1.9 SDL_Surface * creation_case_bureau (SDL_Surface * *ecran*, int *int_type*, int *int_position*)

crÃľer une case de type bureau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de bureau

int_position cÃťtÃľ sur lequel se trouve la case

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 315 du fichier sdl.c.

Référencé par init_case_administration().

```
316 {
317   //Surface du logo
318   SDL_Surface* surf_logo;
319   surf_logo=NULL;
320   //Surface du fond de la case
321   SDL_Surface* surf_fond;
322   surf_fond=NULL;
323   //Surface de la case
324   SDL_Surface* surf_case;
325   surf_case=NULL;
326   //Surface de la première partie du texte
327   SDL_Surface* surf_texte1;
328   surf_texte1=NULL;
329   //Surface de la deuxième partie du texte
330   SDL_Surface* surf_texte2;
331   surf_texte2=NULL;
332   //Position des images
333   SDL_Rect position;
334
335   //Police d'écriture
336   TTF_Font* police;
337   police=NULL;
338
339   //Couleur du texte
340   SDL_Color couleur_texte = {0, 0, 0};
341   //Couleur du fond du texte
342   SDL_Color couleur_fond_texte = {16, 246, 128};
343
344   //Ouvertue de la police d'écriture
345   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
346
347   //Ecriture du premier texte
348   surf_texte1 = TTF_RenderText_Shaded(police, "Bureau de", couleur_texte, couleur_fond_texte);
349   //Ecriture du second texte
350   if(int_type==BUREAU_KRYSTEL) surf_texte2 = TTF_RenderText_Shaded(police, "Krystel", couleur_texte, couleur_fond_te
351   else surf_texte2 = TTF_RenderText_Shaded(police, "Nadege", couleur_texte, couleur_fond_texte);
352
353   //Fermeture de la police d'écriture
354   TTF_CloseFont(police);
355
356   //Création de la surface de la case
357   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
358   //Remplissage de noir pour faire le cadre
359   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
360
361   //Création de la surface de fond
362   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
363   //Remplissage de vert tapis
364   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
365
366   //Chargelent de l'image du logo
367   if(int_type==BUREAU_KRYSTEL) surf_logo = IMG_Load("sdl/images/eisti_logo.png");
368   else surf_logo = IMG_Load("sdl/images/eisti_logo2.png");
369
370   //On modifie la position du fond
371   position.x=CASE_EPAISSEUR;
```

```
372   position.y=CASE_EPAISSEUR;
373   //On colle le fond sur la case
374   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
375
376   //On centre la position de la première partie texte
377   position.x=(CASE_LARGEUR-(*surf_texte1).w)/2;
378   position.y=5;
379   //On colle la première partie du texte
380   SDL_BlitSurface(surf_texte1, NULL, surf_case, &position);
381
382   //On centre la position de la deuxième partie du texte
383   position.x=(CASE_LARGEUR-(*surf_texte2).w)/2;
384   position.y=25;
385   //On colle la deuxième partie du texte
386   SDL_BlitSurface(surf_texte2, NULL, surf_case, &position);
387
388   //On positione le logo
389   position.x=16;
390   position.y=45;
391   //On colle le logo sur la case
392   SDL_BlitSurface(surf_logo, NULL, surf_case, &position);
393
394   //Selon la position
395   switch(int_position)
396   {
397     //Si la case est à droite
398     case POSITION_DROITE:
399       //Rotation de l'image de 90ř
400       surf_case = rotation_90(surf_case);
401       break;
402     //Si la case est en haut
403     case POSITION_HAUT:
404       //Rotation de l'image de 180ř
405       surf_case = rotation_180(surf_case);
406       break;
407     //Si la case est à gauche
408     case POSITION_GAUCHE:
409       //Rotation de l'image de 270ř
410       surf_case = rotation_270(surf_case);
411       break;
412     //Par défaut, la case est en bas
413     default:
414     break;
415   }
416
417   //Libération des surfaces temporaires
418   SDL_FreeSurface(surf_fond);
419   SDL_FreeSurface(surf_logo);
420   SDL_FreeSurface(surf_texte1);
421   SDL_FreeSurface(surf_texte2);
422   //On retourne la case que l'on vient de créer
423   return(surf_case);
424 }
```

8.11.1.10 SDL_Surface * creation_case_lieu_commun (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case de type lieu commun

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de lieu commun

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 426 du fichier sdl.c.

Référencé par init_case_lieu_commun().

```
427 {
428   //Surface de l'image
429   SDL_Surface* surf_image;
430   surf_image=NULL;
431   //Surface du fond de la case
432   SDL_Surface* surf_fond;
433   surf_fond=NULL;
434   //Surface de la case
435   SDL_Surface* surf_case;
436   surf_case=NULL;
437   //Surface du prix
438   SDL_Surface* surf_prix;
439   surf_prix=NULL;
440
441   //Position des images
442   SDL_Rect position;
443
444   //Police d'écriture
445   TTF_Font* police;
446   police=NULL;
447
448   //Couleur du texte
449   SDL_Color couleur_texte = {0, 0, 0};
450   //Couleur du fond du texte
451   SDL_Color couleur_fond_texte = {16, 246, 128};
452
453   //Ouvertue de la police d'écriture
454   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
455
456   //Ecriture du premier texte
457   surf_prix = TTF_RenderText_Shaded(police, "2000 Fintz", couleur_texte, couleur_fond_texte);
458
459   //Fermeture de la police d'écriture
460   TTF_CloseFont(police);
461
462   //Création de la surface de la case
463   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
464   //Remplissage de noir pour faire le cadre
465   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
466
467   //Création de la surface de fond
468   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
469   //Remplissage de vert tapis
470   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
471
472   //Chargement de l'image
473   //if(int_type==BDE) surf_image = IMG_Load("sdl/images/bureau_bde.png");
474   switch(int_type)
475   {
476     case LC_WC:
```

```
477       surf_image = IMG_Load("sdl/images/wc.png");
478       break;
479     case LC_ASCENSEUR:
480       surf_image = IMG_Load("sdl/images/ascenseur.png");
481       break;
482     case LC_RU:
483       surf_image = IMG_Load("sdl/images/ru.png");
484       break;
485     case LC_PARKING:
486       surf_image = IMG_Load("sdl/images/parking.png");
487       break;
488     default:
489       break;
490   }
491   //On modifie la position du fond
492   position.x=CASE_EPAISSEUR;
493   position.y=CASE_EPAISSEUR;
494   //On colle le fond sur la case
495   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
496
497   //On positionne le prix
498   position.x=(CASE_LARGEUR-(*surf_prix).w)/2;
499   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
500   //On colle le prix sur la case
501   SDL_BlitSurface(surf_prix, NULL, surf_case, &position);
502
503   //On positione le logo
504   position.x=10;
505   position.y=8;
506   //On colle le logo sur la case
507   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
508
509   //Selon la position
510   switch(int_type)
511   {
512     //Si la case est à droite
513     case LC_PARKING:
514       //Rotation de l'image de 90ř
515       surf_case = rotation_90(surf_case);
516       break;
517     //Si la case est en haut
518     case LC_RU:
519       //Rotation de l'image de 180ř
520       surf_case = rotation_180(surf_case);
521       break;
522     //Si la case est à gauche
523     case LC_ASCENSEUR:
524       //Rotation de l'image de 270ř
525       surf_case = rotation_270(surf_case);
526       break;
527     //Par défaut, la case est en bas
528     default:
529     break;
530   }
531
532   //Libération des surfaces temporaires
533   SDL_FreeSurface(surf_fond);
534   SDL_FreeSurface(surf_image);
535   SDL_FreeSurface(surf_prix);
536
537   //On retourne la case que l'on vient de créer
538   return(surf_case);
539 }
```

8.11.1.11 SDL_Surface * creation_case_association (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case de type association

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de l'association

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 540 du fichier sdl.c.

Référencé par init_case_association().

```
541 {
542   //Surface du logo
543   SDL_Surface* surf_logo;
544   surf_logo=NULL;
545   //Surface du fond de la case
546   SDL_Surface* surf_fond;
547   surf_fond=NULL;
548   //Surface de la case
549   SDL_Surface* surf_case;
550   surf_case=NULL;
551   //Surface du prix
552   SDL_Surface* surf_prix;
553   surf_prix=NULL;
554
555   //Position des images
556   SDL_Rect position;
557
558   //Police d'écriture
559   TTF_Font* police;
560   police=NULL;
561
562   //Couleur du texte
563   SDL_Color couleur_texte = {0, 0, 0};
564   //Couleur du fond du texte
565   SDL_Color couleur_fond_texte = {16, 246, 128};
566
567   //Ouvertue de la police d'écriture
568   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
569
570   //Ecriture du premier texte
571   surf_prix = TTF_RenderText_Shaded(police, "1500 Fintz", couleur_texte, couleur_fond_texte);
572
573   //Fermeture de la police d'écriture
574   TTF_CloseFont(police);
575
576   //Création de la surface de la case
577   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
578   //Remplissage de noir pour faire le cadre
579   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
```

```
580
581   //Création de la surface de fond
582   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
583   //Remplissage de vert tapis
584   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
585
586   //Chargelent de l'image du logo
587   if(int_type==BDE) surf_logo = IMG_Load("sdl/images/bureau_bde.png");
588   else surf_logo = IMG_Load("sdl/images/bureau_bds.png");
589
590   //On modifie la position du fond
591   position.x=CASE_EPAISSEUR;
592   position.y=CASE_EPAISSEUR;
593   //On colle le fond sur la case
594   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
595
596   //On positionne le prix
597   position.x=(CASE_LARGEUR-(*surf_prix).w)/2;
598   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
599   //On colle le prix sur la case
600   SDL_BlitSurface(surf_prix, NULL, surf_case, &position);
601
602   //On positione le logo
603   position.x=10;
604   position.y=8;
605   //On colle le logo sur la case
606   SDL_BlitSurface(surf_logo, NULL, surf_case, &position);
607
608   //Selon le type de bureau d'association
609   switch(int_type)
610   {
611     //S'il s'agit du bde
612     case BDE:
613       //Rotation de l'image de 180
614       surf_case = rotation_180(surf_case);
615       break;
616     //Par défaut, il s'agit du bds
617     default:
618       //Rotation de l'image de 270ř
619       surf_case = rotation_270(surf_case);
620       break;
621     break;
622   }
623
624   //Libération des surfaces temporaires
625   SDL_FreeSurface(surf_fond);
626   SDL_FreeSurface(surf_logo);
627   SDL_FreeSurface(surf_prix);
628
629   //On retourne la case que l'on vient de créer
630   return(surf_case);
631 }
```

8.11.1.12 SDL_Surface * creation_case_soiree (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case de type soirÃľe

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de la soirÃľe

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 633 du fichier sdl.c.

Référencé par init_case_soiree().

```
634 {
635   //Surface du logo
636   SDL_Surface* surf_logo;
637   surf_logo=NULL;
638   //Surface du fond de la case
639   SDL_Surface* surf_fond;
640   surf_fond=NULL;
641   //Surface de la case
642   SDL_Surface* surf_case;
643   surf_case=NULL;
644   //Surface du prix
645   SDL_Surface* surf_prix;
646   surf_prix=NULL;
647   SDL_Surface* surf_texte;
648   surf_texte=NULL;
649
650   //Position des images
651   SDL_Rect position;
652
653   //Police d'écriture
654   TTF_Font* police;
655   police=NULL;
656
657   //Couleur du texte
658   SDL_Color couleur_texte = {0, 0, 0};
659   //Couleur du fond du texte
660   SDL_Color couleur_fond_texte = {16, 246, 128};
661
662   //Ouvertue de la police d'écriture
663   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
664
665   //Ecriture du premier texte
666   if(int_type==SOIREE_AREA) surf_texte = TTF_RenderText_Shaded(police, "Area club", couleur_texte, couleur_fond_text
667   else surf_texte = TTF_RenderText_Shaded(police, "Gala", couleur_texte, couleur_fond_texte);
668   //Ecriture du premier texte
669   surf_prix = TTF_RenderText_Shaded(police, "1500 Fintz", couleur_texte, couleur_fond_texte);
670
671   //Fermeture de la police d'écriture
672   TTF_CloseFont(police);
673
674   //Création de la surface de la case
675   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
676   //Remplissage de noir pour faire le cadre
677   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
678
679   //Création de la surface de fond
680   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
681   //Remplissage de vert tapis
682   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
683
684   //Chargelent de l'image du logo
685   if(int_type==SOIREE_AREA) surf_logo = IMG_Load("sdl/images/soiree_area.png");
686   else surf_logo = IMG_Load("sdl/images/soiree_gala.png");
```

```
687
688   //On modifie la position du fond
689   position.x=CASE_EPAISSEUR;
690   position.y=CASE_EPAISSEUR;
691   //On colle le fond sur la case
692   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
693
694   //On positionne le prix
695   position.x=(CASE_LARGEUR-(*surf_texte).w)/2;
696   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-40;
697   //On colle le prix sur la case
698   SDL_BlitSurface(surf_texte, NULL, surf_case, &position);
699
700   //On positionne le prix
701   position.x=(CASE_LARGEUR-(*surf_prix).w)/2;
702   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
703   //On colle le prix sur la case
704   SDL_BlitSurface(surf_prix, NULL, surf_case, &position);
705
706   //On positione le logo
707   position.x=10;
708   position.y=8;
709   //On colle le logo sur la case
710   SDL_BlitSurface(surf_logo, NULL, surf_case, &position);
711
712   //S'il s'agit du gala
713   if(int_type==SOIREE_GALA)
714   {
715     //Rotation de l'image de 90ř
716     surf_case = rotation_90(surf_case);
717   }
718
719   //Libération des surfaces temporaires
720   SDL_FreeSurface(surf_fond);
721   SDL_FreeSurface(surf_logo);
722   SDL_FreeSurface(surf_prix);
723   SDL_FreeSurface(surf_texte);
724
725   //On retourne la case que l'on vient de créer
726   return(surf_case);
727 }
```

8.11.1.13 SDL_Surface * creation_case_coin (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case spÃľcial (chacun des coins)

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de la case

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 729 du fichier sdl.c.

Référencé par init_case_special().

```
730 {
731   //Surface de l'image
732   SDL_Surface* surf_image;
733   surf_image=NULL;
734   //Surface du fond de la case
735   SDL_Surface* surf_fond;
736   surf_fond=NULL;
737   //Surface de la case
738   SDL_Surface* surf_case;
739   surf_case=NULL;
740 
741   //Position des images
742   SDL_Rect position;
743 
744   //Création de la surface de la case
745   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_HAUTEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
746   //Remplissage de noir pour faire le cadre
747   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
748 
749   //Création de la surface de fond
750   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_HAUTEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
751   //Remplissage de vert tapis
752   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
753 
754   //Chargelent de l'image du logo
755   switch(int_type)
756   {
757     case SP_APPARTEMENT:
758       surf_image = IMG_Load("sdl/images/case_depart.png");
759       break;
760     case SP_BUREAU_LAURENCE:
761       surf_image = IMG_Load("sdl/images/bureau_laurence.png");
762       break;
763     case SP_MACHINE_CAFE:
764       surf_image = IMG_Load("sdl/images/machine_cafe.png");
765       break;
766     case SP_TABLEAU:
767       surf_image = IMG_Load("sdl/images/go_to_prison.png");
768       break;
769   }
770 
771   //On modifie la position du fond
772   position.x=CASE_EPAISSEUR;
773   position.y=CASE_EPAISSEUR;
774   //On colle le fond sur la case
775   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
776 
777   //On positionne l'image sur le fond
778   position.x=CASE_EPAISSEUR;
779   position.y=CASE_EPAISSEUR;
780   //On colle le logo sur la case
781   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
782 
783   //Libération des surfaces temporaires
784   SDL_FreeSurface(surf_fond);
785   SDL_FreeSurface(surf_image);
786 
787   //On retourne la case que l'on vient de créer
788   return(surf_case);
789 }
```

8.11.1.14 SDL_Surface * creation_case_detail (SDL_Surface * *surf_ecran*, rvb_couleur *couleur*, char * *texte*, int *int_prix*, int *int_prix_niveau*)

crÃľer la surface de la case dÃľtaillÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

couleur couleur du groupe de la case

texte nom de la case

int_prix prix de la case

int_prix_niveau prix d'un niveau de la case

Version :

1.0

Renvoie :

la surface de la case dÃľtaillÃľ

Définition à la ligne 791 du fichier sdl.c.

Référencé par init_case_salle().

```
792 {
793   SDL_Surface* surf_case;
794   surf_case=NULL;
795   SDL_Surface* surf_fond;
796   surf_fond=NULL;
797   SDL_Surface* surf_groupe;
798   surf_groupe=NULL;
799   SDL_Surface* surf_groupe_fond;
800   surf_groupe_fond=NULL;
801   SDL_Surface* surf_sep;
802   surf_sep=NULL;
803   SDL_Surface* surf_texte[22];
804   char texte_temp[256];
805   int i;
806
807   //Police d'écriture
808   TTF_Font* police;
809   police=NULL;
810
811   //Couleur du texte
812   SDL_Color couleur_texte = {0, 0, 0};
813   //Couleur du fond du texte
814   SDL_Color couleur_fond_texte = {225, 255, 225};
815   //Couleur du groupe
816   SDL_Color couleur_groupe = {couleur.rouge, couleur.vert, couleur.bleu};
817
818   //Position
819   SDL_Rect position;
820
821   //Ouverture de la police d'ecriture
822   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 28);
823
824   //Tire de la carte de propriété
825   surf_texte[1] = TTF_RenderText_Shaded(police, texte, couleur_texte, couleur_groupe);
826
```

```
827   //Fermeture de la police d'ecriture
828   TTF_CloseFont(police);
829
830   //Ouverture de la police d'ecriture
831   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
832
833   //Ecriture des informations contenus dans le groupe
834   surf_texte[0] = TTF_RenderText_Shaded(police, "Titre de propriété", couleur_texte, couleur_groupe);
835
836   //Ecriture des textes sur les informations sur les loyers
837   surf_texte[2] = TTF_RenderText_Shaded(police, "LOYER Salle vide", couleur_texte, couleur_fond_texte);
838   surf_texte[3] = TTF_RenderText_Shaded(police, "Niveau CPI 1", couleur_texte, couleur_fond_texte);
839   surf_texte[4] = TTF_RenderText_Shaded(police, "Niveau CPI 2", couleur_texte, couleur_fond_texte);
840   surf_texte[5] = TTF_RenderText_Shaded(police, "Niveau ING 1", couleur_texte, couleur_fond_texte);
841   surf_texte[6] = TTF_RenderText_Shaded(police, "Niveau ING 2", couleur_texte, couleur_fond_texte);
842   surf_texte[7] = TTF_RenderText_Shaded(police, "Niveau ING 3", couleur_texte, couleur_fond_texte);
843
844   //Ecriture de la monnaie utilisé
845   surf_texte[8] = TTF_RenderText_Shaded(police, "Fintz", couleur_texte, couleur_fond_texte);
846
847   //Ecriture des différents prix
848   for(i=0;i<6;i++)
849   {
850     sprintf(texte_temp,"%d",int_prix*(i+1));
851     surf_texte[9+i] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
852   }
853
854   //Fermeture de la police d'ecriture
855   TTF_CloseFont(police);
856
857   //Ouverture de la police d'ecriture
858   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
859
860   //Ecriture des informations sur les loyers
861   surf_texte[15] = TTF_RenderText_Shaded(police, "Si un élève possède toutes les salles", couleur_texte, couleur_fo
862   surf_texte[16] = TTF_RenderText_Shaded(police, "d'un groupe de couleur, le loyer des", couleur_texte, couleur_fon
863   surf_texte[17] = TTF_RenderText_Shaded(police, "salles vides de ce groupe est doublé.", couleur_texte, couleur_fo
864
865   //Ecriture des informations sur l'achat d'un niveau et l'hypothèque
866   sprintf(texte_temp,"Prix d'un niveau                    %d Fintz",int_prix_niveau);
867   surf_texte[18] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
868   surf_texte[19] = TTF_RenderText_Shaded(police, "chacun", couleur_texte, couleur_fond_texte);
869   surf_texte[20] = TTF_RenderText_Shaded(police, "Valeur Hypothécaire de la salle", couleur_texte, couleur_fond_tex
870   sprintf(texte_temp,"%d Fintz",int_prix);
871   surf_texte[21] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
872
873   //Fermeture de la police d'ecriture
874   TTF_CloseFont(police);
875
876   //Création de la surface de la case
877   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_HAUTEUR, 32, 0, 0, 0, 0);
878   //Remplissage de noir pour faire le cadre
879   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
880
881   //Création de la surface de fond
882   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_HAUTEUR-2*DETAIL_EPAISSEUR
883   //Remplissage de vert tapis
884   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 225, 255, 225));
885
886   //Création de la surface du groupe
887   surf_groupe=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_GROUPE_HAUTEUR+2*DETAIL_EPAISSEUR, 32, 0, 0
888   //Remplissage de noir pour faire le cadre
889   SDL_FillRect(surf_groupe, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
890
891   //Création de la surface du groupe
892   surf_groupe_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_GROUPE_HAUTEUR, 32
893   //Remplissage de la couleur du groupe
```

```
894   SDL_FillRect(surf_groupe_fond, NULL, SDL_MapRGB(surf_ecran->format, couleur.rouge, couleur.vert, couleur.bleu));
895
896   //Création de la surface de séparation
897   surf_sep=SDL_CreateRGBSurface(SDL_HWSURFACE, (DETAIL_LARGEUR*9)/10, 3, 32, 0, 0, 0, 0);
898   //Remplissage de noir
899   SDL_FillRect(surf_sep, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
900
901   //On modifie la position du fond
902   position.x=DETAIL_EPAISSEUR;
903   position.y=DETAIL_EPAISSEUR;
904   //On colle le fond du groupe sur ce dernier
905   SDL_BlitSurface(surf_groupe_fond, NULL, surf_groupe, &position);
906
907   //On modifie la position du fond
908   position.x=DETAIL_EPAISSEUR;
909   position.y=DETAIL_EPAISSEUR;
910   //On colle le fond sur la case
911   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
912
913   //On modifie la position du fond
914   position.x=0;
915   position.y=0;
916   //On colle le groupe sur la case
917   SDL_BlitSurface(surf_groupe, NULL, surf_case, &position);
918
919   //Pour le texte contenu dans la coulerur du groupe
920   for(i=0;i<2;i++)
921   {
922     //On centre la position du texte
923     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
924     position.y=10+25*i;
925     //On colle le texte sur la surface
926     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
927   }
928
929   //Modification de la position du texte pour le loyer du terrain nu
930   position.x=20;
931   position.y=105;
932   //Affichage du loyer du terrain nu
933   SDL_BlitSurface(surf_texte[2], NULL, surf_case, &position);
934
935   //Pour les autres informations de la carte
936   for(i=3;i<8;i++)
937   {
938     //On centre la position du texte
939     position.x=95;
940     position.y=55+25*i;
941     //On colle le texte sur la surface
942     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
943   }
944
945   //Pour les informations sur les prix
946   for(i=8;i<15;i++)
947   {
948     //On centre la position du texte
949     position.x=DETAIL_LARGEUR-15-(surf_texte[i]->w);
950     position.y=55+25*(i-7);
951     //On colle le texte sur la surface
952     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
953   }
954
955   //Position du séparateur de texte
956   position.x=DETAIL_LARGEUR/20;
957   position.y=262;
958   //Affichage du séparateur de texte
959   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
960
```

```
961   for(i=15;i<18;i++)
962   {
963     //On centre la position du texte
964     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
965     position.y=90+20*(i-6);
966     //On colle le texte sur la surface
967     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
968   }
969
970   //Position du séparateur de texte
971   position.x=DETAIL_LARGEUR/20;
972   position.y=337;
973   //Affichage du séparateur de texte
974   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
975
976   //Position du texte
977   position.x=10;
978   position.y=345;
979   //On colle le texte sur la surface
980   SDL_BlitSurface(surf_texte[18], NULL, surf_case, &position);
981
982   //Position du texte
983   position.x=(DETAIL_LARGEUR-10-(*surf_texte[19]).w);
984   position.y=365;
985   //On colle le texte sur la surface
986   SDL_BlitSurface(surf_texte[19], NULL, surf_case, &position);
987
988   //Position du texte
989   position.x=(DETAIL_LARGEUR-(*surf_texte[20]).w)/2;
990   position.y=390;
991   //On colle le texte sur la surface
992   SDL_BlitSurface(surf_texte[20], NULL, surf_case, &position);
993
994   //Position du texte
995   position.x=(DETAIL_LARGEUR-(*surf_texte[21]).w)/2;
996   position.y=410;
997   //On colle le texte sur la surface
998   SDL_BlitSurface(surf_texte[21], NULL, surf_case, &position);
999
1000   //Libération des surfaces temporaires
1001   SDL_FreeSurface(surf_fond);
1002   SDL_FreeSurface(surf_groupe_fond);
1003   SDL_FreeSurface(surf_groupe);
1004   for(i=0;i<22;i++) SDL_FreeSurface(surf_texte[i]);
1005
1006   return(surf_case);
1007 }
```

8.11.1.15 SDL_Surface * creation_case_detail_lc (SDL_Surface * *surf_ecran*, int *int_type*)

crÃľer la surface de la case dÃľtaille d'une case lieu commun

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

int_type type de lieu commun

Version :

1.0

Renvoie :

la surface de la case dÃľtaillÃľ

Définition à la ligne 1009 du fichier sdl.c.

Référencé par init_case_lieu_commun().

```
1010 {
1011   SDL_Surface* surf_case;
1012   surf_case=NULL;
1013   SDL_Surface* surf_fond;
1014   surf_fond=NULL;
1015   SDL_Surface* surf_image;
1016   surf_image=NULL;
1017   SDL_Surface* surf_sep;
1018   surf_sep=NULL;
1019   SDL_Surface* surf_texte[14];
1020   char texte_temp[256];
1021   int i;
1022
1023   //Police d'écriture
1024   TTF_Font* police;
1025   police=NULL;
1026
1027   //Couleur du texte
1028   SDL_Color couleur_texte = {0, 0, 0};
1029   //Couleur du fond du texte
1030   SDL_Color couleur_fond_texte = {225, 255, 225};
1031
1032   //Position
1033   SDL_Rect position;
1034
1035   switch(int_type)
1036   {
1037     case LC_WC:
1038       surf_image = IMG_Load("sdl/images/wc.png");
1039       strcpy(texte_temp,"W.C.");
1040       break;
1041     case LC_ASCENSEUR:
1042       surf_image = IMG_Load("sdl/images/ascenseur.png");
1043       strcpy(texte_temp,"Ascenseur");
1044       break;
1045     case LC_RU:
1046       surf_image = IMG_Load("sdl/images/ru.png");
1047       strcpy(texte_temp,"R.U.");
1048       break;
1049     case LC_PARKING:
1050       surf_image = IMG_Load("sdl/images/parking.png");
1051       strcpy(texte_temp,"Parking");
1052       break;
1053     default:
1054       break;
1055   }
1056
1057   //Ouverture de la police d'ecriture
1058   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 28);
1059
1060   //Tire de la carte de propriété
1061   surf_texte[1] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1062
1063   //Fermeture de la police d'ecriture
1064   TTF_CloseFont(police);
1065
1066   //Ouverture de la police d'ecriture
1067   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
1068
1069   //Ecriture des informations contenus dans le groupe
```

```
1070   surf_texte[0] = TTF_RenderText_Shaded(police, "Titre de propriété", couleur_texte, couleur_fond_texte);
1071
1072   //Ecriture des textes sur les informations sur les loyers
1073   surf_texte[2] = TTF_RenderText_Shaded(police, "LOYER si vous avez :", couleur_texte, couleur_fond_texte);
1074   surf_texte[3] = TTF_RenderText_Shaded(police, "1 lieu commun", couleur_texte, couleur_fond_texte);
1075   surf_texte[4] = TTF_RenderText_Shaded(police, "2 lieux commun", couleur_texte, couleur_fond_texte);
1076   surf_texte[5] = TTF_RenderText_Shaded(police, "3 lieux commun", couleur_texte, couleur_fond_texte);
1077   surf_texte[6] = TTF_RenderText_Shaded(police, "4 lieux commun", couleur_texte, couleur_fond_texte);
1078
1079   //Ecriture de la monnaie utilisé
1080   surf_texte[7] = TTF_RenderText_Shaded(police, "Fintz", couleur_texte, couleur_fond_texte);
1081
1082   //Ecriture des différents prix
1083   for(i=0;i<4;i++)
1084   {
1085     sprintf(texte_temp,"%d",250*(i+1));
1086     surf_texte[8+i] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1087   }
1088
1089   //Fermeture de la police d'ecriture
1090   TTF_CloseFont(police);
1091
1092   //Ouverture de la police d'ecriture
1093   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1094
1095   //Ecriture des informations sur l'achat d'un niveau et l'hypothèque
1096   surf_texte[12] = TTF_RenderText_Shaded(police, "Valeur Hypothécaire de la salle", couleur_texte, couleur_fond_te
1097   sprintf(texte_temp,"%d Fintz",1000);
1098   surf_texte[13] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1099
1100   //Fermeture de la police d'ecriture
1101   TTF_CloseFont(police);
1102
1103   //Création de la surface de la case
1104   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_HAUTEUR, 32, 0, 0, 0, 0);
1105   //Remplissage de noir pour faire le cadre
1106   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1107
1108   //Création de la surface de fond
1109   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_HAUTEUR-2*DETAIL_EPAISSEU
1110   //Remplissage de vert tapis
1111   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 225, 255, 225));
1112
1113   //Création de la surface de séparation
1114   surf_sep=SDL_CreateRGBSurface(SDL_HWSURFACE, (DETAIL_LARGEUR*9)/10, 3, 32, 0, 0, 0, 0);
1115   //Remplissage de noir
1116   SDL_FillRect(surf_sep, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1117
1118
1119   //On modifie la position du fond
1120   position.x=DETAIL_EPAISSEUR;
1121   position.y=DETAIL_EPAISSEUR;
1122   //On colle le fond sur la case
1123   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
1124
1125   //Pour le titre
1126   for(i=0;i<2;i++)
1127   {
1128     //On centre la position du texte
1129     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1130     position.y=10+25*i;
1131     //On colle le texte sur la surface
1132     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1133   }
1134
1135   //On centre la position de l'image
1136   position.x=(DETAIL_LARGEUR-(surf_image->w))/2;
```

```
1137   position.y=85;
1138   //On colle l'image sur la surface
1139   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
1140
1141   for(i=2;i<7;i++)
1142   {
1143     //Modification de la position du texte pour le loyer du terrain nu
1144     position.x=20;
1145     position.y=145+25*i;
1146     //Affichage du loyer du terrain nu
1147     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1148   }
1149
1150   //Pour les informations sur les prix
1151   for(i=7;i<12;i++)
1152   {
1153     //On centre la position du texte
1154     position.x=DETAIL_LARGEUR-15-(surf_texte[i]->w);
1155     position.y=145+25*(i-5);
1156     //On colle le texte sur la surface
1157     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1158   }
1159
1160   //Position du séparateur de texte
1161   position.x=DETAIL_LARGEUR/20;
1162   position.y=360;
1163   //Affichage du séparateur de texte
1164   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
1165
1166   //Position du texte
1167   position.x=(DETAIL_LARGEUR-(*surf_texte[12]).w)/2;
1168   position.y=390;
1169   //On colle le texte sur la surface
1170   SDL_BlitSurface(surf_texte[12], NULL, surf_case, &position);
1171
1172   //Position du texte
1173   position.x=(DETAIL_LARGEUR-(*surf_texte[13]).w)/2;
1174   position.y=410;
1175   //On colle le texte sur la surface
1176   SDL_BlitSurface(surf_texte[13], NULL, surf_case, &position);
1177
1178   //Libération des surfaces temporaires
1179   SDL_FreeSurface(surf_fond);
1180   SDL_FreeSurface(surf_image);
1181   for(i=0;i<14;i++) SDL_FreeSurface(surf_texte[i]);
1182
1183   return(surf_case);
1184 }
```

8.11.1.16 creation_case_detail_assoc (SDL_Surface * *surf_ecran*, int *int_type*)

crÃľer la surface de la case dÃľtaille d'une case association

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran
int_type type d'association

Version :

1.0

Renvoie :

la surface de la case dÃľtaillÃľ

Définition à la ligne 1186 du fichier sdl.c.

Référencé par init_case_association().

```
1187 {
1188   SDL_Surface* surf_case;
1189   surf_case=NULL;
1190   SDL_Surface* surf_fond;
1191   surf_fond=NULL;
1192   SDL_Surface* surf_image;
1193   surf_image=NULL;
1194   SDL_Surface* surf_sep;
1195   surf_sep=NULL;
1196   SDL_Surface* surf_texte[10];
1197   char texte_temp[256];
1198   int i;
1199
1200   //Police d'écriture
1201   TTF_Font* police;
1202   police=NULL;
1203
1204   //Couleur du texte
1205   SDL_Color couleur_texte = {0, 0, 0};
1206   //Couleur du fond du texte
1207   SDL_Color couleur_fond_texte = {225, 255, 225};
1208
1209   //Position
1210   SDL_Rect position;
1211
1212   switch(int_type)
1213   {
1214     case BDE:
1215       surf_image = IMG_Load("sdl/images/bureau_bde.png");
1216       strcpy(texte_temp,"B.D.E.");
1217       break;
1218     case BDS:
1219       surf_image = IMG_Load("sdl/images/bureau_bds.png");
1220       strcpy(texte_temp,"B.D.S.");
1221       break;
1222     default:
1223       break;
1224   }
1225
1226   //Ouverture de la police d'ecriture
1227   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 28);
1228
1229   //Tire de la carte de propriété
1230   surf_texte[1] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1231
1232   //Fermeture de la police d'ecriture
1233   TTF_CloseFont(police);
1234
1235   //Ouverture de la police d'ecriture
1236   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
1237
1238   //Ecriture des informations contenus dans le groupe
1239   surf_texte[0] = TTF_RenderText_Shaded(police, "Titre de propriété", couleur_texte, couleur_fond_texte);
1240
1241   //Fermeture de la police d'ecriture
1242   TTF_CloseFont(police);
1243
1244   //Ouverture de la police d'ecriture
1245   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1246
```

```
1247   //Ecriture des textes sur les informations sur les loyers
1248   surf_texte[2] = TTF_RenderText_Shaded(police, "Si l'élève possède UN seul bureau", couleur_texte, couleur_fond_te
1249   surf_texte[3] = TTF_RenderText_Shaded(police, "d'association, le droit de passage est", couleur_texte, couleur_fo
1250   surf_texte[4] = TTF_RenderText_Shaded(police, "40 fois le montant indiqué par les dés", couleur_texte, couleur_fo
1251
1252   //Ecriture des textes sur les informations sur les loyers
1253   surf_texte[5] = TTF_RenderText_Shaded(police, "Si l'élève possède les DEUX bureaux", couleur_texte, couleur_fond_
1254   surf_texte[6] = TTF_RenderText_Shaded(police, "d'association, le droit de passage est", couleur_texte, couleur_fo
1255   surf_texte[7] = TTF_RenderText_Shaded(police, "100 fois le montant indiqué par les dés", couleur_texte, couleur_
1256
1257   //Ecriture des informations sur l'achat d'un niveau et l'hypothèque
1258   surf_texte[8] = TTF_RenderText_Shaded(police, "Valeur Hypothécaire de la salle", couleur_texte, couleur_fond_text
1259   sprintf(texte_temp,"%d Fintz",750);
1260   surf_texte[9] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1261
1262   //Fermeture de la police d'ecriture
1263   TTF_CloseFont(police);
1264
1265   //Création de la surface de la case
1266   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_HAUTEUR, 32, 0, 0, 0, 0);
1267   //Remplissage de noir pour faire le cadre
1268   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1269
1270   //Création de la surface de fond
1271   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_HAUTEUR-2*DETAIL_EPAISSEU
1272   //Remplissage de vert tapis
1273   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 225, 255, 225));
1274
1275   //Création de la surface de séparation
1276   surf_sep=SDL_CreateRGBSurface(SDL_HWSURFACE, (DETAIL_LARGEUR*9)/10, 3, 32, 0, 0, 0, 0);
1277   //Remplissage de noir
1278   SDL_FillRect(surf_sep, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1279
1280
1281   //On modifie la position du fond
1282   position.x=DETAIL_EPAISSEUR;
1283   position.y=DETAIL_EPAISSEUR;
1284   //On colle le fond sur la case
1285   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
1286
1287   //Pour le titre
1288   for(i=0;i<2;i++)
1289   {
1290     //On centre la position du texte
1291     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1292     position.y=10+25*i;
1293     //On colle le texte sur la surface
1294     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1295   }
1296
1297   //On centre la position de l'image
1298   position.x=(DETAIL_LARGEUR-(surf_image->w))/2;
1299   position.y=85;
1300   //On colle l'image sur la surface
1301   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
1302
1303   for(i=2;i<5;i++)
1304   {
1305     //Modification de la position du texte pour le loyer du terrain nu
1306     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1307     position.y=140+20*i;
1308     //Affichage du loyer du terrain nu
1309     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1310   }
1311
1312   //Position du séparateur de texte
1313   position.x=DETAIL_LARGEUR/20;
```

```
1314   position.y=260;
1315   //Affichage du séparateur de texte
1316   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
1317
1318   //Pour les informations sur les prix
1319   for(i=5;i<8;i++)
1320   {
1321     //On centre la position du texte
1322     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1323     position.y=180+20*i;
1324     //On colle le texte sur la surface
1325     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1326   }
1327
1328   //Position du séparateur de texte
1329   position.x=DETAIL_LARGEUR/20;
1330   position.y=360;
1331   //Affichage du séparateur de texte
1332   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
1333
1334   //Position du texte
1335   position.x=(DETAIL_LARGEUR-(*surf_texte[8]).w)/2;
1336   position.y=390;
1337   //On colle le texte sur la surface
1338   SDL_BlitSurface(surf_texte[8], NULL, surf_case, &position);
1339
1340   //Position du texte
1341   position.x=(DETAIL_LARGEUR-(*surf_texte[9]).w)/2;
1342   position.y=410;
1343   //On colle le texte sur la surface
1344   SDL_BlitSurface(surf_texte[9], NULL, surf_case, &position);
1345
1346   //Libération des surfaces temporaires
1347   SDL_FreeSurface(surf_fond);
1348   SDL_FreeSurface(surf_image);
1349   for(i=0;i<10;i++) SDL_FreeSurface(surf_texte[i]);
1350
1351   return(surf_case);
1352 }
```

8.11.1.17 SDL_Surface * creation_case_propriete (SDL_Surface * *surf_ecran*, rvb_couleur *couleur*, char * *texte*)

crÃľer la surface de la propriÃľtÃľ qui s'affichera dans le panneau des propriÃľtÃľs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

couleur couleur du groupe de la case

texte nom de la case

Version :

1.0

Renvoie :

la surface la propriÃľtÃľ

Définition à la ligne 1354 du fichier sdl.c.

