GUI

Programmation d'un éditeur de texte

Difficile de programmer un éditeur de texte.

On va simplement utiliser une API qui fournit tous les outils pour utiliser des éditeurs de texte comme objets.

Un éditeur de texte sera pour nous une généralisation d'un text-field :

-comporte plusieurs lignes

-offre la possibilité de changer la police de caractères

-offre la possibilité d'utiliser du texte en couleur, en gras, etc…

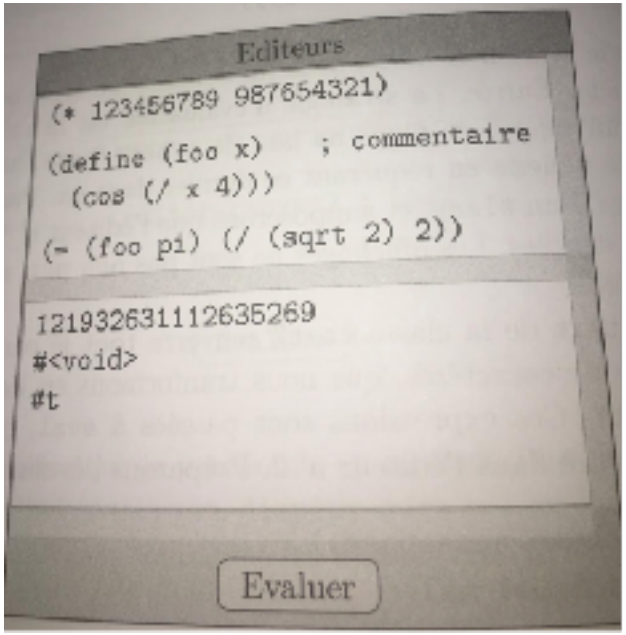
GUI

Un éditeur de texte comporte deux notions :

- la notion de zone de l'écran dans laquelle vont se produire les affichages
- la notion d'écrivain chargé de produire les affichages avec crayons de couleur, etc…

La zone de l'écran est un objet de la classe editor-canvas%

L'écrivain provient de la classe text%



GUI

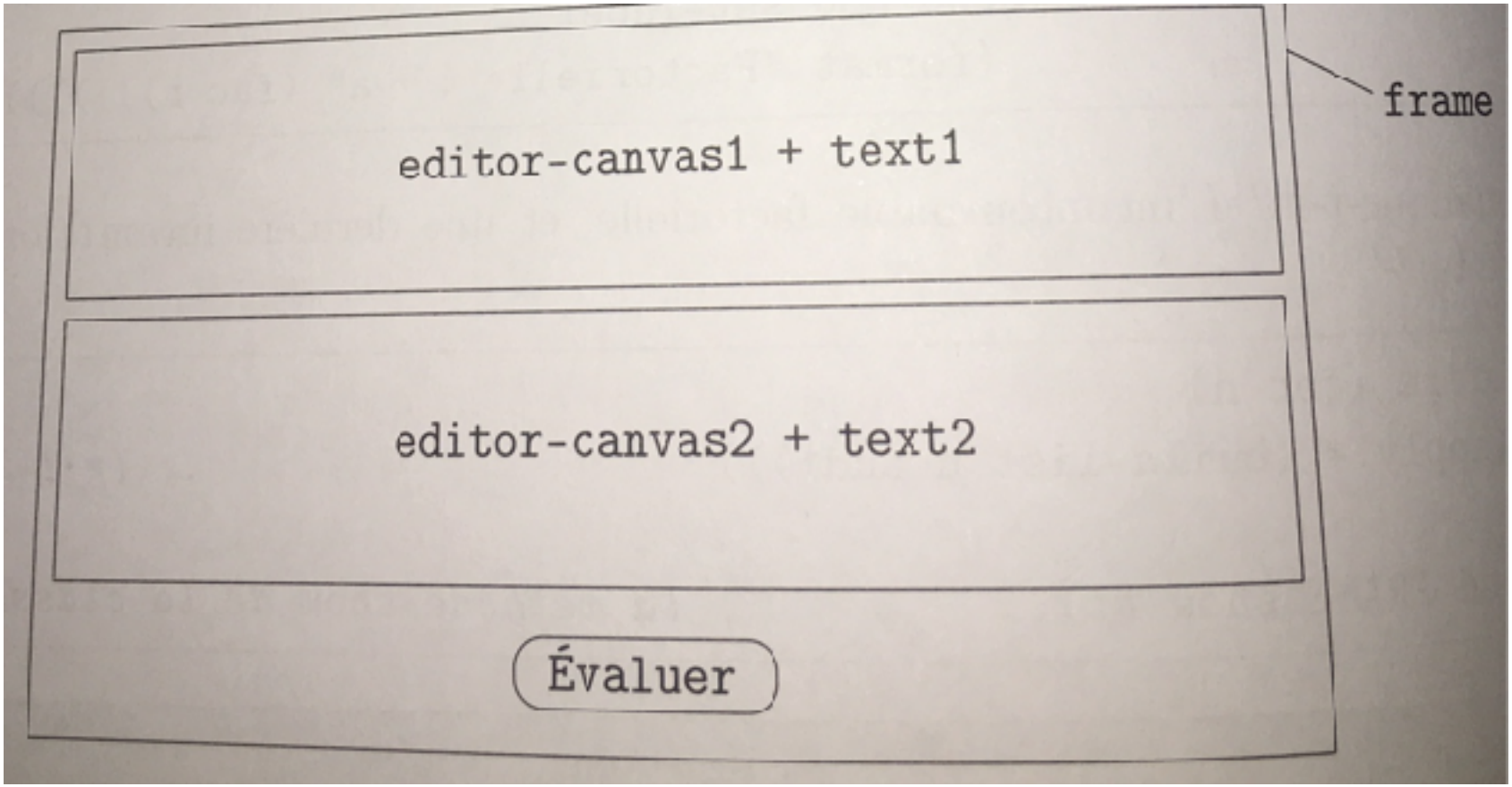
Pour construire l'éditeur de texte, on s'inspirera de l'interface de DrRacket : éditeur de définitions en haut et fenêtre top-level en bas

