Tours de Hanoï : version graphique

L'objectif du TP est de vous faire programmer une version graphique des tours de Hanoï. La première étape est d'observer le fonctionnement de la version avec affichage en mode texte:

```
(define (dohanoi n to from using)
  (if (> n 0)
    (begin
      (dohanoi (- n 1) using from to)
      (display "move ")
      (display from)
      (display " --> ")
      (display to)
      (newline)
      (dohanoi (- n 1) to using from)
      #t
    )
    #f
  )
)

(define (hanoi n)
  (dohanoi n 'C 'A 'B)
)
```

Vous pouvez lancer le programme pour n = 3, pour voir le résultat :

```
> (hanoi 3)
move A --> C
move A --> B
move C --> B
move A --> C
move B --> A
move B --> C
move A --> C
#t
```

Le programme ne modifie rien, il se contente d'afficher les déplacements. Pour la version graphique, nous allons avoir besoin d'une représentation de nos tours de Hanoï, qui sera modifiée par le programme. Pour cela, le plus simple (comme annoncé en TD) est de choisir une représentation par un vector de taille 3, dont chaque élément est une liste contenant les disques ordonnés du haut vers le bas sur chacune des tours, les disques étant représentés par des entiers proportionnels à leur taille. Par exemple, considérant que les tours sont dans l'ordre (départ, intermédiaire, arrivée), la représentation de la situation initiale pour un problème à n = 3 serait :

```
'#((1 2 3) () ())
```

- Ecrire la fonction (init-hanoi n) qui renvoie un vector représentant les tours de hanois en position initiale.

```
> (init-hanoi 3)
'#((1 2 3) () ())
```

- Ecrire la fonction (move-hanoi from to tours) qui renvoie la configuration des tours après le déplacement du premier disque du pilier numéro from vers le pilier numéro to

```
> (move-hanoi 0 1 (init-hanoi 3))
'#((2 3) (1) ())
```

- Il n'y a plus qu'à modifier les fonctions do-hanoi et hanoi pour prendre en compte le déplacement (en violet, la trace de l'exécution, en bleu le résultat)

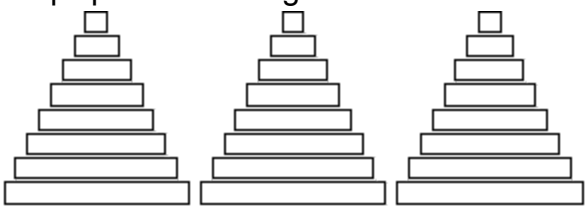
```
> (hanoi-physique 3)
move 0 --> 2#((2 3) () (1))
move 0 --> 1#((3) (2) (1))
move 2 --> 1#((3) (1 2) ())
move 0 --> 2#(() (1 2) (3))
move 1 --> 0#((1) (2) (3))
move 1 --> 2#((1) () (2 3))
move 0 --> 2#(() () (1 2 3))
'#(() () (1 2 3))
```

Nous avons maintenant un objet représentant les tours de hanoï. Pour faire une version graphique, il *suffit* d'écrire une fonction qui affichera graphiquement les tours de hanoï.

- Ecrire une fonction **`(draw-disque dc taille pos-v pilier)`** affichant dans un drawing context **`dc`** un disque de taille **`taille`**, à la position verticale (ordre sur le pilier, en partant du bas) **`pos-v`**, et sur le pilier **`pilier :`**

```
(define dc (new bitmap-dc% [bitmap (make-bitmap 300 100)]))
(map (lambda (x) (draw-disque dc x (- 9 x) 0)) '(1 2 3 4 5 6 7 8))
(map (lambda (x) (draw-disque dc x (- 9 x) 1)) '(1 2 3 4 5 6 7 8))
(map (lambda (x) (draw-disque dc x (- 9 x) 2)) '(1 2 3 4 5 6 7 8))
(make-object image-snip% (send dc get-bitmap))
```

Ce qui produira l'image suivante :



On a fini, ou presque …

- Ecrire une fonction (draw-hanoi tours) qui prend une représentation des tours en paramètre et qui produit l'image associée
- Modifier la fonction move-hanoi pour intégrer l'affichage graphique