Référencé par init_case_association(), init_case_lieu_commun(), et init_case_salle().

```
1355 {
1356   //Surface de la case
1357   SDL_Surface* surf_case;
1358   surf_case=NULL;
1359   //Surface du fond
1360   SDL_Surface* surf_fond;
1361   surf_fond=NULL;
1362   //Surface du texte
1363   SDL_Surface* surf_texte;
1364   surf_texte=NULL;
1365
1366
1367   //Police d'écriture
1368   TTF_Font* police;
1369   police=NULL;
1370
1371   //Couleur du texte
1372   SDL_Color couleur_texte = {0, 0, 0};
1373   //Couleur du groupe
1374   SDL_Color couleur_groupe = {couleur.rouge,couleur.vert,couleur.bleu};
1375
1376   //Position
1377   SDL_Rect position;
1378
1379   //Ouverture de la police d'ecriture
1380   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1381
1382   //Ecriture du texte
1383   surf_texte = TTF_RenderText_Shaded(police, texte, couleur_texte, couleur_groupe);
1384
1385   //Fermeture de la police d'ecriture
1386   TTF_CloseFont(police);
1387
1388   //Création de la surface de la case
1389   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, PROPRIETE_LARGEUR, PROPRIETE_HAUTEUR, 32, 0, 0, 0, 0);
1390   //Remplissage de noir pour faire le cadre
1391   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1392
1393   //Création de la surface de fond
1394   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PROPRIETE_LARGEUR-6, PROPRIETE_HAUTEUR-6, 32, 0, 0, 0, 0);
1395   //Remplissage de la couleur du groupe
1396   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, couleur.rouge,couleur.vert,couleur.bleu));
1397
1398
1399   //On modifie la position du fond
1400   position.x=3;
1401   position.y=3;
1402   //On colle le fond sur la case
1403   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
1404
1405   //Position du texte
1406   position.x=(PROPRIETE_LARGEUR-(surf_texte->w))/2;
1407   position.y=5;
1408   //Collage du texte sur la case
1409   SDL_BlitSurface(surf_texte, NULL, surf_case, &position);
1410
1411   //Libération des surfaces temporaires
1412   SDL_FreeSurface(surf_fond);
1413   SDL_FreeSurface(surf_texte);
1414   //On retourne la case
1415   return(surf_case);
1416 }
```

8.11.1.18 int affich_accueil (SDL_Surface * *surf_ecran*)

affiche la page d'accueil

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãl'cran

Version :

1.0

Renvoie :

une action demandÃl'e du joueur

Définition à la ligne 1418 du fichier sdl.c.

Référencé par main().

```
1419 {
1420   int i;
1421   char char_nombre[16];
1422   int nombre_joueur;
1423   //Surface du numéro
1424   SDL_Surface* surf_nombre[6];
1425   //Surface de la fleche gauche
1426   SDL_Surface* surf_fleche_gauche;
1427   surf_fleche_gauche=NULL;
1428   //Surface de la fleche droite
1429   SDL_Surface* surf_fleche_droite;
1430   surf_fleche_droite=NULL;
1431   //Surface du titre
1432   SDL_Surface* surf_titre;
1433   surf_titre=NULL;
1434   //Surface du texte
1435   SDL_Surface* surf_texte;
1436   surf_texte=NULL;
1437   //Surface des boutons
1438   SDL_Surface* surf_boutons[6];
1439   //Police d'écriture
1440   TTF_Font* police;
1441   police=NULL;
1442   //Position des images
1443   SDL_Rect position;
1444
1445   //Couleur du titre
1446   SDL_Color couleur_titre = {231, 86, 86};
1447   //Couleur du texte
1448   SDL_Color couleur_texte = {0, 155, 126};
1449   //Couleur des nombres
1450   SDL_Color couleur_nombre = {255, 0, 0};
1451
1452   //Remplissage de noir pour faire le fond
1453   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1454
1455   //Chargement de l'image de la fleche gauche
1456   surf_fleche_gauche=IMG_Load("sdl/images/fleche_gauche.gif");
1457   //Chargement de l'image de la fleche droite
1458   surf_fleche_droite=IMG_Load("sdl/images/fleche_droite.gif");
1459
1460   //Création de la surface du bouton valider
```

```
1461   surf_boutons[0]=IMG_Load("sdl/images/suivant.gif");
1462   surf_boutons[1]=IMG_Load("sdl/images/suivant2.gif");
1463   surf_boutons[2]=IMG_Load("sdl/images/quitter.gif");
1464   surf_boutons[3]=IMG_Load("sdl/images/quitter2.gif");
1465   surf_boutons[4]=IMG_Load("sdl/images/charger.gif");
1466   surf_boutons[5]=IMG_Load("sdl/images/charger2.gif");
1467
1468   //Ouverture de la police d'ecriture
1469   police = TTF_OpenFont("sdl/police/police.ttf", 100);
1470
1471   //Ecriture du titre sur la surface de titre
1472   surf_titre = TTF_RenderText_Blended(police, "EISTIOPOLY", couleur_titre);
1473
1474   //Pour chacune des surfaces nombre
1475   for(i=0;i<6;i++)
1476   {
1477     //Enregistrement du numéro
1478     sprintf(char_nombre,"%d",i+1);
1479     //Ecriture du numéro sur la surface
1480     surf_nombre[i] = TTF_RenderText_Blended(police, char_nombre, couleur_nombre);
1481   }
1482
1483   //Fermeture de la police d'ecriture
1484   TTF_CloseFont(police);
1485
1486   //Ouverture de la police d'ecriture
1487   police = TTF_OpenFont("sdl/police/police.ttf", 60);
1488
1489   //Ecriture du texte sur la surface de texte
1490   surf_texte = TTF_RenderText_Blended(police, "Nombre de joueur", couleur_texte);
1491
1492   //Fermeture de la police d'ecriture
1493   TTF_CloseFont(police);
1494
1495   //On modifie la position du titre
1496   position.x=(ECRAN_LARGEUR-(surf_titre->w))/2;
1497   position.y=TITRE_POS_Y;
1498   //On colle le titre sur le fond
1499   SDL_BlitSurface(surf_titre, NULL, surf_ecran, &position);
1500
1501   //Position du texte
1502   position.x=(ECRAN_LARGEUR-(surf_texte->w))/2;
1503   position.y=TEXTE_POS_Y;
1504   //Affichage du texte
1505   SDL_BlitSurface(surf_texte, NULL, surf_ecran, &position);
1506
1507   //On modifie la position du bouton valider
1508   position.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
1509   position.y=BTN_ACCUEIL_POS_Y;
1510   //On colle le bouton valider sur le fond
1511   SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position);
1512
1513   //On modifie la position du bouton chargement
1514   position.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
1515   position.y=BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE;
1516   //On colle le bouton charger sur le fond
1517   SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position);
1518
1519   //On modifie la position du bouton quitter
1520   position.x=ECRAN_LARGEUR-(surf_boutons[2]->w+10);
1521   position.y=10;
1522   //On colle le bouton quitter sur le fond
1523   SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position);
1524
1525   //Fonction d'attente d'événement
1526   nombre_joueur=attente_action_accueil(surf_ecran, surf_boutons, surf_nombre, surf_fleche_gauche, surf_fleche_droit
1527
```

```
1528   //On efface l'écran
1529   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 25, 25, 75));
1530   //Mise à jour de l'écran
1531   SDL_Flip(surf_ecran);
1532
1533   //Libération des surfaces utilisées
1534   for(i=0;i<6;i++) SDL_FreeSurface(surf_nombre[i]);
1535   for(i=0;i<6;i++) SDL_FreeSurface(surf_boutons[i]);
1536   SDL_FreeSurface(surf_fleche_gauche);
1537   SDL_FreeSurface(surf_fleche_droite);
1538   SDL_FreeSurface(surf_titre);
1539
1540   return(nombre_joueur);
1541 }
```

8.11.1.19 int affich_config (SDL_Surface * *surf_ecran*, char ** *str_nom_joueur*, int *nombre_joueur*)

affiche la page de configuration de la partie

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãl'cran

str_nom_joueur tableau des noms des joueurs

nombre_joueur nombre de joueur dans la partie

Version :

1.0

Renvoie :

une action demandÃl'e du joueur

Définition à la ligne 1543 du fichier sdl.c.

Référencé par main().

```
1544 {
1545   int i;
1546   char str_nom[64];
1547   int int_retour;
1548
1549   //Surface d'un champ (sert de cache)
1550   SDL_Surface* surf_champ;
1551   surf_champ=NULL;
1552   //Surfaces des noms
1553   SDL_Surface* surf_nom[nombre_joueur];
1554
1555   //Surface du titre
1556   SDL_Surface* surf_titre;
1557   surf_titre=NULL;
1558   //Surface des boutons
1559   SDL_Surface* surf_boutons[6];
1560   //Police d'écriture
1561   TTF_Font* police;
1562   police=NULL;
1563   //Position des images
```

```
1564   SDL_Rect position;
1565
1566   //Couleur du texte
1567   SDL_Color couleur_titre = {231, 86, 86};
1568   //Couleur des noms
1569   SDL_Color couleur_noms = {255, 0, 0};
1570
1571   //Remplissage de noir pour faire le fond
1572   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1573
1574   //Création des surfaces du bouton retour
1575   surf_boutons[0]=IMG_Load("sdl/images/retour.gif");
1576   surf_boutons[1]=IMG_Load("sdl/images/retour2.gif");
1577   surf_boutons[2]=IMG_Load("sdl/images/jouer.gif");
1578   surf_boutons[3]=IMG_Load("sdl/images/jouer2.gif");
1579   surf_boutons[4]=IMG_Load("sdl/images/quitter.gif");
1580   surf_boutons[5]=IMG_Load("sdl/images/quitter2.gif");
1581
1582   //Ouverture de la police d'ecriture
1583   police = TTF_OpenFont("sdl/police/police.ttf", 100);
1584
1585   //Ecriture du titre sur la surface de titre
1586   surf_titre = TTF_RenderText_Blended(police, "EISTIOPOLY", couleur_titre);
1587
1588   //Fermeture de la police d'ecriture
1589   TTF_CloseFont(police);
1590
1591   //Ouverture de la police d'ecriture
1592   police = TTF_OpenFont("sdl/police/police.ttf", 30);
1593
1594   //Ecriture du numéro sur la surface
1595   surf_champ=SDL_CreateRGBSurface(SDL_HWSURFACE, CHAMP_LARGEUR, CHAMP_HAUTEUR, 32, 0, 0, 0, 0);
1596   //Remplissage du bouton valider
1597   SDL_FillRect(surf_champ, NULL, SDL_MapRGB(surf_ecran->format, 204, 204, 204));
1598
1599   //Pour chacune des surfaces nombre
1600   for(i=0;i<nombre_joueur;i++)
1601   {
1602     //Enregistrement du numéro
1603     sprintf(str_nom,"Joueur %d :",i+1);
1604     surf_nom[i] = TTF_RenderText_Blended(police, str_nom, couleur_noms);
1605   }
1606
1607   //Fermeture de la police d'ecriture
1608   TTF_CloseFont(police);
1609
1610   //On modifie la position du titre
1611   position.x=TITRE_POS_X;
1612   position.y=TITRE_POS_Y;
1613   //On colle le titre sur le fond
1614   SDL_BlitSurface(surf_titre, NULL, surf_ecran, &position);
1615
1616   //On modifie la position du bouton valider
1617   position.x=BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w;
1618   position.y=BTN_ACCUEIL_POS_Y;
1619   //On colle le bouton valider sur le fond
1620   SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position);
1621
1622   //On modifie la position du bouton retour
1623   position.x=BTN_ACCUEIL_POS_X;
1624   position.y=BTN_ACCUEIL_POS_Y;
1625   //On colle le bouton retour sur le fond
1626   SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position);
1627
1628   //On modifie la position du bouton quitter
1629   position.x=ECRAN_LARGEUR-(surf_boutons[4]->w+10);
1630   position.y=10;
```

```
1631   //On colle le bouton quitter sur le fond
1632   SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position);
1633
1634   for(i=0;i<nombre_joueur;i++)
1635   {
1636     position.x=CHAMP_POS_X;
1637     position.y=CHAMP_POS_Y+i*2*CHAMP_HAUTEUR;
1638     SDL_BlitSurface(surf_champ, NULL, surf_ecran, &position);
1639     position.x=NOM_CHAMP_POS_X;
1640     position.y=NOM_CHAMP_POS_Y+i*2*CHAMP_HAUTEUR;
1641     SDL_BlitSurface(surf_nom[i], NULL, surf_ecran, &position);
1642   }
1643
1644   //Fonction d'attente d'événement
1645   int_retour=attente_action_config(surf_ecran, str_nom_joueur, surf_boutons, surf_champ, nombre_joueur);
1646
1647   //On efface l'écran
1648   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 25, 25, 75));
1649   //Mise à jour de l'écran
1650   SDL_Flip(surf_ecran);
1651
1652   //Libération des surfaces utilisées
1653   for(i=0;i<nombre_joueur;i++)
1654   {
1655     SDL_FreeSurface(surf_nom[i]);
1656   }
1657   SDL_FreeSurface(surf_titre);
1658   for(i=0;i<6;i++) SDL_FreeSurface(surf_boutons[i]);
1659   return(int_retour);
1660 }
```

8.11.1.20 void affich_case (SDL_Surface * *surf_ecran*, cases * *pcase*)

affiche une case Ãă l'Ãľcran

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

pcase case Ãă afficher

Version :

1.0

Renvoie :

++

Définition à la ligne 1662 du fichier sdl.c.

Référencé par aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), jeu(), et reculer_jeton().

```
1663 {
1664   //Surface de l'image cpi
1665   SDL_Surface* surf_image_cpi;
1666   surf_image_cpi=NULL;
1667
1668   //Surface de l'image ingé
```

```
1669  SDL_Surface* surf_image_inge;
1670  surf_image_inge=NULL;
1671
1672  //Surface de la case temporaire
1673  SDL_Surface* surf_case;
1674  surf_case=NULL;
1675
1676  //Position
1677  SDL_Rect position;
1678
1679  int int_niveau;
1680
1681  //Chargement de l'image cpi
1682  surf_image_cpi=IMG_Load("sdl/images/cpi.png");
1683
1684  //Chargement de l'image ingé
1685  surf_image_inge=IMG_Load("sdl/images/inge.png");
1686
1687  //Création de la surface temporaire de la case en fonction de la position de la case
1688  if(pcase->int_type>=SP_APPARTEMENT && pcase->int_type<=SP_TABLEAU) surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE,
1689  else if(pcase->int_position==POSITION_GAUCHE || pcase->int_position==POSITION_DROITE) surf_case=SDL_CreateRGBSur
1690  else surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
1691
1692  //Position à 0, 0
1693  position.x=0;
1694  position.y=0;
1695  //Collage de la case sur la surface case temporaire
1696  SDL_BlitSurface((*pcase).surf_image, NULL, surf_case, &position);
1697
1698  //S'il s'agit d'une case salle qui a un niveau supérieur à zéro
1699  if(pcase->int_type==SALLE && pcase->case_salle.int_niveau!=0)
1700  {
1701
1702    //On initalisation le niveau à 0
1703    int_niveau=1;
1704
1705    if(pcase->case_salle.int_niveau<=2)
1706    {
1707      //Selon la position de la case
1708      switch(pcase->int_position)
1709      {
1710        case POSITION_HAUT:
1711          //On tourne l'image de 180 degré
1712          surf_image_cpi=rotation_180(surf_image_cpi);
1713          break;
1714        case POSITION_DROITE:
1715          //On tourne l'image de 90 degré
1716          surf_image_cpi=rotation_90(surf_image_cpi);
1717          break;
1718        case POSITION_GAUCHE:
1719          //On tourne l'image de 270 degré
1720          surf_image_cpi=rotation_270(surf_image_cpi);
1721          break;
1722        default:
1723          //On ne fait rien
1724          break;
1725      }
1726
1727      //Tant que l'image à afficher fait partit du groupe cpi et tant que l'on a pas atteint le niveau de la salle
1728      while(int_niveau<=2 && int_niveau<=(pcase->case_salle.int_niveau))
1729      {
1730        switch(pcase->int_position)
1731        {
1732          case POSITION_HAUT:
1733            //On modifie la position du fond
1734            position.x=(CASE_EPAISSEUR+20*(int_niveau-1));
1735            position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
```

```
1736            break;
1737          case POSITION_DROITE:
1738            //On modifie la position du fond
1739            position.x=CASE_EPAISSEUR;
1740            position.y=CASE_EPAISSEUR+20*(int_niveau-1);
1741            break;
1742          case POSITION_GAUCHE:
1743            //On modifie la position du fond
1744            position.x=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1745            position.y=(CASE_EPAISSEUR+20*(int_niveau-1));
1746            break;
1747          default:
1748            //On modifie la position du fond
1749            position.x=CASE_EPAISSEUR+20*(int_niveau-1);
1750            position.y=CASE_EPAISSEUR;
1751            break;
1752        }
1753        //On colle le fond sur la case
1754        SDL_BlitSurface(surf_image_cpi, NULL, surf_case, &position);
1755        int_niveau++;
1756      }
1757    }
1758    else
1759    {
1760      //Selon la position de la case
1761      switch(pcase->int_position)
1762      {
1763        case POSITION_HAUT:
1764          //On tourne l'image de 180 degré
1765          surf_image_inge=rotation_180(surf_image_inge);
1766          break;
1767        case POSITION_DROITE:
1768          //On tourne l'image de 90 degré
1769          surf_image_inge=rotation_90(surf_image_inge);
1770          break;
1771        case POSITION_GAUCHE:
1772          //On tourne l'image de 270 degré
1773          surf_image_inge=rotation_270(surf_image_inge);
1774          break;
1775        default:
1776          //On ne fait rien
1777          break;
1778      }
1779
1780      //Tant que l'image à afficher fait partit du groupe cpi et tant que l'on a pas atteint le niveau de la salle
1781      while(int_niveau<=3 && int_niveau<=(pcase->case_salle.int_niveau)-2)
1782      {
1783        switch(pcase->int_position)
1784        {
1785          case POSITION_HAUT:
1786            //On modifie la position du fond
1787            position.x=(CASE_EPAISSEUR+20*(int_niveau-1));
1788            position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1789            break;
1790          case POSITION_DROITE:
1791            //On modifie la position du fond
1792            position.x=CASE_EPAISSEUR;
1793            position.y=CASE_EPAISSEUR+20*(int_niveau-1);
1794            break;
1795          case POSITION_GAUCHE:
1796            //On modifie la position du fond
1797            position.x=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1798            position.y=(CASE_EPAISSEUR+20*(int_niveau-1));
1799            break;
1800          default:
1801            //On modifie la position du fond
1802            position.x=CASE_EPAISSEUR+20*(int_niveau-1);
```

```
1803            position.y=CASE_EPAISSEUR;
1804            break;
1805        }
1806        //On colle le fond sur la case
1807        SDL_BlitSurface(surf_image_inge, NULL, surf_case, &position);
1808        int_niveau++;
1809      }
1810    }
1811  }
1812
1813  //Position de la case sur l'écran
1814  position.x=(*pcase).rect_coordonnees.x;
1815  position.y=(*pcase).rect_coordonnees.y;
1816
1817  //Affichage de la case
1818  SDL_BlitSurface(surf_case, NULL, surf_ecran, &position);
1819
1820  //Libération des surfaces temporaire
1821  SDL_FreeSurface(surf_image_inge);
1822  SDL_FreeSurface(surf_image_cpi);
1823  SDL_FreeSurface(surf_case);
1824 }
```

8.11.1.21 void affich_case_detail (SDL_Surface * *surf_ecran*, cases * *pcase*)

affiche le dÃľtail d'une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

pcase case dont on doti afficher le dÃľtail

Version :

1.0

Renvoie :

++

Définition à la ligne 1826 du fichier sdl.c.

```
1827 {
1828   SDL_Rect position;
1829   position.x=POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_LARGEUR)/2;
1830   position.y=POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_HAUTEUR)/2;
1831   SDL_BlitSurface(pcase->case_salle.surf_detail, NULL, surf_ecran, &position); // Collage de la surface sur l'écra
1832 }
```

8.11.1.22 void affich_centre (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*)

affiche le centre du plateau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'Ãľcran

surf_ centre surface du centre

Version :

1.0

Renvoie :

++

Définition à la ligne 1834 du fichier sdl.c.

Référencé par attente_validation_message(), attente_validation_propriete(), et jeu().

```
1835 {
1836   SDL_Rect position;
1837   position.x=POS_X_PLATEAU+CASE_HAUTEUR;
1838   position.y=POS_Y_PLATEAU+CASE_HAUTEUR;
1839   SDL_BlitSurface(surf_centre, NULL, surf_ecran, &position); // Collage de la surface sur l'écran
1840 }
```

8.11.1.23 void affich_panneau_menu (SDL_Surface * *surf_ ecran*, int *int_ etat_ bouton*)

affiche le panneau du menu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'Ãľcran

int_ etat_ bouton Ãľtat des boutons

Version :

1.0

Renvoie :

++

Définition à la ligne 1842 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
1843 {
1844   //Surface du panneau
1845   SDL_Surface* surf_menu;
1846   //Surface du fond du panneau
1847   SDL_Surface* surf_fond;
1848   //Surface des boutons images
1849   SDL_Surface* surf_objet[3];
1850   //Position
1851   SDL_Rect position;
1852   int i;
1853
1854   //Création de la surface du menu
```

```
1855   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_MENU_LARGEUR, PANNEAU_MENU_HAUTEUR, 32, 0, 0, 0, 0);
1856   //Remplissage de noir pour faire le cadre
1857   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1858
1859   //Création de la surface du fond du menu
1860   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_MENU_LARGEUR-6, PANNEAU_MENU_HAUTEUR-6, 32, 0, 0, 0, 0);
1861   //Remplissage de noir pour faire le cadre
1862   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
1863
1864   //Chargement des boutons
1865   if(int_etat_bouton==1) surf_objet[0]=IMG_Load("sdl/images/quitter2.gif");
1866   else surf_objet[0]=IMG_Load("sdl/images/quitter.gif");
1867   if(int_etat_bouton==2) surf_objet[1]=IMG_Load("sdl/images/sauvegarder2.png");
1868   else surf_objet[1]=IMG_Load("sdl/images/sauvegarder.png");
1869   if(int_etat_bouton==3) surf_objet[2]=IMG_Load("sdl/images/tourner.png");
1870   else surf_objet[2]=IMG_Load("sdl/images/tourner.png");
1871
1872   //Positon du fond
1873   position.x=3;
1874   position.y=3;
1875   //Collage du fond sur l'image
1876   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
1877
1878   //Pour chacun des boutons
1879   for(i=0;i<3;i++)
1880   {
1881     //On position le bouton
1882     position.x=(PANNEAU_OBJET_TAILLE+30)*i+30;
1883     position.y=10;
1884     //Affichage du bouton
1885     SDL_BlitSurface(surf_objet[i], NULL, surf_menu, &position);
1886   }
1887
1888   //Position du panneau sur l'écran
1889   position.x=PANNEAU_MENU_POS_X;
1890   position.y=PANNEAU_MENU_POS_Y;
1891   //Affichage du panneau
1892   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
1893
1894   //Mise à jour de l'écran
1895   SDL_Flip(surf_ecran);
1896
1897   //Libération des surfaces temporaires
1898   SDL_FreeSurface(surf_menu);
1899   SDL_FreeSurface(surf_fond);
1900   for(i=0;i<3;i++) SDL_FreeSurface(surf_objet[i]);
1901 }
```

8.11.1.24 void affich_panneau_joueur (SDL_Surface * *surf_ecran*, joueur * *j_anneau_joueurs*)

affiche le panneau d'information sur le joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran
j_anneau_joueurs joueur en cours

Version :

1.0

Renvoie :

++

Définition à la ligne 1903 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
1904 {
1905   SDL_Surface* surf_menu;
1906   SDL_Surface* surf_fond;
1907   SDL_Surface* surf_texte[5];
1908
1909   SDL_Rect position_jeton;
1910
1911   char str_temp[256];
1912   int i;
1913
1914   SDL_Rect position;
1915
1916   //Police d'écriture
1917   TTF_Font* police;
1918   police=NULL;
1919
1920   //Couleur du texte
1921   SDL_Color couleur_texte = {86, 255, 86};
1922   SDL_Color couleur_valeur = {125, 225, 125};
1923
1924   //Création de la surface du menu
1925   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_JOUEUR_LARGEUR, PANNEAU_JOUEUR_HAUTEUR, 32, 0, 0, 0, 0);
1926   //Remplissage de noir pour faire le cadre
1927   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1928
1929   //Création de la surface du fond du menu
1930   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_JOUEUR_LARGEUR-6, PANNEAU_JOUEUR_HAUTEUR-6, 32, 0, 0, 0, 0)
1931   //Remplissage de noir pour faire le cadre
1932   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
1933
1934   //Ouverture de la police d'ecriture
1935   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1936
1937   //Ecriture du texte "nom du joueur"
1938   surf_texte[0] = TTF_RenderText_Blended(police, "Nom du joueur :", couleur_texte);
1939
1940   //Ecriture du nom du joueur
1941   surf_texte[1] = TTF_RenderText_Blended(police, j_anneau_joueurs->str_nom, couleur_valeur);
1942
1943   sprintf(str_temp,"%d Fintz",j_anneau_joueurs->int_argent);
1944   //Ecriture du texte "argent"
1945   surf_texte[2] = TTF_RenderText_Blended(police, "Argent :", couleur_texte);
1946
1947   //Ecriture de la quantité d'argent du joueur
1948   surf_texte[3] = TTF_RenderText_Blended(police, str_temp, couleur_valeur);
1949
1950   sprintf(str_temp,"Nombre de certificat : %d",j_anneau_joueurs->int_certificat);
1951   //Ecriture du nombre de certficat
1952   surf_texte[4] = TTF_RenderText_Blended(police, str_temp, couleur_texte);
1953
1954   //Fermeture de la police d'ecriture
1955   TTF_CloseFont(police);
1956
1957   //on affiche l'image du jeton
1958   position_jeton.x=130;
1959   position_jeton.y=25;
1960   SDL_BlitSurface(j_anneau_joueurs->surf_image, NULL, surf_fond, &position_jeton);
1961
1962   position.x=3;
1963   position.y=3;
```

```
1964   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
1965
1966   for(i=0;i<5;i++)
1967   {
1968     position.x=PANNEAU_JOUEUR_POS_X+5+(15*(i%2));
1969     position.y=PANNEAU_JOUEUR_POS_Y-95+25*i-5*(i%2);
1970     SDL_BlitSurface(surf_texte[i], NULL, surf_menu, &position);
1971   }
1972
1973   position.x=PANNEAU_JOUEUR_POS_X;
1974   position.y=PANNEAU_JOUEUR_POS_Y;
1975   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
1976
1977   SDL_Flip(surf_ecran);
1978   SDL_FreeSurface(surf_menu);
1979   SDL_FreeSurface(surf_fond);
1980   for(i=0;i<3;i++) SDL_FreeSurface(surf_texte[i]);
1981 }
```

8.11.1.25 void affich_panneau_possessions (SDL_Surface * *surf_ecran*, joueur * *j_anneau_joueurs*)

affiche le panneau des possessions du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

j_anneau_joueurs joueur en cours

Version :

1.0

Renvoie :

++

Définition à la ligne 1983 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
1984 {
1985   //Surface du panneau
1986   SDL_Surface* surf_menu;
1987   surf_menu=NULL;
1988   SDL_Surface* surf_fond;
1989   surf_fond=NULL;
1990   SDL_Surface* surf_texte;
1991   surf_texte=NULL;
1992   SDL_Rect position;
1993   SDL_Surface* surf_propriete;
1994   surf_propriete=NULL;
1995   //Police d'écriture
1996   TTF_Font* police;
1997   police=NULL;
1998   possession* propriete;
1999   int nombre_propriete;
2000   int propriete_actuel;
2001   SDL_Color couleur_texte = {86, 255, 86};
```

```
2002   propriete=j_anneau_joueurs->propriete;
2003   propriete_actuel=0;
2004   nombre_propriete=0;
2005
2006   if(!propriete)
2007   {
2008     //Création de la surface du menu
2009     surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR, PANNEAU_POSSESSION_HAUTEUR, 32, 0, 0
2010     //Remplissage de noir pour faire le cadre
2011     SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2012
2013     //Création de la surface du fond du menu
2014     surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR-6, PANNEAU_POSSESSION_HAUTEUR-6, 32, 0
2015     //Remplissage de noir pour faire le cadre
2016     SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2017
2018     //Ouverture de la police d'ecriture
2019     police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
2020
2021     //Ecriture du texte "nom du joueur"
2022     surf_texte = TTF_RenderText_Blended(police, "Aucun titre de propriété.", couleur_texte);
2023
2024     //Fermeture de la police d'ecriture
2025     TTF_CloseFont(police);
2026
2027     position.x=3;
2028     position.y=3;
2029     SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2030
2031   }
2032   else
2033   {
2034     //Pour chaque propriete
2035     while(propriete)
2036     {
2037       //On ajoute 1 au nombre de propriété
2038       nombre_propriete++;
2039       //On passe à la propriété suivante
2040       propriete=propriete->suivant;
2041     }
2042     //Création de la surface du panneau
2043     surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR, 40+((PROPRIETE_HAUTEUR+5)*nombre_prop
2044     //Remplissage de noir pour faire le cadre
2045     SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2046
2047     //Création de la surface du fond du panneau
2048     surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR-6, 40+((PROPRIETE_HAUTEUR+5)*nombre_pr
2049     //Remplissage de noir pour faire le cadre
2050     SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2051
2052     //Ouverture de la police d'ecriture
2053     police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
2054
2055     //Ecriture du texte
2056     surf_texte = TTF_RenderText_Blended(police, "Propriété de l'élève :", couleur_texte);
2057
2058     //Fermeture de la police d'ecriture
2059     TTF_CloseFont(police);
2060
2061     //Position du fond sur le menu
2062     position.x=3;
2063     position.y=3;
2064     //Collage du fond sur le menu
2065     SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2066
2067     //Retour à l'entete de la liste des propriétés du joueur
2068     propriete=j_anneau_joueurs->propriete;
```

```
2069
2070     //tant qu'on a des propriétés
2071     while(propriete)
2072     {
2073       //Chargement de la surface en fonction de la propriété
2074       if(propriete->propriete->int_type==SALLE) surf_propriete=propriete->propriete->case_salle.surf_propriete;
2075       else if(propriete->propriete->int_type==BDE || propriete->propriete->int_type==BDS) surf_propriete=propriete-
2076       else surf_propriete=propriete->propriete->case_lieu_commun.surf_propriete;
2077
2078       //Position de la propriété dans le menu
2079       position.x=10;
2080       position.y=35+propriete_actuel*(PROPRIETE_HAUTEUR+5);
2081
2082       //Affichage de la propriété
2083       SDL_BlitSurface(surf_propriete, NULL, surf_menu, &position);
2084       //Passage à la propriété suivante
2085       propriete=propriete->suivant;
2086       //Incrémentation du nombre de propriété actuel
2087       propriete_actuel++;
2088     }
2089   }
2090
2091   //Position du texte sur le menu
2092   position.x=20;
2093   position.y=10;
2094   //Collage du texte sur le menu
2095   SDL_BlitSurface(surf_texte, NULL, surf_menu, &position);
2096
2097   //Position du panneau à l'écran
2098   position.x=PANNEAU_POSSESSION_POS_X;
2099   position.y=PANNEAU_POSSESSION_POS_Y;
2100   //Affichage du panneau à l'écran
2101   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2102
2103   //Mise à jour de l'écran
2104   SDL_Flip(surf_ecran);
2105
2106   //Libération des surfaces temporaires
2107   SDL_FreeSurface(surf_menu);
2108   SDL_FreeSurface(surf_fond);
2109   SDL_FreeSurface(surf_texte);
2110 }
2111
```

8.11.1.26 void affich_possessions_cache (SDL_Surface * *surf_ecran*)

affiche le cache du panneau de possesion du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

Version :

1.0

Renvoie :

++

Définition à la ligne 2113 du fichier sdl.c.

Référencé par jeu().

```
2113 {
2114   //Surface du cache
2115   SDL_Surface* surf_cache;
2116   surf_cache=NULL;
2117
2118   //Position
2119   SDL_Rect position;
2120
2121   //Création de la surface du menu
2122   surf_cache=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR, ECRAN_HAUTEUR-PANNEAU_POSSESSION_POS_Y
2123   //Remplissage de noir pour faire le cadre
2124   SDL_FillRect(surf_cache, NULL, SDL_MapRGB(surf_ecran->format, 25, 25, 75));
2125
2126   //Position du cache
2127   position.x=PANNEAU_POSSESSION_POS_X;
2128   position.y=PANNEAU_POSSESSION_POS_Y;
2129
2130   //Collage du cache sur l'écran
2131   SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
2132
2133 }
2134
```

8.11.1.27 void affich_panneau_des_bouton (SDL_Surface * *surf_ecran*, int *int_image*)

affiche le panneau contenant le bouton lancer dÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

int_image score du dÃľ

Version :

1.0

Renvoie :

++

Définition à la ligne 2136 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
2136 {
2137   //Surface du panneau
2138   SDL_Surface* surf_menu;
2139   //Surface du fond
2140   SDL_Surface* surf_fond;
2141   //Surface des boutons lancer dés
2142   SDL_Surface* surf_bouton[2];
2143   //Position
2144   SDL_Rect position;
2145
2146   int i;
```

```
2147
2148   //Création de la surface du menu
2149   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR, PANNEAU_DES_HAUTEUR, 32, 0, 0, 0, 0);
2150   //Remplissage de noir pour faire le cadre
2151   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2152
2153   //Création de la surface du fond du menu
2154   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR-6, PANNEAU_DES_HAUTEUR-6, 32, 0, 0, 0, 0);
2155   //Remplissage de noir pour faire le cadre
2156   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2157
2158   //Chargement des images bes boutons
2159   surf_bouton[0]=IMG_Load("sdl/images/lancer.gif");
2160   surf_bouton[1]=IMG_Load("sdl/images/lancer2.gif");
2161
2162   //Position du fond sur le panneau
2163   position.x=3;
2164   position.y=3;
2165   //Collage du fond sur le panneau
2166   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2167
2168   //Position du bouton sur le panneau
2169   position.x=(PANNEAU_DES_LARGEUR-surf_bouton[0]->w)/2;
2170   position.y=(PANNEAU_DES_HAUTEUR-surf_bouton[0]->h)/2;
2171   //Collage du bouton sur le panneau
2172   SDL_BlitSurface(surf_bouton[int_image], NULL, surf_menu, &position);
2173
2174   //Position du panneau sur l'écran
2175   position.x=PANNEAU_DES_POS_X;
2176   position.y=PANNEAU_DES_POS_Y;
2177   //Affichage du panneau
2178   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2179
2180   //Mise à jour de l'écran
2181   SDL_Flip(surf_ecran);
2182
2183   //Libération des surfaces temporaires
2184   SDL_FreeSurface(surf_menu);
2185   SDL_FreeSurface(surf_fond);
2186   for(i=0;i<2;i++) SDL_FreeSurface(surf_bouton[i]);
2187 }
2188
```

8.11.1.28 SDL_Surface ** creation_des (void)

renvoit un tableau de surface contenant chacune des faces du dÃ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Version :

1.0

Renvoie :

le tableau des surfaces des faces du dÃ

Définition à la ligne 2190 du fichier sdl.c.

Référencé par lancer_des().

```
2190 {
```

```
2191   //Surface du dé
2192   SDL_Surface** surf_des;
2193   if((surf_des= new SDL_Surface*[6])==NULL) cout<<"Fesse en boite"<<endl;
2194   //Création de la surface de l'objet nř1
2195   surf_des[0]=IMG_Load("sdl/images/un.png");
2196   surf_des[1]=IMG_Load("sdl/images/deux.png");
2197   surf_des[2]=IMG_Load("sdl/images/trois.png");
2198   surf_des[3]=IMG_Load("sdl/images/quatre.png");
2199   surf_des[4]=IMG_Load("sdl/images/cinq.png");
2200   surf_des[5]=IMG_Load("sdl/images/six.png");
2201   //On retourne la surface du dé
2202   return(surf_des);
2203 }
2204
```

8.11.1.29 void destruction_des (SDL_Surface ** *surf_des*)

dÃľtruit les surfaces des dÃľs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_des tableau des surfaces des faces d'un dÃľ

Version :

1.0

Renvoie :

++

Définition à la ligne 2206 du fichier sdl.c.