Donc on va utiliser 2 éditeurs de texte placés dans un panneau vertical contes dans une fenêtre.

Un bouton Evaluer provoquera évaluation des expressions entrées dans la fenêtre n°1 et leurs valeurs seront affichées dans la fenêtre n°2.

GUI

Voila ce que l'on veut obtenir



GUI

On va commencer par définir la fenêtre que l'on positionne en haut de l'écran (elle occupera le tiers central en largeur).

On écrira la fonction get-display-size qui retournera deux résultats capturés par un let-values et contenant les dimensions de l'écran.

Entete du programme

```
#lang racket/gui

(require (only-in mzlib/string read-from-string-all expr->string))
```

GUI

```
(define FRAME
(let-values (((w h) (get-display-size)))
    (new frame% (min-width (quotient w 3))
    (label « éditeurs »)
    (x (quotient w 3))
    (y  30))))
(define VPANEL
 (new vertical-panel% (parent FRAME) (alignment '(center center))))
```

GUI

```
(define TEXT1 (new text%)
(define ECANVAS1
  (new editor-canvas% (parent VPANEL)
     (min-height 200) (editor TEXT1)))
(define TEXT2 (new text%)
(define ECANVAS2
  (new editor-canvas% (parent VPANEL)
   (min-height 200) (editor TEXT2))))
```

GUI

Bouton Evaluer

Créer un bouton EVAL qui déclenchera l'evaluation des expressions de l'éditeur n°1

Stratégie d'évaluation

Nous allons faire appel à la fonction eval dans un espace de noms ns issu de scheme/base.

Hypothèse : l'éditeur n°1 va contenir des expressions scheme simple.

On va aussi se contenter d'afficher les résultats des évaluations.

GUI

La méthode get-text de la classe text% renverra tout le contenu de l'éditeur n°1 dans une seule chaine de caractères. Il faudra la transformer en liste d'expressions en utilisant read-from-string-all.

Ces expressions sont alors passées à eval puis les résultats sont affichés au fur et à mesure dans l'éditeur n°2.

