

# Le langage Racket

---

- Un langage de programmation fonctionnelle
- 24/11/2021

# Dessin et fractales

1. Formes simples
2. Superposition de dessins
3. Polygones et couleurs
4. Sauvegarder un dessin dans un fichier
5. Exercices

# 1. Formes simples

- L'utilisation du package 2htdp/image permet de dessiner en utilisant des formes simples.
- Les fonctions des base sont:
  - **1. circle**
  - **2. ellipse**
  - **3. line**
  - **4. add-line**
  - **5. add-curve**
  - **6. add-solid-curve**
  - **7. text**
  - **8. text/font**
  - **9. empty-image**

# 1. Formes simples

**1. circle:** attend un rayon / un mode de tracé / une couleur

-Remarque:

Le mode de tracé peut être une valeur de transparence variant de 0 à 255.

- le mode de tracé solide est équivalent à 255
- le mode de tracé 0 est équivalent à invisible
- On peut avoir d'autres formes en utilisant les fonctions de square et rectangle

# 1. Formes simples

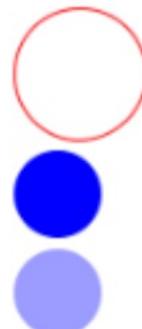
Exemple:

```
1 (require 2htdp/image)
2 (circle 30 "outline" "red")
3 (circle 20 "solid" "blue")
4 (circle 20 100 "blue")
```

# 1. Formes simples

Exemple:

```
1 (require 2htdp/image)
2 (circle 30 "outline" "red")
3 (circle 20 "solid" "blue")
4 (circle 20 100 "blue")
```



# 1. Formes simples

2. **ellipse** : attend une largeur / une hauteur / un mode de tracé / une couleur

**Exemple:** (ellipse 40 70 "solid" "gray")

# 1. Formes simples

**ellipse** : attend une largeur / une hauteur / un mode de tracé / une couleur

**Exemple:** (ellipse 40 70 "solid" "gray")



# 1. Formes simples

**3. line:** attend une valeur en x, une valeur en y et une couleur

**Remarque:**

- Pour une valeur nulle en x, la ligne est verticale.
- Pour une valeur nulle en y, la ligne est vers le haut.

**Exemple:** (line 80 0 "orange")

# 1. Formes simples

**3. add-line:** permet de superposer des lignes sur un dessin en cours

L'image sur laquelle est superposée la ligne, est passée en premier paramètre.

Les paramètres suivants concernent la ligne en question.

(add-line image x1 y1 x2 y2 color)

Ajoute une ligne à l'image, en partant du point (x1,y1) et en allant vers le point (x2,y2).

# 1. Formes simples

## 3. add-line:

Exemple:

```
1 (require 2htdp/image)
2 (add-line (square 40 "solid" "gray")
3           0 40 80 0 "red")
```

# 1. Formes simples

## 3. add-line:

Exemple:

```
1 (require 2htdp/image)
2 (add-line (square 40 "solid" "gray")
3           0 40 80 0 "red")
```



# 1. Formes simples

**4. add-curve:** permet de tracer des courbes (ajoute une courbe à une image).

10 paramètres qui sont:

- L'image concernée par la tracé
- 4 coordonnées de départ : x, y , l'angle en degré, le temps de conservation de cet angle
- 4 coordonnées de fin : x, y , l'angle en degré, le temps de conservation de cet angle
- une couleur de tracé

# 1. Formes simples

**4. add-curve:** permet de tracer des courbes (ajoute une courbe à une image).

## Exemple:

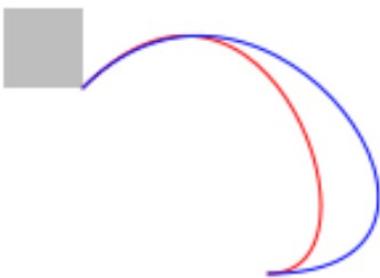
```
1 (require 2htdp/image)
2 (add-curve (add-curve (square 30 "solid" "gray")
3                         30 30 45 1 100 100 180 0.5 "red")
4                         30 30 45 1 100 100 180 1 "blue"))
```

# 1. Formes simples

**4. add-curve:** permet de tracer des courbes (ajoute une courbe à une image).

## Exemple:

```
1 (require 2htdp/image)
2 (add-curve (add-curve (square 30 "solid" "gray")
3                         30 30 45 1 100 100 180 0.5 "red")
4                         30 30 45 1 100 100 180 1 "blue"))
```



# 1. Formes simples

## 6. add-solid-curve

Ajoute une courbe à l'image comme add-curve, sauf qu'il remplit la région à l'intérieur de la courbe.

