Unité graphique minimale

lw

8 juillet 2005

1 Avant de commencer...

1.1 Initialiser

Avant de commencer à dessiner, il faut ouvrir une fenêtre graphique, cela peut être fait grâce à la procédure `initialiser`.

Attention, il est possible que l'opération échoue dans ce cas l'exception `PB_GRAPHIQUE` peut être déclenchée. D'autre part, l'unité `u_graphique_minimale` ne permet d'ouvrir qu'une seule fenêtre graphique à la fois. Un nouvel appel à la procédure `initialiser` déclenche une exception `MAUVAIS_USAGE`. On doit préalablement appeler la procédure `terminer`.

```
// procédure initialiser
// sert à ouvrir la fenêtre graphique ...
// peut eventuellement  déclencher
//  *  l'exception MAUVAIS_USAGE si la fenêtre
//     graphique a déjà été initialisée
//  *  l'exception PB_GRAPHIQUE si la bibliothèque
//     sous-jacente échoue dans l'ouverture
procedure initialiser;
```

1.2 Terminer

Avant de quitter le programme, il convient de fermer la fenêtre graphique. Si on essaie de fermer la fenêtre graphique alors qu'elle n'est pas ouverte, on déclenche l'exception `MAUVAIS_USAGE`.

```
// procédure terminer
// sert à fermer la fenêtre graphique
// peut éventuellement déclencher
//  *  l'exception MAUVAIS_USAGE si la fenêtre
//     graphique n'a pas déjà été initialisée
procedure terminer;
```

1.3 Rafraichir

La procédure `rafraichir` permet de redessiner la fenêtre graphique. Cette fonction n'est pas indispensable car par défaut la variable globale `rafraichit_auto` est positionnée sur la valeur `true`. Il y a alors rafraichissement après chaque appel à une primitive graphique. Toutefois, comme cela peut être assez coûteux, on peut donner à cette variable la valeur `false`. Dès lors, lorsqu'on dessine, l'affichage n'est pas systématiquement rafraichi et il faut le faire manuellement.

```
// ne pas utiliser.
procedure rafraichir;

var rafraichit_auto : BOOLEAN;
```

2 les propriétés de la fenêtre graphique

2.1 hauteur de la fenêtre graphique

La fonction `hauteur` (sans paramètre donc constante) retourne un entier correspondant à la hauteur de la fenêtre graphique. Un appel avant d'avoir initialisé déclenche l'exception `MAUVAIS_USAGE`.

```
// fonction hauteur
// résultat entier correspondant à la hauteur de la
// fenêtre graphique ouverte
// C.U. une fenetre doit être ouverte avant un appel
// à cette fonction, sinon elle déclenche l'exception
// MAUVAIS_USAGE
function hauteur  : CARDINAL;
```

2.2 largeur de la fenêtre graphique

La fonction `largeur` (sans paramètre donc constante) retourne un entier correspondant à la largeur de la fenêtre graphique. Un appel avant d'avoir initialisé déclenche l'exception `MAUVAIS_USAGE`.

```
// fonction largeur
// résultat entier correspondant à la largeur de la
// fenêtre graphique ouverte
// C.U. une fenetre doit être ouverte avant un appel
// à cette fonction, sinon elle déclenche l'exception
// MAUVAIS_USAGE
function largeur  : CARDINAL;
```

2.3 hauteur des caractères

La fonction `hauteur_caractere` (sans paramètre donc constante) retourne un entier correspondant à la hauteur d'un caractère tel qu'il sera dessiné dans la fenêtre graphique. Un appel avant d'avoir initialisé déclenche l'exception `MAUVAIS_USAGE`.

```
// fonction hauteur_car
// (constante) correspondant à la hauteur
// en point d'un caractère
// C.U. cette fonction nécessite que
// la fenêtre graphique ait été initialisée
// peut déclencher MAUVAIS_USAGE sinon
function hauteur_car : CARDINAL;
```

2.4 largeur des caractères

La fonction `largeur_caractere` (sans paramètre donc constante) retourne un entier correspondant à la largeur d'un caractère tel qu'il sera dessiné dans la fenêtre graphique. Un appel avant d'avoir initialisé déclenche l'exception `MAUVAIS_USAGE`.

```
// fonction largeur_car
// (constante) correspondant à la largeur
// en point d'un caractère
// C.U. cette fonction nécessite que
// la fenêtre graphique ait été initialisée
// peut déclencher MAUVAIS_USAGE sinon
function largeur_car : CARDINAL;
```

3 Manipulation des couleurs

3.1 les couleurs prédéfinies

les couleurs suivantes sont définies dans l'unité `u_graphique_minimale` : `NOIR`, `ARGENT`, `GRIS`, `BLANC`, `MARRON`, `ROUGE`, `POURPRE`, `FUSCHIA`, `VERT`, `CITRON_VERT`, `OLIVE`, `JAUNE`, `MARINE`, `BLEU`, `SARCELLE`, `VERT_EAU`. Les couleurs sont composées d'un mélange des couleurs rouge, verte et bleue, l'intensité de chaque couleur est représentée par un nombre réel compris entre 0 et 1 :

