Protocoles de populations & Grandes populations

Olivier Bournez

École Polytechnique
Laboratoire d'Informatique de l'X
Palaiseau, France

Journées SHAMAN
27 Janvier 2009

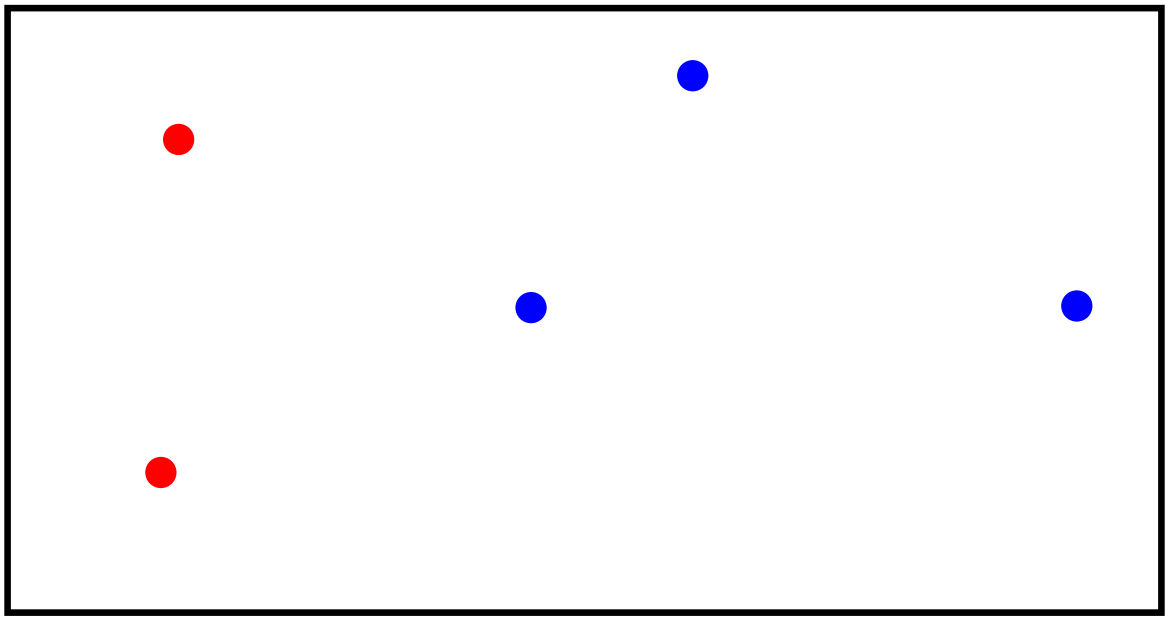
Plan

Protocoles de population[1]

- Un modèle de réseaux de capteurs, avec des hypothèses minimales sur
  - la puissance des entités mobiles :
    - des systèmes à états finis
  - l'infrastructure du réseau :
    - aucune topologie
    - des agents complètement anonymes
  - synchronisme :
    - totalement asynchrone
  - communications :
    - des communications sporadiques entre paires d'agents.

[1] [Angluin,Aspnes,Diamadi,Fischer,Peralta 2004]

Exemple 1 : partant de 3 ●, 2 ●



Program :

