Programmation Fonctionnelle

Séance 7 : 2htdp/image

Alexandros Singh

Université Paris 8

26 novembre 2023

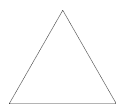
- Il existe différents paquets Racket pour la création d'images.

- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.

- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de “teachpacks” accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

```
(triangle 100 "outline" "black")
```

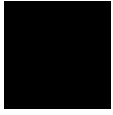


- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

```
(triangle 100 "solid" "red")
```

- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

```
(square 100 "solid" "black")
```



- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

```
(rectangle 200 400 "solid" "black")
```

- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

```
(circle 100 "solid" "black")
```

- Il existe différents paquets Racket pour la création d'images.
- Le paquet `htdp-lib`, qui fait partie d'une collection de "teachpacks" accompagnant le manuel "How To Design Programms", fournit un module `2htdp/image` particulièrement facile à utiliser.
- Il fournit des fonctions pour dessiner des primitives géométriques (formes) telles que :

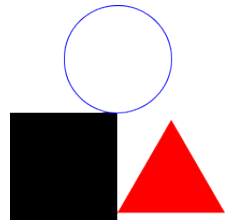
```
(ellipse 200 100 "solid" "black")
```

Il permet également de composer plusieurs images pour en obtenir une nouvelle :

```
> (define i1 (square 100 "solid" "black"))
> (define i2 (triangle 100 "solid" "red"))
> (beside i1 i2)
```

Il permet également de composer plusieurs images pour en obtenir une nouvelle :

```
> (define i1 (square 100 "solid" "black"))
> (define i2 (triangle 100 "solid" "red"))
> (above
    (circle 50 "outline" "blue")
    (beside i1 i2))
```

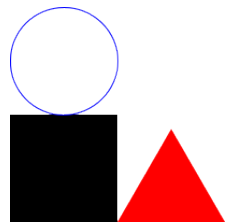


Les versions -/align de ces fonctions permettent d'obtenir un placement plus précis :

```
> (define i1 (square 100 "solid" "black"))
> (define i2 (triangle 100 "solid" "red"))
> (beside/align "bottom" i1 i2)
```

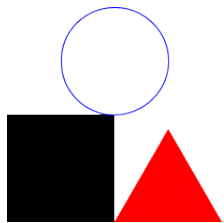
Il permet également de composer plusieurs images pour en obtenir une nouvelle :

```
> (define i1 (square 100 "solid" "black"))
> (define i2 (triangle 100 "solid" "red"))
> (define i3 (beside/align "bottom" i1 i2)
> (above/align "left"
    (circle 50 "outline" "blue")
    i3)
```



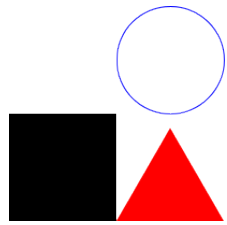
Il permet également de composer plusieurs images pour en obtenir une nouvelle :

```
> (define i1 (square 100 "solid" "black"))
> (define i2 (triangle 100 "solid" "red"))
> (define i3 (beside/align "bottom" i1 i2)
> (above/align "middle"
    (circle 50 "outline" "blue")
    i3)
```



Il permet également de composer plusieurs images pour en obtenir une nouvelle :

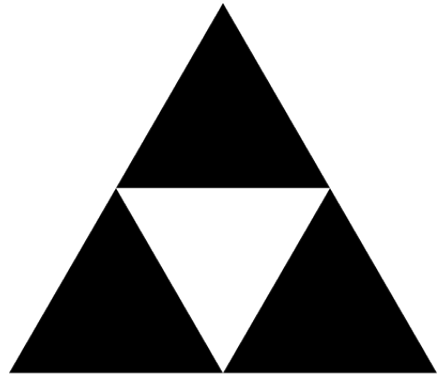
```
> (define i1 (square 100 "solid" "black"))
> (define i2 (triangle 100 "solid" "red"))
> (define i3 (beside/align "bottom" i1 i2)
> (above/align "right"
    (circle 50 "outline" "blue")
    i3)
```



Sierpiski

Essayez de reproduire cette image :

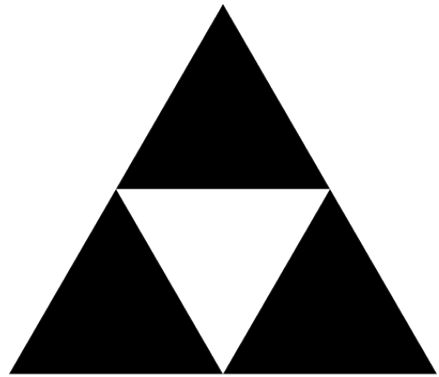
```
???
```



Sierpiski

Essayez de reproduire cette image :