Préparer l'évaluateur

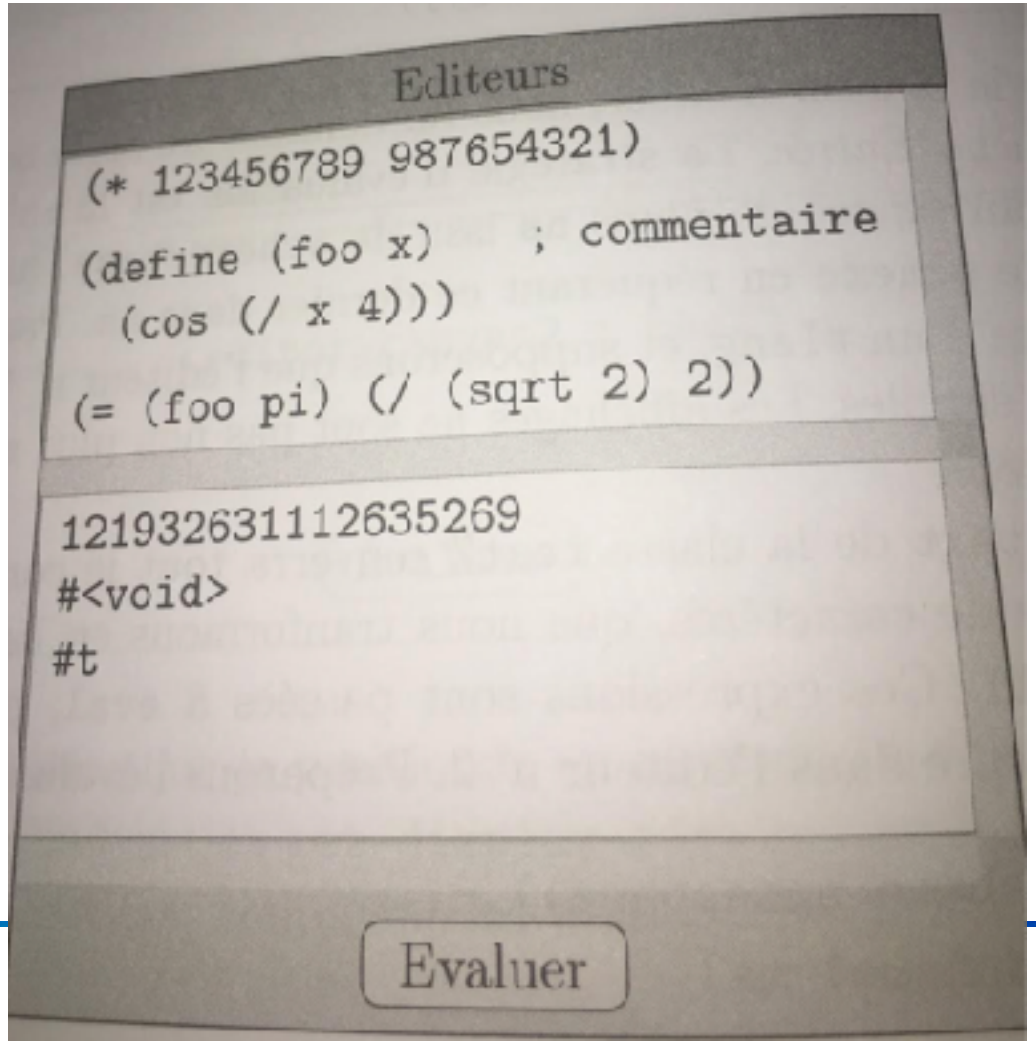
(define ns (make-base-namespace))

(eval '(require scheme) ns)

GUI

```
(define BUTTON-EVAL
(new button% (parent VPANEL)
   (label « EVAL! »)
    (callback
      (lambda(obj evt).    ; obj est le bouton
        (send TEXT2 erase)
        (let ((L (read-from-string-all. (send TEXT1 get-text))))
           (map (lambda (expr)
                        (send TEXT2 insert
                                 expr->string (eval expr ns)))
                        (send TEXT2 insert «\n »)) L))))))
```

GUI


Editeurs
(* 123456789 987654321)
(define (foo x) ; commentaire
(cos (/ x 4)))
(= (foo pi) (/ (sqrt 2) 2))
121932631112635269
#<void>
#t
Evaluer