Contents

1 Module `Glob` : Types et paramètres globaux partagés par les modules `Gui`[2] et `Jeu`[3]. **1**

2 Module `Gui` : Interface graphique pour le jeu *same game*. **2**
2.1 Position et taille du tableau de jeu. 2
2.2 Définition des boutons. 2
2.3 Taille de la fenêtre graphique. 3
2.4 Interaction. 3
2.5 Dessin. 3

3 Module `Jeu` : Le jeu du *same game*. **4**
3.1 Initialisation. 4
3.2 Fin de partie. 4
3.3 Sélections. 4
3.4 Suppression et compactage. 4
3.5 Déroulement du jeu. 5

4 Module `Tli` : Interface texte du jeu. **5**

1 Module `Glob` : Types et paramètres globaux partagés par les modules `Gui`[2] et `Jeu`[3].

```
val gb_width : int
```

```
val gb_height : int
```

```
val nb_lig : int
```

```
val nb_col : int
```

Largeur/nombre de colonnes et hauteur/nombre de lignes du jeu

```
val gb : int array array
```

Tableau de jeu: les pièces sont codées par des entiers; la valeur `0` code l'absence de pièce (case vide); les pièces marquées (voir fonction `Jeu.mark_sel`[3.3]) sont codées par une valeur négative.
Le tableau est un tableau de *colonnes*.

```
type  action =
  | Null
  | Next
  | Quit
  | Sel of int * int
```

Type des (inter)actions de jeu possible.

- `Null` action vide (on ne fait rien);
- `Next` procéder à l'effacement/compactage;
- `Quit` quitter le jeu;
- `Sel(icol, ilig)` marquer la sélection d'origine `(icol, ilig)`.

2 Module `Gui` : Interface graphique pour le jeu *same game*.

```
val gr_unit : int
```
Unité de longueur.

```
val gr_incr : int Pervasives.ref -> unit
```
Incrément d'une unité `gr_unit`.

2.1 Position et taille du tableau de jeu.

```
val gb_width : int
```
Largeur (en pixels).

```
val gb_height : int
```
Hauteur (en pixels).

2.2 Définition des boutons.

```
type  bt_descr = {
  x0 : int ;
  y0 : int ;
  width : int ;
  height : int ;
  color : Graphics.color ;
  label : string ;
}
```
Structure descriptive d'un bouton.

```
val next_bt : bt_descr
```
Bouton "next": activé pour effacer une sélection ou jouer une nouvelle partie.

```
val quit_bt : bt_descr
```
Bouton "quit": activé pour quitter le jeu.

2.3 Taille de la fenêtre graphique.

`val gw_width : int`
`val gw_height : int`

2.4 Interaction.

`val is_in_rect : int * int -> int -> int -> int -> int -> bool`
Vaut `true` si la position `(x,y)` est dans le rectangle `r_x r_y r_w r_h`.

`val is_in_gb : int * int -> bool`
Vaut `true` si la position `(x,y)` est dans le tableau de jeu.

`val is_in_next_bt : int * int -> bool`
Vaut `true` si la position `(x,y)` est dans le bouton "next".

`val is_in_quit_bt : int * int -> bool`
Vaut `true` si la position `(x,y)` est dans le bouton "quit".

`val gb_cell : int * int -> int * int`
Convertit la position `(x,y)` en coordonnées colonne/ligne du tableau de jeu `Glob.gb`.

`val click : unit -> Glob.action`
Renvoie l'action liée à l'évènement souris correspondant à la position d'un "clic" (on a laissé des affichages sur la sortie standard pour debuggage).

2.5 Dessin.

`val ucolor : Graphics.color array`
Table des couleurs des pièces non marquées.

`val mcolor : Graphics.color array`
Table des couleurs des pièces marquées.

`val draw_cell : int -> int -> int -> unit`
Dessine une pièce de couleur `c` dans un rectangle de coin inférieur gauche en `(x,y)`.

`val draw_gb : int array array -> unit`
Dessine le contenu du tableau de jeu `Glob.gb`.

`val redraw_gb : int array array -> unit`
Redessine le contenu du tableau de jeu `Glob.gb`.

`val draw_bt : bt_descr -> unit`

Dessine le bouton décrit par `bt`.

`val draw_next_bt : unit -> unit`
Dessine le bouton "next".

`val draw_quit_bt : unit -> unit`
Dessine le bouton "quit".

3 Module `Jeu` : Le jeu du *same game*.

3.1 Initialisation.

`val init_gb : unit -> unit`
Initialise aléatoirement le tableau de jeu.

3.2 Fin de partie.

`val end_game : unit -> bool`
Vaut `true` si la partie est terminée: il ne reste plus aucune sélection possible.

3.3 Sélections.

`val mark_sel : int * int -> unit`
Marque la sélection déterminée par la pièce en `(icol, ilig)`.

`val valid_sel : int -> int -> bool`
Vaut `true` si la pièce en `(icol, ilig)` n'est pas isolée.

3.4 Suppression et compactage.

`val collapse_down : int array -> unit`
Supprime les pièces marquées de la colonne `col` et compacte les pièces restantes.

`val collapse : unit -> unit`
Supprime les pièces marquées du tableau et compacte les colonnes restantes.

3.5 Déroulement du jeu.

`val last_act : Glob.action Pervasives.ref`

Variable globale: mémorise la dernière action effectuée (voir fonction `Jeu.game_act`[3.5]).

`val game_act : unit -> unit`

Effectue une action de jeu selon l'évènement souris enregistré (`Gui.click`[2.4]) et la dernière action effectuée (`Jeu.last_act`[3.5]).

`val game_loop : unit -> unit`

Boucle de jeu.

`val main : unit -> unit`

Lancement du jeu.

4 Module `Tli` : Interface texte du jeu.

`val ushapes : char array`

Table des caractères des pièces non marquées.

`val mshapes : char array`

Table des caractères des pièces marquées.

`val print_shape : int -> unit`

Affichage d'une pièce.

`val line : unit -> unit`

Affichage d'une ligne de séparation.

`val display_gb : unit -> unit`

Affichage de tableau de jeu.