Référencé par lancer_des().

```
2206 {
2207   int i;
2208   //Libération des surfaces des dés
2209   for(i=0;i<5;i++) SDL_FreeSurface(surf_des[i]);
2210   //Libération de la mémoire
2211   delete(surf_des);
2212 }
2213
```

8.11.1.30 void affich_panneau_fdt (SDL_Surface * *surf_ecran*, bool *bool_etat*)

affiche le panneau du menu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

bool_etat Ãľtat des boutons

Version :

1.0

Renvoie :

++

Définition à la ligne 2215 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
2215 {
2216   //Surface du panneau
2217   SDL_Surface* surf_menu;
2218   //Surface du fond du panneau
2219   SDL_Surface* surf_fond;
2220   //Surface du bouton de fin de tour
2221   SDL_Surface* surf_bouton;
2222   //Position
2223   SDL_Rect position;
2224
2225   //Création de la surface du menu
2226   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_FDT_LARGEUR, PANNEAU_FDT_HAUTEUR, 32, 0, 0, 0, 0);
2227   //Remplissage de noir pour faire le cadre
2228   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2229
2230   //Création de la surface du fond du menu
2231   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_FDT_LARGEUR-6, PANNEAU_FDT_HAUTEUR-6, 32, 0, 0, 0, 0);
2232   //Remplissage de noir pour faire le cadre
2233   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2234
2235   //Bouton en surbrillance
2236   if(bool_etat==true) surf_bouton=IMG_Load("sdl/images/finir2.gif");
2237   //Bouton normal
2238   else surf_bouton=IMG_Load("sdl/images/finir.gif");
2239
2240   //Position du fond sur le panneau
2241   position.x=3;
2242   position.y=3;
2243   //Collage du fond sur le panneau
2244   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2245
2246   //Position du bouton
2247   position.x=(PANNEAU_FDT_LARGEUR-surf_bouton->w)/2;
2248   position.y=(PANNEAU_FDT_HAUTEUR-surf_bouton->h)/2;
2249   //Collage du bouton sur le panneau
2250   SDL_BlitSurface(surf_bouton, NULL, surf_menu, &position);
2251
2252   //Position du panneau à l'écran
2253   position.x=PANNEAU_FDT_POS_X;
2254   position.y=PANNEAU_FDT_POS_Y;
2255   //Collage du panneau sur l'écran
2256   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2257
2258   //Mise à jour de l'écran
2259   SDL_Flip(surf_ecran);
2260
2261   //Libération des surfaces temporaires
2262   SDL_FreeSurface(surf_menu);
2263   SDL_FreeSurface(surf_fond);
2264   SDL_FreeSurface(surf_bouton);
2265 }
2266
```

8.11.1.31 void affich_panneau_des (SDL_Surface * *surf_ecran*, SDL_Surface ** *surf_des*, int *int_de1*, int *int_de2*)

affiche le panneau du menu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

surf_des tableau des surfaces des face du dÃľ

int_de1 score du dÃľ numÃľro 1

int_de2 score du dÃľ numÃľro 2

Version :

1.0

Renvoie :

++

Définition à la ligne 2268 du fichier sdl.c.

Référencé par lancer_des().

```
2268 {
2269   //Surface du menu
2270   SDL_Surface* surf_menu;
2271   //Surface du fond
2272   SDL_Surface* surf_fond;
2273 
2274   //Position
2275   SDL_Rect position;
2276 
2277   //Création de la surface du menu
2278   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR, PANNEAU_DES_HAUTEUR, 32, 0, 0, 0, 0);
2279   //Remplissage de noir pour faire le cadre
2280   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2281 
2282   //Création de la surface du fond du menu
2283   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR-6, PANNEAU_DES_HAUTEUR-6, 32, 0, 0, 0, 0);
2284   //Remplissage de noir pour faire le cadre
2285   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2286 
2287   //Position du fond
2288   position.x=3;
2289   position.y=3;
2290   //Affichage du fond
2291   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2292 
2293   //Position du premier dé
2294   position.x=40;
2295   position.y=(PANNEAU_DES_HAUTEUR-surf_des[0]->h)/2;
2296   //Affichage du premier dé
2297   SDL_BlitSurface(surf_des[int_de1-1], NULL, surf_menu, &position);
2298 
2299   //Position du second dé
2300   position.x=130;
2301   position.y=(PANNEAU_DES_HAUTEUR-surf_des[0]->h)/2;
2302   //Affichage du second dé
2303   SDL_BlitSurface(surf_des[int_de2-1], NULL, surf_menu, &position);
```

```
2304
2305   //Position du menu
2306   position.x=PANNEAU_DES_POS_X;
2307   position.y=PANNEAU_DES_POS_Y;
2308   //Affichage du menu
2309   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2310
2311   //Mise à jour de l'écran
2312   SDL_Flip(surf_ecran);
2313
2314   //Libération des surfaces temporaires
2315   SDL_FreeSurface(surf_menu);
2316   SDL_FreeSurface(surf_fond);
2317 }
2318
```

8.11.1.32 void affich_joueur_depart (SDL_Surface * *surf_ecran*, cases ** *plateau*, joueur * *j_anneau_joueurs*, int *nombre_joueur*)

affiche tout les joueurs sur la case de dÃľpart

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

plateau plateau de jeu

j_anneau_joueurs liste chaÃőnÃľe des joueurs

nombre_joueur nombre de joueur

Version :

1.0

Renvoie :

action ud joueur

Définition à la ligne 2320 du fichier sdl.c.

Référencé par jeu().

```
2320 {
2321   //pour chaque joueurs
2322   for (int i=0;i<nombre_joueur;i++)
2323   {
2324     //on l'affiche sur la case
2325     affich_joueur(surf_ecran,j_anneau_joueurs,i, plateau[j_anneau_joueurs->int_position]);
2326
2327     //on passe au joueur suivant
2328     j_anneau_joueurs=j_anneau_joueurs->pjoueur_suivant;
2329   }
2330   //on depart au premier joueur pour que le jeu puisse commencer avec
2331   j_anneau_joueurs=j_anneau_joueurs->pjoueur_suivant;
2332 }
2333
```

8.11.1.33 int affich_validation_propriete (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_fond*, int *int_etat*, int *int_nombre_boutons*, ...)

affiche le message des propriÃľtÃľ ainsi que les boutons d'actions possibles

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

surf_fond surface de la propriÃľtÃľ

int_etat Ãľtat des boutons

int_nombre_boutons nombre de boutons Ãă afficher

... liste des boutons Ãă afficher

Version :

1.0

Renvoie :

un entier d'Ãľtat

Définition à la ligne 2335 du fichier sdl.c.

Référencé par action_to_boutons().

```
2335 {
2336   //Surface du bouton
2337   SDL_Surface** surf_bouton;
2338 
2339   int i;
2340 
2341   //Nombre de bouton
2342   int nombre_boutons;
2343   nombre_boutons=int_nombre_boutons;
2344 
2345   //Si jamais il n'y a pas de boutons
2346   if(nombre_boutons<=0) return(0);
2347 
2348   //Un crée un tableau de bouton correspondant au nombres de boutons et à leurs états
2349   surf_bouton= new SDL_Surface*[nombre_boutons];
2350 
2351   //Position
2352   SDL_Rect position;
2353 
2354   //Valeur de l'argument
2355   int int_valeur;
2356 
2357   //Liste des arguments
2358   va_list arguments;
2359 
2360   //Initialisation de la liste des arguments
2361   va_start(arguments, int_nombre_boutons);
2362 
2363   //Position du message à l'écran
2364   position.x=DETAIL_MESSAGE_POS_X;
2365   position.y=DETAIL_MESSAGE_POS_Y;
2366   //Affichage du message
2367   SDL_BlitSurface(surf_fond, NULL, surf_ecran, &position); // Collage de la surface sur l'écran
2368 
2369   //Tant qu'on a des boutons
```

```
2370  while(nombre_boutons!=0)
2371  {
2372    //Récupération de l'argument
2373    int_valeur=va_arg(arguments, int);
2374    //Selon le bouton à afficher
2375    switch(int_valeur)
2376    {
2377      //Dans le cas d'un achat
2378      case BTN_ACHAT:
2379        //Charger l'image en surbrillance
2380        if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/achete
2381        //Ou l'image normale
2382        else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/acheter.gif");
2383        break;
2384      //Dans le cas d'un hypothèque
2385      case BTN_HYPOTHEQUE:
2386        //Charger l'image en surbrillance
2387        if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/hypoth
2388        //Ou l'image normale
2389        else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/hypotheque.gif");
2390        break;
2391      //Dans le cas d'une déshypothèque
2392      case BTN_UNHYPOTHEQUE:
2393        //Charger l'image en surbrillance
2394        if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/deshyp
2395        //Ou l'image normale
2396        else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/deshypotheque.gif");
2397        break;
2398      case BTN_MOINS:
2399        //Charger l'image en surbrillance
2400        if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/moins2
2401        //Ou l'image normale
2402        else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/moins.gif");
2403        break;
2404      case BTN_PLUS:
2405        //Charger l'image en surbrillance
2406        if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[0]=IMG_Load("sdl/images/plus2.gif");
2407        //Ou l'image normale
2408        else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/plus.gif");
2409        break;
2410      case BTN_FINIR:
2411        //Charger l'image en surbrillance
2412        if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[0]=IMG_Load("sdl/images/annuler2.gif");
2413        //Ou l'image normale
2414        else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/annuler.gif");
2415        break;
2416    }
2417    //Passage au bouton suivant
2418    nombre_boutons--;
2419    //Position du bouton
2420    position.x=DETAIL_BOUTON_POS_X;
2421    position.y=DETAIL_BOUTON_POS_Y+50*(int_nombre_boutons-(nombre_boutons+1));
2422    //On colle le bouton sur le message
2423    SDL_BlitSurface(surf_bouton[nombre_boutons], NULL, surf_ecran, &position);
2424  }
2425
2426  //Destruction de la liste d'argument
2427  va_end(arguments);
2428
2429  //Mise à jour de l'écran
2430  SDL_Flip(surf_ecran);
2431
2432  //Libération des surfaces temporaire
2433  for(i=0;i<int_nombre_boutons;i++) SDL_FreeSurface(surf_bouton[i]);
2434  //Libération de la mémoire
2435  delete[] surf_bouton;
2436
```

```
2437   //On retourne 0
2438   return(0);
2439 }
2440
```

8.11.1.34 SDL_Surface * creation_message (SDL_Surface * *surf_ecran*, char * *titre*, char ** *message*, int *int_type_message*, int *int_nbre_ligne*)

crÃľer la surface d'un message Ãă afficher

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

titre titre du message

message message Ãă afficher

int_type_message type du message Ãă afficher

int_nbre_ligne nombre de ligne du message

Version :

1.0

Renvoie :

la surface du message

Définition à la ligne 2442 du fichier sdl.c.

Référencé par attente_validation_message().

```
2442 {
2443   //Surface du message
2444   SDL_Surface* surf_message;
2445   surf_message=NULL;
2446
2447   //Surface du fond du message
2448   SDL_Surface* surf_fond;
2449   surf_fond=NULL;
2450
2451   //Surface du texte
2452   SDL_Surface* surf_texte[int_nbre_ligne];
2453
2454   //Surface du titre
2455   SDL_Surface* surf_titre;
2456   surf_titre=NULL;
2457
2458   //Police d'écriture
2459   TTF_Font* police;
2460   police=NULL;
2461
2462   int i;
2463
2464   //Couleur du texte
2465   SDL_Color couleur_texte = {0, 0, 0};
2466   //Couleur du groupe
2467   SDL_Color couleur_fond_texte = {0,0,0};
2468
```

```
2469  //Selon le type de message, on change la couleur de fond du message
2470  switch(int_type_message)
2471  {
2472    //Dans le cas d'un message de Krystel
2473    case MESSAGE_KRYSTEL:
2474      couleur_fond_texte.r=85;
2475      couleur_fond_texte.g=115;
2476      couleur_fond_texte.b=215;
2477      break;
2478    //Dans le cas d'un message de Nadège
2479    case MESSAGE_NADEGE:
2480      couleur_fond_texte.r=215;
2481      couleur_fond_texte.g=85;
2482      couleur_fond_texte.b=45;
2483      break;
2484    //Dans tout les autres cas
2485    default:
2486      couleur_fond_texte.r=185;
2487      couleur_fond_texte.g=235;
2488      couleur_fond_texte.b=140;
2489      break;
2490  }
2491
2492  //Position
2493  SDL_Rect position;
2494
2495  //Si il y à un titre
2496  if(titre)
2497  {
2498    //Ouverture de la police d'ecriture
2499    police = TTF_OpenFont("sdl/police/police2-bold.ttf", 30);
2500
2501    //Ecriture du titre
2502    surf_titre = TTF_RenderText_Shaded(police, titre, couleur_texte, couleur_fond_texte);
2503
2504    //Fermeture de la police d'ecriture
2505    TTF_CloseFont(police);
2506  }
2507
2508  //Ouverture de la police d'ecriture
2509  police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
2510
2511  //Ecriture du texte
2512  for(i=0;i<int_nbre_ligne;i++) surf_texte[i] = TTF_RenderText_Shaded(police, message[i], couleur_texte, couleur_fo
2513
2514  //Fermeture de la police d'ecriture
2515  TTF_CloseFont(police);
2516
2517  //Création de la surface du message
2518  surf_message=SDL_CreateRGBSurface(SDL_HWSURFACE, MESSAGE_LARGEUR, 180+(int_nbre_ligne*20), 32, 0, 0, 0, 0);
2519  //Remplissage de noir pour faire le cadre
2520  SDL_FillRect(surf_message, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2521
2522  //Création de la surface de fond
2523  surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, MESSAGE_LARGEUR-6, 180+(int_nbre_ligne*20)-6, 32, 0, 0, 0, 0);
2524  //Remplissage de la couleur du groupe
2525  SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, couleur_fond_texte.r, couleur_fond_texte.g, couleur_
2526
2527  //On modifie la position du fond
2528  position.x=3;
2529  position.y=3;
2530  //On colle le fond sur la case
2531  SDL_BlitSurface(surf_fond, NULL, surf_message, &position);
2532
2533  //Si il y a un titre
2534  if(titre)
2535  {
```

```
2536     //Position du titre
2537     position.x=position.x=(MESSAGE_LARGEUR-(surf_titre->w))/2;;
2538     position.y=20;
2539     //Affichage du titre
2540     SDL_BlitSurface(surf_titre, NULL, surf_message, &position);
2541   }
2542
2543   //Pour chaque ligne du texte
2544   for(i=0;i<int_nbre_ligne;i++)
2545   {
2546     //Position de la ligne de texte
2547     position.x=(MESSAGE_LARGEUR-(surf_texte[i]->w))/2;
2548     position.y=60+20*i;
2549     //Affichage de la ligne du texte
2550     SDL_BlitSurface(surf_texte[i], NULL, surf_message, &position);
2551   }
2552
2553   //Libération des surfaces temporaires
2554   SDL_FreeSurface(surf_fond);
2555   for(i=0;i<int_nbre_ligne;i++) SDL_FreeSurface(surf_texte[i]);
2556
2557   //Si il y a un titre, on libère la surface du titre
2558   if(titre) SDL_FreeSurface(surf_titre);
2559
2560
2561   //On retourne la surface du message
2562   return(surf_message);
2563
2564 }
2565
```

8.11.1.35 void affich_message (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_message*, int *int_type_message*, int *int_etat*)

affiche un message Ãă l'Ãľcran

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

surf_message surface du message

int_type_message type du message Ãă afficher

int_etat Ãľtat des boutons

Version :

1.0

Renvoie :

rien

Définition à la ligne 2568 du fichier sdl.c.

Référencé par attente_validation_message().

```
2568 {
2569   //Surface du bouton
2570   SDL_Surface* surf_bouton[3];
```

```
2571
2572   int i;
2573
2574   //Position
2575   SDL_Rect position;
2576
2577   //Initialisation des trois boutons
2578   for(i=0;i<3;i++) surf_bouton[i]=NULL;
2579
2580   //S'il s'agit d'un message quitter
2581   if(int_type_message==MESSAGE_QUITTER)
2582   {
2583     //Chargement de l'image en surbrillance
2584     if(int_etat==1) surf_bouton[0]=IMG_Load("sdl/images/exit2.gif");
2585     //En normal
2586     else surf_bouton[0]=IMG_Load("sdl/images/exit.gif");
2587     //Chargement de l'image en surbrillance
2588     if(int_etat==2) surf_bouton[1]=IMG_Load("sdl/images/sauvegarder2.gif");
2589     //En normal
2590     else surf_bouton[1]=IMG_Load("sdl/images/sauvegarder.gif");
2591
2592     //Position du bouton à l'écran
2593     position.x=(MESSAGE_LARGEUR-(surf_bouton[0]->w)*2)/3;
2594     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2595     //Affichage du bouton
2596     SDL_BlitSurface(surf_bouton[0], NULL, surf_message, &position);
2597
2598     //Position du bouton à l'écran
2599     position.x=2*(MESSAGE_LARGEUR-(surf_bouton[0]->w))/3;
2600     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2601     //Affichage du bouton
2602     SDL_BlitSurface(surf_bouton[1], NULL, surf_message, &position);
2603   }
2604   //S'il s'agit d'un message prison
2605   else if(int_type_message==MESSAGE_PRISON)
2606   {
2607     //Chargement de l'image en surbrillance
2608     if(int_etat==1) surf_bouton[0]=IMG_Load("sdl/images/attendre2.gif");
2609     //En normal
2610     else surf_bouton[0]=IMG_Load("sdl/images/attendre.gif");
2611     //Chargement de l'image en surbrillance
2612     if(int_etat==2) surf_bouton[1]=IMG_Load("sdl/images/payer2.gif");
2613     //En normal
2614     else surf_bouton[1]=IMG_Load("sdl/images/payer.gif");
2615     //Chargement de l'image en surbrillance
2616     if(int_etat==3) surf_bouton[2]=IMG_Load("sdl/images/certificat2.gif");
2617     //En normal
2618     else surf_bouton[2]=IMG_Load("sdl/images/certificat.gif");
2619
2620     //Position du bouton à l'écran
2621     position.x=30;
2622     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2623     //Affichage du bouton
2624     SDL_BlitSurface(surf_bouton[0], NULL, surf_message, &position);
2625
2626     //Position du bouton à l'écran
2627     position.x=(MESSAGE_LARGEUR-(surf_bouton[2]->w))/2;
2628     position.y=MESSAGE_HAUTEUR-(surf_bouton[2]->h);
2629     //Affichage du bouton
2630     SDL_BlitSurface(surf_bouton[1], NULL, surf_message, &position);
2631
2632     //Position du bouton à l'écran
2633     position.x=MESSAGE_LARGEUR-30-(surf_bouton[2]->w);
2634     position.y=MESSAGE_HAUTEUR-(surf_bouton[2]->h);
2635     //Affichage du bouton
2636     SDL_BlitSurface(surf_bouton[2], NULL, surf_message, &position);
2637   }
```

```
2638   //S'il s'agit d'un message normal
2639   else
2640   {
2641     //Chargement de l'image en surbrillance
2642     if(int_etat==1) surf_bouton[0]=IMG_Load("sdl/images/valider2.gif");
2643     //En normal
2644     else surf_bouton[0]=IMG_Load("sdl/images/valider.gif");
2645     //Position du bouton à l'écran
2646     position.x=(MESSAGE_LARGEUR-(surf_bouton[0]->w))/2;
2647     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2648     //Affichage du bouton
2649     SDL_BlitSurface(surf_bouton[0], NULL, surf_message, &position);
2650   }
2651
2652   //Position du message à l'écran
2653   position.x=POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_LARGEUR)/2;
2654   position.y=POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_HAUTEUR)/2;
2655   //Affichage du message à l'écran
2656   SDL_BlitSurface(surf_message, NULL, surf_ecran, &position);
2657   //Mise à jour de l'écran
2658   SDL_Flip(surf_ecran);
2659
2660   //Libération des surfaces temporaires
2661   SDL_FreeSurface(surf_bouton[0]);
2662   if(int_type_message==MESSAGE_QUITTER || int_type_message==MESSAGE_PRISON) SDL_FreeSurface(surf_bouton[1]);
2663   if(int_type_message==MESSAGE_PRISON) SDL_FreeSurface(surf_bouton[2]);
2664 }
2665
```

8.12 Référence du fichier sdl.h

Fonctions

- SDL_Surface * **init_sdl** (void)
  fonction d'initialisation de sdl

- void **mapause** (void)
- SDL_Surface * **rotation_90** (SDL_Surface *Depart)
  permet de tourner une image de 90 degrÃ©

- SDL_Surface * **rotation_180** (SDL_Surface *Depart)
  permet de tourner une image de 180 degrÃ©

- SDL_Surface * **rotation_270** (SDL_Surface *Depart)
  permet de tourner une image de 270 degrÃ©

- SDL_Surface * **creation_joueur** (int int_joueur)
  crÃ©er la surface du joueur

- SDL_Surface * **creation_case** (SDL_Surface *surf_ecran, **rvb_couleur** couleur, char *texte1, char *texte2, int int_prix, int int_position)
  crÃ©er une case de type salle

- SDL_Surface * **creation_case_bureau** (SDL_Surface *ecran, int int_type, int int_position)
  crÃ©er une case de type bureau

- SDL_Surface * **creation_case_lieu_commun** (SDL_Surface *ecran, int int_type)
  crÃ©er une case de type lieu commun

- SDL_Surface * **creation_case_association** (SDL_Surface *ecran, int int_type)
  crÃ©er une case de type association

- SDL_Surface * **creation_case_soiree** (SDL_Surface *ecran, int int_type)
  crÃ©er une case de type soirÃ©e

- SDL_Surface * **creation_case_coin** (SDL_Surface *ecran, int int_type)
  crÃ©er une case spÃ©cial (chacun des coins)

- SDL_Surface * **creation_case_detail** (SDL_Surface *surf_ecran, **rvb_couleur** couleur, char *texte, int int_prix, int int_prix_niveau)
  crÃ©er la surface de la case dÃ©taillÃ©

- SDL_Surface * **creation_case_detail_lc** (SDL_Surface *surf_ecran, int int_type)
  crÃ©er la surface de la case dÃ©taille d'une case lieu commun

- SDL_Surface * **creation_case_detail_assoc** (SDL_Surface *surf_ecran, int int_type)
  crÃ©er la surface de la case dÃ©taille d'une case association

- SDL_Surface * **creation_case_propriete** (SDL_Surface *surf_ecran, **rvb_couleur** couleur, char *texte)
  crÃ©er la surface de la propriÃ©tÃ© qui s'affichera dans le panneau des propriÃ©tÃ©s

- SDL_Surface ** **creation_des** (void)
  renvoit un tableau de surface contenant chacune des faces du dÃ©

- SDL_Surface * **creation_message** (SDL_Surface *surf_ecran, char *titre, char **message, int int_type_message, int int_nbre_ligne)

crÃľer la surface d'un message Ãă afficher

- void **destruction_des** (SDL_Surface **surf_des)
  dÃľtruit les surfaces des dÃľs

- void **affich_joueur** (SDL_Surface *surf_ecran, **joueur** *pj_joueur, int int_position, **cases** *pcase)
  affiche un joueur Ãă l'Ãľcran

- void **affich_case** (SDL_Surface *surf_ecran, **cases** *pcase)
  affiche une case Ãă l'Ãľcran

- void **affich_case_detail** (SDL_Surface *surf_ecran, **cases** *pcase)
  affiche le dÃľtail d'une case

- void **affich_centre** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre)
  affiche le centre du plateau

- int **affich_accueil** (SDL_Surface *surf_ecran)
  affiche la page d'accueil

- int **affich_config** (SDL_Surface *surf_ecran, char **str_nom_joueur, int nombre_joueur)
  affiche la page de configuration de la partie

- void **affich_panneau_menu** (SDL_Surface *surf_ecran, int int_etat_bouton)
  affiche le panneau du menu

- void **affich_panneau_joueur** (SDL_Surface *surf_ecran, **joueur** *j_anneau_joueurs)
  affiche le panneau d'information sur le joueur

- void **affich_panneau_possessions** (SDL_Surface *surf_ecran, **joueur** *j_anneau_joueurs)
  affiche le panneau des possessions du joueur

- void **affich_possessions_cache** (SDL_Surface *surf_ecran)
  affiche le cache du panneau de possesion du joueur

- void **affich_panneau_des_bouton** (SDL_Surface *surf_ecran, int int_image)
  affiche le panneau contenant le bouton lancer dÃľ

- void **affich_panneau_des** (SDL_Surface *surf_ecran, SDL_Surface **surf_des, int int_de1, int int_de2)
  affiche le panneau du menu

- void **affich_panneau_fdt** (SDL_Surface *surf_ecran, bool bool_etat)
  affiche le panneau du menu

- int **affich_validation_propriete** (SDL_Surface *surf_ecran, SDL_Surface *surf_fond, int int_etat, int int_nombre_boutons,...)
  affiche le message des propriÃľtÃľ ainsi que les boutons d'actions possibles

- void **affich_joueur_depart** (SDL_Surface *surf_ecran, **cases** **plateau, **joueur** *j_anneau_joueurs, int nombre_joueur)
  affiche tout les joueurs sur la case de dÃľpart

- void **affich_message** (SDL_Surface *surf_ecran, SDL_Surface *surf_message, int int_type_message, int int_etat)
  affiche un message Ãă l'Ãľcran

8.12.1 Documentation des fonctions

8.12.1.1 SDL_Surface* init_sdl (void)

fonction d'initialisation de sdl

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Version :

1.0

Renvoie :

la surface de l'Ãľcran

Définition à la ligne 86 du fichier sdl.c.

Référencé par main().

```
87 {
88   //Surface de l'ecran
89   SDL_Surface *ecran=NULL;
90 
91   //Initialisation en mode video de sdl
92   SDL_Init(SDL_INIT_VIDEO);
93 
94   //Répétition des touches
95   SDL_EnableKeyRepeat(80,80);
96 
97   //Initialisation du mode TTF (Pour l'ecriture)
98   TTF_Init();
99 
100   //Chargement du mode video en resolution 800*600 et 32 couleur, plein ecran.
101   ecran = SDL_SetVideoMode(1280, 1024, 32, SDL_HWSURFACE|SDL_FULLSCREEN);
102 
103   //Chargement du titre de la fenetre
104   SDL_WM_SetCaption("Monopoly", NULL);
105 
106   //On remplit le fond de noir
107   SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 25, 25, 75));
108 
109   //On retourne la surface de l'écran
110   return(ecran);
111 }
```

8.12.1.2 void mapause (void)

Définition à la ligne 4 du fichier sdl.c.

```
5 {
6     int continuer = 1;
7     SDL_Event event;
8     while (continuer)
9     {
10         SDL_WaitEvent(&event);
11         switch(event.type)
12         {
13             case SDL_QUIT:
14                 continuer = 0;
15         break;
```

```
16       case SDL_KEYDOWN:
17         switch(event.key.keysym.sym)
18         {
19           case SDLK_ESCAPE:
20             continuer = 0;
21             break;
22           default:
23           break;
24         }
25         break;
26       default:
27       break;
28         }
29     }
30 }
```

8.12.1.3 SDL_Surface* rotation_90 (SDL_Surface * *Depart*)

permet de tourner une image de 90 degrÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

Depart surface de dÃľpart

Version :

1.0

Renvoie :

la surface tournÃľe

Définition à la ligne 32 du fichier sdl.c.

Référencé par affich_case(), creation_case(), creation_case_bureau(), creation_case_lieu_commun(), et creation_case_soiree().

```
33 {
34   //La surface d'arrivee
35   SDL_Surface* arrivee;
36   arrivee = SDL_CreateRGBSurface(SDL_HWSURFACE,Depart->h,Depart->w,Depart->format->BitsPerPixel,0,0,0,0);
37 
38   int x;
39   int y;
40 
41   for(y=0;y<Depart->h;y+=1)
42   {
43     for(x=0;x<Depart->w;x+=1)
44     {
45       ((Uint32*)arrivee->pixels)[y+((arrivee->h-1)-x)*arrivee->w]=((Uint32*)Depart->pixels)[x+y*Depart->w];
46     }
47   }
48   return(arrivee);
49 }
```

8.12.1.4 SDL_Surface* rotation_180 (SDL_Surface * *Depart*)

permet de tourner une image de 180 degrÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

Depart surface de dÃľpart

Version :

1.0

Renvoie :

la surface tournÃľe

Définition à la ligne 51 du fichier sdl.c.

Référencé par affich_case(), creation_case(), creation_case_association(), creation_case_bureau(), et creation_case_lieu_commun().

```
52 {
53   // La surface d'arrivee
54   SDL_Surface* arrivee;
55   arrivee = SDL_CreateRGBSurface(SDL_HWSURFACE,Depart->w,Depart->h,Depart->format->BitsPerPixel,0,0,0,0);
56 
57   int x;
58   int y;
59   for(y=0;y<Depart->h;y+=1)
60   {
61     for(x=0;x<Depart->w;x+=1)
62     {
63       ((Uint32*)arrivee->pixels)[((arrivee->w-1)-x)+((arrivee->h-1)-y)*arrivee->w]=((Uint32*)Depart->pixels)[x+y*Depa
64     }
65   }
66   return(arrivee);
67 }
```

8.12.1.5 SDL_Surface* rotation_270 (SDL_Surface * *Depart*)

permet de tourner une image de 270 degrÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

Depart surface de dÃľpart

Version :

1.0

Renvoie :

la surface tournÃľe

Définition à la ligne 69 du fichier sdl.c.

Référencé par affich_case(), creation_case(), creation_case_association(), creation_case_bureau(), et creation_case_lieu_commun().

```
70 {
71   // La surface d'arrivee
72   SDL_Surface* arrivee;
73   arrivee = SDL_CreateRGBSurface(SDL_HWSURFACE,Depart->h,Depart->w,Depart->format->BitsPerPixel,0,0,0,0);
74
75   int x,y;
76   for(y=0;y<Depart->h;y+=1)
77   {
78     for(x=0;x<Depart->w;x+=1)
79     {
80       ((Uint32*)arrivee->pixels)[((arrivee->w-1)-y)+x*arrivee->w]=((Uint32*)Depart->pixels)[x+y*Depart->w];
81     }
82   }
83   return(arrivee);
84 }
```

8.12.1.6 SDL_Surface* creation_joueur (int *int_joueur*)

crÃľer la surface du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

int_joueur numÃľro du joueur

Version :

1.0

Renvoie :

la surface du joueur

Définition à la ligne 113 du fichier sdl.c.

Référencé par init_joueur(), et init_joueur_chargement().

```
114 {
115   //Surface du joueur
116   SDL_Surface* surf_joueur;
117   surf_joueur=NULL;
118   //Chemin d'accès à l'image
119   char chemin[256];
120
121   //Stockage du chemin vers l'image du joueur en fonction de son numéro
122   sprintf(chemin,"sdl/images/joueur%d.png",int_joueur);
123
124   //Chargement de l'image
125   surf_joueur=IMG_Load(chemin);
126
127   //On retourne l'image du joueur
128   return(surf_joueur);
129
130 }
```

8.12.1.7 SDL_Surface* creation_case (SDL_Surface * *surf_ecran*, rvb_couleur *couleur*, char * *texte1*, char * *texte2*, int *int_prix*, int *int_position*)

crÃľer une case de type salle

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

couleur couleur du groupe

texte1 premiÃĺre partie du nom de la case

texte2 seconde partie du nom de la case

int_prix prix de la case

int_position cotÃľ sur lequel se trouve la case

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 164 du fichier sdl.c.

Référencé par init_case_salle().

```
165 {
166   //Chaine de caractere du prix
167   char texte_prix[255];
168 
169   //Surface de la case
170   SDL_Surface* surf_case;
171   surf_case=NULL;
172 
173   //Surface du fond de la case
174   SDL_Surface* surf_fond;
175   surf_fond=NULL;
176 
177   //Surface de la couleur du groupe
178   SDL_Surface* surf_groupe;
179   surf_groupe=NULL;
180 
181   //Surface du fond du groupe
182   SDL_Surface* surf_fond_groupe;
183   surf_fond_groupe=NULL;
184 
185   //Surface du texte partie 1
186   SDL_Surface* surf_texte1;
187   surf_texte1=NULL;
188 
189   //Surface du texte partie 2
190   SDL_Surface* surf_texte2;
191   surf_texte2=NULL;
192 
193   //Surface du texte partie 2
194   SDL_Surface* surf_texte_prix;
195   surf_texte_prix=NULL;
196 
197   //Police d'écriture
198   TTF_Font* police;
199   police=NULL;
200 
201   //Couleur du texte
202   SDL_Color couleur_texte = {0, 0, 0};
203   //Couleur du fond du texte
```

```
204   SDL_Color couleur_fond_texte = {16, 246, 128};
205
206   //Position
207   SDL_Rect position;
208
209   //Ouverture de la police d'ecriture
210   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
211
212   //Ecriture de la première partie du texte
213   surf_texte1 = TTF_RenderText_Shaded(police, texte1, couleur_texte, couleur_fond_texte);
214   //Ecriture de la seconde partie du texte
215   surf_texte2 = TTF_RenderText_Shaded(police, texte2, couleur_texte, couleur_fond_texte);
216   //Enregistrement du prix
217   sprintf(texte_prix,"%d Fintz",int_prix);
218   //Ecriture du prix
219   surf_texte_prix = TTF_RenderText_Shaded(police, texte_prix, couleur_texte, couleur_fond_texte);
220
221   //Fermeture de la police d'ecriture
222   TTF_CloseFont(police);
223
224   //Création de la surface de la case
225   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
226   //Remplissage de noir pour faire le cadre
227   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
228
229   //Création de la surface de fond
230   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
231   //Remplissage de vert tapis
232   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 16, 246, 128));
233
234   //Création de la surface du groupe
235   surf_groupe=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_GROUPE_HAUTEUR+2*CASE_EPAISSEUR, 32, 0, 0, 0, 0
236   //Remplissage de noir pour faire le cadre
237   SDL_FillRect(surf_groupe, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
238
239   //Création de la surface de fond du groupe
240   surf_fond_groupe=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_GROUPE_HAUTEUR, 32, 0, 0
241   //Remplissage de la couleur de fond du groupe
242   SDL_FillRect(surf_fond_groupe, NULL, SDL_MapRGB(surf_ecran->format, couleur.rouge, couleur.vert, couleur.bleu));
243
244   //On modifie la position du fond du groupe
245   position.x=CASE_EPAISSEUR;
246   position.y=CASE_EPAISSEUR;
247   //On colle le fond du groupe sur ce dernier
248   SDL_BlitSurface(surf_fond_groupe, NULL, surf_groupe, &position);
249
250   //On modifie la position du fond
251   position.x=CASE_EPAISSEUR;
252   position.y=CASE_EPAISSEUR;
253   //On colle le fond sur la case
254   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
255
256   //On modifie la position du groupe
257   position.x=0;
258   position.y=0;
259   //On colle le groupe sur la case
260   SDL_BlitSurface(surf_groupe, NULL, surf_case, &position);
261
262   //On centre la position de la premiere partie texte
263   position.x=(CASE_LARGEUR-(*surf_texte1).w)/2;
264   position.y=30;
265   //On colle la première partie du texte
266   SDL_BlitSurface(surf_texte1, NULL, surf_case, &position);
267
268   //On centre la position de la deuxieme partie du texte
269   position.x=(CASE_LARGEUR-(*surf_texte2).w)/2;
270   position.y=50;
```

```
271   //On colle la deuxième partie du texte
272   SDL_BlitSurface(surf_texte2, NULL, surf_case, &position);
273
274   //On centre la position du prix
275   position.x=(CASE_LARGEUR-(*surf_texte_prix).w)/2;
276   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
277   //On colle le prix
278   SDL_BlitSurface(surf_texte_prix, NULL, surf_case, &position);
279
280   //Selon la position
281   switch(int_position)
282   {
283     //Si la case est à droite
284     case POSITION_DROITE:
285       //Rotation de l'image de 90ř
286       surf_case = rotation_90(surf_case);
287       break;
288     //Si la case est en haut
289     case POSITION_HAUT:
290       //Rotation de l'image de 180ř
291       surf_case = rotation_180(surf_case);
292       break;
293     //Si la case est à gauche
294     case POSITION_GAUCHE:
295       //Rotation de l'image de 270ř
296       surf_case = rotation_270(surf_case);
297       break;
298     //Par défaut, la case est en bas
299     default:
300     break;
301   }
302
303   //Libération des surfaces temporaires
304   SDL_FreeSurface(surf_fond);
305   SDL_FreeSurface(surf_fond_groupe);
306   SDL_FreeSurface(surf_groupe);
307   SDL_FreeSurface(surf_texte1);
308   SDL_FreeSurface(surf_texte2);
309   SDL_FreeSurface(surf_texte_prix);
310
311   //On retourne la surface de la case
312   return(surf_case);
313 }
```

8.12.1.8 SDL_Surface* creation_case_bureau (SDL_Surface * *ecran*, int *int_type*, int *int_position*)

crÃľer une case de type bureau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de bureau

int_position cÃŕtÃľ sur lequel se trouve la case

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 315 du fichier sdl.c.

Référencé par init_case_administration().

```
316 {
317   //Surface du logo
318   SDL_Surface* surf_logo;
319   surf_logo=NULL;
320   //Surface du fond de la case
321   SDL_Surface* surf_fond;
322   surf_fond=NULL;
323   //Surface de la case
324   SDL_Surface* surf_case;
325   surf_case=NULL;
326   //Surface de la première partie du texte
327   SDL_Surface* surf_texte1;
328   surf_texte1=NULL;
329   //Surface de la deuxième partie du texte
330   SDL_Surface* surf_texte2;
331   surf_texte2=NULL;
332   //Position des images
333   SDL_Rect position;
334
335   //Police d'écriture
336   TTF_Font* police;
337   police=NULL;
338
339   //Couleur du texte
340   SDL_Color couleur_texte = {0, 0, 0};
341   //Couleur du fond du texte
342   SDL_Color couleur_fond_texte = {16, 246, 128};
343
344   //Ouvertue de la police d'écriture
345   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
346
347   //Ecriture du premier texte
348   surf_texte1 = TTF_RenderText_Shaded(police, "Bureau de", couleur_texte, couleur_fond_texte);
349   //Ecriture du second texte
350   if(int_type==BUREAU_KRYSTEL) surf_texte2 = TTF_RenderText_Shaded(police, "Krystel", couleur_texte, couleur_fond_te
351   else surf_texte2 = TTF_RenderText_Shaded(police, "Nadege", couleur_texte, couleur_fond_texte);
352
353   //Fermeture de la police d'écriture
354   TTF_CloseFont(police);
355
356   //Création de la surface de la case
357   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
358   //Remplissage de noir pour faire le cadre
359   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
360
361   //Création de la surface de fond
362   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
363   //Remplissage de vert tapis
364   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
365
366   //Chargelent de l'image du logo
367   if(int_type==BUREAU_KRYSTEL) surf_logo = IMG_Load("sdl/images/eisti_logo.png");
368   else surf_logo = IMG_Load("sdl/images/eisti_logo2.png");
369
370   //On modifie la position du fond
371   position.x=CASE_EPAISSEUR;
372   position.y=CASE_EPAISSEUR;
373   //On colle le fond sur la case
374   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
375
```

```
376   //On centre la position de la première partie texte
377   position.x=(CASE_LARGEUR-(*surf_texte1).w)/2;
378   position.y=5;
379   //On colle la première partie du texte
380   SDL_BlitSurface(surf_texte1, NULL, surf_case, &position);
381
382   //On centre la position de la deuxième partie du texte
383   position.x=(CASE_LARGEUR-(*surf_texte2).w)/2;
384   position.y=25;
385   //On colle la deuxième partie du texte
386   SDL_BlitSurface(surf_texte2, NULL, surf_case, &position);
387
388   //On positione le logo
389   position.x=16;
390   position.y=45;
391   //On colle le logo sur la case
392   SDL_BlitSurface(surf_logo, NULL, surf_case, &position);
393
394   //Selon la position
395   switch(int_position)
396   {
397     //Si la case est à droite
398     case POSITION_DROITE:
399       //Rotation de l'image de 90ř
400       surf_case = rotation_90(surf_case);
401       break;
402     //Si la case est en haut
403     case POSITION_HAUT:
404       //Rotation de l'image de 180ř
405       surf_case = rotation_180(surf_case);
406       break;
407     //Si la case est à gauche
408     case POSITION_GAUCHE:
409       //Rotation de l'image de 270ř
410       surf_case = rotation_270(surf_case);
411       break;
412     //Par défaut, la case est en bas
413     default:
414     break;
415   }
416
417   //Libération des surfaces temporaires
418   SDL_FreeSurface(surf_fond);
419   SDL_FreeSurface(surf_logo);
420   SDL_FreeSurface(surf_texte1);
421   SDL_FreeSurface(surf_texte2);
422   //On retourne la case que l'on vient de créer
423   return(surf_case);
424 }
```

8.12.1.9 SDL_Surface* creation_case_lieu_commun (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case de type lieu commun

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de lieu commun

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 426 du fichier sdl.c.

Référencé par init_case_lieu_commun().