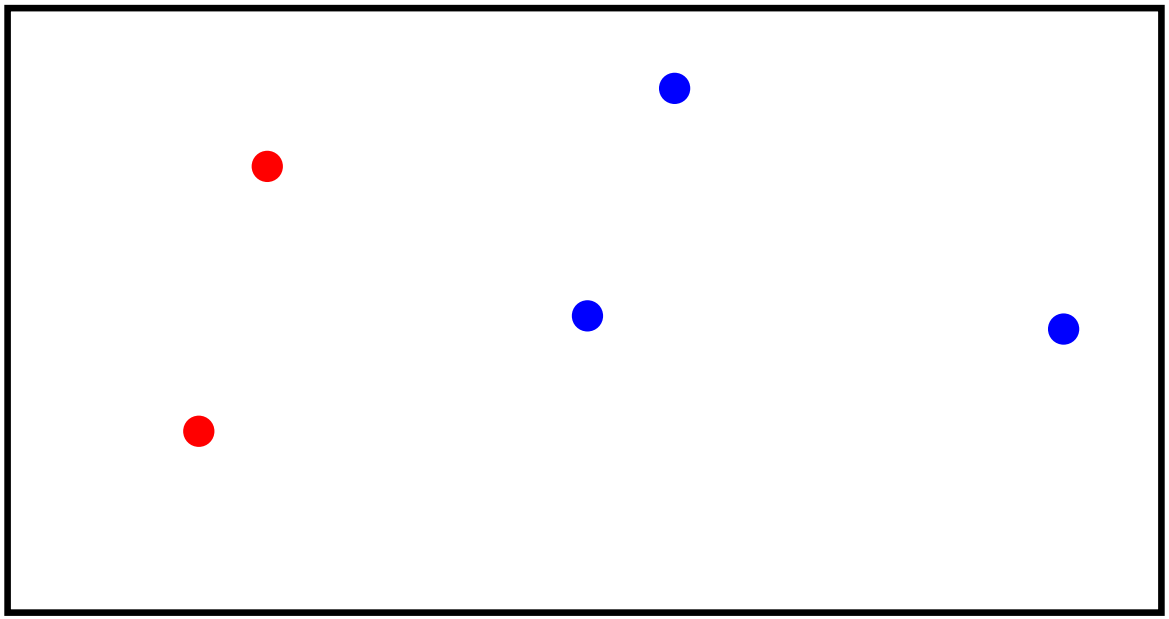
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

