# GUI

Dessin dans une fenêtre graphique.

Nous avons vu un éditeur de texte composé d'une feuille de papier (editor-canvas%) et d'un écrivain (text%).

De meme une zone à dessiner sera formé :

. d'une feuille de papier (canvas%)

. d'un peintre (dc%).

Feuille deja donc on va se concentrer sur le peintre.

# GUI

Exemple : calcul de  pi par une méthode de Monte Carlo

But  : utiliser une suite de simulations (méthode de MonteCarlo) basée sur des probabilités et calculer la moyenne des observations.

Application au calcul de pi :  On inscrit un disque de rayon R dans un carré de coté 2R, pus on tire N points au hasard dans ce carré (N doit être grand - exemple 5000). Et on observe le nombre S de points qui tombe à l'intérieur du disque.

2 facons d'interpreter la probabilité p qu'un point tombe dans le cercle :

- quotient du nb  de succès (S) par le nb d'experiences (N) : p = S/N
- rapport entre l'aire du disque et l'aire du carré: p = pi R*R/4 R*R

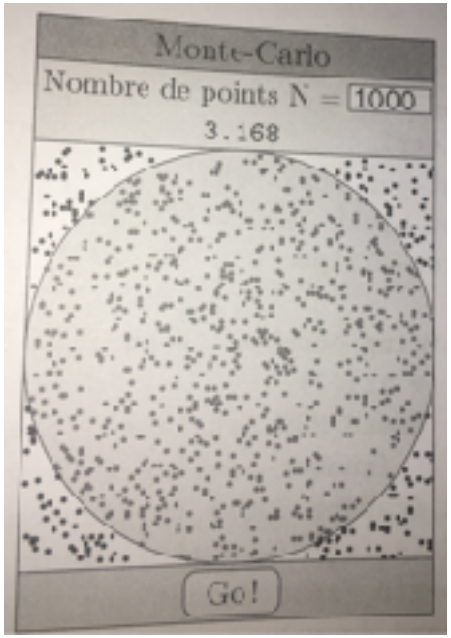
Donc S/N = pi/4. donc pi = 4S/N

# GUI

A faire : procéder à la simulation dans une fenêtre.

La fenêtre contiendra 4 composants :

- un text-field pour entrer le nb de points
- Un message pour afficher l'approximation de pi.
- un canvas, zone à dessiner pour voir le disque et les points.
- un bouton pour relancer plusieurs fois le calcul avec le meme nb de points.



# GUI

Pour dessiner il faut : des classes canvas% et dc%.

Un canvas est une zone pourtant recevoir du graphique et du texte (et des événements - clavier ou souris -

Attention : lorsque la fenêtre est redimensionnée avec la souris, il faudra mettre à jour le dessin pour l'adapter à la nouvelle échelle. Cette mise à jour est réalisée automatiquement par la méthode on-paintd cela classe canvas%. C'est le système qui l'invoque lui-meme (il vous suffit de spécifier dans cette méthode le dessin que vous souhaiter réaliser). On peut aussi l'invoquer soi-même dans le call-back du bouton.

# GUI

Donc 2 solutions :

- fournir un call-back spécifique au canvas (paint-callback) ou
- programmer une sous-classe de canvas% en redéfinissant la méthode on-paint (utile en cas de la gestion de la souris ou du clavier).

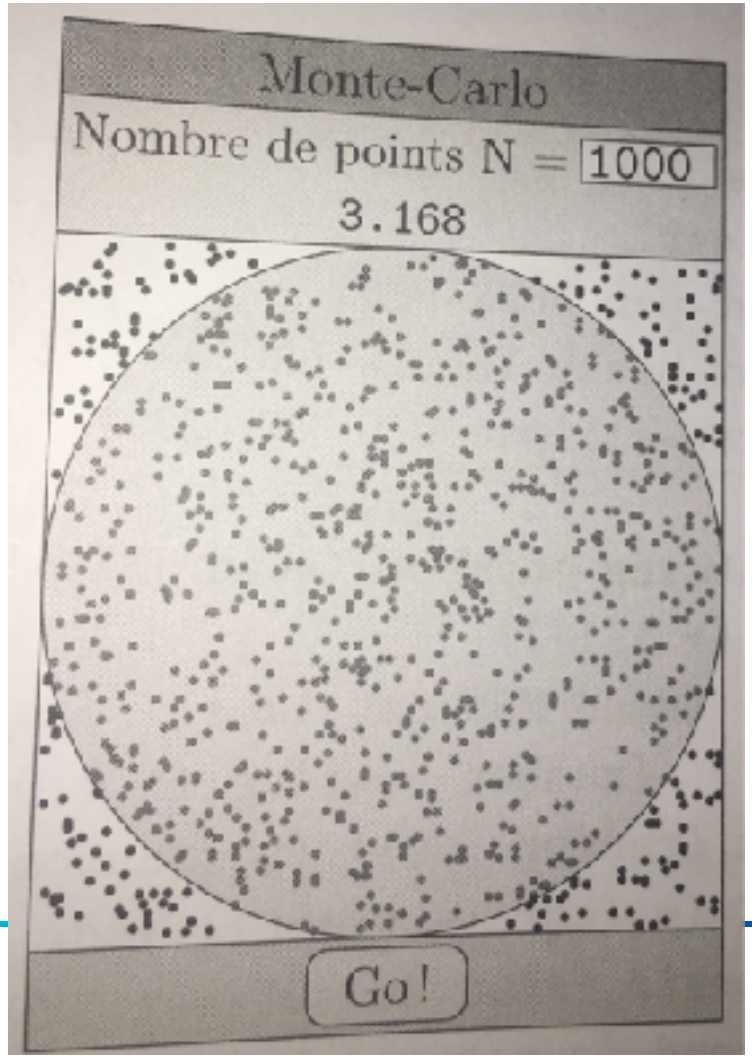
Choisissons la premiere méthode.