### Exemple:

```
> (add-solid-curve (rectangle 100 100 "solid" "black")
                     20 20 0 1
                     80 80 0 1
                     "white")
```



# 1. Formes simples

## 7. Text

(text chaîne de caractère taille couleur): Construit une image qui dessine la chaîne de caractères donnée, en utilisant la taille et la couleur de la police.

### Exemple:

(text "Dessiner avec Racket" 24 "olive")

Dessiner avec Racket

# 1. Formes simples

## 8. text/font

Construit une image qui dessine la chaîne de caractères donnée, en utilisant une spécification de police complète.

; Exemple:

```
(text/font "Hello" 24 "olive"  
          "Gill Sans" 'swiss 'normal 'bold #f)
```

A large, bold, olive green text "Hello" centered on the slide.

## 2. Superpositions de dessins

Les fonctions qui permettent de superposer des dessins de toutes formes sont:

**overlay**: le premier paramètre est une surcouche du second

**underlay**: le premier paramètre est une sous-couche du second

## 2. Superpositions de dessins

**overlay:** le premier paramètre est une surcouche du second

**underlay:** le premier paramètre est une sous-couche du second

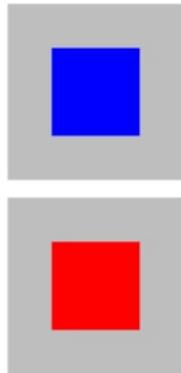
```
1 (require 2htdp/image)
2 (underlay (square 80 "solid" "gray") (square 40 "solid" "blue"))
3 (overlay (square 40 "solid" "red") (square 40 "solid" "gray"))
```

## 2. Superpositions de dessins

**overlay**: le premier paramètre est une surcouche du second

**underlay**: le premier paramètre est une sous-couche du second

```
1 (require 2htdp/image)
2 (underlay (square 80 "solid" "gray") (square 40 "solid" "blue"))
3 (overlay (square 40 "solid" "red") (square 40 "solid" "gray"))
```



## 2. Superpositions de dessins

Avec les fonctions **overlay/xy** il est possible de préciser des décalages entre les dessins superposés.

**(underlay/xy i1 x y i2) → image**

Construit une image en superposant i1 sous i2.

Les images sont initialement alignées sur leurs coins supérieurs gauches,

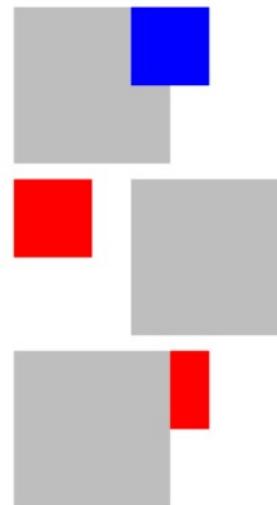
puis i2 est décalé vers la droite de x pixels et vers le bas de y pixels.

## 2. Superpositions de dessins

```
1 (require 2htdp/image)
2 (underlay/xy (square 80 "solid" "gray")
3   60 0
4   (square 40 "solid" "blue"))
5 (overlay/xy (square 40 "solid" "red")
6   60 0
7   (square 80 "solid" "gray"))
8 (overlay/xy (square 80 "solid" "gray")
9   60 0
10  (square 40 "solid" "red"))
```

## 2. Superpositions de dessins

```
1 (require 2htdp/image)
2 (underlay/xy (square 80 "solid" "gray")
3   60 0
4   (square 40 "solid" "blue"))
5 (overlay/xy (square 40 "solid" "red")
6   60 0
7   (square 80 "solid" "gray"))
8 (overlay/xy (square 80 "solid" "gray")
9   60 0
10  (square 40 "solid" "red"))
```