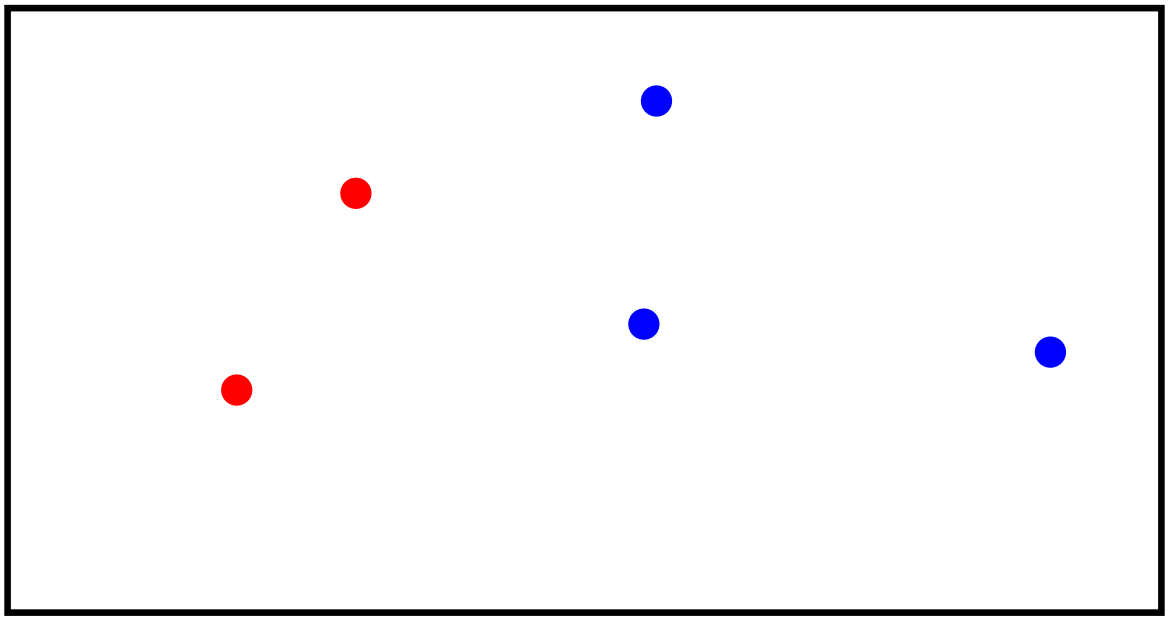
● ● → ● ●

● ● → ● ●

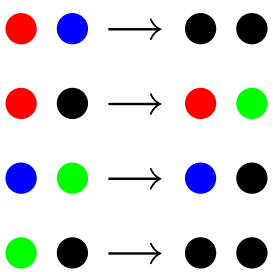
● ● → ● ●

● ● → ● ●

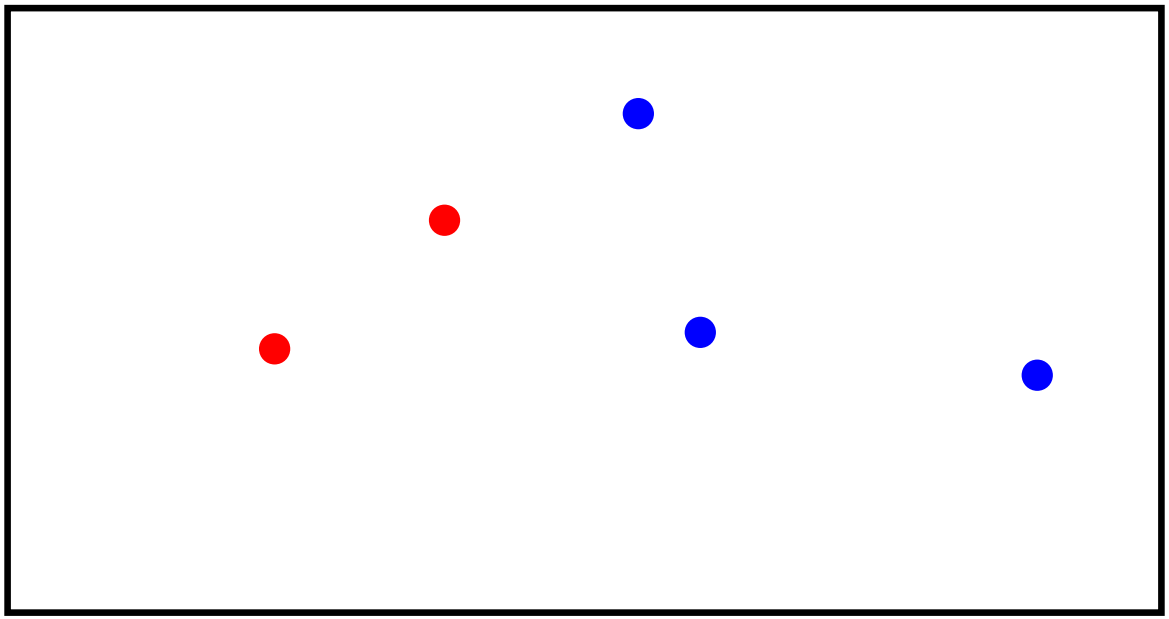
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

