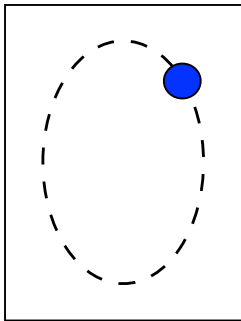


Animations



Livre PCPS, Chap. 4

Simuler le mouvement

- Les animations sont des techniques très utilisées dans les pages Web [par exemple avec HTML5 et son *Canvas*, ou bien Java et ses *applets*].
- Le teachpack [2htdp/universe](#) de Racket (déjà intégré au teachpack **valrose**) va nous permettre de programmer facilement de petites scènes animées. Applications à la géométrie, à la physique, aux jeux, etc.
- Une **horloge** sera automatiquement mise en place pour scander le temps, et donc l'évolution de la scène, par défaut tous les 1/28 de seconde (donc 28 images par seconde).
- Une **animation** se construit comme un **dessin animé** : ce n'est en effet pas autre chose qu'une suite d'images défilant très vite pour donner l'illusion du mouvement !
- La **sonorisation** d'une animation (en Scheme) sera vue plus tard...
- Les **jeux en réseau** (possibles) ne sont pas abordés dans ce cours. 2

Comment programmer une animation ?

- La méthodologie prônée par Racket est **MVC** (**M**odel-**V**iew-**C**ontroller). Le but est d'animer un monde d'objets virtuels (balles, personnages, etc).

1. Préciser le **Modèle mathématique** ou **Monde** : l'ensemble minimum des variables qui décrivent l'état des objets. TRES IMPORTANT !

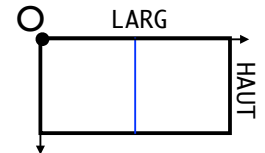
- la position (x,y) pour une balle se dirigeant au hasard.
- l'angle polaire θ pour une balle tournant sur un cercle.
- un paramètre t pour une balle se dirigeant sur une trajectoire d'équation paramétrique $x = f(t)$, $y = g(t)$.

2. Préciser comment le Monde **évolue** à chaque top d'horloge

3. Préciser comment le Monde sera transformé en une **scène** (image rectangulaire) contenant des images : la **Vue**.

4. Préciser (optionnellement) comment ce Monde va **interagir** avec l'utilisateur, via le clavier ou la souris.

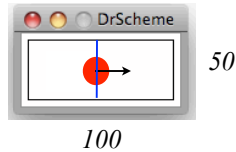
- **Mise en Place de la Scène**. Une **scène** est une image rectangulaire. Le FOND sera une scène contenant les éléments **fixes** de l'animation, qui ne bougeront jamais. Elle est souvent réduite à un rectangle de dimensions LARG et HAUT.



```
(define LARG 100)
(define HAUT 50)
(define X0 (/ LARG 2) ; si j'ai besoin des...
         Y0 (/ HAUT 2)) ; ...coordonnées du centre
(define FOND (underlay (empty-scene LARG HAUT)
                       (line 0 HAUT "blue")))
```

- Pour animer le mouvement d'un objet sur un fond fixe, on **placera** l'image de cet objet à un certain point (x,y) de la scène, et ses coordonnées (x,y) changeront un tout petit peu à chaque image...
- Donc utilisation intensive de (**place-image** *img* *x* *y* *scene*) ainsi que de (**scene+line** *img* *x1* *y1* *x2* *y2* *color*).

- Nous allons animer une balle **rouge** qui oscille entre les murs gauche et droit, avec une vitesse nulle au rebond (une sorte de *pendule horizontale*).



- Plutôt que traiter un problème de *collision* entre le disque et les murs, nous modéliserons la position du centre de la balle sous la forme d'une **fonction périodique** d'un seul paramètre réel m .

- Le **MONDE** sera donc réduit à la seule variable m . On est en présence d'une animation à 1 paramètre.

- Et quoi de mieux qu'un **sinus** pour obtenir un mouvement périodique ?

$$(x, y) = (50 + 50 \sin m, 25) \quad m \in [0, +\infty[$$

- L'abscisse x varie entre 0 et 100, tandis que y est constant.

ici le monde est m

Le MONDE n'est autre que l'ensemble des variables indépendantes gouvernant le phénomène et à partir desquelles on peut dessiner la scène à un instant donné !