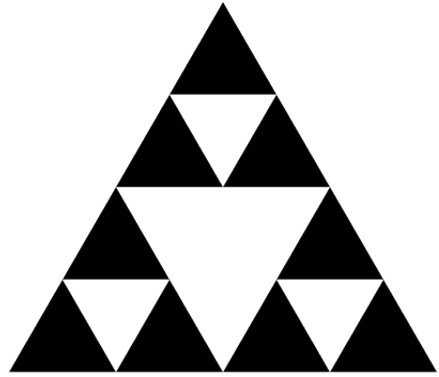
```
> (define t (triangle 100 "solid" "black"))
> (above t (beside t t))
```



Sierpiski

Et celle-ci :

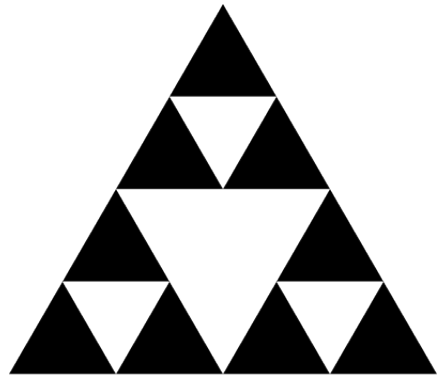
```
???
```



Sierpiski

Et celle-ci :

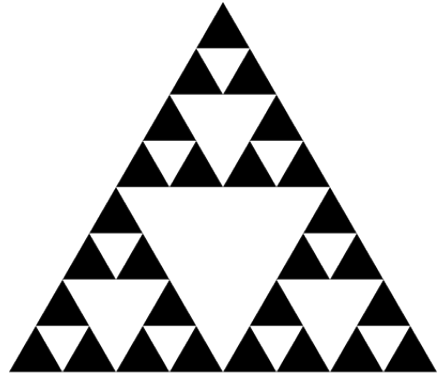
```
> (define t (triangle 100 "solid" "black"))
> (define i1 (above t (beside t t)))
> (above i1 (beside i1 i1))
```



Sierpiski

Et celle-ci (x2) :

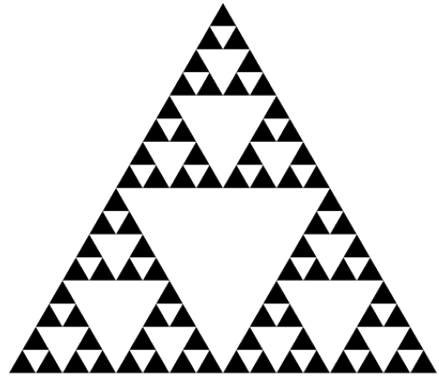
```
oh non...
```



Sierpiski

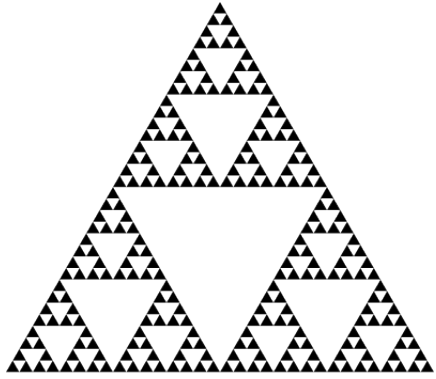
Et celle-ci (x3) :

```
c'est trop de travail !
```



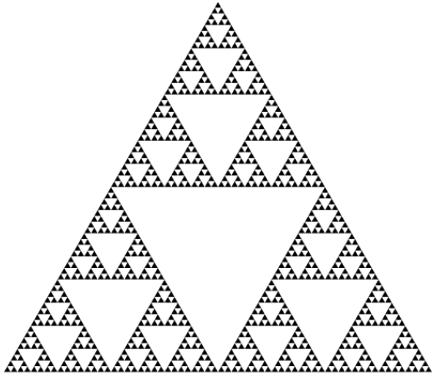
Et celle-ci (x4) :

```
si seulement il y avait un moyen...
```



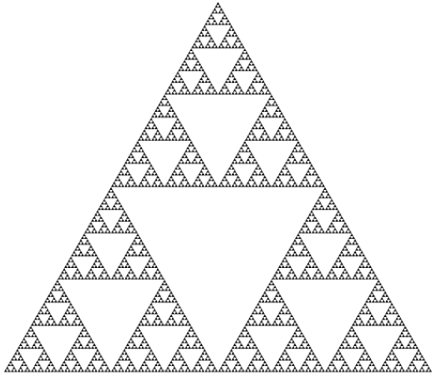
Et celle-ci (x5) :

```
de faire la même chose...
```