```
427 {
428   //Surface de l'image
429   SDL_Surface* surf_image;
430   surf_image=NULL;
431   //Surface du fond de la case
432   SDL_Surface* surf_fond;
433   surf_fond=NULL;
434   //Surface de la case
435   SDL_Surface* surf_case;
436   surf_case=NULL;
437   //Surface du prix
438   SDL_Surface* surf_prix;
439   surf_prix=NULL;
440
441   //Position des images
442   SDL_Rect position;
443
444   //Police d'écriture
445   TTF_Font* police;
446   police=NULL;
447
448   //Couleur du texte
449   SDL_Color couleur_texte = {0, 0, 0};
450   //Couleur du fond du texte
451   SDL_Color couleur_fond_texte = {16, 246, 128};
452
453   //Ouvertue de la police d'écriture
454   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
455
456   //Ecriture du premier texte
457   surf_prix = TTF_RenderText_Shaded(police, "2000 Fintz", couleur_texte, couleur_fond_texte);
458
459   //Fermeture de la police d'écriture
460   TTF_CloseFont(police);
461
462   //Création de la surface de la case
463   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
464   //Remplissage de noir pour faire le cadre
465   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
466
467   //Création de la surface de fond
468   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
469   //Remplissage de vert tapis
470   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
471
472   //Chargement de l'image
473   //if(int_type==BDE) surf_image = IMG_Load("sdl/images/bureau_bde.png");
474   switch(int_type)
475   {
476     case LC_WC:
477       surf_image = IMG_Load("sdl/images/wc.png");
478       break;
479     case LC_ASCENSEUR:
480       surf_image = IMG_Load("sdl/images/ascenseur.png");
481       break;
482     case LC_RU:
```

```
483       surf_image = IMG_Load("sdl/images/ru.png");
484       break;
485     case LC_PARKING:
486       surf_image = IMG_Load("sdl/images/parking.png");
487       break;
488     default:
489       break;
490   }
491   //On modifie la position du fond
492   position.x=CASE_EPAISSEUR;
493   position.y=CASE_EPAISSEUR;
494   //On colle le fond sur la case
495   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
496
497   //On positionne le prix
498   position.x=(CASE_LARGEUR-(*surf_prix).w)/2;
499   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
500   //On colle le prix sur la case
501   SDL_BlitSurface(surf_prix, NULL, surf_case, &position);
502
503   //On positione le logo
504   position.x=10;
505   position.y=8;
506   //On colle le logo sur la case
507   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
508
509   //Selon la position
510   switch(int_type)
511   {
512     //Si la case est à droite
513     case LC_PARKING:
514       //Rotation de l'image de 90ř
515       surf_case = rotation_90(surf_case);
516       break;
517     //Si la case est en haut
518     case LC_RU:
519       //Rotation de l'image de 180ř
520       surf_case = rotation_180(surf_case);
521       break;
522     //Si la case est à gauche
523     case LC_ASCENSEUR:
524       //Rotation de l'image de 270ř
525       surf_case = rotation_270(surf_case);
526       break;
527     //Par défaut, la case est en bas
528     default:
529     break;
530   }
531
532   //Libération des surfaces temporaires
533   SDL_FreeSurface(surf_fond);
534   SDL_FreeSurface(surf_image);
535   SDL_FreeSurface(surf_prix);
536
537   //On retourne la case que l'on vient de créer
538   return(surf_case);
539 }
```

8.12.1.10 SDL_Surface* creation_case_association (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case de type association

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãl'cran

int_type type de l'association

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 540 du fichier sdl.c.

Référencé par init_case_association().

```
541 {
542   //Surface du logo
543   SDL_Surface* surf_logo;
544   surf_logo=NULL;
545   //Surface du fond de la case
546   SDL_Surface* surf_fond;
547   surf_fond=NULL;
548   //Surface de la case
549   SDL_Surface* surf_case;
550   surf_case=NULL;
551   //Surface du prix
552   SDL_Surface* surf_prix;
553   surf_prix=NULL;
554
555   //Position des images
556   SDL_Rect position;
557
558   //Police d'écriture
559   TTF_Font* police;
560   police=NULL;
561
562   //Couleur du texte
563   SDL_Color couleur_texte = {0, 0, 0};
564   //Couleur du fond du texte
565   SDL_Color couleur_fond_texte = {16, 246, 128};
566
567   //Ouvertue de la police d'écriture
568   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
569
570   //Ecriture du premier texte
571   surf_prix = TTF_RenderText_Shaded(police, "1500 Fintz", couleur_texte, couleur_fond_texte);
572
573   //Fermeture de la police d'écriture
574   TTF_CloseFont(police);
575
576   //Création de la surface de la case
577   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
578   //Remplissage de noir pour faire le cadre
579   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
580
581   //Création de la surface de fond
582   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
583   //Remplissage de vert tapis
584   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
585
586   //Chargelent de l'image du logo
```

```
587   if(int_type==BDE) surf_logo = IMG_Load("sdl/images/bureau_bde.png");
588   else surf_logo = IMG_Load("sdl/images/bureau_bds.png");
589
590   //On modifie la position du fond
591   position.x=CASE_EPAISSEUR;
592   position.y=CASE_EPAISSEUR;
593   //On colle le fond sur la case
594   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
595
596   //On positionne le prix
597   position.x=(CASE_LARGEUR-(*surf_prix).w)/2;
598   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
599   //On colle le prix sur la case
600   SDL_BlitSurface(surf_prix, NULL, surf_case, &position);
601
602   //On positione le logo
603   position.x=10;
604   position.y=8;
605   //On colle le logo sur la case
606   SDL_BlitSurface(surf_logo, NULL, surf_case, &position);
607
608   //Selon le type de bureau d'association
609   switch(int_type)
610   {
611     //S'il s'agit du bde
612     case BDE:
613       //Rotation de l'image de 180
614       surf_case = rotation_180(surf_case);
615       break;
616     //Par défaut, il s'agit du bds
617     default:
618       //Rotation de l'image de 270ř
619       surf_case = rotation_270(surf_case);
620       break;
621     break;
622   }
623
624   //Libération des surfaces temporaires
625   SDL_FreeSurface(surf_fond);
626   SDL_FreeSurface(surf_logo);
627   SDL_FreeSurface(surf_prix);
628
629   //On retourne la case que l'on vient de créer
630   return(surf_case);
631 }
```

8.12.1.11 SDL_Surface* creation_case_soiree (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case de type soirÃľe

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de la soirÃľe

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 633 du fichier sdl.c.

Référencé par init_case_soiree().

```
634 {
635   //Surface du logo
636   SDL_Surface* surf_logo;
637   surf_logo=NULL;
638   //Surface du fond de la case
639   SDL_Surface* surf_fond;
640   surf_fond=NULL;
641   //Surface de la case
642   SDL_Surface* surf_case;
643   surf_case=NULL;
644   //Surface du prix
645   SDL_Surface* surf_prix;
646   surf_prix=NULL;
647   SDL_Surface* surf_texte;
648   surf_texte=NULL;
649
650   //Position des images
651   SDL_Rect position;
652
653   //Police d'écriture
654   TTF_Font* police;
655   police=NULL;
656
657   //Couleur du texte
658   SDL_Color couleur_texte = {0, 0, 0};
659   //Couleur du fond du texte
660   SDL_Color couleur_fond_texte = {16, 246, 128};
661
662   //Ouvertue de la police d'écriture
663   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 14);
664
665   //Ecriture du premier texte
666   if(int_type==SOIREE_AREA) surf_texte = TTF_RenderText_Shaded(police, "Area club", couleur_texte, couleur_fond_text
667   else surf_texte = TTF_RenderText_Shaded(police, "Gala", couleur_texte, couleur_fond_texte);
668   //Ecriture du premier texte
669   surf_prix = TTF_RenderText_Shaded(police, "1500 Fintz", couleur_texte, couleur_fond_texte);
670
671   //Fermeture de la police d'écriture
672   TTF_CloseFont(police);
673
674   //Création de la surface de la case
675   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
676   //Remplissage de noir pour faire le cadre
677   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
678
679   //Création de la surface de fond
680   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
681   //Remplissage de vert tapis
682   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
683
684   //Chargelent de l'image du logo
685   if(int_type==SOIREE_AREA) surf_logo = IMG_Load("sdl/images/soiree_area.png");
686   else surf_logo = IMG_Load("sdl/images/soiree_gala.png");
687
688   //On modifie la position du fond
689   position.x=CASE_EPAISSEUR;
690   position.y=CASE_EPAISSEUR;
691   //On colle le fond sur la case
692   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
693
```

```
694   //On positionne le prix
695   position.x=(CASE_LARGEUR-(*surf_texte).w)/2;
696   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-40;
697   //On colle le prix sur la case
698   SDL_BlitSurface(surf_texte, NULL, surf_case, &position);
699
700   //On positionne le prix
701   position.x=(CASE_LARGEUR-(*surf_prix).w)/2;
702   position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
703   //On colle le prix sur la case
704   SDL_BlitSurface(surf_prix, NULL, surf_case, &position);
705
706   //On positione le logo
707   position.x=10;
708   position.y=8;
709   //On colle le logo sur la case
710   SDL_BlitSurface(surf_logo, NULL, surf_case, &position);
711
712   //S'il s'agit du gala
713   if(int_type==SOIREE_GALA)
714   {
715     //Rotation de l'image de 90ř
716     surf_case = rotation_90(surf_case);
717   }
718
719   //Libération des surfaces temporaires
720   SDL_FreeSurface(surf_fond);
721   SDL_FreeSurface(surf_logo);
722   SDL_FreeSurface(surf_prix);
723   SDL_FreeSurface(surf_texte);
724
725   //On retourne la case que l'on vient de créer
726   return(surf_case);
727 }
```

8.12.1.12 SDL_Surface* creation_case_coin (SDL_Surface * *ecran*, int *int_type*)

crÃľer une case spÃľcial (chacun des coins)

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

ecran surface de l'Ãľcran

int_type type de la case

Version :

1.0

Renvoie :

la surface de la case

Définition à la ligne 729 du fichier sdl.c.

Référencé par init_case_special().

```
730 {
731   //Surface de l'image
732   SDL_Surface* surf_image;
```

```
733   surf_image=NULL;
734   //Surface du fond de la case
735   SDL_Surface* surf_fond;
736   surf_fond=NULL;
737   //Surface de la case
738   SDL_Surface* surf_case;
739   surf_case=NULL;
740
741   //Position des images
742   SDL_Rect position;
743
744   //Création de la surface de la case
745   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_HAUTEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
746   //Remplissage de noir pour faire le cadre
747   SDL_FillRect(surf_case, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
748
749   //Création de la surface de fond
750   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_HAUTEUR-2*CASE_EPAISSEUR, CASE_HAUTEUR-2*CASE_EPAISSEUR, 32, 0
751   //Remplissage de vert tapis
752   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(ecran->format, 16, 246, 128));
753
754   //Chargelent de l'image du logo
755   switch(int_type)
756   {
757     case SP_APPARTEMENT:
758       surf_image = IMG_Load("sdl/images/case_depart.png");
759       break;
760     case SP_BUREAU_LAURENCE:
761       surf_image = IMG_Load("sdl/images/bureau_laurence.png");
762       break;
763     case SP_MACHINE_CAFE:
764       surf_image = IMG_Load("sdl/images/machine_cafe.png");
765       break;
766     case SP_TABLEAU:
767       surf_image = IMG_Load("sdl/images/go_to_prison.png");
768       break;
769   }
770
771   //On modifie la position du fond
772   position.x=CASE_EPAISSEUR;
773   position.y=CASE_EPAISSEUR;
774   //On colle le fond sur la case
775   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
776
777   //On positionne l'image sur le fond
778   position.x=CASE_EPAISSEUR;
779   position.y=CASE_EPAISSEUR;
780   //On colle le logo sur la case
781   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
782
783   //Libération des surfaces temporaires
784   SDL_FreeSurface(surf_fond);
785   SDL_FreeSurface(surf_image);
786
787   //On retourne la case que l'on vient de créer
788   return(surf_case);
789 }
```

8.12.1.13 SDL_Surface* creation_case_detail (SDL_Surface * *surf_ecran*, rvb_couleur *couleur*, char * *texte*, int *int_prix*, int *int_prix_niveau*)

crÃľer la surface de la case dÃľtaillÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

couleur couleur du groupe de la case

texte nom de la case

int_prix prix de la case

int_prix_niveau prix d'un niveau de la case

Version :

1.0

Renvoie :

la surface de la case dÃľtaillÃľ

Définition à la ligne 791 du fichier sdl.c.

Référencé par init_case_salle().

```
792 {
793   SDL_Surface* surf_case;
794   surf_case=NULL;
795   SDL_Surface* surf_fond;
796   surf_fond=NULL;
797   SDL_Surface* surf_groupe;
798   surf_groupe=NULL;
799   SDL_Surface* surf_groupe_fond;
800   surf_groupe_fond=NULL;
801   SDL_Surface* surf_sep;
802   surf_sep=NULL;
803   SDL_Surface* surf_texte[22];
804   char texte_temp[256];
805   int i;
806
807   //Police d'écriture
808   TTF_Font* police;
809   police=NULL;
810
811   //Couleur du texte
812   SDL_Color couleur_texte = {0, 0, 0};
813   //Couleur du fond du texte
814   SDL_Color couleur_fond_texte = {225, 255, 225};
815   //Couleur du groupe
816   SDL_Color couleur_groupe = {couleur.rouge, couleur.vert, couleur.bleu};
817
818   //Position
819   SDL_Rect position;
820
821   //Ouverture de la police d'ecriture
822   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 28);
823
824   //Tire de la carte de propriété
825   surf_texte[1] = TTF_RenderText_Shaded(police, texte, couleur_texte, couleur_groupe);
826
827   //Fermeture de la police d'ecriture
828   TTF_CloseFont(police);
829
830   //Ouverture de la police d'ecriture
831   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
832
```

```
833   //Ecriture des informations contenus dans le groupe
834   surf_texte[0] = TTF_RenderText_Shaded(police, "Titre de propriété", couleur_texte, couleur_groupe);
835
836   //Ecriture des textes sur les informations sur les loyers
837   surf_texte[2] = TTF_RenderText_Shaded(police, "LOYER Salle vide", couleur_texte, couleur_fond_texte);
838   surf_texte[3] = TTF_RenderText_Shaded(police, "Niveau CPI 1", couleur_texte, couleur_fond_texte);
839   surf_texte[4] = TTF_RenderText_Shaded(police, "Niveau CPI 2", couleur_texte, couleur_fond_texte);
840   surf_texte[5] = TTF_RenderText_Shaded(police, "Niveau ING 1", couleur_texte, couleur_fond_texte);
841   surf_texte[6] = TTF_RenderText_Shaded(police, "Niveau ING 2", couleur_texte, couleur_fond_texte);
842   surf_texte[7] = TTF_RenderText_Shaded(police, "Niveau ING 3", couleur_texte, couleur_fond_texte);
843
844   //Ecriture de la monnaie utilisé
845   surf_texte[8] = TTF_RenderText_Shaded(police, "Fintz", couleur_texte, couleur_fond_texte);
846
847   //Ecriture des différents prix
848   for(i=0;i<6;i++)
849   {
850     sprintf(texte_temp,"%d",int_prix*(i+1));
851     surf_texte[9+i] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
852   }
853
854   //Fermeture de la police d'ecriture
855   TTF_CloseFont(police);
856
857   //Ouverture de la police d'ecriture
858   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
859
860   //Ecriture des informations sur les loyers
861   surf_texte[15] = TTF_RenderText_Shaded(police, "Si un élève possède toutes les salles", couleur_texte, couleur_for
862   surf_texte[16] = TTF_RenderText_Shaded(police, "d'un groupe de couleur, le loyer des", couleur_texte, couleur_fond
863   surf_texte[17] = TTF_RenderText_Shaded(police, "salles vides de ce groupe est doublé.", couleur_texte, couleur_for
864
865   //Ecriture des informations sur l'achat d'un niveau et l'hypothèque
866   sprintf(texte_temp,"Prix d'un niveau                    %d Fintz",int_prix_niveau);
867   surf_texte[18] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
868   surf_texte[19] = TTF_RenderText_Shaded(police, "chacun", couleur_texte, couleur_fond_texte);
869   surf_texte[20] = TTF_RenderText_Shaded(police, "Valeur Hypothécaire de la salle", couleur_texte, couleur_fond_text
870   sprintf(texte_temp,"%d Fintz",int_prix);
871   surf_texte[21] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
872
873   //Fermeture de la police d'ecriture
874   TTF_CloseFont(police);
875
876   //Création de la surface de la case
877   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_HAUTEUR, 32, 0, 0, 0, 0);
878   //Remplissage de noir pour faire le cadre
879   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
880
881   //Création de la surface de fond
882   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_HAUTEUR-2*DETAIL_EPAISSEUR
883   //Remplissage de vert tapis
884   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 225, 255, 225));
885
886   //Création de la surface du groupe
887   surf_groupe=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_GROUPE_HAUTEUR+2*DETAIL_EPAISSEUR, 32, 0, 0
888   //Remplissage de noir pour faire le cadre
889   SDL_FillRect(surf_groupe, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
890
891   //Création de la surface du groupe
892   surf_groupe_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_GROUPE_HAUTEUR, 32
893   //Remplissage de la couleur du groupe
894   SDL_FillRect(surf_groupe_fond, NULL, SDL_MapRGB(surf_ecran->format, couleur.rouge, couleur.vert, couleur.bleu));
895
896   //Création de la surface de séparation
897   surf_sep=SDL_CreateRGBSurface(SDL_HWSURFACE, (DETAIL_LARGEUR*9)/10, 3, 32, 0, 0, 0, 0);
898   //Remplissage de noir
899   SDL_FillRect(surf_sep, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
```

```
900
901   //On modifie la position du fond
902   position.x=DETAIL_EPAISSEUR;
903   position.y=DETAIL_EPAISSEUR;
904   //On colle le fond du groupe sur ce dernier
905   SDL_BlitSurface(surf_groupe_fond, NULL, surf_groupe, &position);
906
907   //On modifie la position du fond
908   position.x=DETAIL_EPAISSEUR;
909   position.y=DETAIL_EPAISSEUR;
910   //On colle le fond sur la case
911   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
912
913   //On modifie la position du fond
914   position.x=0;
915   position.y=0;
916   //On colle le groupe sur la case
917   SDL_BlitSurface(surf_groupe, NULL, surf_case, &position);
918
919   //Pour le texte contenu dans la coulerur du groupe
920   for(i=0;i<2;i++)
921   {
922     //On centre la position du texte
923     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
924     position.y=10+25*i;
925     //On colle le texte sur la surface
926     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
927   }
928
929   //Modification de la position du texte pour le loyer du terrain nu
930   position.x=20;
931   position.y=105;
932   //Affichage du loyer du terrain nu
933   SDL_BlitSurface(surf_texte[2], NULL, surf_case, &position);
934
935   //Pour les autres informations de la carte
936   for(i=3;i<8;i++)
937   {
938     //On centre la position du texte
939     position.x=95;
940     position.y=55+25*i;
941     //On colle le texte sur la surface
942     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
943   }
944
945   //Pour les informations sur les prix
946   for(i=8;i<15;i++)
947   {
948     //On centre la position du texte
949     position.x=DETAIL_LARGEUR-15-(surf_texte[i]->w);
950     position.y=55+25*(i-7);
951     //On colle le texte sur la surface
952     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
953   }
954
955   //Position du séparateur de texte
956   position.x=DETAIL_LARGEUR/20;
957   position.y=262;
958   //Affichage du séparateur de texte
959   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
960
961   for(i=15;i<18;i++)
962   {
963     //On centre la position du texte
964     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
965     position.y=90+20*(i-6);
966     //On colle le texte sur la surface
```

```
967     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
968   }
969
970   //Position du séparateur de texte
971   position.x=DETAIL_LARGEUR/20;
972   position.y=337;
973   //Affichage du séparateur de texte
974   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
975
976   //Position du texte
977   position.x=10;
978   position.y=345;
979   //On colle le texte sur la surface
980   SDL_BlitSurface(surf_texte[18], NULL, surf_case, &position);
981
982   //Position du texte
983   position.x=(DETAIL_LARGEUR-10-(*surf_texte[19]).w);
984   position.y=365;
985   //On colle le texte sur la surface
986   SDL_BlitSurface(surf_texte[19], NULL, surf_case, &position);
987
988   //Position du texte
989   position.x=(DETAIL_LARGEUR-(*surf_texte[20]).w)/2;
990   position.y=390;
991   //On colle le texte sur la surface
992   SDL_BlitSurface(surf_texte[20], NULL, surf_case, &position);
993
994   //Position du texte
995   position.x=(DETAIL_LARGEUR-(*surf_texte[21]).w)/2;
996   position.y=410;
997   //On colle le texte sur la surface
998   SDL_BlitSurface(surf_texte[21], NULL, surf_case, &position);
999
1000   //Libération des surfaces temporaires
1001   SDL_FreeSurface(surf_fond);
1002   SDL_FreeSurface(surf_groupe_fond);
1003   SDL_FreeSurface(surf_groupe);
1004   for(i=0;i<22;i++) SDL_FreeSurface(surf_texte[i]);
1005
1006   return(surf_case);
1007 }
```

8.12.1.14 SDL_Surface* creation_case_detail_lc (SDL_Surface * *surf_ecran*, int *int_type*)

crÃľer la surface de la case dÃľtaille d'une case lieu commun

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

int_type type de lieu commun

Version :

1.0

Renvoie :

la surface de la case dÃľtaillÃľ

Définition à la ligne 1009 du fichier sdl.c.

Référencé par init_case_lieu_commun().

```
1010 {
1011   SDL_Surface* surf_case;
1012   surf_case=NULL;
1013   SDL_Surface* surf_fond;
1014   surf_fond=NULL;
1015   SDL_Surface* surf_image;
1016   surf_image=NULL;
1017   SDL_Surface* surf_sep;
1018   surf_sep=NULL;
1019   SDL_Surface* surf_texte[14];
1020   char texte_temp[256];
1021   int i;
1022
1023   //Police d'écriture
1024   TTF_Font* police;
1025   police=NULL;
1026
1027   //Couleur du texte
1028   SDL_Color couleur_texte = {0, 0, 0};
1029   //Couleur du fond du texte
1030   SDL_Color couleur_fond_texte = {225, 255, 225};
1031
1032   //Position
1033   SDL_Rect position;
1034
1035   switch(int_type)
1036   {
1037     case LC_WC:
1038       surf_image = IMG_Load("sdl/images/wc.png");
1039       strcpy(texte_temp,"W.C.");
1040       break;
1041     case LC_ASCENSEUR:
1042       surf_image = IMG_Load("sdl/images/ascenseur.png");
1043       strcpy(texte_temp,"Ascenseur");
1044       break;
1045     case LC_RU:
1046       surf_image = IMG_Load("sdl/images/ru.png");
1047       strcpy(texte_temp,"R.U.");
1048       break;
1049     case LC_PARKING:
1050       surf_image = IMG_Load("sdl/images/parking.png");
1051       strcpy(texte_temp,"Parking");
1052       break;
1053     default:
1054       break;
1055   }
1056
1057   //Ouverture de la police d'ecriture
1058   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 28);
1059
1060   //Tire de la carte de propriété
1061   surf_texte[1] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1062
1063   //Fermeture de la police d'ecriture
1064   TTF_CloseFont(police);
1065
1066   //Ouverture de la police d'ecriture
1067   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
1068
1069   //Ecriture des informations contenus dans le groupe
1070   surf_texte[0] = TTF_RenderText_Shaded(police, "Titre de propriété", couleur_texte, couleur_fond_texte);
1071
1072   //Ecriture des textes sur les informations sur les loyers
```

```
1073   surf_texte[2] = TTF_RenderText_Shaded(police, "LOYER si vous avez :", couleur_texte, couleur_fond_texte);
1074   surf_texte[3] = TTF_RenderText_Shaded(police, "1 lieu commun", couleur_texte, couleur_fond_texte);
1075   surf_texte[4] = TTF_RenderText_Shaded(police, "2 lieux commun", couleur_texte, couleur_fond_texte);
1076   surf_texte[5] = TTF_RenderText_Shaded(police, "3 lieux commun", couleur_texte, couleur_fond_texte);
1077   surf_texte[6] = TTF_RenderText_Shaded(police, "4 lieux commun", couleur_texte, couleur_fond_texte);
1078
1079   //Ecriture de la monnaie utilisé
1080   surf_texte[7] = TTF_RenderText_Shaded(police, "Fintz", couleur_texte, couleur_fond_texte);
1081
1082   //Ecriture des différents prix
1083   for(i=0;i<4;i++)
1084   {
1085     sprintf(texte_temp,"%d",250*(i+1));
1086     surf_texte[8+i] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1087   }
1088
1089   //Fermeture de la police d'ecriture
1090   TTF_CloseFont(police);
1091
1092   //Ouverture de la police d'ecriture
1093   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1094
1095   //Ecriture des informations sur l'achat d'un niveau et l'hypothèque
1096   surf_texte[12] = TTF_RenderText_Shaded(police, "Valeur Hypothécaire de la salle", couleur_texte, couleur_fond_tex
1097   sprintf(texte_temp,"%d Fintz",1000);
1098   surf_texte[13] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1099
1100   //Fermeture de la police d'ecriture
1101   TTF_CloseFont(police);
1102
1103   //Création de la surface de la case
1104   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_HAUTEUR, 32, 0, 0, 0, 0);
1105   //Remplissage de noir pour faire le cadre
1106   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1107
1108   //Création de la surface de fond
1109   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_HAUTEUR-2*DETAIL_EPAISSEU
1110   //Remplissage de vert tapis
1111   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 225, 255, 225));
1112
1113   //Création de la surface de séparation
1114   surf_sep=SDL_CreateRGBSurface(SDL_HWSURFACE, (DETAIL_LARGEUR*9)/10, 3, 32, 0, 0, 0, 0);
1115   //Remplissage de noir
1116   SDL_FillRect(surf_sep, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1117
1118
1119   //On modifie la position du fond
1120   position.x=DETAIL_EPAISSEUR;
1121   position.y=DETAIL_EPAISSEUR;
1122   //On colle le fond sur la case
1123   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
1124
1125   //Pour le titre
1126   for(i=0;i<2;i++)
1127   {
1128     //On centre la position du texte
1129     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1130     position.y=10+25*i;
1131     //On colle le texte sur la surface
1132     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1133   }
1134
1135   //On centre la position de l'image
1136   position.x=(DETAIL_LARGEUR-(surf_image->w))/2;
1137   position.y=85;
1138   //On colle l'image sur la surface
1139   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
```

```
1140
1141   for(i=2;i<7;i++)
1142   {
1143     //Modification de la position du texte pour le loyer du terrain nu
1144     position.x=20;
1145     position.y=145+25*i;
1146     //Affichage du loyer du terrain nu
1147     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1148   }
1149
1150   //Pour les informations sur les prix
1151   for(i=7;i<12;i++)
1152   {
1153     //On centre la position du texte
1154     position.x=DETAIL_LARGEUR-15-(surf_texte[i]->w);
1155     position.y=145+25*(i-5);
1156     //On colle le texte sur la surface
1157     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1158   }
1159
1160   //Position du séparateur de texte
1161   position.x=DETAIL_LARGEUR/20;
1162   position.y=360;
1163   //Affichage du séparateur de texte
1164   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
1165
1166   //Position du texte
1167   position.x=(DETAIL_LARGEUR-(*surf_texte[12]).w)/2;
1168   position.y=390;
1169   //On colle le texte sur la surface
1170   SDL_BlitSurface(surf_texte[12], NULL, surf_case, &position);
1171
1172   //Position du texte
1173   position.x=(DETAIL_LARGEUR-(*surf_texte[13]).w)/2;
1174   position.y=410;
1175   //On colle le texte sur la surface
1176   SDL_BlitSurface(surf_texte[13], NULL, surf_case, &position);
1177
1178   //Libération des surfaces temporaires
1179   SDL_FreeSurface(surf_fond);
1180   SDL_FreeSurface(surf_image);
1181   for(i=0;i<14;i++) SDL_FreeSurface(surf_texte[i]);
1182
1183   return(surf_case);
1184 }
```

8.12.1.15 SDL_Surface* creation_case_detail_assoc (SDL_Surface * *surf_ecran*, int *int_type*)

crÃľer la surface de la case dÃľtaille d'une case association

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

int_type type d'association

Version :

1.0

Renvoie :

la surface de la case dÃŕtaillÃŕ

Définition à la ligne 1186 du fichier sdl.c.

Référencé par init_case_association().

```
1187 {
1188   SDL_Surface* surf_case;
1189   surf_case=NULL;
1190   SDL_Surface* surf_fond;
1191   surf_fond=NULL;
1192   SDL_Surface* surf_image;
1193   surf_image=NULL;
1194   SDL_Surface* surf_sep;
1195   surf_sep=NULL;
1196   SDL_Surface* surf_texte[10];
1197   char texte_temp[256];
1198   int i;
1199
1200   //Police d'écriture
1201   TTF_Font* police;
1202   police=NULL;
1203
1204   //Couleur du texte
1205   SDL_Color couleur_texte = {0, 0, 0};
1206   //Couleur du fond du texte
1207   SDL_Color couleur_fond_texte = {225, 255, 225};
1208
1209   //Position
1210   SDL_Rect position;
1211
1212   switch(int_type)
1213   {
1214     case BDE:
1215       surf_image = IMG_Load("sdl/images/bureau_bde.png");
1216       strcpy(texte_temp,"B.D.E.");
1217       break;
1218     case BDS:
1219       surf_image = IMG_Load("sdl/images/bureau_bds.png");
1220       strcpy(texte_temp,"B.D.S.");
1221       break;
1222     default:
1223       break;
1224   }
1225
1226   //Ouverture de la police d'ecriture
1227   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 28);
1228
1229   //Tire de la carte de propriété
1230   surf_texte[1] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1231
1232   //Fermeture de la police d'ecriture
1233   TTF_CloseFont(police);
1234
1235   //Ouverture de la police d'ecriture
1236   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
1237
1238   //Ecriture des informations contenus dans le groupe
1239   surf_texte[0] = TTF_RenderText_Shaded(police, "Titre de propriété", couleur_texte, couleur_fond_texte);
1240
1241   //Fermeture de la police d'ecriture
1242   TTF_CloseFont(police);
1243
1244   //Ouverture de la police d'ecriture
1245   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1246
```

```
1247   //Ecriture des textes sur les informations sur les loyers
1248   surf_texte[2] = TTF_RenderText_Shaded(police, "Si l'élève possède UN seul bureau", couleur_texte, couleur_fond_te
1249   surf_texte[3] = TTF_RenderText_Shaded(police, "d'association, le droit de passage est", couleur_texte, couleur_fo
1250   surf_texte[4] = TTF_RenderText_Shaded(police, "40 fois le montant indiqué par les dés", couleur_texte, couleur_fo
1251
1252   //Ecriture des textes sur les informations sur les loyers
1253   surf_texte[5] = TTF_RenderText_Shaded(police, "Si l'élève possède les DEUX bureaux", couleur_texte, couleur_fond_
1254   surf_texte[6] = TTF_RenderText_Shaded(police, "d'association, le droit de passage est", couleur_texte, couleur_fo
1255   surf_texte[7] = TTF_RenderText_Shaded(police, "100 fois le montant indiqué par les dés", couleur_texte, couleur_
1256
1257   //Ecriture des informations sur l'achat d'un niveau et l'hypothèque
1258   surf_texte[8] = TTF_RenderText_Shaded(police, "Valeur Hypothécaire de la salle", couleur_texte, couleur_fond_text
1259   sprintf(texte_temp,"%d Fintz",750);
1260   surf_texte[9] = TTF_RenderText_Shaded(police, texte_temp, couleur_texte, couleur_fond_texte);
1261
1262   //Fermeture de la police d'ecriture
1263   TTF_CloseFont(police);
1264
1265   //Création de la surface de la case
1266   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR, DETAIL_HAUTEUR, 32, 0, 0, 0, 0);
1267   //Remplissage de noir pour faire le cadre
1268   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1269
1270   //Création de la surface de fond
1271   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, DETAIL_LARGEUR-2*DETAIL_EPAISSEUR, DETAIL_HAUTEUR-2*DETAIL_EPAISSEU
1272   //Remplissage de vert tapis
1273   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 225, 255, 225));
1274
1275   //Création de la surface de séparation
1276   surf_sep=SDL_CreateRGBSurface(SDL_HWSURFACE, (DETAIL_LARGEUR*9)/10, 3, 32, 0, 0, 0, 0);
1277   //Remplissage de noir
1278   SDL_FillRect(surf_sep, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1279
1280
1281   //On modifie la position du fond
1282   position.x=DETAIL_EPAISSEUR;
1283   position.y=DETAIL_EPAISSEUR;
1284   //On colle le fond sur la case
1285   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
1286
1287   //Pour le titre
1288   for(i=0;i<2;i++)
1289   {
1290     //On centre la position du texte
1291     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1292     position.y=10+25*i;
1293     //On colle le texte sur la surface
1294     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1295   }
1296
1297   //On centre la position de l'image
1298   position.x=(DETAIL_LARGEUR-(surf_image->w))/2;
1299   position.y=85;
1300   //On colle l'image sur la surface
1301   SDL_BlitSurface(surf_image, NULL, surf_case, &position);
1302
1303   for(i=2;i<5;i++)
1304   {
1305     //Modification de la position du texte pour le loyer du terrain nu
1306     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1307     position.y=140+20*i;
1308     //Affichage du loyer du terrain nu
1309     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1310   }
1311
1312   //Position du séparateur de texte
1313   position.x=DETAIL_LARGEUR/20;
```

```
1314   position.y=260;
1315   //Affichage du séparateur de texte
1316   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
1317
1318   //Pour les informations sur les prix
1319   for(i=5;i<8;i++)
1320   {
1321     //On centre la position du texte
1322     position.x=(DETAIL_LARGEUR-(*surf_texte[i]).w)/2;
1323     position.y=180+20*i;
1324     //On colle le texte sur la surface
1325     SDL_BlitSurface(surf_texte[i], NULL, surf_case, &position);
1326   }
1327
1328   //Position du séparateur de texte
1329   position.x=DETAIL_LARGEUR/20;
1330   position.y=360;
1331   //Affichage du séparateur de texte
1332   SDL_BlitSurface(surf_sep, NULL, surf_case, &position);
1333
1334   //Position du texte
1335   position.x=(DETAIL_LARGEUR-(*surf_texte[8]).w)/2;
1336   position.y=390;
1337   //On colle le texte sur la surface
1338   SDL_BlitSurface(surf_texte[8], NULL, surf_case, &position);
1339
1340   //Position du texte
1341   position.x=(DETAIL_LARGEUR-(*surf_texte[9]).w)/2;
1342   position.y=410;
1343   //On colle le texte sur la surface
1344   SDL_BlitSurface(surf_texte[9], NULL, surf_case, &position);
1345
1346   //Libération des surfaces temporaires
1347   SDL_FreeSurface(surf_fond);
1348   SDL_FreeSurface(surf_image);
1349   for(i=0;i<10;i++) SDL_FreeSurface(surf_texte[i]);
1350
1351   return(surf_case);
1352 }
```

8.12.1.16 SDL_Surface* creation_case_propriete (SDL_Surface * *surf_ecran*, rvb_couleur *couleur*, char * *texte*)

crÃľer la surface de la propriÃľtÃľ qui s'affichera dans le panneau des propriÃľtÃľs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

couleur couleur du groupe de la case

texte nom de la case

Version :

1.0

Renvoie :

la surface la propriÃľtÃľ

Définition à la ligne 1354 du fichier sdl.c.

Référencé par init_case_association(), init_case_lieu_commun(), et init_case_salle().