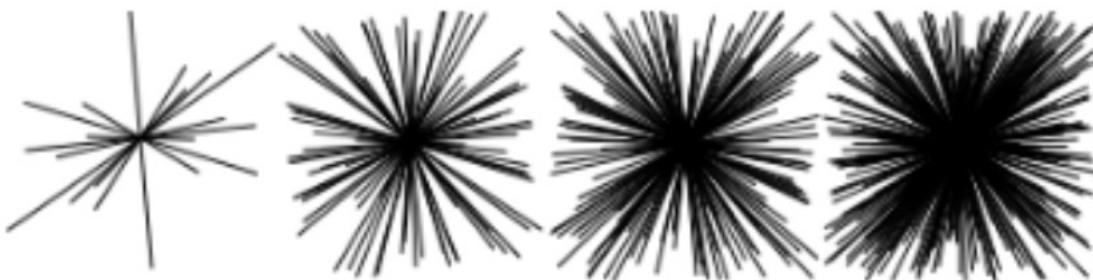


## 2. Superpositions de dessins

Le programme suivant présente un autre exemple d'utilisation des fonctions de superposition.

```
1 (require 2htdp/image)
2 (define (rline n)
3   (if (= n 0) empty-image
4       (underlay
5         (line (random 100) (- (random 200) 100) "black")
6         (rline (- n 1)))
7       )))
8 (underlay/xy (underlay/xy (underlay/xy
9                           (rline 10)
10                          100 0 (rline 50))
11                         200 0 (rline 100))
12                         300 0 (rline 200))
```

```
1 (require 2htdp/image)
2 (define (rline n)
3   (if (= n 0) empty-image
4       (underlay
5         (line (random 100) (- (random 200) 100) "black")
6         (rline (- n 1)))
7       )))
8 (underlay/xy (underlay/xy (underlay/xy
9                           (rline 10)
10                          100 0 (rline 50))
11                          200 0 (rline 100))
12                          300 0 (rline 200))
```



## 2. Superpositions de dessins

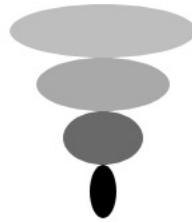
Il est également possible de placer les images les unes par rapport aux autres à l'aide des fonctions:

1. **above**: verticalement vers le bas.
2. **above/align** : verticalement vers le bas avec un alignement à gauche ou à droite.
3. **beside**: horizontalement vers la gauche

## 2. Superpositions de dessins

### Exemples:

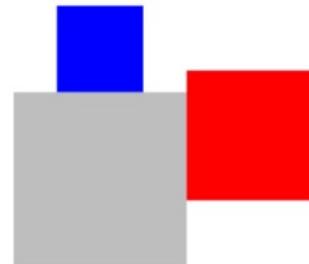
```
> (above (ellipse 70 20 "solid" "gray")
          (ellipse 50 20 "solid" "darkgray")
          (ellipse 30 20 "solid" "dimgray")
          (ellipse 10 20 "solid" "black"))
```



## 2. Superpositions de dessins

### Exemples:

```
1 (require 2htdp/image)
2 (beside (above (square 40 "solid" "blue")
3                   (square 80 "solid" "gray")))
4   (square 60 "solid" "red"))
```



```
> (above/align "right"
  (ellipse 70 20 "solid" "gold")
  (ellipse 50 20 "solid" "goldenrod")
  (ellipse 30 20 "solid" "darkgoldenrod")
  (ellipse 10 20 "solid" "sienna"))
```



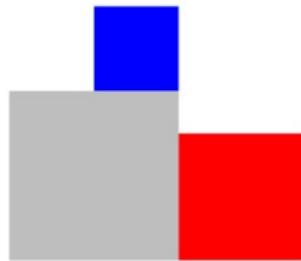
```
> (above/align "left"
  (ellipse 70 20 "solid" "yellowgreen")
  (ellipse 50 20 "solid" "olivedrab")
  (ellipse 30 20 "solid" "darkolivegreen")
  (ellipse 10 20 "solid" "darkgreen"))
```



## 2. Superpositions de dessins

### Exemples:

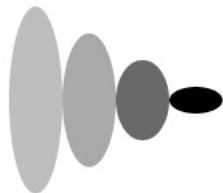
```
1 (require 2htdp/image)
2 (beside/align "bottom"
3   (above/align "right"
4     (square 40 "solid" "blue")
5     (square 80 "solid" "gray")))
6   (square 60 "solid" "red"))
```