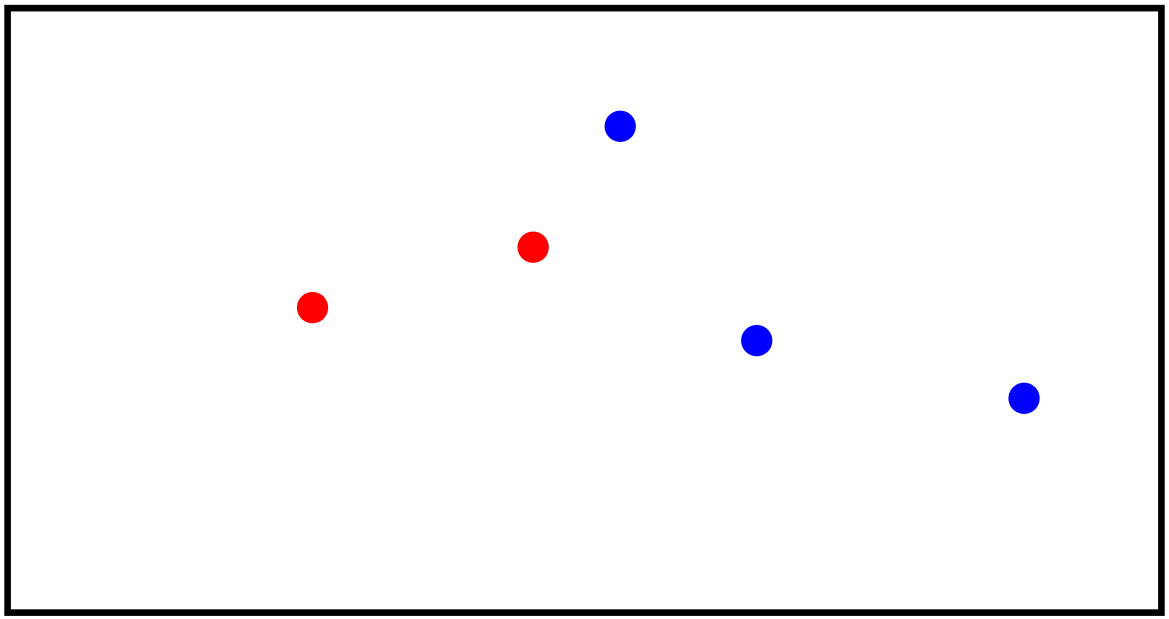
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

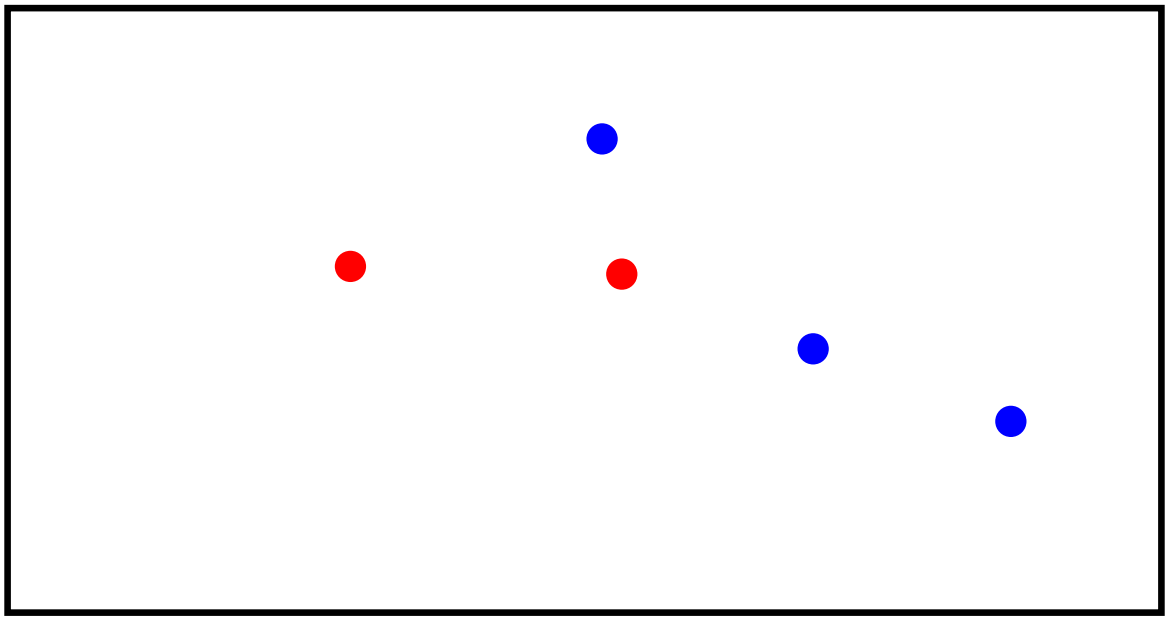
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