```
1355 {
1356   //Surface de la case
1357   SDL_Surface* surf_case;
1358   surf_case=NULL;
1359   //Surface du fond
1360   SDL_Surface* surf_fond;
1361   surf_fond=NULL;
1362   //Surface du texte
1363   SDL_Surface* surf_texte;
1364   surf_texte=NULL;
1365
1366
1367   //Police d'écriture
1368   TTF_Font* police;
1369   police=NULL;
1370
1371   //Couleur du texte
1372   SDL_Color couleur_texte = {0, 0, 0};
1373   //Couleur du groupe
1374   SDL_Color couleur_groupe = {couleur.rouge,couleur.vert,couleur.bleu};
1375
1376   //Position
1377   SDL_Rect position;
1378
1379   //Ouverture de la police d'ecriture
1380   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1381
1382   //Ecriture du texte
1383   surf_texte = TTF_RenderText_Shaded(police, texte, couleur_texte, couleur_groupe);
1384
1385   //Fermeture de la police d'ecriture
1386   TTF_CloseFont(police);
1387
1388   //Création de la surface de la case
1389   surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, PROPRIETE_LARGEUR, PROPRIETE_HAUTEUR, 32, 0, 0, 0, 0);
1390   //Remplissage de noir pour faire le cadre
1391   SDL_FillRect(surf_case, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1392
1393   //Création de la surface de fond
1394   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PROPRIETE_LARGEUR-6, PROPRIETE_HAUTEUR-6, 32, 0, 0, 0, 0);
1395   //Remplissage de la couleur du groupe
1396   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, couleur.rouge,couleur.vert,couleur.bleu));
1397
1398
1399   //On modifie la position du fond
1400   position.x=3;
1401   position.y=3;
1402   //On colle le fond sur la case
1403   SDL_BlitSurface(surf_fond, NULL, surf_case, &position);
1404
1405   //Position du texte
1406   position.x=(PROPRIETE_LARGEUR-(surf_texte->w))/2;
1407   position.y=5;
1408   //Collage du texte sur la case
1409   SDL_BlitSurface(surf_texte, NULL, surf_case, &position);
1410
1411   //Libération des surfaces temporaires
1412   SDL_FreeSurface(surf_fond);
1413   SDL_FreeSurface(surf_texte);
1414   //On retourne la case
1415   return(surf_case);
1416 }
```

8.12.1.17 SDL_Surface** creation_des (void)

renvoit un tableau de surface contenant chacune des faces du dÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Version :

1.0

Renvoie :

le tableau des surfaces des faces du dÃľ

Définition à la ligne 2190 du fichier sdl.c.

Référencé par lancer_des().

```
2190 {
2191   //Surface du dé
2192   SDL_Surface** surf_des;
2193   if((surf_des= new SDL_Surface*[6])==NULL) cout<<"Fesse en boite"<<endl;
2194   //Création de la surface de l'objet nř1
2195   surf_des[0]=IMG_Load("sdl/images/un.png");
2196   surf_des[1]=IMG_Load("sdl/images/deux.png");
2197   surf_des[2]=IMG_Load("sdl/images/trois.png");
2198   surf_des[3]=IMG_Load("sdl/images/quatre.png");
2199   surf_des[4]=IMG_Load("sdl/images/cinq.png");
2200   surf_des[5]=IMG_Load("sdl/images/six.png");
2201   //On retourne la surface du dé
2202   return(surf_des);
2203 }
2204
```

8.12.1.18 SDL_Surface* creation_message (SDL_Surface * *surf_ecran*, char * *titre*, char ** *message*, int *int_type_message*, int *int_nbre_ligne*)

crÃľer la surface d'un message Ãă afficher

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

titre titre du message

message message Ãă afficher

int_type_message type du message Ãă afficher

int_nbre_ligne nombre de ligne du message

Version :

1.0

Renvoie :

la surface du message

Définition à la ligne 2442 du fichier sdl.c.

Référencé par attente_validation_message().

```
2442 {
2443   //Surface du message
2444   SDL_Surface* surf_message;
2445   surf_message=NULL;
2446
2447   //Surface du fond du message
2448   SDL_Surface* surf_fond;
2449   surf_fond=NULL;
2450
2451   //Surface du texte
2452   SDL_Surface* surf_texte[int_nbre_ligne];
2453
2454   //Surface du titre
2455   SDL_Surface* surf_titre;
2456   surf_titre=NULL;
2457
2458   //Police d'écriture
2459   TTF_Font* police;
2460   police=NULL;
2461
2462   int i;
2463
2464   //Couleur du texte
2465   SDL_Color couleur_texte = {0, 0, 0};
2466   //Couleur du groupe
2467   SDL_Color couleur_fond_texte = {0,0,0};
2468
2469   //Selon le type de message, on change la couleur de fond du message
2470   switch(int_type_message)
2471   {
2472     //Dans le cas d'un message de Krystel
2473     case MESSAGE_KRYSTEL:
2474       couleur_fond_texte.r=85;
2475       couleur_fond_texte.g=115;
2476       couleur_fond_texte.b=215;
2477       break;
2478     //Dans le cas d'un message de Nadège
2479     case MESSAGE_NADEGE:
2480       couleur_fond_texte.r=215;
2481       couleur_fond_texte.g=85;
2482       couleur_fond_texte.b=45;
2483       break;
2484     //Dans tout les autres cas
2485     default:
2486       couleur_fond_texte.r=185;
2487       couleur_fond_texte.g=235;
2488       couleur_fond_texte.b=140;
2489       break;
2490   }
2491
2492   //Position
2493   SDL_Rect position;
2494
2495   //Si il y à un titre
2496   if(titre)
2497   {
2498     //Ouverture de la police d'ecriture
2499     police = TTF_OpenFont("sdl/police/police2-bold.ttf", 30);
2500
2501     //Ecriture du titre
2502     surf_titre = TTF_RenderText_Shaded(police, titre, couleur_texte, couleur_fond_texte);
2503
2504     //Fermeture de la police d'ecriture
```

```
2505     TTF_CloseFont(police);
2506   }
2507
2508   //Ouverture de la police d'ecriture
2509   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 20);
2510
2511   //Ecriture du texte
2512   for(i=0;i<int_nbre_ligne;i++) surf_texte[i] = TTF_RenderText_Shaded(police, message[i], couleur_texte, couleur_fo
2513
2514   //Fermeture de la police d'ecriture
2515   TTF_CloseFont(police);
2516
2517   //Création de la surface du message
2518   surf_message=SDL_CreateRGBSurface(SDL_HWSURFACE, MESSAGE_LARGEUR, 180+(int_nbre_ligne*20), 32, 0, 0, 0, 0);
2519   //Remplissage de noir pour faire le cadre
2520   SDL_FillRect(surf_message, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2521
2522   //Création de la surface de fond
2523   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, MESSAGE_LARGEUR-6, 180+(int_nbre_ligne*20)-6, 32, 0, 0, 0, 0);
2524   //Remplissage de la couleur du groupe
2525   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, couleur_fond_texte.r, couleur_fond_texte.g, couleur_
2526
2527   //On modifie la position du fond
2528   position.x=3;
2529   position.y=3;
2530   //On colle le fond sur la case
2531   SDL_BlitSurface(surf_fond, NULL, surf_message, &position);
2532
2533   //Si il y a un titre
2534   if(titre)
2535   {
2536     //Position du titre
2537     position.x=position.x=(MESSAGE_LARGEUR-(surf_titre->w))/2;;
2538     position.y=20;
2539     //Affichage du titre
2540     SDL_BlitSurface(surf_titre, NULL, surf_message, &position);
2541   }
2542
2543   //Pour chaque ligne du texte
2544   for(i=0;i<int_nbre_ligne;i++)
2545   {
2546     //Position de la ligne de texte
2547     position.x=(MESSAGE_LARGEUR-(surf_texte[i]->w))/2;
2548     position.y=60+20*i;
2549     //Affichage de la ligne du texte
2550     SDL_BlitSurface(surf_texte[i], NULL, surf_message, &position);
2551   }
2552
2553   //Libération des surfaces temporaires
2554   SDL_FreeSurface(surf_fond);
2555   for(i=0;i<int_nbre_ligne;i++) SDL_FreeSurface(surf_texte[i]);
2556
2557   //Si il y a un titre, on libère la surface du titre
2558   if(titre) SDL_FreeSurface(surf_titre);
2559
2560
2561   //On retourne la surface du message
2562   return(surf_message);
2563
2564 }
2565
```

8.12.1.19 void destruction_des (SDL_Surface ** *surf_des*)

dÃľtruit les surfaces des dÃŀs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ des tableau des surfaces des faces d'un dÃľ

Version :

1.0

Renvoie :

++

Définition à la ligne 2206 du fichier sdl.c.

Référencé par lancer_des().

```
2206 {
2207   int i;
2208   //Libération des surfaces des dés
2209   for(i=0;i<5;i++) SDL_FreeSurface(surf_des[i]);
2210   //Libération de la mémoire
2211   delete(surf_des);
2212 }
2213
```

8.12.1.20 void affich_joueur (SDL_Surface * *surf_ ecran*, joueur * *pj_ joueur*, int *int_ position*, cases * *pcase*)

affiche un joueur Ãă l'Ãľcran

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'Ãľcran

pj_ joueur joueur en cours

int_ position position du joueur sur la case

pcase case sur laquelle se trouve le joueur

Version :

1.0

Renvoie :

++

Définition à la ligne 132 du fichier sdl.c.

Référencé par affich_joueur_depart(), aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), et reculer_jeton().

```
133 {
134   //Position du joueur
135   SDL_Rect position;
```

```
136
137   //Selon la position de la case
138   switch((*pcase).int_position)
139   {
140     //On calcul les coordonnées de l'image du joueur
141     case POSITION_GAUCHE:
142       position.x=(*pcase).rect_coordonnees.x+25+30*(int_position%2);
143       position.y=(*pcase).rect_coordonnees.y+5+25*(int_position%3);
144       break;
145     case POSITION_DROITE:
146       position.x=(*pcase).rect_coordonnees.x+35+30*(int_position%2);
147       position.y=(*pcase).rect_coordonnees.y+5+25*(int_position%3);
148       break;
149     case POSITION_HAUT:
150       position.x=(*pcase).rect_coordonnees.x+5+25*(int_position%3);
151       position.y=(*pcase).rect_coordonnees.y+25+30*(int_position%2);
152       break;
153     default:
154       position.x=(*pcase).rect_coordonnees.x+5+25*(int_position%3);
155       position.y=(*pcase).rect_coordonnees.y+35+30*(int_position%2);
156       break;
157   }
158   //On colle l'image du joueur sur le fond
159   SDL_BlitSurface(pj_joueur->surf_image, NULL, surf_ecran, &position);
160   //On met à jour l'écran
161   SDL_Flip(surf_ecran);
162 }
```

8.12.1.21 void affich_case (SDL_Surface * *surf_ecran*, cases * *pcase*)

affiche une case Ãă l'Ãľcran

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

pcase case Ãă afficher

Version :

1.0

Renvoie :

++

Définition à la ligne 1662 du fichier sdl.c.

Référencé par aller_a_jeton(), aller_en_prison_jeton(), attente_validation_propriete(), avancer_jeton(), jeu(), et reculer_jeton().

```
1663 {
1664   //Surface de l'image cpi
1665   SDL_Surface* surf_image_cpi;
1666   surf_image_cpi=NULL;
1667
1668   //Surface de l'image ingé
1669   SDL_Surface* surf_image_inge;
1670   surf_image_inge=NULL;
1671
```

```
1672  //Surface de la case temporaire
1673  SDL_Surface* surf_case;
1674  surf_case=NULL;
1675
1676  //Position
1677  SDL_Rect position;
1678
1679  int int_niveau;
1680
1681  //Chargement de l'image cpi
1682  surf_image_cpi=IMG_Load("sdl/images/cpi.png");
1683
1684  //Chargement de l'image ingé
1685  surf_image_inge=IMG_Load("sdl/images/inge.png");
1686
1687  //Création de la surface temporaire de la case en fonction de la position de la case
1688  if(pcase->int_type>=SP_APPARTEMENT && pcase->int_type<=SP_TABLEAU) surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE,
1689  else if(pcase->int_position==POSITION_GAUCHE || pcase->int_position==POSITION_DROITE) surf_case=SDL_CreateRGBSur
1690  else surf_case=SDL_CreateRGBSurface(SDL_HWSURFACE, CASE_LARGEUR, CASE_HAUTEUR, 32, 0, 0, 0, 0);
1691
1692  //Position à 0, 0
1693  position.x=0;
1694  position.y=0;
1695  //Collage de la case sur la surface case temporaire
1696  SDL_BlitSurface((*pcase).surf_image, NULL, surf_case, &position);
1697
1698  //S'il s'agit d'une case salle qui a un niveau supérieur à zéro
1699  if(pcase->int_type==SALLE && pcase->case_salle.int_niveau!=0)
1700  {
1701
1702    //On initalisation le niveau à 0
1703    int_niveau=1;
1704
1705    if(pcase->case_salle.int_niveau<=2)
1706    {
1707      //Selon la position de la case
1708      switch(pcase->int_position)
1709      {
1710        case POSITION_HAUT:
1711          //On tourne l'image de 180 degré
1712          surf_image_cpi=rotation_180(surf_image_cpi);
1713          break;
1714        case POSITION_DROITE:
1715          //On tourne l'image de 90 degré
1716          surf_image_cpi=rotation_90(surf_image_cpi);
1717          break;
1718        case POSITION_GAUCHE:
1719          //On tourne l'image de 270 degré
1720          surf_image_cpi=rotation_270(surf_image_cpi);
1721          break;
1722        default:
1723          //On ne fait rien
1724          break;
1725      }
1726
1727      //Tant que l'image à afficher fait partit du groupe cpi et tant que l'on a pas atteint le niveau de la salle
1728      while(int_niveau<=2 && int_niveau<=(pcase->case_salle.int_niveau))
1729      {
1730        switch(pcase->int_position)
1731        {
1732          case POSITION_HAUT:
1733            //On modifie la position du fond
1734            position.x=(CASE_EPAISSEUR+20*(int_niveau-1));
1735            position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1736            break;
1737          case POSITION_DROITE:
1738            //On modifie la position du fond
```

```
1739            position.x=CASE_EPAISSEUR;
1740            position.y=CASE_EPAISSEUR+20*(int_niveau-1);
1741            break;
1742          case POSITION_GAUCHE:
1743            //On modifie la position du fond
1744            position.x=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1745            position.y=(CASE_EPAISSEUR+20*(int_niveau-1));
1746            break;
1747          default:
1748            //On modifie la position du fond
1749            position.x=CASE_EPAISSEUR+20*(int_niveau-1);
1750            position.y=CASE_EPAISSEUR;
1751            break;
1752        }
1753        //On colle le fond sur la case
1754        SDL_BlitSurface(surf_image_cpi, NULL, surf_case, &position);
1755        int_niveau++;
1756      }
1757    }
1758    else
1759    {
1760      //Selon la position de la case
1761      switch(pcase->int_position)
1762      {
1763        case POSITION_HAUT:
1764          //On tourne l'image de 180 degré
1765          surf_image_inge=rotation_180(surf_image_inge);
1766          break;
1767        case POSITION_DROITE:
1768          //On tourne l'image de 90 degré
1769          surf_image_inge=rotation_90(surf_image_inge);
1770          break;
1771        case POSITION_GAUCHE:
1772          //On tourne l'image de 270 degré
1773          surf_image_inge=rotation_270(surf_image_inge);
1774          break;
1775        default:
1776          //On ne fait rien
1777          break;
1778      }
1779
1780      //Tant que l'image à afficher fait partit du groupe cpi et tant que l'on a pas atteint le niveau de la salle
1781      while(int_niveau<=3 && int_niveau<=(pcase->case_salle.int_niveau)-2)
1782      {
1783        switch(pcase->int_position)
1784        {
1785          case POSITION_HAUT:
1786            //On modifie la position du fond
1787            position.x=(CASE_EPAISSEUR+20*(int_niveau-1));
1788            position.y=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1789            break;
1790          case POSITION_DROITE:
1791            //On modifie la position du fond
1792            position.x=CASE_EPAISSEUR;
1793            position.y=CASE_EPAISSEUR+20*(int_niveau-1);
1794            break;
1795          case POSITION_GAUCHE:
1796            //On modifie la position du fond
1797            position.x=CASE_HAUTEUR-CASE_EPAISSEUR-20;
1798            position.y=(CASE_EPAISSEUR+20*(int_niveau-1));
1799            break;
1800          default:
1801            //On modifie la position du fond
1802            position.x=CASE_EPAISSEUR+20*(int_niveau-1);
1803            position.y=CASE_EPAISSEUR;
1804            break;
1805        }
```

```
1806         //On colle le fond sur la case
1807         SDL_BlitSurface(surf_image_inge, NULL, surf_case, &position);
1808         int_niveau++;
1809       }
1810     }
1811   }
1812
1813   //Position de la case sur l'écran
1814   position.x=(*pcase).rect_coordonnees.x;
1815   position.y=(*pcase).rect_coordonnees.y;
1816
1817   //Affichage de la case
1818   SDL_BlitSurface(surf_case, NULL, surf_ecran, &position);
1819
1820   //Libération des surfaces temporaire
1821   SDL_FreeSurface(surf_image_inge);
1822   SDL_FreeSurface(surf_image_cpi);
1823   SDL_FreeSurface(surf_case);
1824 }
```

8.12.1.22 void affich_case_detail (SDL_Surface * *surf_ecran*, cases * *pcase*)

affiche le dÃľtail d'une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

pcase case dont on doti afficher le dÃľtail

Version :

1.0

Renvoie :

++

Définition à la ligne 1826 du fichier sdl.c.

```
1827 {
1828   SDL_Rect position;
1829   position.x=POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_LARGEUR)/2;
1830   position.y=POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-DETAIL_HAUTEUR)/2;
1831   SDL_BlitSurface(pcase->case_salle.surf_detail, NULL, surf_ecran, &position); // Collage de la surface sur l'écran
1832 }
```

8.12.1.23 void affich_centre (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*)

affiche le centre du plateau

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'Ãľcran

surf_ centre surface du centre

Version :

1.0

Renvoie :

++

Définition à la ligne 1834 du fichier sdl.c.

Référencé par attente_validation_message(), attente_validation_propriete(), et jeu().

```
1835 {
1836   SDL_Rect position;
1837   position.x=POS_X_PLATEAU+CASE_HAUTEUR;
1838   position.y=POS_Y_PLATEAU+CASE_HAUTEUR;
1839   SDL_BlitSurface(surf_centre, NULL, surf_ecran, &position); // Collage de la surface sur l'écran
1840 }
```

8.12.1.24 int affich_accueil (SDL_Surface * *surf_ ecran*)

affiche la page d'accueil

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'Ãľcran

Version :

1.0

Renvoie :

une action demandÃľe du joueur

Définition à la ligne 1418 du fichier sdl.c.

Référencé par main().

```
1419 {
1420   int i;
1421   char char_nombre[16];
1422   int nombre_joueur;
1423   //Surface du numéro
1424   SDL_Surface* surf_nombre[6];
1425   //Surface de la fleche gauche
1426   SDL_Surface* surf_fleche_gauche;
1427   surf_fleche_gauche=NULL;
1428   //Surface de la fleche droite
1429   SDL_Surface* surf_fleche_droite;
1430   surf_fleche_droite=NULL;
1431   //Surface du titre
1432   SDL_Surface* surf_titre;
1433   surf_titre=NULL;
```

```
1434   //Surface du texte
1435   SDL_Surface* surf_texte;
1436   surf_texte=NULL;
1437   //Surface des boutons
1438   SDL_Surface* surf_boutons[6];
1439   //Police d'écriture
1440   TTF_Font* police;
1441   police=NULL;
1442   //Position des images
1443   SDL_Rect position;
1444
1445   //Couleur du titre
1446   SDL_Color couleur_titre = {231, 86, 86};
1447   //Couleur du texte
1448   SDL_Color couleur_texte = {0, 155, 126};
1449   //Couleur des nombres
1450   SDL_Color couleur_nombre = {255, 0, 0};
1451
1452   //Remplissage de noir pour faire le fond
1453   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1454
1455   //Chargement de l'image de la fleche gauche
1456   surf_fleche_gauche=IMG_Load("sdl/images/fleche_gauche.gif");
1457   //Chargement de l'image de la fleche droite
1458   surf_fleche_droite=IMG_Load("sdl/images/fleche_droite.gif");
1459
1460   //Création de la surface du bouton valider
1461   surf_boutons[0]=IMG_Load("sdl/images/suivant.gif");
1462   surf_boutons[1]=IMG_Load("sdl/images/suivant2.gif");
1463   surf_boutons[2]=IMG_Load("sdl/images/quitter.gif");
1464   surf_boutons[3]=IMG_Load("sdl/images/quitter2.gif");
1465   surf_boutons[4]=IMG_Load("sdl/images/charger.gif");
1466   surf_boutons[5]=IMG_Load("sdl/images/charger2.gif");
1467
1468   //Ouverture de la police d'ecriture
1469   police = TTF_OpenFont("sdl/police/police.ttf", 100);
1470
1471   //Ecriture du titre sur la surface de titre
1472   surf_titre = TTF_RenderText_Blended(police, "EISTIOPOLY", couleur_titre);
1473
1474   //Pour chacune des surfaces nombre
1475   for(i=0;i<6;i++)
1476   {
1477     //Enregistrement du numéro
1478     sprintf(char_nombre,"%d",i+1);
1479     //Ecriture du numéro sur la surface
1480     surf_nombre[i] = TTF_RenderText_Blended(police, char_nombre, couleur_nombre);
1481   }
1482
1483   //Fermeture de la police d'ecriture
1484   TTF_CloseFont(police);
1485
1486   //Ouverture de la police d'ecriture
1487   police = TTF_OpenFont("sdl/police/police.ttf", 60);
1488
1489   //Ecriture du texte sur la surface de texte
1490   surf_texte = TTF_RenderText_Blended(police, "Nombre de joueur", couleur_texte);
1491
1492   //Fermeture de la police d'ecriture
1493   TTF_CloseFont(police);
1494
1495   //On modifie la position du titre
1496   position.x=(ECRAN_LARGEUR-(surf_titre->w))/2;
1497   position.y=TITRE_POS_Y;
1498   //On colle le titre sur le fond
1499   SDL_BlitSurface(surf_titre, NULL, surf_ecran, &position);
1500
```

```
1501   //Position du texte
1502   position.x=(ECRAN_LARGEUR-(surf_texte->w))/2;
1503   position.y=TEXTE_POS_Y;
1504   //Affichage du texte
1505   SDL_BlitSurface(surf_texte, NULL, surf_ecran, &position);
1506
1507   //On modifie la position du bouton valider
1508   position.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
1509   position.y=BTN_ACCUEIL_POS_Y;
1510   //On colle le bouton valider sur le fond
1511   SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position);
1512
1513   //On modifie la position du bouton chargement
1514   position.x=(ECRAN_LARGEUR-BTN_ACCUEIL_LARGEUR)/2;
1515   position.y=BTN_ACCUEIL_POS_Y+BTN_ACCUEIL_ESPACE;
1516   //On colle le bouton charger sur le fond
1517   SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position);
1518
1519   //On modifie la position du bouton quitter
1520   position.x=ECRAN_LARGEUR-(surf_boutons[2]->w+10);
1521   position.y=10;
1522   //On colle le bouton quitter sur le fond
1523   SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position);
1524
1525   //Fonction d'attente d'événement
1526   nombre_joueur=attente_action_accueil(surf_ecran, surf_boutons, surf_nombre, surf_fleche_gauche, surf_fleche_droit
1527
1528   //On efface l'écran
1529   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 25, 25, 75));
1530   //Mise à jour de l'écran
1531   SDL_Flip(surf_ecran);
1532
1533   //Libération des surfaces utilisées
1534   for(i=0;i<6;i++) SDL_FreeSurface(surf_nombre[i]);
1535   for(i=0;i<6;i++) SDL_FreeSurface(surf_boutons[i]);
1536   SDL_FreeSurface(surf_fleche_gauche);
1537   SDL_FreeSurface(surf_fleche_droite);
1538   SDL_FreeSurface(surf_titre);
1539
1540   return(nombre_joueur);
1541 }
```

8.12.1.25 int affich_config (SDL_Surface * *surf_ecran*, char ** *str_nom_joueur*, int *nombre_joueur*)

affiche la page de configuration de la partie

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

str_nom_joueur tableau des noms des joueurs

nombre_joueur nombre de joueur dans la partie

Version :

1.0

Renvoie :

une action demandÃľe du joueur

Définition à la ligne 1543 du fichier sdl.c.

Référencé par main().

```
1544 {
1545   int i;
1546   char str_nom[64];
1547   int int_retour;
1548
1549   //Surface d'un champ (sert de cache)
1550   SDL_Surface* surf_champ;
1551   surf_champ=NULL;
1552   //Surfaces des noms
1553   SDL_Surface* surf_nom[nombre_joueur];
1554
1555   //Surface du titre
1556   SDL_Surface* surf_titre;
1557   surf_titre=NULL;
1558   //Surface des boutons
1559   SDL_Surface* surf_boutons[6];
1560   //Police d'écriture
1561   TTF_Font* police;
1562   police=NULL;
1563   //Position des images
1564   SDL_Rect position;
1565
1566   //Couleur du texte
1567   SDL_Color couleur_titre = {231, 86, 86};
1568   //Couleur des noms
1569   SDL_Color couleur_noms = {255, 0, 0};
1570
1571   //Remplissage de noir pour faire le fond
1572   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1573
1574   //Création des surfaces du bouton retour
1575   surf_boutons[0]=IMG_Load("sdl/images/retour.gif");
1576   surf_boutons[1]=IMG_Load("sdl/images/retour2.gif");
1577   surf_boutons[2]=IMG_Load("sdl/images/jouer.gif");
1578   surf_boutons[3]=IMG_Load("sdl/images/jouer2.gif");
1579   surf_boutons[4]=IMG_Load("sdl/images/quitter.gif");
1580   surf_boutons[5]=IMG_Load("sdl/images/quitter2.gif");
1581
1582   //Ouverture de la police d'ecriture
1583   police = TTF_OpenFont("sdl/police/police.ttf", 100);
1584
1585   //Ecriture du titre sur la surface de titre
1586   surf_titre = TTF_RenderText_Blended(police, "EISTIOPOLY", couleur_titre);
1587
1588   //Fermeture de la police d'ecriture
1589   TTF_CloseFont(police);
1590
1591   //Ouverture de la police d'ecriture
1592   police = TTF_OpenFont("sdl/police/police.ttf", 30);
1593
1594   //Ecriture du numéro sur la surface
1595   surf_champ=SDL_CreateRGBSurface(SDL_HWSURFACE, CHAMP_LARGEUR, CHAMP_HAUTEUR, 32, 0, 0, 0, 0);
1596   //Remplissage du bouton valider
1597   SDL_FillRect(surf_champ, NULL, SDL_MapRGB(surf_ecran->format, 204, 204, 204));
1598
1599   //Pour chacune des surfaces nombre
1600   for(i=0;i<nombre_joueur;i++)
1601   {
1602     //Enregistrement du numéro
1603     sprintf(str_nom,"Joueur %d :",i+1);
1604     surf_nom[i] = TTF_RenderText_Blended(police, str_nom, couleur_noms);
1605   }
1606
```

```
1607   //Fermeture de la police d'ecriture
1608   TTF_CloseFont(police);
1609
1610   //On modifie la position du titre
1611   position.x=TITRE_POS_X;
1612   position.y=TITRE_POS_Y;
1613   //On colle le titre sur le fond
1614   SDL_BlitSurface(surf_titre, NULL, surf_ecran, &position);
1615
1616   //On modifie la position du bouton valider
1617   position.x=BTN_ACCUEIL_POS_X+20+(surf_boutons[0])->w;
1618   position.y=BTN_ACCUEIL_POS_Y;
1619   //On colle le bouton valider sur le fond
1620   SDL_BlitSurface(surf_boutons[2], NULL, surf_ecran, &position);
1621
1622   //On modifie la position du bouton retour
1623   position.x=BTN_ACCUEIL_POS_X;
1624   position.y=BTN_ACCUEIL_POS_Y;
1625   //On colle le bouton retour sur le fond
1626   SDL_BlitSurface(surf_boutons[0], NULL, surf_ecran, &position);
1627
1628   //On modifie la position du bouton quitter
1629   position.x=ECRAN_LARGEUR-(surf_boutons[4]->w+10);
1630   position.y=10;
1631   //On colle le bouton quitter sur le fond
1632   SDL_BlitSurface(surf_boutons[4], NULL, surf_ecran, &position);
1633
1634   for(i=0;i<nombre_joueur;i++)
1635   {
1636     position.x=CHAMP_POS_X;
1637     position.y=CHAMP_POS_Y+i*2*CHAMP_HAUTEUR;
1638     SDL_BlitSurface(surf_champ, NULL, surf_ecran, &position);
1639     position.x=NOM_CHAMP_POS_X;
1640     position.y=NOM_CHAMP_POS_Y+i*2*CHAMP_HAUTEUR;
1641     SDL_BlitSurface(surf_nom[i], NULL, surf_ecran, &position);
1642   }
1643
1644   //Fonction d'attente d'événement
1645   int_retour=attente_action_config(surf_ecran, str_nom_joueur, surf_boutons, surf_champ, nombre_joueur);
1646
1647   //On efface l'écran
1648   SDL_FillRect(surf_ecran, NULL, SDL_MapRGB(surf_ecran->format, 25, 25, 75));
1649   //Mise à jour de l'écran
1650   SDL_Flip(surf_ecran);
1651
1652   //Libération des surfaces utilisées
1653   for(i=0;i<nombre_joueur;i++)
1654   {
1655     SDL_FreeSurface(surf_nom[i]);
1656   }
1657   SDL_FreeSurface(surf_titre);
1658   for(i=0;i<6;i++) SDL_FreeSurface(surf_boutons[i]);
1659   return(int_retour);
1660 }
```

8.12.1.26 void affich_panneau_menu (SDL_Surface * *surf_ecran*, int *int_etat_bouton*)

affiche le panneau du menu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

int_etat_bouton Ãľtat des boutons

Version :

1.0

Renvoie :

++

Définition à la ligne 1842 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
1843 {
1844   //Surface du panneau
1845   SDL_Surface* surf_menu;
1846   //Surface du fond du panneau
1847   SDL_Surface* surf_fond;
1848   //Surface des boutons images
1849   SDL_Surface* surf_objet[3];
1850   //Position
1851   SDL_Rect position;
1852   int i;
1853 
1854   //Création de la surface du menu
1855   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_MENU_LARGEUR, PANNEAU_MENU_HAUTEUR, 32, 0, 0, 0, 0);
1856   //Remplissage de noir pour faire le cadre
1857   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1858 
1859   //Création de la surface du fond du menu
1860   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_MENU_LARGEUR-6, PANNEAU_MENU_HAUTEUR-6, 32, 0, 0, 0, 0);
1861   //Remplissage de noir pour faire le cadre
1862   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
1863 
1864   //Chargement des boutons
1865   if(int_etat_bouton==1) surf_objet[0]=IMG_Load("sdl/images/quitter2.gif");
1866   else surf_objet[0]=IMG_Load("sdl/images/quitter.gif");
1867   if(int_etat_bouton==2) surf_objet[1]=IMG_Load("sdl/images/sauvegarder2.png");
1868   else surf_objet[1]=IMG_Load("sdl/images/sauvegarder.png");
1869   if(int_etat_bouton==3) surf_objet[2]=IMG_Load("sdl/images/tourner.png");
1870   else surf_objet[2]=IMG_Load("sdl/images/tourner.png");
1871 
1872   //Positon du fond
1873   position.x=3;
1874   position.y=3;
1875   //Collage du fond sur l'image
1876   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
1877 
1878   //Pour chacun des boutons
1879   for(i=0;i<3;i++)
1880   {
1881     //On position le bouton
1882     position.x=(PANNEAU_OBJET_TAILLE+30)*i+30;
1883     position.y=10;
1884     //Affichage du bouton
1885     SDL_BlitSurface(surf_objet[i], NULL, surf_menu, &position);
1886   }
1887 
1888   //Position du panneau sur l'écran
1889   position.x=PANNEAU_MENU_POS_X;
1890   position.y=PANNEAU_MENU_POS_Y;
1891   //Affichage du panneau
1892   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
```

```
1893
1894   //Mise à jour de l'écran
1895   SDL_Flip(surf_ecran);
1896
1897   //Libération des surfaces temporaires
1898   SDL_FreeSurface(surf_menu);
1899   SDL_FreeSurface(surf_fond);
1900   for(i=0;i<3;i++) SDL_FreeSurface(surf_objet[i]);
1901 }
```

8.12.1.27 void affich_panneau_joueur (SDL_Surface * *surf_ecran*, joueur * *j_anneau_joueurs*)

affiche le panneau d'information sur le joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

j_anneau_joueurs joueur en cours

Version :

1.0

Renvoie :

++

Définition à la ligne 1903 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
1904 {
1905   SDL_Surface* surf_menu;
1906   SDL_Surface* surf_fond;
1907   SDL_Surface* surf_texte[5];
1908
1909   SDL_Rect position_jeton;
1910
1911   char str_temp[256];
1912   int i;
1913
1914   SDL_Rect position;
1915
1916   //Police d'écriture
1917   TTF_Font* police;
1918   police=NULL;
1919
1920   //Couleur du texte
1921   SDL_Color couleur_texte = {86, 255, 86};
1922   SDL_Color couleur_valeur = {125, 225, 125};
1923
1924   //Création de la surface du menu
1925   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_JOUEUR_LARGEUR, PANNEAU_JOUEUR_HAUTEUR, 32, 0, 0, 0, 0);
1926   //Remplissage de noir pour faire le cadre
1927   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
1928
1929   //Création de la surface du fond du menu
1930   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_JOUEUR_LARGEUR-6, PANNEAU_JOUEUR_HAUTEUR-6, 32, 0, 0, 0, 0
```

```
1931   //Remplissage de noir pour faire le cadre
1932   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
1933
1934   //Ouverture de la police d'ecriture
1935   police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
1936
1937   //Ecriture du texte "nom du joueur"
1938   surf_texte[0] = TTF_RenderText_Blended(police, "Nom du joueur :", couleur_texte);
1939
1940   //Ecriture du nom du joueur
1941   surf_texte[1] = TTF_RenderText_Blended(police, j_anneau_joueurs->str_nom, couleur_valeur);
1942
1943   sprintf(str_temp,"%d Fintz",j_anneau_joueurs->int_argent);
1944   //Ecriture du texte "argent"
1945   surf_texte[2] = TTF_RenderText_Blended(police, "Argent :", couleur_texte);
1946
1947   //Ecriture de la quantité d'argent du joueur
1948   surf_texte[3] = TTF_RenderText_Blended(police, str_temp, couleur_valeur);
1949
1950   sprintf(str_temp,"Nombre de certificat : %d",j_anneau_joueurs->int_certificat);
1951   //Ecriture du nombre de certficat
1952   surf_texte[4] = TTF_RenderText_Blended(police, str_temp, couleur_texte);
1953
1954   //Fermeture de la police d'ecriture
1955   TTF_CloseFont(police);
1956
1957   //on affiche l'image du jeton
1958   position_jeton.x=130;
1959   position_jeton.y=25;
1960   SDL_BlitSurface(j_anneau_joueurs->surf_image, NULL, surf_fond, &position_jeton);
1961
1962   position.x=3;
1963   position.y=3;
1964   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
1965
1966   for(i=0;i<5;i++)
1967   {
1968     position.x=PANNEAU_JOUEUR_POS_X+5+(15*(i%2));
1969     position.y=PANNEAU_JOUEUR_POS_Y-95+25*i-5*(i%2);
1970     SDL_BlitSurface(surf_texte[i], NULL, surf_menu, &position);
1971   }
1972
1973   position.x=PANNEAU_JOUEUR_POS_X;
1974   position.y=PANNEAU_JOUEUR_POS_Y;
1975   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
1976
1977   SDL_Flip(surf_ecran);
1978   SDL_FreeSurface(surf_menu);
1979   SDL_FreeSurface(surf_fond);
1980   for(i=0;i<3;i++) SDL_FreeSurface(surf_texte[i]);
1981 }
```

8.12.1.28 void affich_panneau_possessions (SDL_Surface * *surf_ecran*, joueur * *j_anneau_joueurs*)

affiche le panneau des possessions du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

j_ anneau_ joueurs joueur en cours

Version :

1.0

Renvoie :

++

Définition à la ligne 1983 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
1984 {
1985   //Surface du panneau
1986   SDL_Surface* surf_menu;
1987   surf_menu=NULL;
1988   SDL_Surface* surf_fond;
1989   surf_fond=NULL;
1990   SDL_Surface* surf_texte;
1991   surf_texte=NULL;
1992   SDL_Rect position;
1993   SDL_Surface* surf_propriete;
1994   surf_propriete=NULL;
1995   //Police d'écriture
1996   TTF_Font* police;
1997   police=NULL;
1998   possession* propriete;
1999   int nombre_propriete;
2000   int propriete_actuel;
2001   SDL_Color couleur_texte = {86, 255, 86};
2002   propriete=j_anneau_joueurs->propriete;
2003   propriete_actuel=0;
2004   nombre_propriete=0;
2005
2006   if(!propriete)
2007   {
2008     //Création de la surface du menu
2009     surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR, PANNEAU_POSSESSION_HAUTEUR, 32, 0, 0
2010     //Remplissage de noir pour faire le cadre
2011     SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2012
2013     //Création de la surface du fond du menu
2014     surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR-6, PANNEAU_POSSESSION_HAUTEUR-6, 32, 0
2015     //Remplissage de noir pour faire le cadre
2016     SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2017
2018     //Ouverture de la police d'ecriture
2019     police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
2020
2021     //Ecriture du texte "nom du joueur"
2022     surf_texte = TTF_RenderText_Blended(police, "Aucun titre de propriété.", couleur_texte);
2023
2024     //Fermeture de la police d'ecriture
2025     TTF_CloseFont(police);
2026
2027     position.x=3;
2028     position.y=3;
2029     SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2030
2031   }
2032   else
2033   {
2034     //Pour chaque propriete
2035     while(propriete)
2036     {
```

```
2037      //On ajoute 1 au nombre de propriété
2038      nombre_propriete++;
2039      //On passe à la propriété suivante
2040      propriete=propriete->suivant;
2041    }
2042    //Création de la surface du panneau
2043    surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR, 40+((PROPRIETE_HAUTEUR+5)*nombre_prop
2044    //Remplissage de noir pour faire le cadre
2045    SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2046
2047    //Création de la surface du fond du panneau
2048    surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR-6, 40+((PROPRIETE_HAUTEUR+5)*nombre_pr
2049    //Remplissage de noir pour faire le cadre
2050    SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2051
2052    //Ouverture de la police d'ecriture
2053    police = TTF_OpenFont("sdl/police/police2-bold.ttf", 17);
2054
2055    //Ecriture du texte
2056    surf_texte = TTF_RenderText_Blended(police, "Propriété de l'élève :", couleur_texte);
2057
2058    //Fermeture de la police d'ecriture
2059    TTF_CloseFont(police);
2060
2061    //Position du fond sur le menu
2062    position.x=3;
2063    position.y=3;
2064    //Collage du fond sur le menu
2065    SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2066
2067    //Retour à l'entete de la liste des propriétés du joueur
2068    propriete=j_anneau_joueurs->propriete;
2069
2070    //tant qu'on a des propriétés
2071    while(propriete)
2072    {
2073      //Chargement de la surface en fonction de la propriété
2074      if(propriete->propriete->int_type==SALLE) surf_propriete=propriete->propriete->case_salle.surf_propriete;
2075      else if(propriete->propriete->int_type==BDE || propriete->propriete->int_type==BDS) surf_propriete=propriete
2076      else surf_propriete=propriete->propriete->case_lieu_commun.surf_propriete;
2077
2078      //Position de la propriété dans le menu
2079      position.x=10;
2080      position.y=35+propriete_actuel*(PROPRIETE_HAUTEUR+5);
2081
2082      //Affichage de la propriété
2083      SDL_BlitSurface(surf_propriete, NULL, surf_menu, &position);
2084      //Passage à la propriété suivante
2085      propriete=propriete->suivant;
2086      //Incrémentation du nombre de propriété actuel
2087      propriete_actuel++;
2088    }
2089  }
2090
2091  //Position du texte sur le menu
2092  position.x=20;
2093  position.y=10;
2094  //Collage du texte sur le menu
2095  SDL_BlitSurface(surf_texte, NULL, surf_menu, &position);
2096
2097  //Position du panneau à l'écran
2098  position.x=PANNEAU_POSSESSION_POS_X;
2099  position.y=PANNEAU_POSSESSION_POS_Y;
2100  //Affichage du panneau à l'écran
2101  SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2102
2103  //Mise à jour de l'écran
```

```
2104   SDL_Flip(surf_ecran);
2105
2106   //Libération des surfaces temporaires
2107   SDL_FreeSurface(surf_menu);
2108   SDL_FreeSurface(surf_fond);
2109   SDL_FreeSurface(surf_texte);
2110 }
2111
```

8.12.1.29 void affich_possessions_cache (SDL_Surface * *surf_ecran*)

affiche le cache du panneau de possesion du joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

Version :

1.0

Renvoie :

++

Définition à la ligne 2113 du fichier sdl.c.