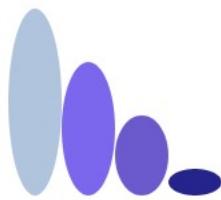
## 2. Superpositions de dessins

### Exemples:

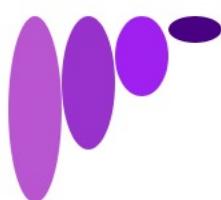
```
> (beside (ellipse 20 70 "solid" "gray")
            (ellipse 20 50 "solid" "darkgray")
            (ellipse 20 30 "solid" "dimgray")
            (ellipse 20 10 "solid" "black"))
```



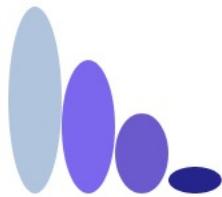
```
> (beside/align "bottom"
  (ellipse 20 70 "solid" "lightsteelblue")
  (ellipse 20 50 "solid" "mediumslateblue")
  (ellipse 20 30 "solid" "slateblue")
  (ellipse 20 10 "solid" "navy"))
```



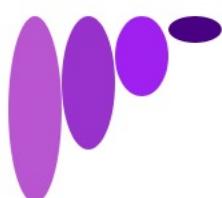
```
> (beside/align "top"
  (ellipse 20 70 "solid" "mediumorchid")
  (ellipse 20 50 "solid" "darkorchid")
  (ellipse 20 30 "solid" "purple")
  (ellipse 20 10 "solid" "indigo"))
```



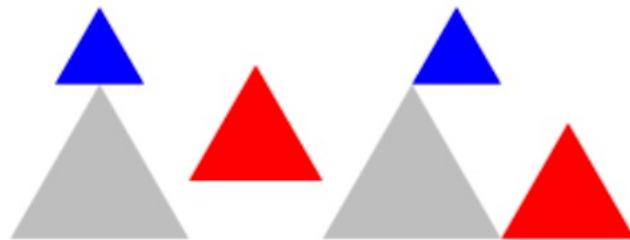
```
> (beside/align "bottom"
  (ellipse 20 70 "solid" "lightsteelblue")
  (ellipse 20 50 "solid" "mediumslateblue")
  (ellipse 20 30 "solid" "slateblue")
  (ellipse 20 10 "solid" "navy"))
```



```
> (beside/align "top"
  (ellipse 20 70 "solid" "mediumorchid")
  (ellipse 20 50 "solid" "darkorchid")
  (ellipse 20 30 "solid" "purple")
  (ellipse 20 10 "solid" "indigo"))
```



```
1 (require 2htdp/image)
2 (beside
3   (beside
4     (above (triangle 40 "solid" "blue")
5           (triangle 80 "solid" "gray")))
6     (triangle 60 "solid" "red"))
7   (beside/align "bottom"
8     (above/align "right"
9       (triangle 40 "solid" "blue")
10      (triangle 80 "solid" "gray")))
11     (triangle 60 "solid" "red"))
12 ))
```



## 2. Superpositions de dessins

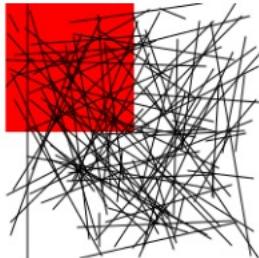
### Exemples:

```
1 (require 2htdp/image)
2 (define (rline2 input n dx dy)
3   (if (= n 0)
4     input
5     (add-line (rline2 input (- n 1) dx dy)
6               (random dx) (random dy)
7               (random dx) (random dy) "black")
8   )))
9 (overlay/xy (rline2 empty-image 100 200 200) 0 0
10  (square 100 "solid" "red"))
```

## 2. Superpositions de dessins

### Exemples:

```
1 (require 2htdp/image)
2 (define (rline2 input n dx dy)
3   (if (= n 0)
4     input
5     (add-line (rline2 input (- n 1) dx dy)
6               (random dx) (random dy)
7               (random dx) (random dy) "black")
8   ))
9 (overlay/xy (rline2 empty-image 100 200 200) 0 0
10  (square 100 "solid" "red"))
```