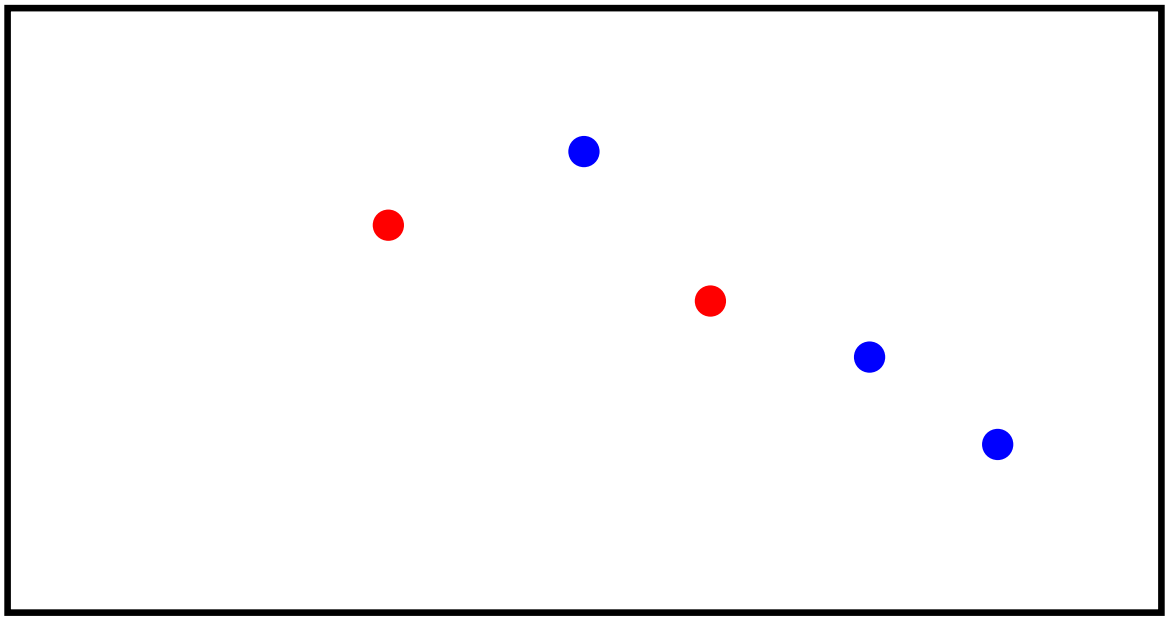
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