Référencé par jeu().

```
2113 {
2114   //Surface du cache
2115   SDL_Surface* surf_cache;
2116   surf_cache=NULL;
2117
2118   //Position
2119   SDL_Rect position;
2120
2121   //Création de la surface du menu
2122   surf_cache=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_POSSESSION_LARGEUR, ECRAN_HAUTEUR-PANNEAU_POSSESSION_POS_Y
2123   //Remplissage de noir pour faire le cadre
2124   SDL_FillRect(surf_cache, NULL, SDL_MapRGB(surf_ecran->format, 25, 25, 75));
2125
2126   //Position du cache
2127   position.x=PANNEAU_POSSESSION_POS_X;
2128   position.y=PANNEAU_POSSESSION_POS_Y;
2129
2130   //Collage du cache sur l'écran
2131   SDL_BlitSurface(surf_cache, NULL, surf_ecran, &position);
2132
2133 }
2134
```

8.12.1.30 void affich_panneau_des_bouton (SDL_Surface * *surf_ecran*, int *int_image*)

affiche le panneau contenant le bouton lancer dÃľ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran
int_image score du dÃľ

Version :

1.0

Renvoie :

++

Définition à la ligne 2136 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
2136 {
2137   //Surface du panneau
2138   SDL_Surface* surf_menu;
2139   //Surface du fond
2140   SDL_Surface* surf_fond;
2141   //Surface des boutons lancer dés
2142   SDL_Surface* surf_bouton[2];
2143   //Position
2144   SDL_Rect position;
2145
2146   int i;
2147
2148   //Création de la surface du menu
2149   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR, PANNEAU_DES_HAUTEUR, 32, 0, 0, 0, 0);
2150   //Remplissage de noir pour faire le cadre
2151   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2152
2153   //Création de la surface du fond du menu
2154   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR-6, PANNEAU_DES_HAUTEUR-6, 32, 0, 0, 0, 0);
2155   //Remplissage de noir pour faire le cadre
2156   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2157
2158   //Chargement des images bes boutons
2159   surf_bouton[0]=IMG_Load("sdl/images/lancer.gif");
2160   surf_bouton[1]=IMG_Load("sdl/images/lancer2.gif");
2161
2162   //Position du fond sur le panneau
2163   position.x=3;
2164   position.y=3;
2165   //Collage du fond sur le panneau
2166   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2167
2168   //Position du bouton sur le panneau
2169   position.x=(PANNEAU_DES_LARGEUR-surf_bouton[0]->w)/2;
2170   position.y=(PANNEAU_DES_HAUTEUR-surf_bouton[0]->h)/2;
2171   //Collage du bouton sur le panneau
2172   SDL_BlitSurface(surf_bouton[int_image], NULL, surf_menu, &position);
2173
2174   //Position du panneau sur l'écran
2175   position.x=PANNEAU_DES_POS_X;
2176   position.y=PANNEAU_DES_POS_Y;
2177   //Affichage du panneau
2178   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2179
2180   //Mise à jour de l'écran
2181   SDL_Flip(surf_ecran);
```

```
2182
2183   //Libération des surfaces temporaires
2184   SDL_FreeSurface(surf_menu);
2185   SDL_FreeSurface(surf_fond);
2186   for(i=0;i<2;i++) SDL_FreeSurface(surf_bouton[i]);
2187 }
2188
```

8.12.1.31 void affich_panneau_des (SDL_Surface * *surf_ecran*, SDL_Surface ** *surf_des*, int *int_de1*, int *int_de2*)

affiche le panneau du menu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

surf_des tableau des surfaces des face du dÃľ

int_de1 score du dÃľ numÃľro 1

int_de2 score du dÃľ numÃľro 2

Version :

1.0

Renvoie :

++

Définition à la ligne 2268 du fichier sdl.c.

Référencé par lancer_des().

```
2268 {
2269   //Surface du menu
2270   SDL_Surface* surf_menu;
2271   //Surface du fond
2272   SDL_Surface* surf_fond;
2273
2274   //Position
2275   SDL_Rect position;
2276
2277   //Création de la surface du menu
2278   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR, PANNEAU_DES_HAUTEUR, 32, 0, 0, 0, 0);
2279   //Remplissage de noir pour faire le cadre
2280   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2281
2282   //Création de la surface du fond du menu
2283   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_DES_LARGEUR-6, PANNEAU_DES_HAUTEUR-6, 32, 0, 0, 0, 0);
2284   //Remplissage de noir pour faire le cadre
2285   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2286
2287   //Position du fond
2288   position.x=3;
2289   position.y=3;
2290   //Affichage du fond
2291   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2292
2293   //Position du premier dé
```

```
2294   position.x=40;
2295   position.y=(PANNEAU_DES_HAUTEUR-surf_des[0]->h)/2;
2296   //Affichage du premier dé
2297   SDL_BlitSurface(surf_des[int_de1-1], NULL, surf_menu, &position);
2298
2299   //Position du second dé
2300   position.x=130;
2301   position.y=(PANNEAU_DES_HAUTEUR-surf_des[0]->h)/2;
2302   //Affichage du second dé
2303   SDL_BlitSurface(surf_des[int_de2-1], NULL, surf_menu, &position);
2304
2305   //Position du menu
2306   position.x=PANNEAU_DES_POS_X;
2307   position.y=PANNEAU_DES_POS_Y;
2308   //Affichage du menu
2309   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2310
2311   //Mise à jour de l'écran
2312   SDL_Flip(surf_ecran);
2313
2314   //Libération des surfaces temporaires
2315   SDL_FreeSurface(surf_menu);
2316   SDL_FreeSurface(surf_fond);
2317 }
2318
```

8.12.1.32 void affich_panneau_fdt (SDL_Surface * *surf_ecran*, bool *bool_etat*)

affiche le panneau du menu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

bool_etat Ãľtat des boutons

Version :

1.0

Renvoie :

++

Définition à la ligne 2215 du fichier sdl.c.

Référencé par attente_clic(), et jeu().

```
2215 {
2216   //Surface du panneau
2217   SDL_Surface* surf_menu;
2218   //Surface du fond du panneau
2219   SDL_Surface* surf_fond;
2220   //Surface du bouton de fin de tour
2221   SDL_Surface* surf_bouton;
2222   //Position
2223   SDL_Rect position;
2224
2225   //Création de la surface du menu
2226   surf_menu=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_FDT_LARGEUR, PANNEAU_FDT_HAUTEUR, 32, 0, 0, 0, 0);
```

```
2227   //Remplissage de noir pour faire le cadre
2228   SDL_FillRect(surf_menu, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 0));
2229
2230   //Création de la surface du fond du menu
2231   surf_fond=SDL_CreateRGBSurface(SDL_HWSURFACE, PANNEAU_FDT_LARGEUR-6, PANNEAU_FDT_HAUTEUR-6, 32, 0, 0, 0, 0);
2232   //Remplissage de noir pour faire le cadre
2233   SDL_FillRect(surf_fond, NULL, SDL_MapRGB(surf_ecran->format, 0, 0, 125));
2234
2235   //Bouton en surbrillance
2236   if(bool_etat==true) surf_bouton=IMG_Load("sdl/images/finir2.gif");
2237   //Bouton normal
2238   else surf_bouton=IMG_Load("sdl/images/finir.gif");
2239
2240   //Position du fond sur le panneau
2241   position.x=3;
2242   position.y=3;
2243   //Collage du fond sur le panneau
2244   SDL_BlitSurface(surf_fond, NULL, surf_menu, &position);
2245
2246   //Position du bouton
2247   position.x=(PANNEAU_FDT_LARGEUR-surf_bouton->w)/2;
2248   position.y=(PANNEAU_FDT_HAUTEUR-surf_bouton->h)/2;
2249   //Collage du bouton sur le panneau
2250   SDL_BlitSurface(surf_bouton, NULL, surf_menu, &position);
2251
2252   //Position du panneau à l'écran
2253   position.x=PANNEAU_FDT_POS_X;
2254   position.y=PANNEAU_FDT_POS_Y;
2255   //Collage du panneau sur l'écran
2256   SDL_BlitSurface(surf_menu, NULL, surf_ecran, &position);
2257
2258   //Mise à jour de l'écran
2259   SDL_Flip(surf_ecran);
2260
2261   //Libération des surfaces temporaires
2262   SDL_FreeSurface(surf_menu);
2263   SDL_FreeSurface(surf_fond);
2264   SDL_FreeSurface(surf_bouton);
2265 }
2266
```

8.12.1.33 int affich_validation_propriete (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_fond*, int *int_etat*, int *int_nombre_boutons*, ...)

affiche le message des propriÃľtÃľ ainsi que les boutons d'actions possibles

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

surf_fond surface de la propriÃľtÃľ

int_etat Ãľtat des boutons

int_nombre_boutons nombre de boutons Ãă afficher

... liste des boutons Ãă afficher

Version :

1.0

Renvoie :

un entier d'Ãl'tat

Définition à la ligne 2335 du fichier sdl.c.

Référencé par action_to_boutons().

```
2335 {
2336   //Surface du bouton
2337   SDL_Surface** surf_bouton;
2338 
2339   int i;
2340 
2341   //Nombre de bouton
2342   int nombre_boutons;
2343   nombre_boutons=int_nombre_boutons;
2344 
2345   //Si jamais il n'y a pas de boutons
2346   if(nombre_boutons<=0) return(0);
2347 
2348   //Un crée un tableau de bouton correspondant au nombres de boutons et à leurs états
2349   surf_bouton= new SDL_Surface*[nombre_boutons];
2350 
2351   //Position
2352   SDL_Rect position;
2353 
2354   //Valeur de l'argument
2355   int int_valeur;
2356 
2357   //Liste des arguments
2358   va_list arguments;
2359 
2360   //Initialisation de la liste des arguments
2361   va_start(arguments, int_nombre_boutons);
2362 
2363   //Position du message à l'écran
2364   position.x=DETAIL_MESSAGE_POS_X;
2365   position.y=DETAIL_MESSAGE_POS_Y;
2366   //Affichage du message
2367   SDL_BlitSurface(surf_fond, NULL, surf_ecran, &position); // Collage de la surface sur l'écran
2368 
2369   //Tant qu'on a des boutons
2370   while(nombre_boutons!=0)
2371   {
2372     //Récupération de l'argument
2373     int_valeur=va_arg(arguments, int);
2374     //Selon le bouton à afficher
2375     switch(int_valeur)
2376     {
2377       //Dans le cas d'un achat
2378       case BTN_ACHAT:
2379         //Charger l'image en surbrillance
2380         if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/achete
2381         //Ou l'image normale
2382         else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/acheter.gif");
2383         break;
2384       //Dans le cas d'un hypothèque
2385       case BTN_HYPOTHEQUE:
2386         //Charger l'image en surbrillance
2387         if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/hypoth
2388         //Ou l'image normale
2389         else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/hypotheque.gif");
2390         break;
2391       //Dans le cas d'une déshypothèque
2392       case BTN_UNHYPOTHEQUE:
2393         //Charger l'image en surbrillance
2394         if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/deshyp
```

```
2395         //Ou l'image normale
2396         else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/deshypotheque.gif");
2397         break;
2398       case BTN_MOINS:
2399         //Charger l'image en surbrillance
2400         if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/moins2
2401         //Ou l'image normale
2402         else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/moins.gif");
2403         break;
2404       case BTN_PLUS:
2405         //Charger l'image en surbrillance
2406         if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[0]=IMG_Load("sdl/images/plus2.gif");
2407         //Ou l'image normale
2408         else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/plus.gif");
2409         break;
2410       case BTN_FINIR:
2411         //Charger l'image en surbrillance
2412         if(int_etat==int_nombre_boutons-nombre_boutons+1) surf_bouton[0]=IMG_Load("sdl/images/annuler2.gif");
2413         //Ou l'image normale
2414         else surf_bouton[nombre_boutons-1]=IMG_Load("sdl/images/annuler.gif");
2415         break;
2416     }
2417     //Passage au bouton suivant
2418     nombre_boutons--;
2419     //Position du bouton
2420     position.x=DETAIL_BOUTON_POS_X;
2421     position.y=DETAIL_BOUTON_POS_Y+50*(int_nombre_boutons-(nombre_boutons+1));
2422     //On colle le bouton sur le message
2423     SDL_BlitSurface(surf_bouton[nombre_boutons], NULL, surf_ecran, &position);
2424   }
2425
2426   //Destruction de la liste d'argument
2427   va_end(arguments);
2428
2429   //Mise à jour de l'écran
2430   SDL_Flip(surf_ecran);
2431
2432   //Libération des surfaces temporaire
2433   for(i=0;i<int_nombre_boutons;i++) SDL_FreeSurface(surf_bouton[i]);
2434   //Libération de la mémoire
2435   delete[] surf_bouton;
2436
2437   //On retourne 0
2438   return(0);
2439 }
2440
```

8.12.1.34 void affich_joueur_depart (SDL_Surface * *surf_ecran*, cases ** *plateau*, joueur * *j_anneau_joueurs*, int *nombre_joueur*)

affiche tout les joueurs sur la case de dÃľpart

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

plateau plateau de jeu

j_anneau_joueurs liste chaÃőnÃľe des joueurs

nombre_joueur nombre de joueur

Version :

1.0

Renvoie :

action ud joueur

Définition à la ligne 2320 du fichier sdl.c.

Référencé par jeu().

```
2320 {
2321   //pour chaque joueurs
2322   for (int i=0;i<nombre_joueur;i++)
2323   {
2324     //on l'affiche sur la case
2325     affich_joueur(surf_ecran,j_anneau_joueurs,i, plateau[j_anneau_joueurs->int_position]);
2326
2327     //on passe au joueur suivant
2328     j_anneau_joueurs=j_anneau_joueurs->pjoueur_suivant;
2329   }
2330   //on depart au premier joueur pour que le jeu puisse commencer avec
2331   j_anneau_joueurs=j_anneau_joueurs->pjoueur_suivant;
2332 }
2333
```

8.12.1.35 void affich_message (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_message*, int *int_type_message*, int *int_etat*)

affiche un message Ãă l'Ãľcran

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'Ãľcran

surf_message surface du message

int_type_message type du message Ãă afficher

int_etat Ãľtat des boutons

Version :

1.0

Renvoie :

rien

Définition à la ligne 2568 du fichier sdl.c.

Référencé par attente_validation_message().

```
2568 {
2569   //Surface du bouton
2570   SDL_Surface* surf_bouton[3];
2571
2572   int i;
2573
2574   //Position
```

```
2575   SDL_Rect position;
2576
2577   //Initialisation des trois boutons
2578   for(i=0;i<3;i++) surf_bouton[i]=NULL;
2579
2580   //S'il s'agit d'un message quitter
2581   if(int_type_message==MESSAGE_QUITTER)
2582   {
2583     //Chargement de l'image en surbrillance
2584     if(int_etat==1) surf_bouton[0]=IMG_Load("sdl/images/exit2.gif");
2585     //En normal
2586     else surf_bouton[0]=IMG_Load("sdl/images/exit.gif");
2587     //Chargement de l'image en surbrillance
2588     if(int_etat==2) surf_bouton[1]=IMG_Load("sdl/images/sauvegarder2.gif");
2589     //En normal
2590     else surf_bouton[1]=IMG_Load("sdl/images/sauvegarder.gif");
2591
2592     //Position du bouton à l'écran
2593     position.x=(MESSAGE_LARGEUR-(surf_bouton[0]->w)*2)/3;
2594     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2595     //Affichage du bouton
2596     SDL_BlitSurface(surf_bouton[0], NULL, surf_message, &position);
2597
2598     //Position du bouton à l'écran
2599     position.x=2*(MESSAGE_LARGEUR-(surf_bouton[0]->w))/3;
2600     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2601     //Affichage du bouton
2602     SDL_BlitSurface(surf_bouton[1], NULL, surf_message, &position);
2603   }
2604   //S'il s'agit d'un message prison
2605   else if(int_type_message==MESSAGE_PRISON)
2606   {
2607     //Chargement de l'image en surbrillance
2608     if(int_etat==1) surf_bouton[0]=IMG_Load("sdl/images/attendre2.gif");
2609     //En normal
2610     else surf_bouton[0]=IMG_Load("sdl/images/attendre.gif");
2611     //Chargement de l'image en surbrillance
2612     if(int_etat==2) surf_bouton[1]=IMG_Load("sdl/images/payer2.gif");
2613     //En normal
2614     else surf_bouton[1]=IMG_Load("sdl/images/payer.gif");
2615     //Chargement de l'image en surbrillance
2616     if(int_etat==3) surf_bouton[2]=IMG_Load("sdl/images/certificat2.gif");
2617     //En normal
2618     else surf_bouton[2]=IMG_Load("sdl/images/certificat.gif");
2619
2620     //Position du bouton à l'écran
2621     position.x=30;
2622     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2623     //Affichage du bouton
2624     SDL_BlitSurface(surf_bouton[0], NULL, surf_message, &position);
2625
2626     //Position du bouton à l'écran
2627     position.x=(MESSAGE_LARGEUR-(surf_bouton[2]->w))/2;
2628     position.y=MESSAGE_HAUTEUR-(surf_bouton[2]->h);
2629     //Affichage du bouton
2630     SDL_BlitSurface(surf_bouton[1], NULL, surf_message, &position);
2631
2632     //Position du bouton à l'écran
2633     position.x=MESSAGE_LARGEUR-30-(surf_bouton[2]->w);
2634     position.y=MESSAGE_HAUTEUR-(surf_bouton[2]->h);
2635     //Affichage du bouton
2636     SDL_BlitSurface(surf_bouton[2], NULL, surf_message, &position);
2637   }
2638   //S'il s'agit d'un message normal
2639   else
2640   {
2641     //Chargement de l'image en surbrillance
```

```
2642     if(int_etat==1) surf_bouton[0]=IMG_Load("sdl/images/valider2.gif");
2643     //En normal
2644     else surf_bouton[0]=IMG_Load("sdl/images/valider.gif");
2645     //Position du bouton à l'écran
2646     position.x=(MESSAGE_LARGEUR-(surf_bouton[0]->w))/2;
2647     position.y=MESSAGE_HAUTEUR-(surf_bouton[0]->h);
2648     //Affichage du bouton
2649     SDL_BlitSurface(surf_bouton[0], NULL, surf_message, &position);
2650   }
2651
2652   //Position du message à l'écran
2653   position.x=POS_X_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_LARGEUR)/2;
2654   position.y=POS_Y_PLATEAU+CASE_HAUTEUR+(TAILLE_CENTRE-MESSAGE_HAUTEUR)/2;
2655   //Affichage du message à l'écran
2656   SDL_BlitSurface(surf_message, NULL, surf_ecran, &position);
2657   //Mise à jour de l'écran
2658   SDL_Flip(surf_ecran);
2659
2660   //Libération des surfaces temporaires
2661   SDL_FreeSurface(surf_bouton[0]);
2662   if(int_type_message==MESSAGE_QUITTER || int_type_message==MESSAGE_PRISON) SDL_FreeSurface(surf_bouton[1]);
2663   if(int_type_message==MESSAGE_PRISON) SDL_FreeSurface(surf_bouton[2]);
2664 }
2665
```

8.13 Référence du fichier structure.h

toutes les structures necessaires pour le monopoly

Structures de données

- struct **rvb_couleur**
- struct **information**
- struct **possession**
- struct **joueur**
- struct **cases**

8.13.1 Description détaillée

toutes les structures necessaires pour le monopoly

Auteur :

Franck.trey <franck.trey@eisti.fr>

Date :

03/12/2007

Version :

1.0

Définition dans le fichier **structure.h**.

8.14 Référence du fichier traitement.c

code source des fonctions de traitement.

Fonctions

- void **jeu** (SDL_Surface *surf_ecran, **joueur** *pj_joueur_encours, **cases** **plateau, int nombre_joueur, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  initialise le jeu

- int **lancer_des** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau)
  fonction chargée de simuler le lancement des dés

- void **reculer_jeton** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  recule le joueur de 3 cases

- void **avancer_jeton** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, int int_nb_tire, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  avance le jeton case par case, traite le passage par la case départ

- void **aller_a_jeton** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, int int_position_voulue, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16], int delai)
  avance le jeton case par case, traite le passage par la case départ

- void **aller_en_prison_jeton** (SDL_Surface *surf_ecran, **joueur** *pj_joueur, **cases** **plateau)
  envoi le joueur directement en prison

- bool **traitement_achat** (**cases** *case_achetee, **joueur** *joueur_actuel)
  traitement l'achat d'une propriété

- void **traitement_case_depart** (SDL_Surface *surf_ecran, **joueur** *pj_joueur)
  ajoute l'argent au joueur

- bool **traitement_payer_loyer** (**joueur** *joueur_qui_paye, **joueur** *joueur_paye, int int_montant)
  paye les loyer d'une salle

- bool **traitement_mise_en_hypotheque** (**joueur** *joueur_actuel, **cases** *case_hypothequee)
  met en place une hypothèque

- int **traitement_loyer_association** (**joueur** *joueur_paye, int int_nb_tire)
  calcul le loyer d'une association

- int **traitement_loyer_lieu_commun** (**joueur** *pj_joueur)
  calcul le loyer pour les lieux communs

- bool **traitement_rachat_hypotheque** (**joueur** *joueur_actuel, **cases** *case_hypothequee)
  rachète une hypothèque

- bool **traitement_augmentation_niveau** (**cases** *pcase, **joueur** *pj_joueur)
  augmente le niveau d'une case

- void **traitement_diminution_niveau** (**cases** *pcase, **joueur** *pj_joueur)
  diminue le niveau d'une case

- void **traitement_arrive_case** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16], int int_nombre_tire)
  effectue les action necessaire lors de l'arrivée sur une case en fonction de son type

- void **traitement_soiree** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau)
  fait payer le joueur suivant la case soirée sur laquelle il est

- void **traitement_machine_a_cafe** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur)
  donne au joueur l'argent de la machine à café

- void **traitement_bureau_laurence** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  effectue les traitement necessaire lorsqu'un joueur est dans le bureau de laurence

- void **traitement_bureau** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur, **information** *le_bureau[16], **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16], int int_id_bureau)
  tire une carte du bureau voulue et effectue le traitement en fonction de la carte tirée

- void **traitement_elimination_joueur** (**joueur** *joueur_eliminer)
  libère les propriété et l'espace mémoire du joueur et de ses possessions

- void **traitement_perdu** (**joueur** *pj_joueur)

8.14.1 Description détaillée

code source des fonctions de traitement.

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **traitement.c**.

8.14.2 Documentation des fonctions

8.14.2.1 void jeu (SDL_Surface * *surf_ecran*, joueur * *pj_joueur_encours*, cases ** *plateau*, int *nombre_joueur*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

initialise le jeu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pj_joueur_encours joueur qui joue actuellement

plateau tableau contenant les cases du plateau

nombre_joueur nombre de joueurs qui jouent

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 12 du fichier traitement.c.

Référencé par main().

```
13 {
14   int i;
15
16   //Valeur de retour
17   int int_retour;
18
19   //Surface du centre
20   SDL_Surface* surf_centre = NULL;
21
22   //Création du centre
23   surf_centre=SDL_LoadBMP("sdl/images/logo.bmp");
24
25   //Affichage des cases du plateau
26   for(i=0;i<40;i++) affich_case(surf_ecran, plateau[i]);
27
28   //joueur_possede_tout(plateau, pj_joueur_encours);                          //<<<==================================
29
30   //on affiche les joueurs sur la cases depart
31   affich_joueur_depart (surf_ecran,plateau,pj_joueur_encours,nombre_joueur);
32   //Affichage du centre
33   affich_centre(surf_ecran, surf_centre);
34
35   //Affichage du panneau du menu
36   affich_panneau_menu(surf_ecran,0);
37
38   //Affichage du panneau contenant les informations du premier joueur
39   affich_panneau_joueur(surf_ecran,pj_joueur_encours);
40
41   //Affichage du panneau du bouton lancer dés
42   affich_panneau_des_bouton(surf_ecran,0);
43
44   //Affichage du panneau de fin de tour
45   affich_panneau_fdt(surf_ecran,false);
46
47   //Affichage contenant les informations sur les possessions du joueur
48   affich_panneau_possessions(surf_ecran,pj_joueur_encours);
49
50   //Attente du premier événement souris
51   int_retour=attente_clic(surf_ecran, surf_centre, pj_joueur_encours,plateau,bureau_de_krystel,bureau_de_nadege);
```

```
52   if(int_retour==0)
53   {
54     int_retour=attente_validation_message(surf_ecran, surf_centre, NULL, "Etes vous sûr de vouloir réellement envisag
55
56   }
57   //Tant que le joueur ne décide pas de quitter
58   while(int_retour)
59   {
60     //si le joueur n'a pas fait de double on passe au joueur suivant
61     if (pj_joueur_encours->int_double_tire == 0)
62     {
63       //Passage au joueur suivant
64       pj_joueur_encours=pj_joueur_encours->pjoueur_suivant;
65     }
66
67     //on affiche le panneau du nouveau joueur
68     affich_panneau_joueur(surf_ecran,pj_joueur_encours);
69     //Affichage du cache pour le panneau des possession
70     affich_possessions_cache(surf_ecran);
71     //On réaffiche le panneau des possessions
72     affich_panneau_possessions(surf_ecran,pj_joueur_encours);
73     //Affichage du panneau du bouton lancer dés
74     affich_panneau_des_bouton(surf_ecran,0);
75     //Mise à jour de l'écran
76     SDL_Flip(surf_ecran);
77     //Attente d'un événement souris
78     int_retour=attente_clic(surf_ecran, surf_centre, pj_joueur_encours, plateau, bureau_de_krystel, bureau_de_nadege)
79     //Si le joueur souhaite quitter
80     if(int_retour==0)
81     {
82       //Demande de confirmation d'action
83       int_retour=attente_validation_message(surf_ecran, surf_centre, NULL, "Etes vous sûr de vouloir réellement envis
84       //Si le joueur demande de sauvegarder
85       if(int_retour==3)
86       {
87         //On sauvegarde
88         sauvegarde(plateau, pj_joueur_encours);
89         //Message de confirmation de sauvegarde
90         attente_validation_message(surf_ecran, surf_centre, "Sauvegarde", "La sauvegarde à été effectué avec un très
91         //Fin du jeu
92         int_retour=0;
93       }
94     }
95     //on vérifie que le joueur n'a pas gagné
96     if(verification_victoire(pj_joueur_encours))
97     {
98       //Message de victoire
99       attente_validation_message(surf_ecran,surf_centre,"VICTOIRE", "Félicitation vous avez ruiné tous vos adversaire
100       //Fin du jeu
101       int_retour=0;
102     }
103   }
104 }
```

8.14.2.2 int lancer_des (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*)

fonction chargée de simuler le lancement des dés

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

le nombre obtenue par les dés

Définition à la ligne 106 du fichier traitement.c.

Référencé par attente_clic(), et traitement_bureau_laurence().

```
107 {
108   int int_1; //valeur du premier dé
109   int int_2; //valeur du second dé
110   int int_rebond1; //nb de rebonds du premier dé
111   int int_rebond2; //nb de rebonds du second dé
112 
113   int i=0;
114 
115   SDL_Surface** surf_des;
116 
117   //calcul du nombre de rebond du premier dé
118   int_rebond1=rand()%10+5;
119 
120   //calcul de nombre de rebond du second dé
121   int_rebond2=rand()%10+5;
122 
123   //initialisation a 0 des deux dés
124   int_1=0;
125   int_2=0;
126 
127   surf_des=creation_des();
128 
129   if (int_rebond1 > int_rebond2)
130   {
131     while (i < int_rebond2)
132     {
133       int_1=(rand()%6)+1;
134       int_2=(rand()%6)+1;
135 
136       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
137       SDL_Delay(DELAY);
138       i++;
139     }
140     for(i=0;i<(int_rebond1-int_rebond2-1);i++)
141     {
142       int_1=(rand()%6)+1;
143 
144       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
145       SDL_Delay(DELAY);
146     }
147   }
148   else
149   {
150     while (i < int_rebond1)
151     {
152       int_1=(rand()%6)+1;
153       int_2=(rand()%6)+1;
```

```
154
155       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
156       SDL_Delay(DELAY);
157       i++;
158     }
159     for(i=0;i<(int_rebond1-int_rebond2-1);i++)
160     {
161       int_1=(rand()%6)+1;
162
163       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
164       SDL_Delay(DELAY);
165     }
166   }
167   if (int_1==int_2)
168   {
169     pj_joueur->int_double_tire++;
170   }
171   else
172   {
173     pj_joueur->int_double_tire=0;
174   }
175
176   //si le joueur a fait 3 doubles il va en prison
177   if (pj_joueur->int_double_tire == 3)
178   {
179     attente_validation_message(surf_ecran, surf_centre, "PRISON" , "Vous avez fait 3 doubles consecutif", MESSAGE_NO
180     aller_en_prison_jeton(surf_ecran, pj_joueur,plateau);
181     pj_joueur->int_double_tire = 0;
182   }
183
184   destruction_des(surf_des);
185   return(int_1+int_2);
186 }
```

8.14.2.3 void reculer_jeton (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

recule le joueur de 3 cases

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 188 du fichier traitement.c.

Référencé par traitement_bureau().

```
189 {
190   int int_position;
191   int int_position2;
192   int i;
193 
194   i=0;
195 
196   joueur* pj_joueur_test;
197 
198   //on le fait avancer case par case
199   for (i=0;i<3;i++)
200   {
201     //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
202     affich_case(surf_ecran,plateau[pj_joueur->int_position]);
203     //on avance le joueur de 1 en faisant attention que la position reste dans les 40 cases du plateau
204     pj_joueur->int_position=(pj_joueur->int_position+39)%40;
205     //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
206     int_position=nombre_joueur_case(pj_joueur);
207     //on affiche le joueur sur la nouvelle case
208     affich_joueur(surf_ecran, pj_joueur,int_position+1, plateau[pj_joueur->int_position]);
209     // on lui indique que le premier joueur test est le suivant
210     pj_joueur_test=pj_joueur->pjoueur_suivant;
211     int_position2=0;
212     //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
213     while (pj_joueur_test != pj_joueur)
214     {
215       if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
216       {
217         int_position2++;
218         //on l'affiche sur la case:
219         affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
220       }
221       pj_joueur_test=pj_joueur_test->pjoueur_suivant;
222     }
223     //on rafraichi l'affichage
224     SDL_Flip(surf_ecran);
225     //petit delay pour que l'utilisateur puisse apprécier l'affichage
226     SDL_Delay(DELAY);
227   }
228 
229   traitement_arrive_case(surf_ecran, surf_centre,plateau,pj_joueur, bureau_de_krystel, bureau_de_nadege, 0);
230 }
```

8.14.2.4 void avancer_jeton (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, int *int_nb_tire*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

avance le jeton case par case, traite le passage par la case départ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

int_nb_tire nombre obtenu lors du lancer des dés

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 233 du fichier traitement.c.

Référencé par attente_clic(), et traitement_bureau_laurence().

```
234 {
235   int int_position;
236   int int_position2;
237
238   joueur* pj_joueur_test;
239
240   //on le fait avancer case par case
241   for (int i=0;i<int_nb_tire;i++)
242   {
243     //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
244     affich_case(surf_ecran,plateau[pj_joueur->int_position]);
245     //on avance le joueur de 1 en faisant attention que la position reste dans les 40 cases du plateau
246     pj_joueur->int_position=(pj_joueur->int_position+1)%40;
247     //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
248     int_position=nombre_joueur_case(pj_joueur);
249     //on affiche le joueur sur la nouvelle case
250     affich_joueur(surf_ecran, pj_joueur,int_position+1, plateau[pj_joueur->int_position]);
251     // on lui indique que le premier joueur test est le suivant
252     pj_joueur_test=pj_joueur->pjoueur_suivant;
253     int_position2=0;
254     //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
255     while (pj_joueur_test != pj_joueur)
256     {
257       if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
258       {
259         int_position2++;
260         //on l'affiche sur la case:
261         affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
262       }
263       pj_joueur_test=pj_joueur_test->pjoueur_suivant;
264     }
265     //on rafraichi l'affichage
266     SDL_Flip(surf_ecran);
267     //petit delay pour que l'utilisateur puisse apprécier l'affichage
268     SDL_Delay(DELAY);
269
270     //si il se trouve sur la case départ
271     if (pj_joueur->int_position==0)
272     {
273       //on effectue le traitement adequat
274       traitement_case_depart (surf_ecran,pj_joueur);
275     }
276   }
277   traitement_arrive_case(surf_ecran, surf_centre,plateau,pj_joueur, bureau_de_krystel, bureau_de_nadege, int_nb_tire
278 }
```

8.14.2.5 void aller_a_jeton (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, int *int_position_voulue*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16], int *delai*)

avance le jeton case par case, traite le passage par la case départ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface du centre du plateau

int_position_voulue position voulue

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

delai délai du déplacement entre deux cases

Version :

1.0

Renvoie :

rien

Définition à la ligne 280 du fichier traitement.c.

Référencé par traitement_bureau().

```
281 {
282   int int_position;
283   int int_position2;
284   joueur* pj_joueur_test;
285 
286   //on le fait avancer case par case
287   while (pj_joueur->int_position != int_position_voulue)
288   {
289     //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
290     affich_case(surf_ecran,plateau[pj_joueur->int_position]);
291     //on avance le joueur de 1 en faisant attention que la position reste dans les 40 cases du plateau
292     pj_joueur->int_position=(pj_joueur->int_position+1)%40;
293     //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
294     int_position=nombre_joueur_case(pj_joueur);
295     //on affiche le joueur sur la nouvelle case
296     affich_joueur(surf_ecran, pj_joueur, int_position+1, plateau[pj_joueur->int_position]);
297 
298     // on lui indique que le premier joueur test est le suivant
299     pj_joueur_test=pj_joueur->pjoueur_suivant;
300     int_position2=0;
301     //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
302     while (pj_joueur_test != pj_joueur)
303     {
304       if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
305       {
306         int_position2++;
307         //on l'affiche sur la case:
```

```
308         affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
309       }
310       pj_joueur_test=pj_joueur_test->pjoueur_suivant;
311     }
312     //on rafraichi l'affichage
313     SDL_Flip(surf_ecran);
314     //petit delay pour que l'utilisateur puisse apprécier l'affichage
315     SDL_Delay(delai);
316     //si il se trouve sur la case départ
317     if (pj_joueur->int_position==0)
318     {
319       //on effectue le traitement adequat
320       traitement_case_depart (surf_ecran,pj_joueur);
321     }
322   }
323   //on lance la fonction traitement associé à la case ou il s'arrete
324   //plateau[pj_joueur->int_position]->ptraitement(pj_joueur->int_position);
325
326   traitement_arrive_case(surf_ecran, surf_centre,plateau,pj_joueur, bureau_de_krystel, bureau_de_nadege, 0);
327 }
```

8.14.2.6 void aller_en_prison_jeton (SDL_Surface * *surf_ecran*, joueur * *pj_joueur*, cases ** *plateau*)

envoi le joueur directement en prison

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

rien

Définition à la ligne 329 du fichier traitement.c.

Référencé par lancer_des(), traitement_arrive_case(), et traitement_bureau().

```
330 {
331   int int_position;
332   int int_position2;
333
334   joueur* pj_joueur_test;
335
336
337   //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
338   affich_case(surf_ecran,plateau[pj_joueur->int_position]);
339
340   // on lui indique que le premier joueur test est le suivant
341   pj_joueur_test=pj_joueur->pjoueur_suivant;
342   int_position2=0;
343   //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
```

```
344   while (pj_joueur_test != pj_joueur)
345   {
346     if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
347     {
348       int_position2++;
349       //on l'affiche sur la case:
350       affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
351     }
352     pj_joueur_test=pj_joueur_test->pjoueur_suivant;
353   }
354
355   //on envoi le jeton directement en prison
356   pj_joueur->int_position=10;
357   //on indique qu'il est emprisonner
358   pj_joueur->bool_laurence=true;
359
360   //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
361   int_position=nombre_joueur_case(pj_joueur);
362
363   //on affiche le joueur sur la nouvelle case
364   affich_joueur(surf_ecran, pj_joueur, int_position+1, plateau[pj_joueur->int_position]);
365
366   //on rafraichi l'affichage
367   SDL_Flip(surf_ecran);
368 }
```

8.14.2.7 bool traitement_achat (cases * *case_achetee*, joueur * *joueur_actuel*)

traitement l'achat d'une propriété

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

case_achetee indice des la case achetée

joueur_actuel l'acheteur

Version :

1.0

Renvoie :

booléen qui indique si la fonction s'est bbien déroulée

Définition à la ligne 370 du fichier traitement.c.

Référencé par attente_validation_propriete(), et joueur_possede_tout().