## 2. Superpositions de dessins

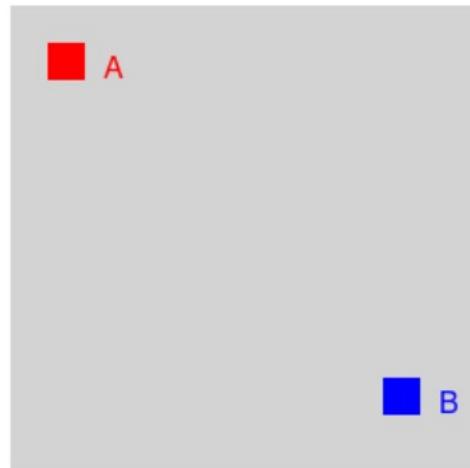
### Exemples:

La fonction `text` permet d'ajouter du texte<sup>53</sup> dans un dessin, comme présenté dans le programme suivant :

```
1 (require 2htdp/image)
2 (underlay/xy (underlay/xy (underlay/xy (underlay/xy
3   (square 250 "solid" "lightgray")
4     20 20 (square 20 "solid" "red"))
5       50 20 (text "A" 16 "red"))
6         200 200 (square 20 "solid" "blue"))
7           230 200 (text "B" 16 "blue"))
```

## 2. Superpositions de dessins

**Exemples:**



### 3. Polygones et couleurs

Les polygones sont définis par :  
données / mode / couleur

Données : peut être composée d'une ou plusieurs valeurs.

Mode= définit le remplissage du dessin qui est plein (i.e. pour “solid” ou ‘solid) ou filiforme (i.e. pour “outline” ou ‘outline) ;

### 3. Polygones et couleurs

la fonction rectangle utilise comme données les valeurs de largeur et de hauteur.

la fonction triangle définit un triangle équilatéral avec pour unique donnée une taille de côté.

### 3. Polygones et couleurs

(triangle side-length mode color)	pour un triangle équilatéral
(right-triangle side-length1 side-length2 mode color)	pour un triangle rectangle
(isosceles-triangle side-length angle mode color)	pour un triangle isocèle
(square side-len mode color)	pour un carré
(rectangle width height mode color)	pour un rectangle

Fig. 25 – Principaux polygones possibles.

### 3. Polygones et couleurs

Tomato, Red, Pink, Brown, Orange, Gold, Yellow, Green, Turquoise, Cyan, Blue, Indigo, Purple, White, LightGray, Silver, Gray, DarkGray, Black

Fig. 26 – Principaux mots-clés possibles pour la partie *couleur*.

## 4. Sauvegarder un dessin dans un fichier

```
1 (require 2htdp/image)
2 (save-image
3   (overlay/xy (rline2 empty-image 100 200 200)
4     0 0
5     (square 100 "solid" "red"))
6   "image1.png")
7 (save-image
8   (overlay/xy
9     (overlay/xy (rline2 empty-image 100 200 200)
10       0 0
11       (square 100 "solid" "red"))
12       0 0
13       (square 200 "solid" "lightgray"))
14   "image2.png")
```

```
#t
```

```
#t
```

## 4. Sauvegarder un dessin dans un fichier

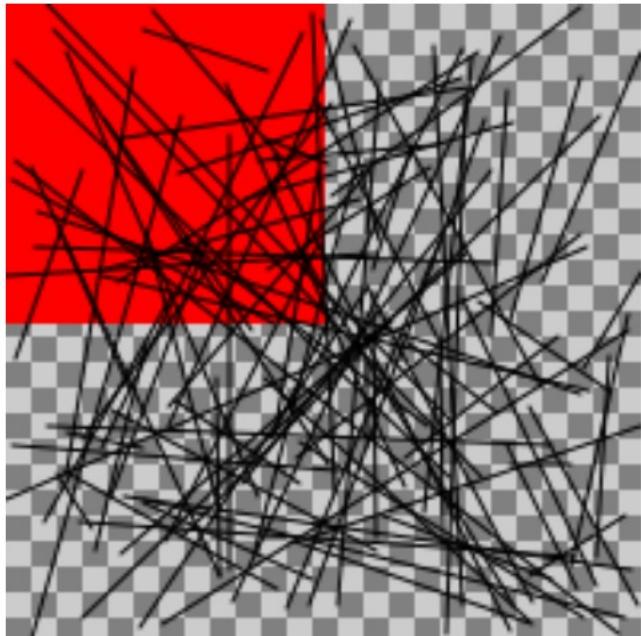


Fig. 27 – `image1.png`.