5

- **L'affichage du monde.** Etant donné un Monde m , comment le transformer en une **scène** (image rectangulaire) pour le visualiser ?

```
; l'image de la balle ne dépend pas de sa position !
(define BALLE (circle 10 'solid "red"))

(define (dessiner m) ; monde → scène
  (place-image BALLE (* X0 (+ 1 (sin m))) Y0 FOND))
```

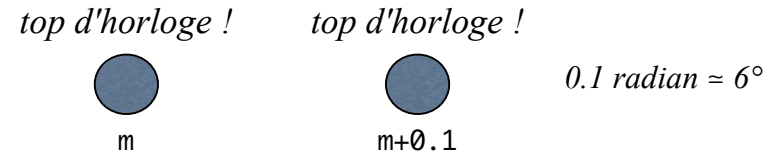
Pour ceux qui ont déjà un peu programmé : bien voir que l'on ne décrit pas l'animation par une **boucle**, on exprime seulement **dans des fonctions séparées** a) comment le monde évolue et b) comment il se visualise de manière statique à un moment donné ! **Ceci est CAPITAL...**

NB. La méthodologie **MVC** sépare en principe le calcul du monde et son rendu graphique. Mais parfois l'image du monde fait partie du monde !

- Il reste à voir comment va s'organiser la simulation dans le temps... En réalité, comme dans un dessin animé : **image par image...**

7

- **La mise à jour du monde.** Etant donné un Monde m (ensemble de variables), quel est le Monde suivant [au prochain top d'horloge] ? J'essaye empiriquement :



- On dit qu'on a **discrétisé** la fonction $m \mapsto 50 + 50 \sin(m)$. On ne la calcule **pas en continu**, mais par pas **discrets** de 0.1 :

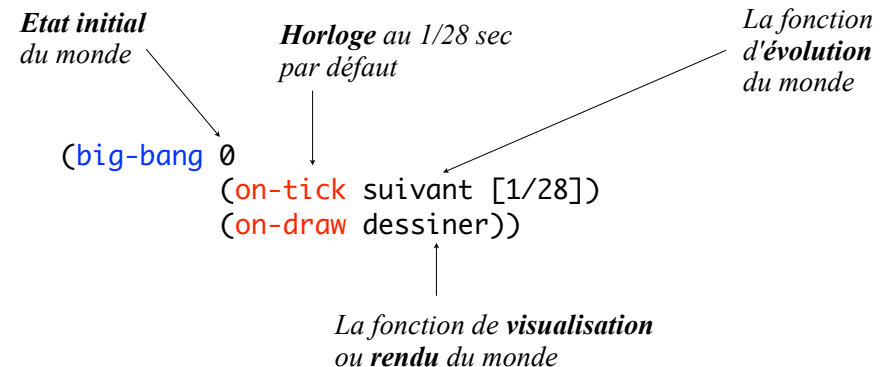
```
(define (suivant m) ; monde → monde
  (+ m 0.1))
```

N.B. On pourrait prendre pour m le temps t , mais rester indépendant procure plus de souplesse pour régler la vitesse de la simulation. Pensez que le monde est décrit par un seul paramètre, que ce soit le temps ou autre chose !

6

- **Les tops d'horloge.** Une horloge *invisible* et *silencieuse* va automatiquement être mise en place, à une certaine fréquence, 1/28 sec par défaut. **Donc 28 fois par seconde, le modèle mathématique sera mis à jour, et transformé en une scène.** Ces deux actions sont bien distinctes et décrites par les fonctions **suivant** et **dessiner**.

- L'animation **démarre** lorsque le **big-bang** crée le monde !



8

MOUVEMENT RECTILIGNE SINUSOIDAL – Version 1

- On peut livrer l'animation dans une seule fonction, avec des variables et fonctions locales.

```
(define (balle-rect-sin)
  (local [(define LARG 100)           ; largeur de la scène
          (define HAUT 50)           ; hauteur de la scène
          (define X0 (/ LARG 2))     ; abscisse du point de départ
          (define Y0 (/ HAUT 2))     ; ordonnée du point de départ
          (define FOND (rectangle LARG HAUT 'solid "yellow"))
          (define BALLE (circle 10 'solid "red"))
          (define INIT 0)            ; le monde m initial
          (define (suivant m)        ; monde → monde
            (+ m 0.1))
          (define (dessiner m)       ; monde → scène
            (place-image BALLE (* X0 (+ 1 (sin m))) Y0 FOND))]
    (big-bang INIT
      (on-tick suivant)
      (on-draw dessiner) )))
```



9

MOUVEMENT RECTILIGNE SINUSOIDAL – Version 2

- **La Fin du Monde.** La simulation précédente tourne indéfiniment ! On peut la stopper en fermant sa fenêtre, mais aussi par programme en exprimant que le monde a atteint un **état final**.
- On écrit donc un prédicat (`final? m`) retournant `#t` si et seulement si le monde `m` est en état final. Par exemple, stoppons l'animation après trois aller-retours - donc si $m \geq 6\pi$:

```
(define (balle-rect-sin2)
  (local [(define LARG 100)
          .....
          (define (dessiner m)       ; monde → scène
            (place-image BALLE (* X0 (+ 1 (sin m))) Y0 SCENE))
          (define (final? m)         ; monde → boolean
            (>= m (* 6 pi)))]
    (big-bang INIT
      (on-tick suivant)
      (on-draw dessiner)
      (stop-when final?))))
```

10

MOUVEMENT RECTILIGNE SINUSOIDAL – Version 3

- **La Gestion du Clavier.** Il est parfois intéressant (*jeu vidéo*) d'**interagir avec l'utilisateur**, qui va piloter le jeu en pressant des touches du clavier pendant l'animation.