```
371 {
372   //on verifie si la case n'appartient a aucun joueur
373
374   //on détermine le type de la case
375   if (case_achetee->int_type==SALLE)
376   {
377     //on verifie si le joueur a assez d'argent pour l'acheter et si la case n'appartient a personne
378     if ((joueur_actuel->int_argent >= case_achetee->case_salle.int_prix) && (case_achetee->case_salle.pjoueur_joueu
379     {
380       //on effectue la transaction
381       joueur_actuel->int_argent=joueur_actuel->int_argent-case_achetee->case_salle.int_prix;
382       joueur_actuel->propriete=insertion_bonne_place_propriete(joueur_actuel->propriete,case_achetee);
```

```
383       case_achetee->case_salle.pjoueur_joueur=joueur_actuel;
384
385       return(true);
386     }
387     else
388     {
389       return(false);
390     }
391   }
392   else
393   {
394     if((case_achetee->int_type == BDE) || (case_achetee->int_type == BDS))
395     {
396       //on verifie si le joueur a assez d'argent pour l'acheter
397       if ((joueur_actuel->int_argent >= case_achetee->case_association.int_prix) && (case_achetee->case_association
398       {
399         //on effectue la transaction
400         joueur_actuel->int_argent=joueur_actuel->int_argent-case_achetee->case_association.int_prix;
401         joueur_actuel->propriete=insertion_bonne_place_propriete(joueur_actuel->propriete,case_achetee);
402         case_achetee->case_association.pjoueur_joueur=joueur_actuel;
403
404         return(true);
405       }
406     }
407     else
408     {
409       if(    (case_achetee->int_type == LC_WC)
410         || (case_achetee->int_type == LC_RU)
411         || (case_achetee->int_type == LC_PARKING)
412         || (case_achetee->int_type == LC_ASCENSEUR))
413         {
414           //on verifie si le joueur a assez d'argent pour l'acheter
415           if ((joueur_actuel->int_argent >= case_achetee->case_lieu_commun.int_prix) && (case_achetee->case_lieu_co
416           {
417             //on effectue la transaction
418             joueur_actuel->int_argent=joueur_actuel->int_argent-case_achetee->case_lieu_commun.int_prix;
419             joueur_actuel->propriete=insertion_bonne_place_propriete(joueur_actuel->propriete,case_achetee);
420             case_achetee->case_lieu_commun.pjoueur_joueur=joueur_actuel;
421             return(true);
422           }
423           else
424           {
425             return(false);
426           }
427         }
428       }
429     }
430     return(false);
431 }
```

8.14.2.8 void traitement_case_depart (SDL_Surface * *surf_ecran*, joueur * *pj_joueur*)

ajoute l'argent au joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

rien

Définition à la ligne 434 du fichier traitement.c.

Référencé par aller_a_jeton(), et avancer_jeton().

```
435 {
436   //on ajoute 2000 Fintz au compte du joueur
437   pj_joueur->int_argent=(pj_joueur->int_argent)+2000;
438   pj_joueur->bool_debut=false;
439 }
```

8.14.2.9 bool traitement_payer_loyer (joueur * *joueur_qui_paye*, joueur * *joueur_paye*, int *int_montant*)

paye les loyer d'une salle

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_qui_paye joueur à qui on prend les sous

joueur_paye joueur à qui on donne les sous

int_montant montant de la transaction

Version :

1.0

Renvoie :

booléen indiquant si la fonction c'est bien dérouler

Définition à la ligne 442 du fichier traitement.c.

Référencé par traitement_arrive_case().

```
443 {
444   //si le joueur a assez d'argent
445   if (joueur_qui_paye->int_argent >= int_montant)
446   {
447     //il paye
448     joueur_qui_paye->int_argent=joueur_qui_paye->int_argent-int_montant;
449     joueur_paye->int_argent=joueur_paye->int_argent+int_montant;
450     return(true);
451   }
452   else
453   {
454     return(false);
455   }
456 }
```

8.14.2.10 bool traitement_mise_en_hypotheque (joueur * *joueur_actuel*, cases * *case_hypothequee*)

met en place une hypothèque

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_actuel joueur qui joue actuellement

case_hypothequee indice de la case à hypothéquer

Version :

1.0

Renvoie :

booléen indiquant si la fonction c'est bien dérouler

Définition à la ligne 458 du fichier traitement.c.

```
459 {
460   //on agit selon le type de la case
461   switch (case_hypothequee->int_type)
462   {
463     case SALLE:
464       //si la case n'est pas déja hypotheque
465       if (!(case_hypothequee->case_salle.bool_hypotheque))
466       {
467         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_salle.int_valeur_hypotheque;
468         case_hypothequee->case_salle.bool_hypotheque=true;
469         return(case_hypothequee->case_salle.bool_hypotheque);
470       }
471       break;
472 
473     case BDS:
474       if (!(case_hypothequee->case_association.bool_hypotheque))
475       {
476         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_association.int_valeur_hypothec
477         case_hypothequee->case_association.bool_hypotheque=true;
478         return(case_hypothequee->case_association.bool_hypotheque);
479       }
480       break;
481 
482     case BDE:
483       if (!(case_hypothequee->case_association.bool_hypotheque))
484       {
485         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_association.int_valeur_hypothec
486         case_hypothequee->case_association.bool_hypotheque=true;
487         return(case_hypothequee->case_association.bool_hypotheque);
488       }
489       break;
490 
491     case LC_WC:
492       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
493       {
494         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
495         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
496         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
497       }
498       break;
499 
```

```
500     case LC_ASCENSEUR:
501       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
502       {
503         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
504         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
505         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
506       }
507       break;
508
509     case LC_RU:
510       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
511       {
512         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
513         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
514         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
515       }
516       break;
517
518     case LC_PARKING:
519       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
520       {
521         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
522         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
523         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
524       }
525       break;
526
527     default:
528       return(false);
529       break;
530   }
531   return(false);
532 }
```

8.14.2.11 int traitement_loyer_association (joueur * *joueur_paye*, int *int_nb_tire*)

calcul le loyer d'une association

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_paye joueur à qui on doit les sous

int_nb_tire nombre indiqué par le dés

Version :

1.0

Renvoie :

le loyer à payer

Définition à la ligne 534 du fichier traitement.c.

Référencé par action_possible(), et traitement_arrive_case().

```
535 {
536   possession* propriete_test;
537   int i; //compteur
```

```
538
539   i=0;
540
541   propriete_test=joueur_paye->propriete;
542   //on comptabilise le nombre de case association du joueur payé
543   while (propriete_test != NULL)
544   {
545     if ((propriete_test->propriete->int_type == BDS) || (propriete_test->propriete->int_type == BDE))
546     {
547       i++;
548     }
549
550     propriete_test=propriete_test->suivant;
551   }
552   //on a ainsi compatibilisé le nombe d'association du joueur
553   //si il n'y a qu'une on multiplie les nombre des dés par 40
554   if (i == 1)
555   {
556     return(int_nb_tire*40);
557   }
558   else
559   {
560     return(int_nb_tire*100);
561   }
562 }
```

8.14.2.12 int traitement_loyer_lieu_commun (joueur * *pj_joueur*)

calcul le loyer pour les lieux communs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur à qui on doit les sous

Version :

1.0

Renvoie :

le montant du loyer

Définition à la ligne 564 du fichier traitement.c.

Référencé par action_possible(), et traitement_arrive_case().

```
565 {
566
567   possession* propriete_temp;
568   int i;
569   i=0;
570
571   propriete_temp=pj_joueur->propriete;
572
573   while(propriete_temp!=NULL)
574   {
575     i++;
576     propriete_temp=propriete_temp->suivant;
577   }
578
```

```
579   return(i*250);
580 }
```

8.14.2.13 bool traitement_rachat_hypotheque (joueur * *joueur_actuel*, cases * *case_hypothequee*)

rachète une hypothèque

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_actuel joueur qui joue actuellement

case_hypothequee indice de la case à déshypothéquer

Version :

1.0

Renvoie :

booléen indiquant si la fonction c'est bien dérouler

Définition à la ligne 582 du fichier traitement.c.

```
583 {
584   //on agit selon le type de la case
585   switch (case_hypothequee->int_type)
586   {
587     case SALLE:
588       //si le joueur a assez d'argent
589       if (joueur_actuel->int_argent >= case_hypothequee->case_salle.int_valeur_hypotheque+case_hypothequee->case_sal
590       {
591       //il rachete l'hypothèque
592       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_salle.int_valeur_hypothequ
593       case_hypothequee->case_salle.bool_hypotheque=false;
594       }
595       return(true);
596       break;
597
598     case BDS:
599       //si le joueur a assez d'argent
600       if (joueur_actuel->int_argent >= (case_hypothequee->case_association.int_valeur_hypotheque+case_hypothequee->c
601       {
602       //il rachete l'hypothèque
603       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_association.int_valeur_hyp
604       case_hypothequee->case_association.bool_hypotheque=false;
605       }
606       return(true);
607       break;
608
609     case BDE:
610       //si le joueur a assez d'argent
611       if (joueur_actuel->int_argent >= (case_hypothequee->case_association.int_valeur_hypotheque+case_hypothequee->c
612       {
613       //il rachete l'hypothèque
614       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_association.int_valeur_hyp
615       case_hypothequee->case_association.bool_hypotheque=false;
616       }
617       return(true);
618       break;
```

```
619
620     case LC_WC:
621       //si le joueur a assez d'argent
622       if (joueur_actuel->int_argent >=(case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->ca
623       {
624       //il rachete l'hypothèque
625       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
626       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
627       }
628       return(true);
629       break;
630
631     case LC_ASCENSEUR:
632       //si le joueur a assez d'argent
633       if (joueur_actuel->int_argent >= (case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->
634       {
635       //il rachete l'hypothèque
636       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
637       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
638       }
639       return(true);
640       break;
641
642     case LC_RU:
643       //si le joueur a assez d'argent
644       if (joueur_actuel->int_argent >= (case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->
645       {
646       //il rachete l'hypothèque
647       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
648       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
649       }
650       return(true);
651       break;
652
653     case LC_PARKING:
654       //si le joueur a assez d'argent
655       if (joueur_actuel->int_argent >= (case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->
656       {
657       //il rachete l'hypothèque
658       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
659       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
660       }
661       return(true);
662       break;
663
664     default:
665       return(false);
666       break;
667   }
668   return(false);
669 }
```

8.14.2.14 bool traitement_augmentation_niveau (cases * *pcase*, joueur * *pj_joueur*)

augmente le niveau d'une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

pcase case dont on doit changer le niveau

Version :

1.0

Renvoie :

booléen indiquant si la fonction s'est bien déroulée

Définition à la ligne 671 du fichier traitement.c.

Référencé par attente_validation_propriete().

```
672 {
673   //on vérifie si le joueur a assez d'argent
674   if (pj_joueur->int_argent >= pcase->case_salle.int_prix_niveau)
675   {
676     //il les achète
677     pj_joueur->int_argent=(pj_joueur->int_argent) - (pcase->case_salle.int_prix_niveau);
678     pcase->case_salle.int_niveau=pcase->case_salle.int_niveau + 1;
679     return(true);
680   }
681   else
682   {
683     return(false);
684   }
685 }
```

8.14.2.15 void traitement_diminution_niveau (cases * *pcase*, joueur * *pj_joueur*)

diminue le niveau d'une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

pcase case dont on doit changer le niveau

Version :

1.0

Renvoie :

booléen indiquant si la fonction s'est bien déroulée

Définition à la ligne 687 du fichier traitement.c.

Référencé par attente_validation_propriete().

```
688 {
689     //il les vend
690     pj_joueur->int_argent=(pj_joueur->int_argent) + ((pcase->case_salle.int_prix_niveau)/2);
691     pcase->case_salle.int_niveau=pcase->case_salle.int_niveau - 1;
692 }
```

8.14.2.16 void traitement_arrive_case (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16], int *int_nombre_tire*)

effectue les action necessaire lors de l'arrivée sur une case en fonction de son type

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

int_nombre_tire nombre obtenu lors du lancer des dés

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 694 du fichier traitement.c.

Référencé par aller_a_jeton(), avancer_jeton(), et reculer_jeton().

```
695 {
696   char message[512];
697 
698   int action_possible_resultat;
699 
700   action_possible_resultat=action_possible(plateau, 0, pj_joueur, ARRIVE_CASE, int_nombre_tire);
701 
702   if(plateau[pj_joueur->int_position]->int_type==SALLE)
703   {
704     if ((plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur!=pj_joueur) && (plateau[pj_joueur->int_positio
705     {
706       //on lui affiche un message lui indiquant ce qu'il paye et a qui
707       sprintf(message, "Vous êtes chez %s : vous payer le loyer %d Fintz", plateau[pj_joueur->int_position]->case_sa
708       attente_validation_message(surf_ecran,surf_centre,"Loyer",message,MESSAGE_NORMAL);
709     }
710     else
711     {
712       attente_validation_propriete(surf_ecran, plateau, surf_centre, plateau[pj_joueur->int_position],pj_joueur, act
713     }
714   }
715   else if (plateau[pj_joueur->int_position]->int_type==BUREAU_KRYSTEL)
716   {
717     //le joueur tire une carte et agit en fonction d'elle
718     traitement_bureau(surf_ecran,surf_centre,plateau, pj_joueur,bureau_de_krystel,bureau_de_krystel, bureau_de_nadeg
719     //reculer_jeton(surf_ecran, surf_centre, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
720   }
721   else if (plateau[pj_joueur->int_position]->int_type==BUREAU_NADEGE)
722   {
723     //le joueur tire une carte et agit en fonction d'elle
```

```
724     traitement_bureau(surf_ecran,surf_centre,plateau, pj_joueur,bureau_de_nadege, bureau_de_krystel, bureau_de_nadeg
725   }
726   else if (plateau[pj_joueur->int_position]->int_type==SP_BUREAU_LAURENCE)
727   {
728     traitement_bureau_laurence(surf_ecran, surf_centre, plateau, pj_joueur, bureau_de_krystel, bureau_de_nadege);
729   }
730   else if (plateau[pj_joueur->int_position]->int_type == SP_TABLEAU)
731   {
732     //on envoi le joueur directement en prison
733     aller_en_prison_jeton(surf_ecran, pj_joueur,plateau);
734   }
735   else if (plateau[pj_joueur->int_position]->int_type == SP_MACHINE_CAFE)
736   {
737     //le joueur récolte l'argent de la machine a café
738     traitement_machine_a_cafe(surf_ecran, surf_centre, plateau, pj_joueur);
739   }
740   else if (plateau[pj_joueur->int_position]->int_type == SOIREE_GALA || plateau[pj_joueur->int_position]->int_type =
741   {
742     traitement_soiree(surf_ecran, surf_centre ,pj_joueur, plateau);
743   }
744   else if (  (plateau[pj_joueur->int_position]->int_type == LC_WC)
745       || (plateau[pj_joueur->int_position]->int_type == LC_ASCENSEUR)
746       || (plateau[pj_joueur->int_position]->int_type == LC_RU)
747       || (plateau[pj_joueur->int_position]->int_type == LC_PARKING))
748   {
749     //si la case appartient a un adversaire
750     if ((plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur!=pj_joueur) && (plateau[pj_joueur->int_po
751     {
752       //on prépare le message et il paye en même temps
753       sprintf(message, "Vous êtes chez %s : vous payer le loyer %d Fintz", plateau[pj_joueur->int_position]->case_li
754       //on lui fait payer le loyer en fonction du nombre de
755       traitement_payer_loyer(pj_joueur, plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur, traitemen
756       //on affiche le message
757       attente_validation_message(surf_ecran,surf_centre,"Loyer",message,MESSAGE_NORMAL);
758     }
759     else
760     {
761       attente_validation_propriete(surf_ecran,plateau, surf_centre, plateau[pj_joueur->int_position],pj_joueur, acti
762     }
763   }
764   else if (  (plateau[pj_joueur->int_position]->int_type == BDE)
765       || (plateau[pj_joueur->int_position]->int_type == BDS))
766   {
767     //si la case appartient à un adversaire
768     if ((plateau[pj_joueur->int_position]->case_association.pjoueur_joueur!=pj_joueur) && (plateau[pj_joueur->int_po
769     {
770       //on prépare le message
771       sprintf(message, "Vous êtes chez %s : vous payer le loyer %d Fintz", plateau[pj_joueur->int_position]->case_as
772       //il paye
773       traitement_payer_loyer(pj_joueur, plateau[pj_joueur->int_position]->case_association.pjoueur_joueur,traitement
774
775       //on affiche le message
776       attente_validation_message(surf_ecran,surf_centre,"Loyer",message,MESSAGE_NORMAL);
777     }
778     else
779     {
780       attente_validation_propriete(surf_ecran,plateau, surf_centre, plateau[pj_joueur->int_position],pj_joueur, acti
781     }
782
783   }
784 }
```

8.14.2.17 void traitement_soiree (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*)

fait payer le joueur suivant la case soirée sur laquelle il est

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

rien

Définition à la ligne 786 du fichier traitement.c.

Référencé par traitement_arrive_case().

```
787 {
788   //le joueur peut-être soit sur le gala soit sur l'area
789   if (pj_joueur->int_position==38)
790   {
791     //le joueur se trouve à la soirée du gala
792     attente_validation_message(surf_ecran,surf_centre,"Cotisation pour le Gala","Payez 1500 Fintz",MESSAGE_NORMAL);
793   }
794   else
795   {
796     //le joueur se trouve sur la case area
797     attente_validation_message(surf_ecran,surf_centre,"Soirée à l'Area !","Payez 1500 Fintz",MESSAGE_NORMAL);
798   }
799   //on retire au joueur l'argent
800   pj_joueur->int_argent=pj_joueur->int_argent - plateau[pj_joueur->int_position]->case_soiree.int_prix;
801   //on ajoute l'argent à la machine a café
802   plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent + plateau[pj_joueur->int_position]->
803
804 }
```

8.14.2.18 void traitement_machine_a_cafe (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*)

donne au joueur l'argent de la machine à café

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

rien

Définition à la ligne 806 du fichier traitement.c.

Référencé par traitement_arrive_case().

```
807 {
808   char message[128];
809 
810   //on donne l'argent au joueur
811   pj_joueur->int_argent=pj_joueur->int_argent+plateau[20]->machine_a_cafe.int_argent;
812   sprintf(message,"Vous recevez %d Fintz de la machine à café", plateau[20]->machine_a_cafe.int_argent);
813   //on vide la machine a café
814   plateau[20]->machine_a_cafe.int_argent=0;
815 
816 
817 
818   attente_validation_message(surf_ecran,surf_centre,"Machine à café",message,MESSAGE_NORMAL);
819 
820 }
```

8.14.2.19 void traitement_bureau_laurence (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

effectue les traitement necessaire lorsqu'un joueur est dans le bureau de laurence

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

rien

Définition à la ligne 821 du fichier traitement.c.

Référencé par attente_clic(), et traitement_arrive_case().

```
822 {
823   int int_des;
824   int int_choix;
825 
826   //on verifie si le joueur est emprisonner
827   if (pj_joueur->bool_laurence==true)
828   {
829     //il lance les dés afin d'essayer de faire un double
830     int_des=lancer_des(surf_ecran, surf_centre,pj_joueur, plateau);
831 
832     if (pj_joueur->int_double_tire == 1)
833     {
834       //le jeton est libéré
835       pj_joueur->bool_laurence=false;
836       pj_joueur->int_laurence=0;
837       avancer_jeton(surf_ecran, surf_centre, int_des, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
838     }
839     //sinon on verifie si les trois tours de prison n'ont pas aboutis
840     else if (pj_joueur->int_laurence == 3)
841     {
842       //on le libère
843       pj_joueur->bool_laurence=false;
844       pj_joueur->int_laurence=0;
845       avancer_jeton(surf_ecran, surf_centre, int_des, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
846     }
847     else
848     {
849       int_choix=attente_validation_message(surf_ecran,surf_centre,"Bureau de Laurence","Que vous voulez vous faire 
850       switch (int_choix)
851       {
852         case 4:
853           //on indique qu'il a passe un tour de plus
854           pj_joueur->int_laurence++;
855           break;
856         case 5:
857           pj_joueur->bool_laurence=false;
858           pj_joueur->int_laurence=0;
859           //on lui fait payer les 500 Fintz à la machine a café
860           //on retire au joueur l'argent
861           pj_joueur->int_argent=pj_joueur->int_argent - 500;
862           //on ajoute l'argent à la machine a café
863           plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent + 500;
864           break;
865         case 6:
866           if (pj_joueur->int_certificat > 0)
867             {
868               pj_joueur->bool_laurence=false;
869               pj_joueur->int_laurence=0;
870               pj_joueur->int_certificat-=1;
871             }
872           else
873             {
874               //on indique qu'il a passe un tour de plus
875               pj_joueur->int_laurence++;
876             }
877           break;
878         default:
879           break;
880       }
881 
882     }
883   }
884 }
885 
```

8.14.2.20 void traitement_bureau (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*, information * *le_bureau*[16], information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16], int *int_id_bureau*)

tire une carte du bureau voulue et effectue le traitement en fonction de la carte tirée

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

le_bureau tableau des cartes bureau à utiliser

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

int_id_bureau id du bureau à utiliser

Version :

1.0

Renvoie :

rien

Définition à la ligne 888 du fichier traitement.c.

Référencé par traitement_arrive_case(), et traitement_bureau().

```
888 {
889   information* carte;
890   joueur* pjoueur_encours;
891   possession* propriete_encours;
892   int i;
893   int k;
894 
895   k=0;
896 
897   //on traite toute cette partie avec la première carte du tableau
898 
899   if (int_id_bureau == BUREAU_KRYSTEL)
900   {
901     attente_validation_message(surf_ecran,surf_centre,"Bureau de Krystel",le_bureau[0]->texte,MESSAGE_KRYSTEL);
902   }
903   else
904   {
905     attente_validation_message(surf_ecran,surf_centre,"Bureau de Nadege",le_bureau[0]->texte,MESSAGE_NADEGE);
906   }
907 
908   //traitement selon la carte
909   if (le_bureau[0]->type == ARGENT)
910   {
911     //ajoute ou enlève de l'argent au joueur
912     pj_joueur->int_argent=pj_joueur->int_argent+le_bureau[0]->valeur;
913     //on ajoute l'argent à la machine a café
914     plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent-le_bureau[0]->valeur;
```

```
915   }
916   else if(le_bureau[0]->type == POSITION)
917   {
918     if (le_bureau[0]->valeur < 0)
919     {
920       reculer_jeton(surf_ecran, surf_centre, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
921     }
922     else
923     {
924       //on avance le jeton jusqu'a la case voulue
925       aller_a_jeton(surf_ecran, surf_centre, le_bureau[0]->valeur, pj_joueur,plateau,bureau_de_krystel, bureau_de_na
926     }
927   }
928   else if(le_bureau[0]->type == ASTRID)
929   {
930     traitement_bureau(surf_ecran,surf_centre,plateau, pj_joueur,bureau_de_nadege, bureau_de_krystel, bureau_de_nadeg
931   }
932   else if(le_bureau[0]->type == ANNIVERSAIRE)
933   {
934     pjoueur_encours=pj_joueur->pjoueur_suivant;
935     //on prend l'argent à chaque joueur et on le donne à l'actuel
936     while (pjoueur_encours != pj_joueur)
937     {
938       pjoueur_encours->int_argent = pjoueur_encours->int_argent-le_bureau[0]->valeur;
939       k++;
940       pjoueur_encours=pjoueur_encours->pjoueur_suivant;
941     }
942     //on donne l'argent récolté au joueur
943     pj_joueur->int_argent = pj_joueur->int_argent+((le_bureau[0]->valeur)*k);
944   }
945   else if(le_bureau[0]->type == CTI)
946   {
947     propriete_encours=pj_joueur->propriete;
948     //on regarde chaque propriété du joueur
949
950     k=0;
951
952     while (propriete_encours != NULL)
953     {
954       if (propriete_encours->propriete->int_type == SALLE)
955       {
956         k=k+(propriete_encours->propriete->case_salle.int_niveau)-1;
957         propriete_encours=propriete_encours->suivant;
958       }
959     }
960     //on retire l'argent au joueur
961     pj_joueur->int_argent = pj_joueur->int_argent+(-(le_bureau[0]->valeur)*k);
962     //on l'ajoute à la machine à café
963     plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent-(-(le_bureau[0]->valeur)*k);
964   }
965   else if (le_bureau[0]->type == CARTE_LAURENCE)
966   {
967     //on envoie le joueur directement en prison
968     aller_en_prison_jeton(surf_ecran, pj_joueur, plateau);
969   }
970   else if(le_bureau[0]->type == CERTIFICAT)
971   {
972     //le joueur possède un certificat de plus
973     pj_joueur->int_certificat++;
974   }
975
976   //puis on remet la carte à la fin du tableau
977
978   carte=le_bureau[0];
979
980   for (i = 0; i < 15; i++)
981   {
```

```
982     le_bureau[i]=le_bureau[i+1];
983     le_bureau[15]=carte;
984   }
985   //si le solde du joueur devient négatif il perd
986   if (pj_joueur->int_argent < 0)
987   {
988     traitement_perdu(pj_joueur);
989   }
990
991 }
992
```

8.14.2.21 void traitement_elimination_joueur (joueur * *joueur_eliminer*)

libère les propriété et l'espace mémoire du joueur et de ses possessions

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_eliminer le joueur à éliminer

Version :

1.0

Renvoie :

rien

Définition à la ligne 994 du fichier traitement.c.

Référencé par traitement_perdu().

```
994 {
995   possession* propriete_joueur_encours;
996   possession* propriete_joueur_precedent;
997   joueur* pj_joueur_precedent;
998   joueur* pj_joueur_encours;
999
1000   propriete_joueur_encours=joueur_eliminer->propriete;
1001
1002   //il faut libérer tous les propriétés du joueur
1003   //pour chaque propriété du joueur
1004   while (propriete_joueur_encours != NULL)
1005   {
1006     //on test le type de la case
1007     switch (propriete_joueur_encours->propriete->int_type)
1008     {
1009       case SALLE:
1010         //la case n'a plus de propriétère
1011         propriete_joueur_encours->propriete->case_salle.pjoueur_joueur=NULL;
1012         //la case n'a plus de maison
1013         propriete_joueur_encours->propriete->case_salle.int_niveau=0;
1014         break;
1015       case BDE:
1016         //la case n'a plus de propriétère
1017         propriete_joueur_encours->propriete->case_association.pjoueur_joueur=NULL;
1018         break;
1019       case BDS:
1020         //la case n'a plus de propriétère
1021         propriete_joueur_encours->propriete->case_association.pjoueur_joueur=NULL;
```

```
1022         break;
1023       case LC_WC:
1024         //la case n'a plus de propriétère
1025         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1026         break;
1027       case LC_PARKING:
1028         //la case n'a plus de propriétère
1029         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1030         break;
1031       case LC_RU:
1032         //la case n'a plus de propriétère
1033         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1034         break;
1035       case LC_ASCENSEUR:
1036         //la case n'a plus de propriétère
1037         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1038         break;
1039       default:
1040         break;
1041     }
1042
1043     propriete_joueur_precedent = propriete_joueur_encours;
1044     propriete_joueur_encours = propriete_joueur_encours->suivant;
1045     delete propriete_joueur_precedent;
1046   }
1047   //ensuite on libère la place en mémoire du joueur
1048   //on trouve le joueur précédent celui éliminer
1049   pj_joueur_precedent=joueur_eliminer;
1050   pj_joueur_encours=joueur_eliminer->pjoueur_suivant;
1051
1052   while (pj_joueur_encours != joueur_eliminer)
1053   {
1054     pj_joueur_precedent=pj_joueur_encours;
1055     pj_joueur_encours=pj_joueur_encours->pjoueur_suivant;
1056   }
1057   pj_joueur_precedent->pjoueur_suivant=joueur_eliminer->pjoueur_suivant;
1058
1059   delete joueur_eliminer;
1060 }
1061
```

8.14.2.22 void traitement_perdu (joueur * *pj_joueur*)

Définition à la ligne 1063 du fichier traitement.c.

Référencé par traitement_bureau().

```
1063 {
1064   //on affiche le message
1065
1066   //on l'élimine
1067   traitement_elimination_joueur(pj_joueur);
1068 }
1069 }
```

8.15 Référence du fichier traitement.h

en tete des fonctions de traitement.

Fonctions

- int **lancer_des** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau)
  fonction chargée de simuler le lancement des dés

- void **avancer_jeton** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, int int_nb_tire, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  avance le jeton case par case, traite le passage par la case départ

- void **jeu** (SDL_Surface *surf_ecran, **joueur** *pj_joueur_encours, **cases** **plateau, int nombre_joueur, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  initialise le jeu

- void **aller_a_jeton** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, int int_position_voulue, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16], int delai)
  avance le jeton case par case, traite le passage par la case départ

- void **aller_en_prison_jeton** (SDL_Surface *surf_ecran, **joueur** *pj_joueur, **cases** **plateau)
  envoi le joueur directement en prison

- void **reculer_jeton** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  recule le joueur de 3 cases

- void **traitement_case_depart** (SDL_Surface *surf_ecran, **joueur** *pj_joueur)
  ajoute l'argent au joueur

- bool **traitement_rachat_hypotheque** (**joueur** *joueur_actuel, **cases** *case_hypothequee)
  rachète une hypothèque

- bool **traitement_mise_en_hypotheque** (**joueur** *joueur_actuel, **cases** *case_hypothequee)
  met en place une hypothèque

- bool **traitement_payer_loyer** (**joueur** *joueur_qui_paye, **joueur** *joueur_paye, int int_montant)
  paye les loyer d'une salle

- int **traitement_loyer_lieu_commun** (**joueur** *pj_joueur)
  calcul le loyer pour les lieux communs

- int **traitement_loyer_association** (**joueur** *joueur_paye, int int_nb_tire)
  calcul le loyer d'une association

- bool **traitement_achat** (**cases** *case_achetee, **joueur** *joueur_actuel)
  traitement l'achat d'une propriété

- bool **traitement_augmentation_niveau** (**cases** *pcase, **joueur** *pj_joueur)
  augmente le niveau d'une case

- void **traitement_diminution_niveau** (**cases** *pcase, **joueur** *pj_joueur)
  diminue le niveau d'une case

- void **traitement_arrive_case** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16], int int_nombre_tire)
  effectue les action necessaire lors de l'arrivée sur une case en fonction de son type

- void **traitement_elimination_joueur** (**joueur** *joueur_eliminer)
  libère les propriété et l'espace mémoire du joueur et de ses possessions

- void **traitement_machine_a_cafe** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur)
  donne au joueur l'argent de la machine à café

- void **traitement_bureau** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur, **information** *le_bureau[16], **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16], int int_id_bureau)
  tire une carte du bureau voulue et effectue le traitement en fonction de la carte tirée

- void **traitement_soiree** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **joueur** *pj_joueur, **cases** **plateau)
  fait payer le joueur suivant la case soirée sur laquelle il est

- void **traitement_bureau_laurence** (SDL_Surface *surf_ecran, SDL_Surface *surf_centre, **cases** **plateau, **joueur** *pj_joueur, **information** *bureau_de_krystel[16], **information** *bureau_de_nadege[16])
  effectue les traitement necessaire lorsqu'un joueur est dans le bureau de laurence

- void **traitement_perdu** (**joueur** *pj_joueur)

8.15.1 Description détaillée

en tete des fonctions de traitement.

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Date :

09/01/2006

Version :

1.0

Définition dans le fichier **traitement.h**.

8.15.2 Documentation des fonctions

8.15.2.1 int lancer_des (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*)

fonction chargée de simuler le lancement des dés

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ ecran surface de l'écran

surf_ centre surface centrale du plateau

pj_ joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

le nombre obtenue par les dés

Définition à la ligne 106 du fichier traitement.c.

Référencé par attente_clic(), et traitement_bureau_laurence().

```
107 {
108   int int_1; //valeur du premier dé
109   int int_2; //valeur du second dé
110   int int_rebond1; //nb de rebonds du premier dé
111   int int_rebond2; //nb de rebonds du second dé
112 
113   int i=0;
114 
115   SDL_Surface** surf_des;
116 
117   //calcul du nombre de rebond du premier dé
118   int_rebond1=rand()%10+5;
119 
120   //calcul de nombre de rebond du second dé
121   int_rebond2=rand()%10+5;
122 
123   //initialisation a 0 des deux dés
124   int_1=0;
125   int_2=0;
126 
127   surf_des=creation_des();
128 
129   if (int_rebond1 > int_rebond2)
130   {
131     while (i < int_rebond2)
132     {
133       int_1=(rand()%6)+1;
134       int_2=(rand()%6)+1;
135 
136       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
137       SDL_Delay(DELAY);
138       i++;
139     }
140     for(i=0;i<(int_rebond1-int_rebond2-1);i++)
141     {
142       int_1=(rand()%6)+1;
143 
144       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
145       SDL_Delay(DELAY);
146     }
147   }
148   else
149   {
```

```
150     while (i < int_rebond1)
151     {
152       int_1=(rand()%6)+1;
153       int_2=(rand()%6)+1;
154
155       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
156       SDL_Delay(DELAY);
157       i++;
158     }
159     for(i=0;i<(int_rebond1-int_rebond2-1);i++)
160     {
161       int_1=(rand()%6)+1;
162
163       affich_panneau_des(surf_ecran,surf_des,int_1,int_2);
164       SDL_Delay(DELAY);
165     }
166   }
167   if (int_1==int_2)
168   {
169     pj_joueur->int_double_tire++;
170   }
171   else
172   {
173     pj_joueur->int_double_tire=0;
174   }
175
176   //si le joueur a fait 3 doubles il va en prison
177   if (pj_joueur->int_double_tire == 3)
178   {
179     attente_validation_message(surf_ecran, surf_centre, "PRISON" , "Vous avez fait 3 doubles consecutif", MESSAGE_NO
180     aller_en_prison_jeton(surf_ecran, pj_joueur,plateau);
181     pj_joueur->int_double_tire = 0;
182   }
183
184   destruction_des(surf_des);
185   return(int_1+int_2);
186 }
```

8.15.2.2 void avancer_jeton (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, int *int_nb_tire*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

avance le jeton case par case, traite le passage par la case départ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran
surf_centre surface centrale du plateau
int_nb_tire nombre obtenu lors du lancer des dés
pj_joueur joueur qui joue actuellement
plateau tableau contenant les cases du plateau
bureau_de_krystel tableau des cartes bureau de krystel
bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 233 du fichier traitement.c.

Référencé par attente_clic(), et traitement_bureau_laurence().

```
234 {
235   int int_position;
236   int int_position2;
237 
238   joueur* pj_joueur_test;
239 
240   //on le fait avancer case par case
241   for (int i=0;i<int_nb_tire;i++)
242   {
243     //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
244     affich_case(surf_ecran,plateau[pj_joueur->int_position]);
245     //on avance le joueur de 1 en faisant attention que la position reste dans les 40 cases du plateau
246     pj_joueur->int_position=(pj_joueur->int_position+1)%40;
247     //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
248     int_position=nombre_joueur_case(pj_joueur);
249     //on affiche le joueur sur la nouvelle case
250     affich_joueur(surf_ecran, pj_joueur,int_position+1, plateau[pj_joueur->int_position]);
251     // on lui indique que le premier joueur test est le suivant
252     pj_joueur_test=pj_joueur->pjoueur_suivant;
253     int_position2=0;
254     //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
255     while (pj_joueur_test != pj_joueur)
256     {
257       if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
258       {
259         int_position2++;
260         //on l'affiche sur la case:
261         affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
262       }
263       pj_joueur_test=pj_joueur_test->pjoueur_suivant;
264     }
265     //on rafraichi l'affichage
266     SDL_Flip(surf_ecran);
267     //petit delay pour que l'utilisateur puisse apprécier l'affichage
268     SDL_Delay(DELAY);
269 
270     //si il se trouve sur la case départ
271     if (pj_joueur->int_position==0)
272     {
273       //on effectue le traitement adequat
274       traitement_case_depart (surf_ecran,pj_joueur);
275     }
276   }
277   traitement_arrive_case(surf_ecran, surf_centre,plateau,pj_joueur, bureau_de_krystel, bureau_de_nadege, int_nb_tire
278 }
```

8.15.2.3 void jeu (SDL_Surface * *surf_ecran*, joueur * *pj_joueur_encours*, cases ** *plateau*, int *nombre_joueur*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

initialise le jeu

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pj_joueur_encours joueur qui joue actuellement

plateau tableau contenant les cases du plateau

nombre_joueur nombre de joueurs qui jouent

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 12 du fichier traitement.c.

Référencé par main().