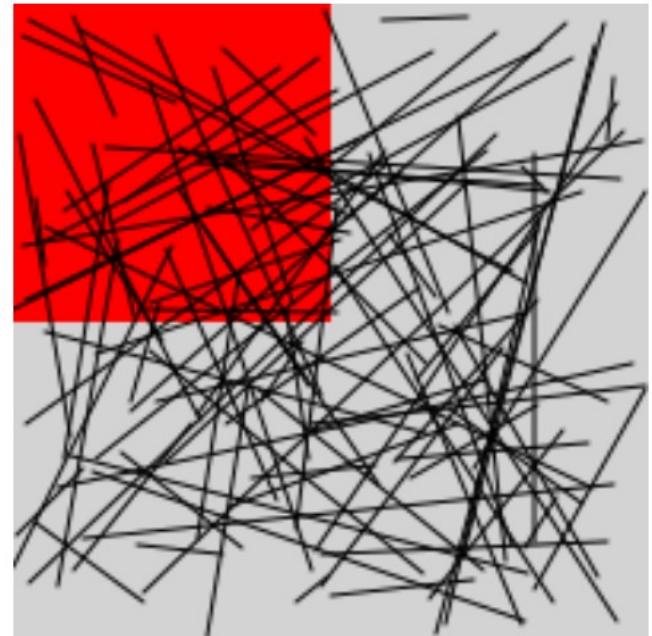


Fig. 28 – `image2.png`.

## 5. Exercices

### Question 1

---

Réaliser le dessin présenté ci-dessous en figure 39, dans lequel :

- Le grand carré gris-clair fait 160 par 160.
- Le petit carré noir fait 80 par 80.
- La couleur gris-clair est définie par "lightgray".
- La couleur noir est définie par "black".



Fig. 39 – Deux carrés superposés.

## 5. Exercices

### Solution 1

---

```
1 (require 2htdp/image)
2 (overlay/xy
3   (square 80 "solid" "black") -80 -80
4   (square 160 "solid" "gray"))
5 )
```

On peut également utiliser underlay/xy en inversant l'ordre des figures.

## Question 2

---

Réaliser les dessins présentés ci-dessous en figures 40 et 41, dans lesquels :

- Un carré rouge de 50 par 50 est dans la partie supérieure gauche d'un carré noir de 100 par 100 (figure 40 en haut).
- Un carré rouge de 50 par 50 est dans la partie inférieure droite d'un carré noir de 100 par 100 (figure 40 en bas).
- Des carrés rouges de 50 par 50 sont sur des carrés noirs de 100 par 100 et on place également un carré rouge de 150 par 150 (figure 41).



Fig. 40 – Carrés rouges de 50 par 50 sur carrés noirs.

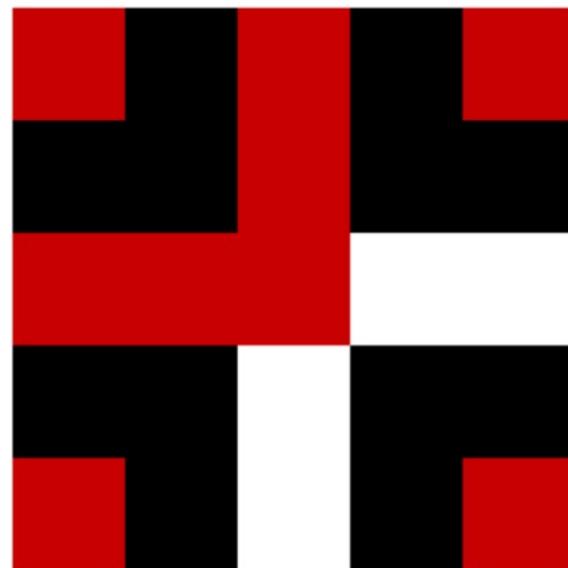


Fig. 41 – Avec un carré rouge de 150 par 150.

