# Projet

ENS Lyon - Janvier 2001 - mai 2001

Laure Danthony et Christophe Delage

## Visualisation de graphes et d'automates cellulaires

PLAN :

- Affichage esthétique de graphes
- Visualisation d'automates cellulaires au travail
- Quelques automates cellulaires particuliers
- Interfaces pour l'utilisateur

# Affichage esthétique de graphes

## Calcul des positions

- **Idée géniale** : utiliser la mécanique élémentaire (du point) et la résolution numérique des équations différentielles.
- Calcul plus performant en C avec une librairie personnelle `complex.c`.
- Paramètres de la fonction `calcule_positions` : le graphe (de type *value*) et les coordonnées de la fenêtre graphique CAML.

## Le type graph et son codage

- C'est le suivant (CAML) :

```
type graph = { tab : int array array;
               mutable pos : (int * int) array
             }
```

- Le nombre de sommets et le degré maximum sont calculables $\Rightarrow$ pas besoin de charger le type.
- La fonction `file_read` permet de lire des graphes sous forme de syntaxe“concrète” :

```
4 2
# 1
0 2          ou          0-1 1-2 2-3 3-4 4-5 5-6 6-0
1 3
2 #
```

## L'affichage graphique

- Pas de commentaire sur le code.
- Possibilité de sortir du postscript ![a]
- Des résultats magnifiques !

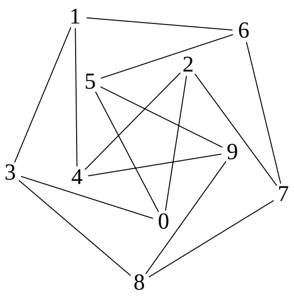


Fig. 1 – Le graphe de Petersen

[a] Idéal pour LaTeX !

# Visualisation d'automates cellulaires au travail

## Qu'est-ce-qu'un automate cellulaire ?

- La donnée d'un graphe (degré maximum $d$ et $n$ nœuds).
- La donnée d'une fonction de transition valable pour les automates situés aux nœuds du graphe.
- Une situation initiale ...

*Objectif : Visualiser les transitions successives des automates du graphe !*

## Codage d'automates

On décide de coder des fonctions de signature :

- (fonction de transition)

```
type tfun = int -> int array -> int
```

- (fonction de coloration)

```
type colfun = int -> color
```

qui sont passées en argument à la fonction `draw_step` et qui déterminent l'évolution.

## Un premier exemple (sans intérêt)

- La fonction de transition : la somme des valeurs du sommet considéré et de ses voisins.
- Le résultat :

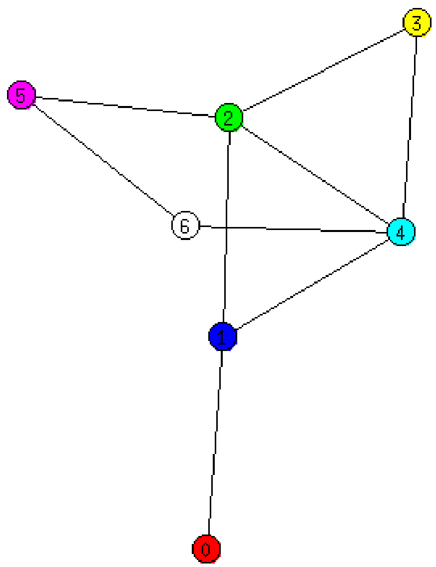


Fig. 2 – Un instantané de l'évolution ...

# Quelques automates cellulaires

## Compter le nombre de nœuds du graphe

- **Objectif** : étant donné un graphe linéaire, compter son nombre de nœuds.
- **Idée** : du point de vue du nœud situé au bord : faire partir un signal qui renverra dans l'autre sens un autre signal à chaque nœud rencontré ; lorsque qu'un exemplaire de ce deuxième signal arrive, on incrémente un compteur binaire (facile).

## Détecter les nombres premiers

- **Objectif** : dire si le nombre de nœuds du graphe est premier.
- **Idée** : utiliser le crible d'Érathostène : on part du $k^{ieme}$ nœud, un signal “raquette” se déplace à la vitesse 1/3 par rapport à une “table” immobile, et un signal “balle” fait un ping-pong entre “raquette” et “table” (*cf :* figure 3) la balle frappe la raquette pour tous les nœuds multiples de $k$.

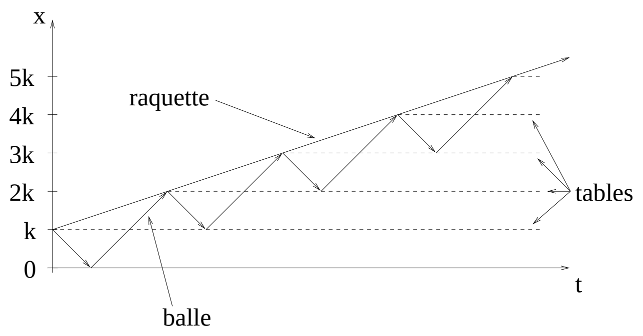


FIG. 3 – Tennis de table

**Des photos ?**

Voir la démo sur machine . . .

# Interfaces pour l'utilisateur

## Idée 1 : utilisation de java

- Une interface utile et très intuitive pour l'utilisateur.
- Une interface facile à programmer.
- **MAIS** lenteur du chargement et difficulté de codage des fonctions de transition.

## Idée 2 : toplevel CAML

- La solution adoptée, même si esthétiquement on peut faire mieux.
- Une invite du type :

```
          Objective Caml version 3

type "help ();;" for help.
#
```

- Une bibliothèque d'exemples déjà chargés . . .
- Inconvénient ( ?) : apprendre CAML.

# Et la suite ?

## Perspectives

- impression des graphes dans du poscript (déjà fait !)
- décompte des nœuds dans un graphe quelconque (utiliser un parcours).
- ...