```
13 {
14   int i;
15
16   //Valeur de retour
17   int int_retour;
18
19   //Surface du centre
20   SDL_Surface* surf_centre = NULL;
21
22   //Création du centre
23   surf_centre=SDL_LoadBMP("sdl/images/logo.bmp");
24
25   //Affichage des cases du plateau
26   for(i=0;i<40;i++) affich_case(surf_ecran, plateau[i]);
27
28   //joueur_possede_tout(plateau, pj_joueur_encours);                          //<<<==================================
29
30   //on affiche les joueurs sur la cases depart
31   affich_joueur_depart (surf_ecran,plateau,pj_joueur_encours,nombre_joueur);
32   //Affichage du centre
33   affich_centre(surf_ecran, surf_centre);
34
35   //Affichage du panneau du menu
36   affich_panneau_menu(surf_ecran,0);
37
38   //Affichage du panneau contenant les informations du premier joueur
39   affich_panneau_joueur(surf_ecran,pj_joueur_encours);
40
41   //Affichage du panneau du bouton lancer dés
42   affich_panneau_des_bouton(surf_ecran,0);
43
44   //Affichage du panneau de fin de tour
45   affich_panneau_fdt(surf_ecran,false);
46
47   //Affichage contenant les informations sur les possessions du joueur
48   affich_panneau_possessions(surf_ecran,pj_joueur_encours);
49
50   //Attente du premier événement souris
51   int_retour=attente_clic(surf_ecran, surf_centre, pj_joueur_encours,plateau,bureau_de_krystel,bureau_de_nadege);
52   if(int_retour==0)
53   {
54     int_retour=attente_validation_message(surf_ecran, surf_centre, NULL, "Etes vous sûr de vouloir réellement envisag
55
```

```
56   }
57   //Tant que le joueur ne décide pas de quitter
58   while(int_retour)
59   {
60     //si le joueur n'a pas fait de double on passe au joueur suivant
61     if (pj_joueur_encours->int_double_tire == 0)
62     {
63       //Passage au joueur suivant
64       pj_joueur_encours=pj_joueur_encours->pjoueur_suivant;
65     }
66
67     //on affiche le panneau du nouveau joueur
68     affich_panneau_joueur(surf_ecran,pj_joueur_encours);
69     //Affichage du cache pour le panneau des possession
70     affich_possessions_cache(surf_ecran);
71     //On réaffiche le panneau des possessions
72     affich_panneau_possessions(surf_ecran,pj_joueur_encours);
73     //Affichage du panneau du bouton lancer dés
74     affich_panneau_des_bouton(surf_ecran,0);
75     //Mise à jour de l'écran
76     SDL_Flip(surf_ecran);
77     //Attente d'un événement souris
78     int_retour=attente_clic(surf_ecran, surf_centre, pj_joueur_encours, plateau, bureau_de_krystel, bureau_de_nadege)
79     //Si le joueur souhaite quitter
80     if(int_retour==0)
81     {
82       //Demande de confirmation d'action
83       int_retour=attente_validation_message(surf_ecran, surf_centre, NULL, "Etes vous sûr de vouloir réellement envis
84       //Si le joueur demande de sauvegarder
85       if(int_retour==3)
86       {
87         //On sauvegarde
88         sauvegarde(plateau, pj_joueur_encours);
89         //Message de confirmation de sauvegarde
90         attente_validation_message(surf_ecran, surf_centre, "Sauvegarde", "La sauvegarde à été effectué avec un très
91         //Fin du jeu
92         int_retour=0;
93       }
94     }
95     //on vérifie que le joueur n'a pas gagné
96     if(verification_victoire(pj_joueur_encours))
97     {
98       //Message de victoire
99       attente_validation_message(surf_ecran,surf_centre,"VICTOIRE", "Félicitation vous avez ruiné tous vos adversaire
100       //Fin du jeu
101       int_retour=0;
102     }
103   }
104 }
```

8.15.2.4 void aller_a_jeton (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, int *int_position_voulue*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16], int *delai*)

avance le jeton case par case, traite le passage par la case départ

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface du centre du plateau

int_position_voulue position voulue

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

delai délai du déplacement entre deux cases

Version :

1.0

Renvoie :

rien

Définition à la ligne 280 du fichier traitement.c.

Référencé par traitement_bureau().

```
281 {
282   int int_position;
283   int int_position2;
284   joueur* pj_joueur_test;
285
286   //on le fait avancer case par case
287   while (pj_joueur->int_position != int_position_voulue)
288   {
289     //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
290     affich_case(surf_ecran,plateau[pj_joueur->int_position]);
291     //on avance le joueur de 1 en faisant attention que la position reste dans les 40 cases du plateau
292     pj_joueur->int_position=(pj_joueur->int_position+1)%40;
293     //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
294     int_position=nombre_joueur_case(pj_joueur);
295     //on affiche le joueur sur la nouvelle case
296     affich_joueur(surf_ecran, pj_joueur, int_position+1, plateau[pj_joueur->int_position]);
297
298     // on lui indique que le premier joueur test est le suivant
299     pj_joueur_test=pj_joueur->pjoueur_suivant;
300     int_position2=0;
301     //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
302     while (pj_joueur_test != pj_joueur)
303     {
304       if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
305       {
306         int_position2++;
307         //on l'affiche sur la case:
308         affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
309       }
310       pj_joueur_test=pj_joueur_test->pjoueur_suivant;
311     }
312     //on rafraichi l'affichage
313     SDL_Flip(surf_ecran);
314     //petit delay pour que l'utilisateur puisse apprécier l'affichage
315     SDL_Delay(delai);
316     //si il se trouve sur la case départ
317     if (pj_joueur->int_position==0)
318     {
319       //on effectue le traitement adequat
320       traitement_case_depart (surf_ecran,pj_joueur);
321     }
322   }
323   //on lance la fonction traitement associé à la case ou il s'arrete
```

```
324   //plateau[pj_joueur->int_position]->ptraitement(pj_joueur->int_position);
325
326   traitement_arrive_case(surf_ecran, surf_centre,plateau,pj_joueur, bureau_de_krystel, bureau_de_nadege, 0);
327 }
```

8.15.2.5 void aller_en_prison_jeton (SDL_Surface * *surf_ecran*, joueur * *pj_joueur*, cases ** *plateau*)

envoi le joueur directement en prison

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

rien

Définition à la ligne 329 du fichier traitement.c.

Référencé par lancer_des(), traitement_arrive_case(), et traitement_bureau().

```
330 {
331   int int_position;
332   int int_position2;
333
334   joueur* pj_joueur_test;
335
336
337   //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
338   affich_case(surf_ecran,plateau[pj_joueur->int_position]);
339
340   // on lui indique que le premier joueur test est le suivant
341   pj_joueur_test=pj_joueur->pjoueur_suivant;
342   int_position2=0;
343   //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
344   while (pj_joueur_test != pj_joueur)
345   {
346     if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
347     {
348       int_position2++;
349       //on l'affiche sur la case:
350       affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
351     }
352     pj_joueur_test=pj_joueur_test->pjoueur_suivant;
353   }
354
355   //on envoi le jeton directement en prison
356   pj_joueur->int_position=10;
357   //on indique qu'il est emprisonner
358   pj_joueur->bool_laurence=true;
359
```

```
360   //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
361   int_position=nombre_joueur_case(pj_joueur);
362
363   //on affiche le joueur sur la nouvelle case
364   affich_joueur(surf_ecran, pj_joueur, int_position+1, plateau[pj_joueur->int_position]);
365
366   //on rafraichi l'affichage
367   SDL_Flip(surf_ecran);
368 }
```

8.15.2.6 void reculer_jeton (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

recule le joueur de 3 cases

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 188 du fichier traitement.c.

Référencé par traitement_bureau().

```
189 {
190   int int_position;
191   int int_position2;
192   int i;
193
194   i=0;
195
196   joueur* pj_joueur_test;
197
198   //on le fait avancer case par case
199   for (i=0;i<3;i++)
200   {
201     //on réactualise la case ou se trouve le joueur afin qu'il efface le jeton
202     affich_case(surf_ecran,plateau[pj_joueur->int_position]);
203     //on avance le joueur de 1 en faisant attention que la position reste dans les 40 cases du plateau
204     pj_joueur->int_position=(pj_joueur->int_position+39)%40;
205     //on compte le nombre de personne sur la case ou doit être afficher le joueur actuel
206     int_position=nombre_joueur_case(pj_joueur);
207     //on affiche le joueur sur la nouvelle case
```

```
208     affich_joueur(surf_ecran, pj_joueur,int_position+1, plateau[pj_joueur->int_position]);
209     // on lui indique que le premier joueur test est le suivant
210     pj_joueur_test=pj_joueur->pjoueur_suivant;
211     int_position2=0;
212     //on verifie pour chaque joueur s'il se trouvait là où était le joueur avant
213     while (pj_joueur_test != pj_joueur)
214     {
215       if (pj_joueur_test->int_position==((pj_joueur->int_position)-1))
216       {
217         int_position2++;
218         //on l'affiche sur la case:
219         affich_joueur(surf_ecran, pj_joueur_test,int_position2, plateau[pj_joueur_test->int_position]);
220       }
221       pj_joueur_test=pj_joueur_test->pjoueur_suivant;
222     }
223     //on rafraichi l'affichage
224     SDL_Flip(surf_ecran);
225     //petit delay pour que l'utilisateur puisse apprécier l'affichage
226     SDL_Delay(DELAY);
227   }
228
229   traitement_arrive_case(surf_ecran, surf_centre,plateau,pj_joueur, bureau_de_krystel, bureau_de_nadege, 0);
230 }
```

8.15.2.7 void traitement_case_depart (SDL_Surface * *surf_ecran*, joueur * *pj_joueur*)

ajoute l'argent au joueur

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

pj_joueur joueur qui joue actuellement

Version :

1.0

Renvoie :

rien

Définition à la ligne 434 du fichier traitement.c.

Référencé par aller_a_jeton(), et avancer_jeton().

```
435 {
436   //on ajoute 2000 Fintz au compte du joueur
437   pj_joueur->int_argent=(pj_joueur->int_argent)+2000;
438   pj_joueur->bool_debut=false;
439 }
```

8.15.2.8 bool traitement_rachat_hypotheque (joueur * *joueur_actuel*, cases * *case_hypothequee*)

rachète une hypothèque

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_actuel joueur qui joue actuellement

case_hypothequee indice de la case à déshypothéquer

Version :

1.0

Renvoie :

booléen indiquant si la fonction c'est bien dérouler

Définition à la ligne 582 du fichier traitement.c.

```
583 {
584   //on agit selon le type de la case
585   switch (case_hypothequee->int_type)
586   {
587     case SALLE:
588       //si le joueur a assez d'argent
589       if (joueur_actuel->int_argent >= case_hypothequee->case_salle.int_valeur_hypotheque+case_hypothequee->case_sal
590       {
591       //il rachete l'hypothèque
592       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_salle.int_valeur_hypothequ
593       case_hypothequee->case_salle.bool_hypotheque=false;
594       }
595       return(true);
596       break;
597
598     case BDS:
599       //si le joueur a assez d'argent
600       if (joueur_actuel->int_argent >= (case_hypothequee->case_association.int_valeur_hypotheque+case_hypothequee->c
601       {
602       //il rachete l'hypothèque
603       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_association.int_valeur_hyp
604       case_hypothequee->case_association.bool_hypotheque=false;
605       }
606       return(true);
607       break;
608
609     case BDE:
610       //si le joueur a assez d'argent
611       if (joueur_actuel->int_argent >= (case_hypothequee->case_association.int_valeur_hypotheque+case_hypothequee->c
612       {
613       //il rachete l'hypothèque
614       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_association.int_valeur_hyp
615       case_hypothequee->case_association.bool_hypotheque=false;
616       }
617       return(true);
618       break;
619
620     case LC_WC:
621       //si le joueur a assez d'argent
622       if (joueur_actuel->int_argent >=(case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->ca
623       {
624       //il rachete l'hypothèque
625       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
626       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
627       }
628       return(true);
629       break;
630
```

```
631     case LC_ASCENSEUR:
632       //si le joueur a assez d'argent
633       if (joueur_actuel->int_argent >= (case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->
634       {
635       //il rachete l'hypothèque
636       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
637       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
638       }
639       return(true);
640       break;
641
642     case LC_RU:
643       //si le joueur a assez d'argent
644       if (joueur_actuel->int_argent >= (case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->
645       {
646       //il rachete l'hypothèque
647       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
648       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
649       }
650       return(true);
651       break;
652
653     case LC_PARKING:
654       //si le joueur a assez d'argent
655       if (joueur_actuel->int_argent >= (case_hypothequee->case_lieu_commun.int_valeur_hypotheque+case_hypothequee->
656       {
657       //il rachete l'hypothèque
658       joueur_actuel->int_argent=(int)(joueur_actuel->int_argent - (case_hypothequee->case_lieu_commun.int_valeur_hyp
659       case_hypothequee->case_lieu_commun.bool_hypotheque=false;
660       }
661       return(true);
662       break;
663
664     default:
665       return(false);
666       break;
667   }
668   return(false);
669 }
```

8.15.2.9 bool traitement_mise_en_hypotheque (joueur * *joueur_actuel*, cases * *case_hypothequee*)

met en place une hypothèque

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_actuel joueur qui joue actuellement

case_hypothequee indice de la case à hypothéquer

Version :

1.0

Renvoie :

booléen indiquant si la fonction c'est bien dérouler

Définition à la ligne 458 du fichier traitement.c.

```
459 {
460   //on agit selon le type de la case
461   switch (case_hypothequee->int_type)
462   {
463     case SALLE:
464       //si la case n'est pas déja hypotheque
465       if (!(case_hypothequee->case_salle.bool_hypotheque))
466       {
467         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_salle.int_valeur_hypotheque;
468         case_hypothequee->case_salle.bool_hypotheque=true;
469         return(case_hypothequee->case_salle.bool_hypotheque);
470       }
471       break;
472
473     case BDS:
474       if (!(case_hypothequee->case_association.bool_hypotheque))
475       {
476         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_association.int_valeur_hypothec
477         case_hypothequee->case_association.bool_hypotheque=true;
478         return(case_hypothequee->case_association.bool_hypotheque);
479       }
480       break;
481
482     case BDE:
483       if (!(case_hypothequee->case_association.bool_hypotheque))
484       {
485         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_association.int_valeur_hypothec
486         case_hypothequee->case_association.bool_hypotheque=true;
487         return(case_hypothequee->case_association.bool_hypotheque);
488       }
489       break;
490
491     case LC_WC:
492       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
493       {
494         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
495         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
496         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
497       }
498       break;
499
500     case LC_ASCENSEUR:
501       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
502       {
503         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
504         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
505         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
506       }
507       break;
508
509     case LC_RU:
510       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
511       {
512         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
513         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
514         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
515       }
516       break;
517
518     case LC_PARKING:
519       if (!(case_hypothequee->case_lieu_commun.bool_hypotheque))
520       {
521         joueur_actuel->int_argent=joueur_actuel->int_argent + case_hypothequee->case_lieu_commun.int_valeur_hypothec
522         case_hypothequee->case_lieu_commun.bool_hypotheque=true;
523         return(case_hypothequee->case_lieu_commun.bool_hypotheque);
524       }
525       break;
```

```
526
527     default:
528       return(false);
529       break;
530   }
531   return(false);
532 }
```

8.15.2.10 bool traitement_payer_loyer (joueur * *joueur_qui_paye*, joueur * *joueur_paye*, int *int_montant*)

paye les loyer d'une salle

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_qui_paye joueur à qui on prend les sous

joueur_paye joueur à qui on donne les sous

int_montant montant de la transaction

Version :

1.0

Renvoie :

booléen indiquant si la fonction c'est bien dérouler

Définition à la ligne 442 du fichier traitement.c.

Référencé par traitement_arrive_case().

```
443 {
444   //si le joueur a assez d'argent
445   if (joueur_qui_paye->int_argent >= int_montant)
446   {
447     //il paye
448     joueur_qui_paye->int_argent=joueur_qui_paye->int_argent-int_montant;
449     joueur_paye->int_argent=joueur_paye->int_argent+int_montant;
450     return(true);
451   }
452   else
453   {
454     return(false);
455   }
456 }
```

8.15.2.11 int traitement_loyer_lieu_commun (joueur * *pj_joueur*)

calcul le loyer pour les lieux communs

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur à qui on doit les sous

Version :

1.0

Renvoie :

le montant du loyer

Définition à la ligne 564 du fichier traitement.c.

Référencé par action_possible(), et traitement_arrive_case().

```
565 {
566
567   possession* propriete_temp;
568   int i;
569   i=0;
570
571   propriete_temp=pj_joueur->propriete;
572
573   while(propriete_temp!=NULL)
574   {
575     i++;
576     propriete_temp=propriete_temp->suivant;
577   }
578
579   return(i*250);
580 }
```

8.15.2.12 int traitement_loyer_association (joueur * *joueur_paye*, int *int_nb_tire*)

calcul le loyer d'une association

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_paye joueur à qui on doit les sous

int_nb_tire nombre indiqué par le dés

Version :

1.0

Renvoie :

le loyer à payer

Définition à la ligne 534 du fichier traitement.c.

Référencé par action_possible(), et traitement_arrive_case().

```
535 {
536   possession* propriete_test;
537   int i; //compteur
538
539   i=0;
540
541   propriete_test=joueur_paye->propriete;
542   //on comptabilise le nombre de case association du joueur payé
```

```
543   while (propriete_test != NULL)
544   {
545     if ((propriete_test->propriete->int_type == BDS) || (propriete_test->propriete->int_type == BDE))
546     {
547       i++;
548     }
549
550     propriete_test=propriete_test->suivant;
551   }
552   //on a ainsi compatibilisé le nombe d'association du joueur
553   //si il n'y a qu'une on multiplie les nombre des dés par 40
554   if (i == 1)
555   {
556     return(int_nb_tire*40);
557   }
558   else
559   {
560     return(int_nb_tire*100);
561   }
562 }
```

8.15.2.13 bool traitement_achat (cases * *case_achetee*, joueur * *joueur_actuel*)

traitement l'achat d'une propriété

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

case_achetee indice des la case achetée

joueur_actuel l'acheteur

Version :

1.0

Renvoie :

booléen qui indique si la fonction s'est bbien déroulée

Définition à la ligne 370 du fichier traitement.c.

Référencé par attente_validation_propriete(), et joueur_possede_tout().

```
371 {
372   //on verifie si la case n'appartient a aucun joueur
373
374   //on détermine le type de la case
375   if (case_achetee->int_type==SALLE)
376   {
377     //on verifie si le joueur a assez d'argent pour l'acheter et si la case n'appartient a personne
378     if ((joueur_actuel->int_argent >= case_achetee->case_salle.int_prix) && (case_achetee->case_salle.pjoueur_joueu
379     {
380       //on effectue la transaction
381       joueur_actuel->int_argent=joueur_actuel->int_argent-case_achetee->case_salle.int_prix;
382       joueur_actuel->propriete=insertion_bonne_place_propriete(joueur_actuel->propriete,case_achetee);
383       case_achetee->case_salle.pjoueur_joueur=joueur_actuel;
384
385       return(true);
386     }
387     else
```

```
388     {
389       return(false);
390     }
391   }
392   else
393   {
394     if((case_achetee->int_type == BDE) || (case_achetee->int_type == BDS))
395     {
396       //on verifie si le joueur a assez d'argent pour l'acheter
397       if ((joueur_actuel->int_argent >= case_achetee->case_association.int_prix) && (case_achetee->case_association
398       {
399         //on effectue la transaction
400         joueur_actuel->int_argent=joueur_actuel->int_argent-case_achetee->case_association.int_prix;
401         joueur_actuel->propriete=insertion_bonne_place_propriete(joueur_actuel->propriete,case_achetee);
402         case_achetee->case_association.pjoueur_joueur=joueur_actuel;
403 
404         return(true);
405       }
406     }
407     else
408     {
409       if(    (case_achetee->int_type == LC_WC)
410         || (case_achetee->int_type == LC_RU)
411         || (case_achetee->int_type == LC_PARKING)
412         || (case_achetee->int_type == LC_ASCENSEUR))
413         {
414           //on verifie si le joueur a assez d'argent pour l'acheter
415           if ((joueur_actuel->int_argent >= case_achetee->case_lieu_commun.int_prix) && (case_achetee->case_lieu_co
416           {
417             //on effectue la transaction
418             joueur_actuel->int_argent=joueur_actuel->int_argent-case_achetee->case_lieu_commun.int_prix;
419             joueur_actuel->propriete=insertion_bonne_place_propriete(joueur_actuel->propriete,case_achetee);
420             case_achetee->case_lieu_commun.pjoueur_joueur=joueur_actuel;
421             return(true);
422           }
423           else
424           {
425             return(false);
426           }
427         }
428       }
429     }
430     return(false);
431 }
```

8.15.2.14 bool traitement_augmentation_niveau (cases * *pcase*, joueur * *pj_joueur*)

augmente le niveau d'une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

pcase case dont on doit changer le niveau

Version :

1.0

Renvoie :

booléen indiquant si la fonction s'est bien déroulée

Définition à la ligne 671 du fichier traitement.c.

Référencé par attente_validation_propriete().

```
672 {
673   //on vérifie si le joueur a assez d'argent
674   if (pj_joueur->int_argent >= pcase->case_salle.int_prix_niveau)
675   {
676     //il les achète
677     pj_joueur->int_argent=(pj_joueur->int_argent) - (pcase->case_salle.int_prix_niveau);
678     pcase->case_salle.int_niveau=pcase->case_salle.int_niveau + 1;
679     return(true);
680   }
681   else
682   {
683     return(false);
684   }
685 }
```

8.15.2.15 void traitement_diminution_niveau (cases * *pcase*, joueur * *pj_joueur*)

diminue le niveau d'une case

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

pj_joueur joueur qui joue actuellement

pcase case dont on doit changer le niveau

Version :

1.0

Renvoie :

booléen indiquant si la fonction s'est bien déroulée

Définition à la ligne 687 du fichier traitement.c.

Référencé par attente_validation_propriete().

```
688 {
689     //il les vend
690     pj_joueur->int_argent=(pj_joueur->int_argent) + ((pcase->case_salle.int_prix_niveau)/2);
691     pcase->case_salle.int_niveau=pcase->case_salle.int_niveau - 1;
692 }
```

8.15.2.16 void traitement_arrive_case (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16], int *int_nombre_tire*)

effectue les action necessaire lors de l'arrivée sur une case en fonction de son type

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran
surf_centre surface centrale du plateau
pj_joueur joueur qui joue actuellement
int_nombre_tire nombre obtenu lors du lancer des dés
plateau tableau contenant les cases du plateau
bureau_de_krystel tableau des cartes bureau de krystel
bureau_de_nadege tableau des cartes bureau de nadège

Version :

1.0

Renvoie :

rien

Définition à la ligne 694 du fichier traitement.c.

Référencé par aller_a_jeton(), avancer_jeton(), et reculer_jeton().

```
695 {
696   char message[512];
697
698   int action_possible_resultat;
699
700   action_possible_resultat=action_possible(plateau, 0, pj_joueur, ARRIVE_CASE, int_nombre_tire);
701
702   if(plateau[pj_joueur->int_position]->int_type==SALLE)
703   {
704     if ((plateau[pj_joueur->int_position]->case_salle.pjoueur_joueur!=pj_joueur) && (plateau[pj_joueur->int_positio
705     {
706       //on lui affiche un message lui indiquant ce qu'il paye et a qui
707       sprintf(message, "Vous êtes chez %s : vous payer le loyer %d Fintz", plateau[pj_joueur->int_position]->case_sa
708       attente_validation_message(surf_ecran,surf_centre,"Loyer",message,MESSAGE_NORMAL);
709     }
710     else
711     {
712       attente_validation_propriete(surf_ecran, plateau, surf_centre, plateau[pj_joueur->int_position],pj_joueur, act
713     }
714   }
715   else if (plateau[pj_joueur->int_position]->int_type==BUREAU_KRYSTEL)
716   {
717     //le joueur tire une carte et agit en fonction d'elle
718     traitement_bureau(surf_ecran,surf_centre,plateau, pj_joueur,bureau_de_krystel,bureau_de_krystel, bureau_de_nadeg
719     //reculer_jeton(surf_ecran, surf_centre, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
720   }
721   else if (plateau[pj_joueur->int_position]->int_type==BUREAU_NADEGE)
722   {
723     //le joueur tire une carte et agit en fonction d'elle
724     traitement_bureau(surf_ecran,surf_centre,plateau, pj_joueur,bureau_de_nadege, bureau_de_krystel, bureau_de_nadeg
725   }
726   else if (plateau[pj_joueur->int_position]->int_type==SP_BUREAU_LAURENCE)
727   {
728     traitement_bureau_laurence(surf_ecran, surf_centre, plateau, pj_joueur, bureau_de_krystel, bureau_de_nadege);
729   }
730   else if (plateau[pj_joueur->int_position]->int_type == SP_TABLEAU)
731   {
732     //on envoi le joueur directement en prison
```

```
733     aller_en_prison_jeton(surf_ecran, pj_joueur,plateau);
734   }
735   else if (plateau[pj_joueur->int_position]->int_type == SP_MACHINE_CAFE)
736   {
737     //le joueur récolte l'argent de la machine a café
738     traitement_machine_a_cafe(surf_ecran, surf_centre, plateau, pj_joueur);
739   }
740   else if (plateau[pj_joueur->int_position]->int_type == SOIREE_GALA || plateau[pj_joueur->int_position]->int_type =
741   {
742     traitement_soiree(surf_ecran, surf_centre ,pj_joueur, plateau);
743   }
744   else if (  (plateau[pj_joueur->int_position]->int_type == LC_WC)
745       || (plateau[pj_joueur->int_position]->int_type == LC_ASCENSEUR)
746       || (plateau[pj_joueur->int_position]->int_type == LC_RU)
747       || (plateau[pj_joueur->int_position]->int_type == LC_PARKING))
748   {
749     //si la case appartient a un adversaire
750     if ((plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur!=pj_joueur) && (plateau[pj_joueur->int_po
751     {
752       //on prépare le message et il paye en même temps
753       sprintf(message, "Vous êtes chez %s : vous payer le loyer %d Fintz", plateau[pj_joueur->int_position]->case_li
754       //on lui fait payer le loyer en fonction du nombre de
755       traitement_payer_loyer(pj_joueur, plateau[pj_joueur->int_position]->case_lieu_commun.pjoueur_joueur, traitemen
756       //on affiche le message
757       attente_validation_message(surf_ecran,surf_centre,"Loyer",message,MESSAGE_NORMAL);
758     }
759     else
760     {
761       attente_validation_propriete(surf_ecran,plateau, surf_centre, plateau[pj_joueur->int_position],pj_joueur, acti
762     }
763   }
764   else if (  (plateau[pj_joueur->int_position]->int_type == BDE)
765       || (plateau[pj_joueur->int_position]->int_type == BDS))
766   {
767     //si la case appartient à un adversaire
768     if ((plateau[pj_joueur->int_position]->case_association.pjoueur_joueur!=pj_joueur) && (plateau[pj_joueur->int_po
769     {
770       //on prépare le message
771       sprintf(message, "Vous êtes chez %s : vous payer le loyer %d Fintz", plateau[pj_joueur->int_position]->case_as
772       //il paye
773       traitement_payer_loyer(pj_joueur, plateau[pj_joueur->int_position]->case_association.pjoueur_joueur,traitement
774
775       //on affiche le message
776       attente_validation_message(surf_ecran,surf_centre,"Loyer",message,MESSAGE_NORMAL);
777     }
778     else
779     {
780       attente_validation_propriete(surf_ecran,plateau, surf_centre, plateau[pj_joueur->int_position],pj_joueur, acti
781     }
782
783   }
784 }
```

8.15.2.17 void traitement_elimination_joueur (joueur * *joueur_eliminer*)

libère les propriété et l'espace mémoire du joueur et de ses possessions

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

joueur_eliminer le joueur à éliminer

Version :

1.0

Renvoie :

rien

Définition à la ligne 994 du fichier traitement.c.

Référencé par traitement_perdu().

```
994 {
995   possession* propriete_joueur_encours;
996   possession* propriete_joueur_precedent;
997   joueur* pj_joueur_precedent;
998   joueur* pj_joueur_encours;
999
1000   propriete_joueur_encours=joueur_eliminer->propriete;
1001
1002   //il faut libérer tous les propriétés du joueur
1003   //pour chaque propriété du joueur
1004   while (propriete_joueur_encours != NULL)
1005   {
1006     //on test le type de la case
1007     switch (propriete_joueur_encours->propriete->int_type)
1008     {
1009       case SALLE:
1010         //la case n'a plus de propriétère
1011         propriete_joueur_encours->propriete->case_salle.pjoueur_joueur=NULL;
1012         //la case n'a plus de maison
1013         propriete_joueur_encours->propriete->case_salle.int_niveau=0;
1014         break;
1015       case BDE:
1016         //la case n'a plus de propriétère
1017         propriete_joueur_encours->propriete->case_association.pjoueur_joueur=NULL;
1018         break;
1019       case BDS:
1020         //la case n'a plus de propriétère
1021         propriete_joueur_encours->propriete->case_association.pjoueur_joueur=NULL;
1022         break;
1023       case LC_WC:
1024         //la case n'a plus de propriétère
1025         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1026         break;
1027       case LC_PARKING:
1028         //la case n'a plus de propriétère
1029         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1030         break;
1031       case LC_RU:
1032         //la case n'a plus de propriétère
1033         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1034         break;
1035       case LC_ASCENSEUR:
1036         //la case n'a plus de propriétère
1037         propriete_joueur_encours->propriete->case_lieu_commun.pjoueur_joueur=NULL;
1038         break;
1039       default:
1040         break;
1041     }
1042
1043     propriete_joueur_precedent = propriete_joueur_encours;
1044     propriete_joueur_encours = propriete_joueur_encours->suivant;
1045     delete propriete_joueur_precedent;
1046   }
1047   //ensuite on libère la place en mémoire du joueur
1048   //on trouve le joueur précédent celui éliminer
1049   pj_joueur_precedent=joueur_eliminer;
```

```
1050   pj_joueur_encours=joueur_eliminer->pjoueur_suivant;
1051
1052   while (pj_joueur_encours != joueur_eliminer)
1053   {
1054     pj_joueur_precedent=pj_joueur_encours;
1055     pj_joueur_encours=pj_joueur_encours->pjoueur_suivant;
1056   }
1057   pj_joueur_precedent->pjoueur_suivant=joueur_eliminer->pjoueur_suivant;
1058
1059   delete joueur_eliminer;
1060 }
1061
```

8.15.2.18 void traitement_machine_a_cafe (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*)

donne au joueur l'argent de la machine à café

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

rien

Définition à la ligne 806 du fichier traitement.c.

Référencé par traitement_arrive_case().

```
807 {
808   char message[128];
809
810   //on donne l'argent au joueur
811   pj_joueur->int_argent=pj_joueur->int_argent+plateau[20]->machine_a_cafe.int_argent;
812   sprintf(message,"Vous recevez %d Fintz de la machine à café", plateau[20]->machine_a_cafe.int_argent);
813   //on vide la machine a café
814   plateau[20]->machine_a_cafe.int_argent=0;
815
816
817
818   attente_validation_message(surf_ecran,surf_centre,"Machine à café",message,MESSAGE_NORMAL);
819
820 }
```

8.15.2.19 void traitement_bureau (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*, information * *le_bureau*[16], information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16], int *int_id_bureau*)

tire une carte du bureau voulue et effectue le traitement en fonction de la carte tirée

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

le_bureau tableau des cartes bureau à utiliser

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

int_id_bureau id du bureau à utiliser

Version :

1.0

Renvoie :

rien

Définition à la ligne 888 du fichier traitement.c.

Référencé par traitement_arrive_case(), et traitement_bureau().

```
888 {
889   information* carte;
890   joueur* pjoueur_encours;
891   possession* propriete_encours;
892   int i;
893   int k;
894 
895   k=0;
896 
897   //on traite toute cette partie avec la première carte du tableau
898 
899   if (int_id_bureau == BUREAU_KRYSTEL)
900   {
901     attente_validation_message(surf_ecran,surf_centre,"Bureau de Krystel",le_bureau[0]->texte,MESSAGE_KRYSTEL);
902   }
903   else
904   {
905     attente_validation_message(surf_ecran,surf_centre,"Bureau de Nadege",le_bureau[0]->texte,MESSAGE_NADEGE);
906   }
907 
908   //traitement selon la carte
909   if (le_bureau[0]->type == ARGENT)
910   {
911     //ajoute ou enlève de l'argent au joueur
912     pj_joueur->int_argent=pj_joueur->int_argent+le_bureau[0]->valeur;
913     //on ajoute l'argent à la machine a café
914     plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent-le_bureau[0]->valeur;
```

```
915   }
916   else if(le_bureau[0]->type == POSITION)
917   {
918     if (le_bureau[0]->valeur < 0)
919     {
920       reculer_jeton(surf_ecran, surf_centre, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
921     }
922     else
923     {
924       //on avance le jeton jusqu'a la case voulue
925       aller_a_jeton(surf_ecran, surf_centre, le_bureau[0]->valeur, pj_joueur,plateau,bureau_de_krystel, bureau_de_na
926     }
927   }
928   else if(le_bureau[0]->type == ASTRID)
929   {
930     traitement_bureau(surf_ecran,surf_centre,plateau, pj_joueur,bureau_de_nadege, bureau_de_krystel, bureau_de_nadeg
931   }
932   else if(le_bureau[0]->type == ANNIVERSAIRE)
933   {
934     pjoueur_encours=pj_joueur->pjoueur_suivant;
935     //on prend l'argent à chaque joueur et on le donne à l'actuel
936     while (pjoueur_encours != pj_joueur)
937     {
938       pjoueur_encours->int_argent = pjoueur_encours->int_argent-le_bureau[0]->valeur;
939       k++;
940       pjoueur_encours=pjoueur_encours->pjoueur_suivant;
941     }
942     //on donne l'argent récolté au joueur
943     pj_joueur->int_argent = pj_joueur->int_argent+((le_bureau[0]->valeur)*k);
944   }
945   else if(le_bureau[0]->type == CTI)
946   {
947     propriete_encours=pj_joueur->propriete;
948     //on regarde chaque propriété du joueur
949
950     k=0;
951
952     while (propriete_encours != NULL)
953     {
954       if (propriete_encours->propriete->int_type == SALLE)
955       {
956         k=k+(propriete_encours->propriete->case_salle.int_niveau)-1;
957         propriete_encours=propriete_encours->suivant;
958       }
959     }
960     //on retire l'argent au joueur
961     pj_joueur->int_argent = pj_joueur->int_argent+(-(le_bureau[0]->valeur)*k);
962     //on l'ajoute à la machine à café
963     plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent-(-(le_bureau[0]->valeur)*k);
964   }
965   else if (le_bureau[0]->type == CARTE_LAURENCE)
966   {
967     //on envoie le joueur directement en prison
968     aller_en_prison_jeton(surf_ecran, pj_joueur, plateau);
969   }
970   else if(le_bureau[0]->type == CERTIFICAT)
971   {
972     //le joueur possède un certificat de plus
973     pj_joueur->int_certificat++;
974   }
975
976   //puis on remet la carte à la fin du tableau
977
978   carte=le_bureau[0];
979
980   for (i = 0; i < 15; i++)
981   {
```

```
982     le_bureau[i]=le_bureau[i+1];
983     le_bureau[15]=carte;
984   }
985   //si le solde du joueur devient négatif il perd
986   if (pj_joueur->int_argent < 0)
987   {
988     traitement_perdu(pj_joueur);
989   }
990
991 }
992
```

8.15.2.20 void traitement_soiree (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, joueur * *pj_joueur*, cases ** *plateau*)

fait payer le joueur suivant la case soirée sur laquelle il est

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

Version :

1.0

Renvoie :

rien

Définition à la ligne 786 du fichier traitement.c.

Référencé par traitement_arrive_case().

```
787 {
788   //le joueur peut-être soit sur le gala soit sur l'area
789   if (pj_joueur->int_position==38)
790   {
791     //le joueur se trouve à la soirée du gala
792     attente_validation_message(surf_ecran,surf_centre,"Cotisation pour le Gala","Payez 1500 Fintz",MESSAGE_NORMAL);
793   }
794   else
795   {
796     //le joueur se trouve sur la case area
797     attente_validation_message(surf_ecran,surf_centre,"Soirée à l'Area !","Payez 1500 Fintz",MESSAGE_NORMAL);
798   }
799   //on retire au joueur l'argent
800   pj_joueur->int_argent=pj_joueur->int_argent - plateau[pj_joueur->int_position]->case_soiree.int_prix;
801   //on ajoute l'argent à la machine a café
802   plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent + plateau[pj_joueur->int_position]->
803
804 }
```

8.15.2.21 void traitement_bureau_laurence (SDL_Surface * *surf_ecran*, SDL_Surface * *surf_centre*, cases ** *plateau*, joueur * *pj_joueur*, information * *bureau_de_krystel*[16], information * *bureau_de_nadege*[16])

effectue les traitement necessaire lorsqu'un joueur est dans le bureau de laurence

Auteur :

Franck Trey <franck.trey@eisti.fr>, Florian Lefevre <Florian.lefevre@eisti.fr>

Paramètres :

surf_ecran surface de l'écran

surf_centre surface centrale du plateau

pj_joueur joueur qui joue actuellement

plateau tableau contenant les cases du plateau

bureau_de_krystel tableau des cartes bureau de krystel

bureau_de_nadege tableau des cartes bureau de nadège

Version :

rien

Définition à la ligne 821 du fichier traitement.c.

Référencé par attente_clic(), et traitement_arrive_case().

```
822 {
823   int int_des;
824   int int_choix;
825 
826   //on verifie si le joueur est emprisonner
827   if (pj_joueur->bool_laurence==true)
828   {
829     //il lance les dés afin d'essayer de faire un double
830     int_des=lancer_des(surf_ecran, surf_centre,pj_joueur, plateau);
831 
832     if (pj_joueur->int_double_tire == 1)
833     {
834       //le jeton est libéré
835       pj_joueur->bool_laurence=false;
836       pj_joueur->int_laurence=0;
837       avancer_jeton(surf_ecran, surf_centre, int_des, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
838     }
839     //sinon on verifie si les trois tours de prison n'ont pas aboutis
840     else if (pj_joueur->int_laurence == 3)
841     {
842       //on le libère
843       pj_joueur->bool_laurence=false;
844       pj_joueur->int_laurence=0;
845       avancer_jeton(surf_ecran, surf_centre, int_des, pj_joueur,plateau,bureau_de_krystel, bureau_de_nadege);
846     }
847     else
848     {
849       int_choix=attente_validation_message(surf_ecran,surf_centre,"Bureau de Laurence","Que vous voulez vous faire 
850       switch (int_choix)
851       {
852         case 4:
853           //on indique qu'il a passe un tour de plus
854           pj_joueur->int_laurence++;
855           break;
```

```
856         case 5:
857           pj_joueur->bool_laurence=false;
858           pj_joueur->int_laurence=0;
859           //on lui fait payer les 500 Fintz à la machine a café
860           //on retire au joueur l'argent
861           pj_joueur->int_argent=pj_joueur->int_argent - 500;
862           //on ajoute l'argent à la machine a café
863           plateau[20]->machine_a_cafe.int_argent=plateau[20]->machine_a_cafe.int_argent + 500;
864           break;
865         case 6:
866           if (pj_joueur->int_certificat > 0)
867             {
868               pj_joueur->bool_laurence=false;
869               pj_joueur->int_laurence=0;
870               pj_joueur->int_certificat-=1;
871             }
872           else
873             {
874               //on indique qu'il a passe un tour de plus
875               pj_joueur->int_laurence++;
876             }
877           break;
878         default:
879           break;
880       }
881
882     }
883   }
884 }
885
```

8.15.2.22 void traitement_perdu (joueur * *pj_joueur*)

Définition à la ligne 1063 du fichier traitement.c.

Référencé par traitement_bureau().

```
1063 {
1064   //on affiche le message
1065
1066   //on l'élimine
1067   traitement_elimination_joueur(pj_joueur);
1068 }
1069 }
```