NOM	rouge	vert	bleu
NOIR	0	0	0
ARGENT	0,75	0,75	0,75
GRIS	0,5	0,5	0,5
BLANC	1	1	1
MARRON	0,5	0	0
ROUGE	1	0	0
POURPRE	0,5	0	0,5
FUSCHIA	1	0	1
VERT	0	0,5	0
CITRON_VERT	0	1	0
OLIVE	0,5	0,5	0
JAUNE	0,5	0,5	0
MARINE	0	0	0,5
BLEU	0	0	1
SARCELLE	0	0,5	0,5
VERT_EAU	0	1	1

3.2 fabrication d'autres couleurs

Une fonction `couleur` permet de fabriquer une couleur en mélangeant les couleurs de base rouge, verte et bleue, en précisant pour chacune d'elles un nombre réel entre 0 et 1 correspondant à l'intensité (1 correspond à l'intensité maximale et 0 à l'intensité minimale)

```
// fonction couleur
// constructeur
//  [0,1]^3 ---> T_COULEURS
//associe à un triplet de nombre réels de l'intervalle [0,1]
```

```
// Chaque élément du triplet représentant
// l'intensité lumineuse relative pour chacune des couleurs
// rouge vert bleu pour fabriquer une couleur par
// synthèse additive
function couleur(const r,v,b:REAL):T_COULEURS;
```

3.3 analyse d'une couleur

On peut connaître l'intensité de bleu qui compose une couleur, grâce à la fonction `intensite_bleue`. De même il existe une fonction `intensite_verte` et une fonction `intensite_rouge`. Ces trois fonctions prennent comme argument une expression de type `T_COULEURS` et produisent comme résultat un nombre réel compris entre 0 et 1.

```
// 3 fonctions intensité_... (resp bleue, rouge et verte)
// selecteur
// T_COULEURS ---> [0,1]
// permettent d'obtenir la composition d'une couleur
// en bleu, rouge et vert
function intensite_bleue(const c : T_COULEURS) : REAL;
function intensite_rouge(const c : T_COULEURS) : REAL;
function intensite_verte(const c : T_COULEURS) : REAL;
```

4 Primitives graphiques

4.1 tracer un point

la procédure `tracer_point` permet de tracer un point sur la fenêtre graphique avec une couleur précisée. Un appel avant d'avoir initialisé déclenche l'exception `MAUVAIS_USAGE`.

```
// procédure tracer_point
// allume le point de coordonnée x,y
// avec la couleur la plus proche de c
// réalisable par le périphérique graphique utilisé
procedure tracer_point(     const x,y   : CARDINAL;
                            const c     : T_COULEURS);
```

4.2 tracer une ligne

la procédure `tracer_ligne` permet de tracer une ligne sur la fenêtre graphique avec une couleur précisée. Un appel avant d'avoir initialisé déclenche l'exception `MAUVAIS_USAGE`.

```
// procédure tracer_ligne
// allume le segment dont les extremitées sont
// proches respectivement de coordonnées x1,y1 et
// x2,y2
// avec la couleur la plus proche de c
// réalisable par le périphérique graphique utilisé
procedure tracer_ligne(     const x1,y1 : CARDINAL;
                            const x2,y2 : CARDINAL;
                            const c     : T_COULEURS);
```

4.3 remplir l'écran

```
// procédure remplir écran
// remplit l'écran
// avec la couleur la plus proche de c
// réalisable par le périphérique graphique utilisé
procedure remplir_ecran(    const c     : T_COULEURS);
```

4.4 tracer une chaîne

```
// procédure tracer_chaine
// dessine la chaine s au point de coordonnée x,y
// en utilisant la couleur c pour l'encre
// et la couleur f pour le fond
procedure tracer_chaine(    const x,y   : CARDINAL;
                            const s     : STRING;
                            const c,f   : T_COULEURS);
```

5 divers

5.1 tester la couleur d'un point

```
// prédicat test
// permet de savoir si le point de coordonnée
// x,y est de la meme couleur que la couleur
// la plus proche de c...
function test(const x,y       : CARDINAL ;
              const c         : T_COULEURS):BOOLEAN;
```

5.2 attendre la pression d'une touche

```
// procédure attendre_pression_touche
// attend que la *fenetre graphique*
// recoive une pression de touche

// cette procédure est à utilisée plutot
// que de lire un caractère dans la console
// car lors de la lecture d'un caractère dans
// la console, l'appel est bloquant, et
// la fenêtre graphique n'est pas rafraichie
// pendant ce temps...

procedure attendre_pression_touche;
```

6 les exceptions

Deux exceptions sont définies dans l'unité `u_graphique_minimale`. Elles se nomment l'exception `MAUVAIS_USAGE` et l'exception `PB_GRAPHIQUE`

```
// l'exception PB_GRAPHIQUE se déclenche lorsqu'un
// appel à la bibliothèque graphique sous-jacente
// ne s'est pas déroulé convenablement
PB_GRAPHIQUE = class(exception);

// l'exception MAUVAIS_USAGE se déclenche lorsque
// l'utilisateur n'a pas respecté les consignes
// par exemple en cas d'utilisation sans avoir
// préalablement initialisé
MAUVAIS_USAGE = class(exception);
```