## Solution 2

---

Partie commune :

```
1 (require 2htdp/image)
2 (define (S x col) (square x "solid" (color col 0 0 255)))
```

Pour la figure 40 en haut :

```
(overlay/xy (S 50 200) 0 0 (S 100 0))
```

Pour la figure 40 en bas :

```
(overlay/xy (S 50 200) -50 0 (S 100 0))
```

Pour la figure 41 :

```
1 (overlay/xy
2   (overlay/xy
3     (overlay/xy (overlay/xy (S 50 200) 0 0 (S 100 0))
4       150 0 (overlay/xy (S 50 200) -50 0 (S 100 0)))
5       0 150 (overlay/xy (overlay/xy (S 50 200) 0 -50 (S 100 0))
6         150 0 (overlay/xy (S 50 200) -50 -50 (S 100 0))))
7       0 0 (S 150 200))
```

### Question 3

---

Définir une fonction  $f$  permettant de dessiner un motif regroupant un cercle plein sur un carré plein ; le cercle et le carré sont de couleur aléatoire.

Définir une fonction  $g$  permettant de dessiner  $n$  fois  $f$  en ligne.

Définir une fonction  $h$  permettant de dessiner  $m$  fois  $g$  en colonne.

Appeler la fonction `(h n m side)` permettant de dessiner des motifs de côté `side` en utilisant  $f$  sur  $n$  lignes et  $m$  colonnes comme sur la figure 42.

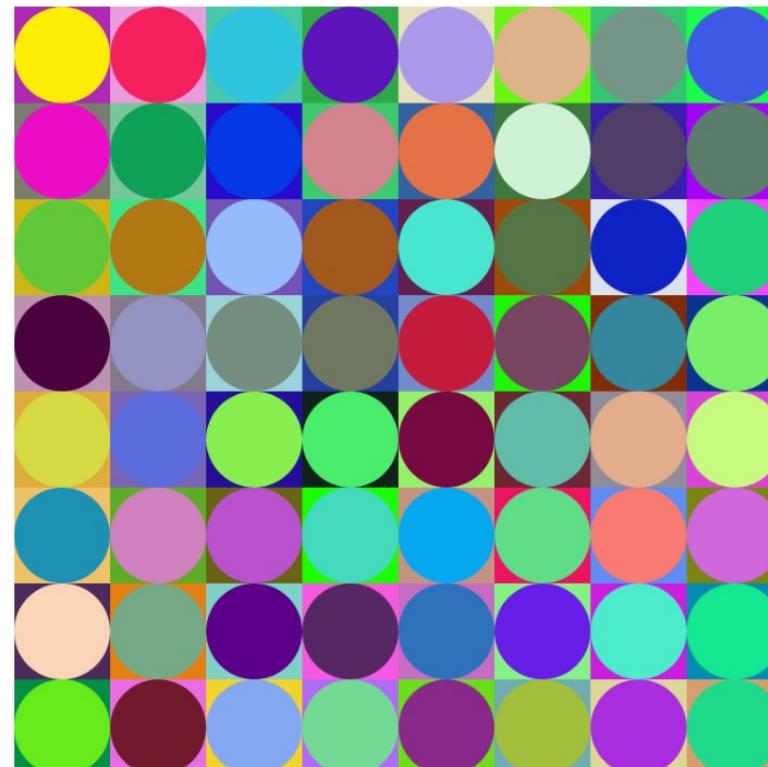


Fig. 42 – `(h 8 8 100)`.

## Solution 3

---

```
1 (require 2htdp/image)
2 (define (f side)
3   (let ([C1 (color (random 255) (random 255) (random 255) 255)]
4       [C2 (color (random 255) (random 255) (random 255) 255)])
5     (underlay (square side "solid" C1) (circle (/ side 2) "solid" C2)))
6 )
7 (define (g n side)
8   (if (= n 1) (f side)
9       (beside (f side) (g (sub1 n) side)))
10  ))
11 (define (h col line side)
12   (if (= line 1) (g col side)
13       (above (g col side) (h col (sub1 line) side)))
14  ))
15 (h 8 8 100)
```