- Jusqu'à présent, nous avons installé dans une animation :
 - le **chef d'orchestre** : (`big-bang ...`)
 - un **contrôleur du temps** qui fait évoluer le Monde : (`on-tick ...`)
 - un **contrôleur d'affichage** qui construit une image : (`on-draw ...`)
 - un **contrôleur de la fin des temps** : (`stop-when ...`)
- Il reste à installer un **contrôleur de clavier** : (`on-key ...`)
- S'il n'est pas installé, l'animation n'écouterait pas le clavier !

11

- La fonction clavier passée au contrôleur (`on-key clavier`) va traiter chacun de ces événements clavier :

clavier : Monde × Key → Monde

où Key est une chaîne de caractères [string] qui peut être :

- une touche usuelle comme "a" "A" "!" " " 
- une touche directionnelle "up", "down", "left", "right".   

- Les touches se comparent avec (`key=? k1 k2`). On pourra par exemple tester la pression sur la touche *flèche en haut* par (`key=? k "up"`).

- Il faut donc programmer une fonction de gestion du clavier (`clavier m k`) qui va calculer le **nouveau monde obtenu à partir du monde m sur pression de la touche k**. Elle envisagera tous les cas intéressants pour la touche k.

- L'utilisateur intervient donc via le clavier pour **forcer l'animation à évoluer vers un autre futur !** Comme dans la série "Sliders" !

12

- Ajoutons un *reset*, qui remet le monde à son origine INIT sur pression de la touche r ou R (majuscule ou minuscule) :

```
(define (balle-rect-sin3)
  (local [(define LARG 100)
          .....
          (define (clavier m key) ; monde × key → monde
            (if (or (key=? key "r") (key=? key "R"))
                INIT
                m))
          (define (final? m) ; monde → boolean
            (>= m (* 6 pi)))]
    (big-bang INIT
              (on-tick suivant)
              (on-draw dessiner)
              (on-key clavier)
              (stop-when final?))))
```

13

- Réagissons à un *clac* sur un bouton de la souris, qui aura le même effet que le *reset* obtenu par pression de la touche r ou R du clavier :

```
(define (balle-rect-sin4)
  (local [(define LARG 100)
          .....
          (define (clavier m key) ; monde × Key → Monde
            (if (key=? key "r") INIT m))
          (define (souris m x y evt) ; je n'écoute que cet évènement
            (if (mouse=? evt "button-down")
                INIT
                m)) ]
    (big-bang INIT
              (on-tick suivant)
              (on-draw dessiner)
              (on-key clavier)
              (on-mouse souris)
              (stop-when final?))))
```

16

MOUVEMENT RECTILIGNE SINUSOIDAL – Version 4

- **Gestion de la Souris.** La communication d'un programme avec l'utilisateur se fait principalement à travers le clavier et la souris. Quid de la souris ?
- La pression d'une touche du clavier génère un *évènement-clavier*.
- Quels sont les *évènements-souris* ? Ouvrons la doc :

MouseEvent

(one-of/c "button-down" "button-up" "drag" "move")

- La fonction f passée au contrôleur (on-mouse f) va traiter les évènements-souris !

$f : \text{Monde} \times \text{integer} \times \text{integer} \times \text{mouse-event} \rightarrow \text{Monde}$

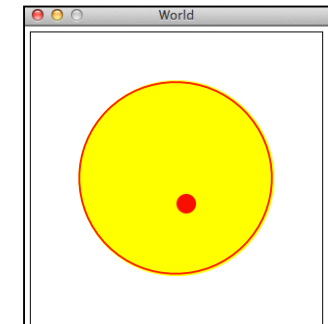


14

Animations à plusieurs paramètres

- Jusqu'à présent, notre animation était à 1 paramètre réel. Or la plupart des animations dépendent de plusieurs paramètres. Par exemple de la position (x,y) d'une balle et du nombre de tours d'horloge, etc.
- Lorsque l'état du phénomène à animer dépend de N variables indépendantes, on les regroupera dans une *structure* et on parlera d'une *animation à N paramètres*.

- Exemple : une balle qui se déplace de manière aléatoire en essayant de s'échapper d'une disque entouré d'une clôture électrifiée ! L'état du système est déterminé par la seule position (x y) de la balle à un instant donné. C'est donc une *animation à 2 paramètres*.