De plus il faut définir un crayon pour dessiner (classe pen%), noir par defaut. Pour remplir les formes fermées, il faut une grosse (calsse brush%), noire par défaut.

Nous allons définir 3 crayons de couleurs et une brosse jaune (ppour peindre l'interieur du disque. (la documentation nous dit d'utiliser make-objets et pas new dans les classe pen% et brush%)

# GUI

Ce que l'on veut



# GUI

```
83 ;;; api3.rkt -- Simulation graphique
84 #lang racket/gui

85 (define RED-PEN (make-object pen% "red" 3 'solid))
86 (define BLACK-PEN (make-object pen% "black" 3 'solid))
87 (define BLUE-PEN (make-object pen% "blue" 1 'solid))
88 (define YELLOW-BRUSH (make-object brush% "yellow" 'solid))

89 (define FRAME
90   (new frame% (label "Monte-Carlo")
91     (stretchable-width #f) (stretchable-height #f)))

92 (define VPANEL (new vertical-panel% (parent FRAME)))      ; centré...
```

# GUI

On peut placer ensuite  le text-field avec 2000 points par défaut.

Son call-back lancera la méthode on-paint du canvas sur pression de la touche ENTREE.

```
93  (define TEXT-FIELD
94    (new text-field% (parent VPANEL)
95      (label "Nombre de points N =")
96      (init-value "5000")
97      (callback (lambda (obj evt)
98                  (when (equal? (send evt get-event-type)
99                                'text-field-enter)
100                   (send CANVAS on-paint))))))
```

# GUI

Ensuite il faut placer le message contenant le résultat et le canvas.

La taille du canvas va fixer celle de la fenêtre (non redimensionnable).

Le paint-callback est une fonction prenant l'objet canvas ainsi que l'événement qui a lancé le callback (les 2 sont généralement ignorés).

IL faut ensuite demander le peintre associé dc (device context). Ce dernier va effacer le canvas, prendre un crayon bleu et une brosse jaune, dessiner et peindre  le disque.

Puis fin peut débute le calcul numérique sur N points tirés au hasard, N étant lu dans le text-field. &

Le point (x,y) tombe dans le disque  si $(x-150)^2+ (y-150)^2< 150^2$

```
101 (define CANVAS
102   (new canvas% (parent VPANEL)
103     (min-width 300) (min-height 300) (style '(border))
104     (paint-callback
105       (lambda (obj evt)
106         (let ((dc (send obj get-dc)))       ; récupération du peintre
107           (send dc clear)                   ; effacement du canvas
108           (send dc set-pen BLUE-PEN)        ; le bord du disque
109           (send dc set-brush YELLOW-BRUSH) ; l'intérieur du disque
110           (send dc draw-ellipse 0 0 299 299)  ; dessin du disque
111           (let ((s 0)                         ; nombre de succès
112                 (N (string->number (send TEXT-FIELD get-value))))
113             (do ((i 0 (+ i 1)))
114               ((= i N) (send MSG set-label  ; affichage du résultat
115                              (number->string (* 4.0 (/ s N)))))
116               (let ((x (random 300)) (y (random 300)))
117                 (if (< (+ (sqr (- x 150)) (sqr (- y 150))) (sqr 150))
118                     (begin (send dc set-pen RED-PEN) (set! s (+ s 1)))
119                     (send dc set-pen BLACK-PEN))
120                 (send dc draw-point x y)))))))))
```

# GUI

```
121 (define BUTTON
122   (new button% (parent VPANEL) (label "Go !")
123     (callback (lambda (obj evt) (send CANVAS on-paint)))))

124 (send FRAME show #t)
```