Et celle-ci (x6) :

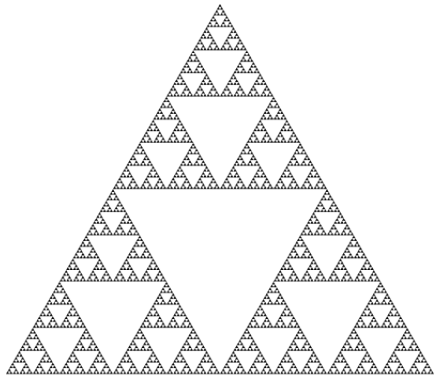
```
avec des instances plus petites de
la même image...
```



Sierpiski

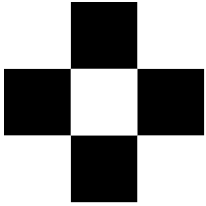
La récursivité à la rescousse !

```
(define (trig-stack s)
  (let ([t (triangle s "solid" "black")])
    (above
     t
     (beside t t))))
(define (sierpinski-trig s n)
  (cond
    [(eq? n 0) (trig-stack s)]
    [else
     (let
       ([rec (sierpinski-trig s (- n 1))])
       (above
        rec
        (beside rec rec)))]))
```



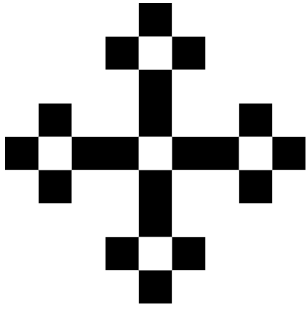
Encore un

Essayez de recréer celui-ci :



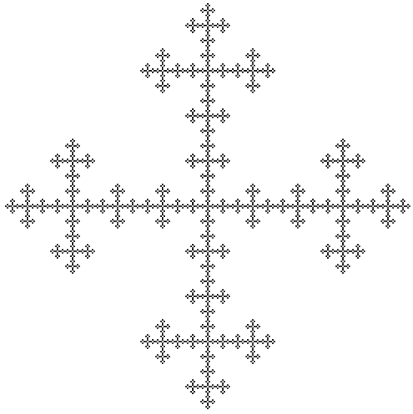
Encore un

Essayez de recréer celui-ci :

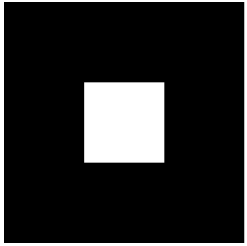


Encore un

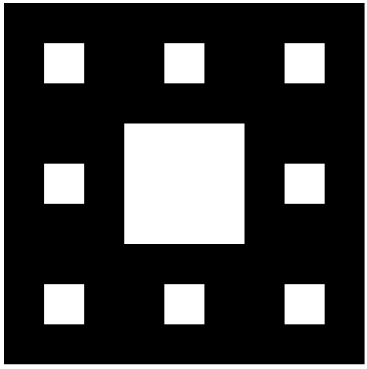
Essayez de recréer celui-ci :



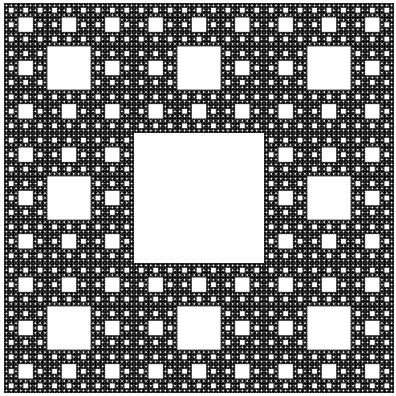
Encore, encore un



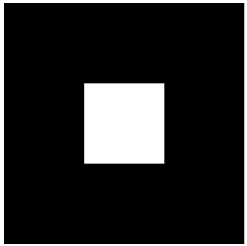
Encore, encore un



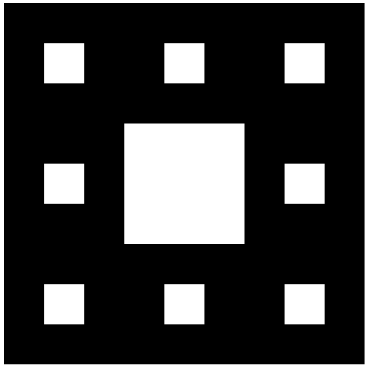
Encore, encore un



Encore, encore un



Encore, encore un



Encore, encore un