18

```

(define (electric-ball SIZE R) ; une animation à 2 paramètres !
  (local [(define S (/ SIZE 2))
          (define FOND (underlay (empty-scene SIZE SIZE)
                                (circle R 'solid "yellow")
                                (circle R 'outline "red")))
          (define BALLE (circle 10 'solid "red"))
          ; Le monde m est le point <x;y> à la position de la balle
          (define INIT (make-posn S S)) ; le centre du disque
          (define (distance x1 y1 x2 y2)
            (sqrt (+ (sqr (- x1 x2)) (sqr (- y1 y2))))
            sqrt ?
          (define (suivant m) ; monde --> monde
            (local [(define x (posn-x m))
                    (define y (posn-y m))]
              ← déstructuration
              (make-posn (+ x (- (random 9) 4)) ; petit déplacement...
                          (+ y (- (random 9) 4)))) ; ...aléatoire
            (define (dessiner m) ; monde --> scène
              (place-image BALLE (posn-x m) (posn-y m) FOND))
            (define (final? m) ; monde --> boolean
              (>= (distance (posn-x m) (posn-y m) S S) R))
            (big-bang INIT
                      (on-tick suivant)
                      (on-draw dessiner)
                      (stop-when final?)
                      (name "Electric Ball"))))

```

17

Déstructuration rapide : match-define

- Structure de bille de rayon r, de position x,y et de vitesse dx,dy :
(define-struct bille (r x y dx dy))
- Il est pénible d'extraire tous les champs d'une bille b :

```

(define (bouger b)
  (local [(define r (bille-r b))
          (define x (bille-x b))
          (define y (bille-y b))
          (define dx (bille-dx b))
          (define dy (bille-dy b))]
    (if (> r 10)
        (make-bille (* r 2) (+ x dx) ...)
        (make-bille r (+ x dx) ...)))

```

déstructuration
de b

- La forme (match-define motif expr) permet de définir d'un seul coup tous les champs de la valeur de l'expression expr. Pour une structure :

```

(local [(match-define (bille r x y dx dy) b)
        (if (> r 10) ...)]

```

18

- Un dernier exemple célèbre : **dessin à main levée avec la souris !**
A tout moment nous devons connaître les coordonnées précédentes px, py de la souris, et l'image img déjà tracée (en réalité un polygone).

```

(define (dessiner-a-la-souris)
  (local [(define SIZE 200)
          ; Le Monde est le triplet (img,px,py)
          (define-struct monde (img px py))
          (define INIT
            (make-monde (rectangle SIZE SIZE 'solid "white") 0 0))
          (define (dessiner m)
            (monde-img m))
          (define (souris m x y evt)
            (if (mouse=? evt "drag")
                (make-monde (add-line (monde-img m)
                                      (monde-px m) (monde-py m) x y
                                      "black")
                            x y)
                (make-monde (monde-img m) x y)))
          (big-bang INIT
                    (on-draw dessiner) ; aucune horloge !
                    (on-mouse souris)))

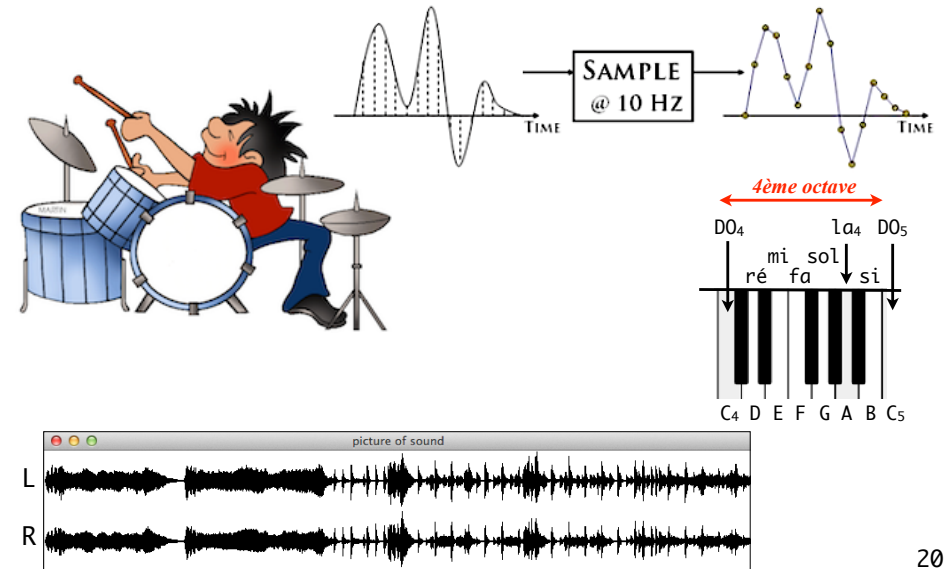
```



19

Sonorisation d'une animation

- Ceci sera traité ultérieurement si le temps le permet...



20