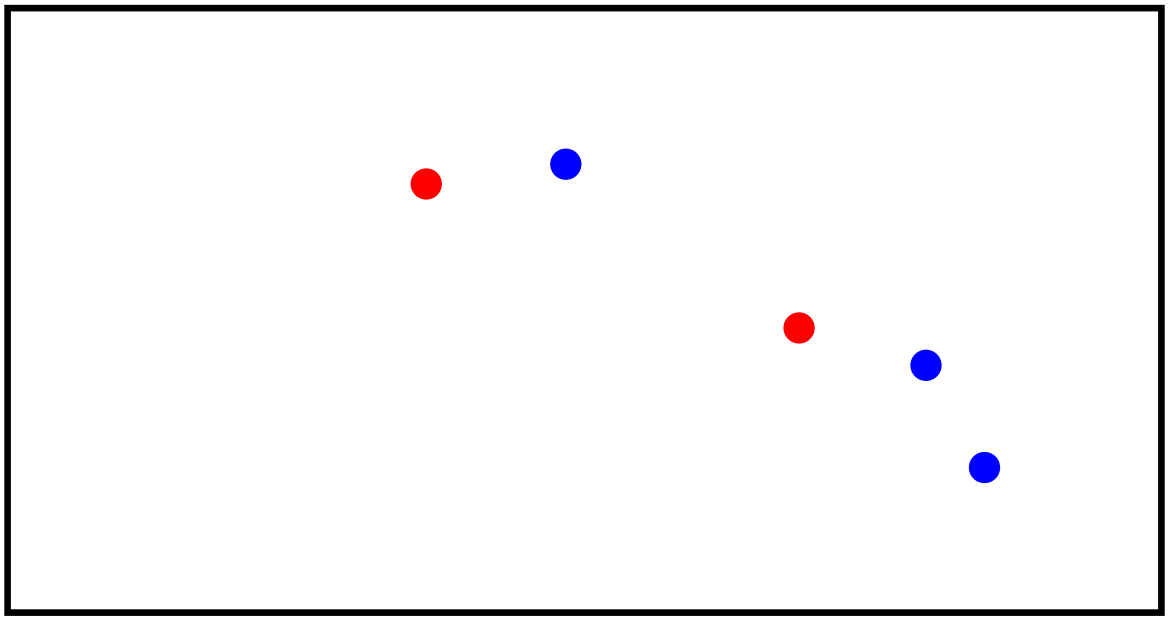
● ● → ● ●

● ● → ● ●

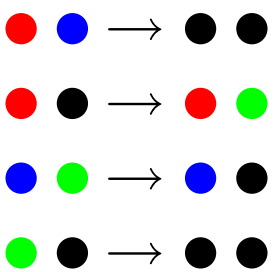
● ● → ● ●

● ● → ● ●

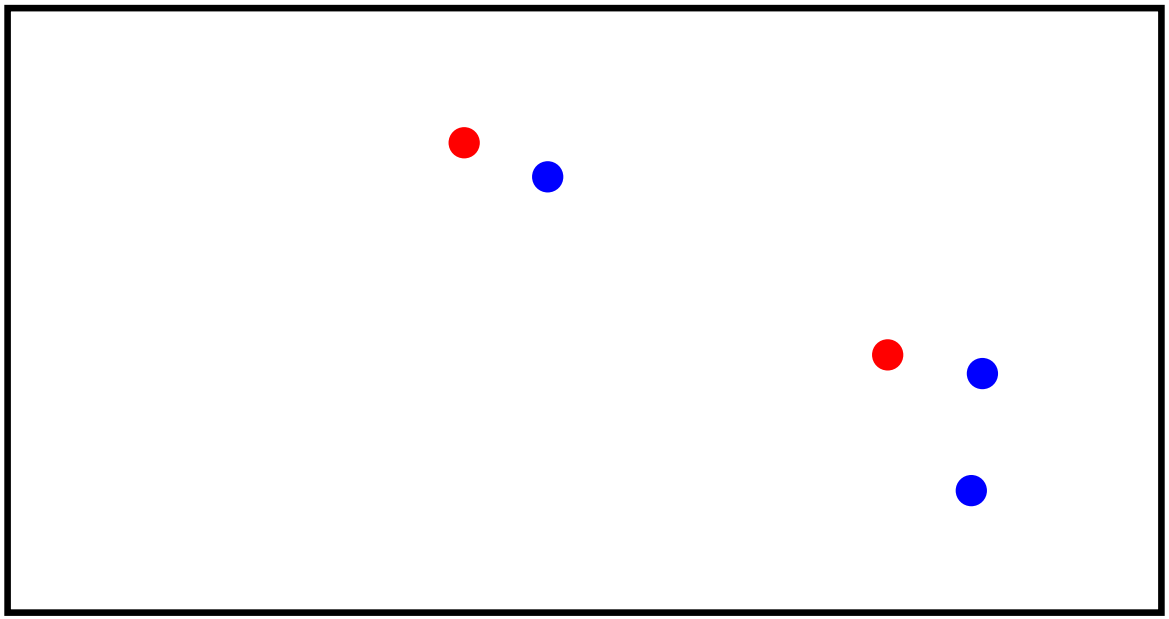
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

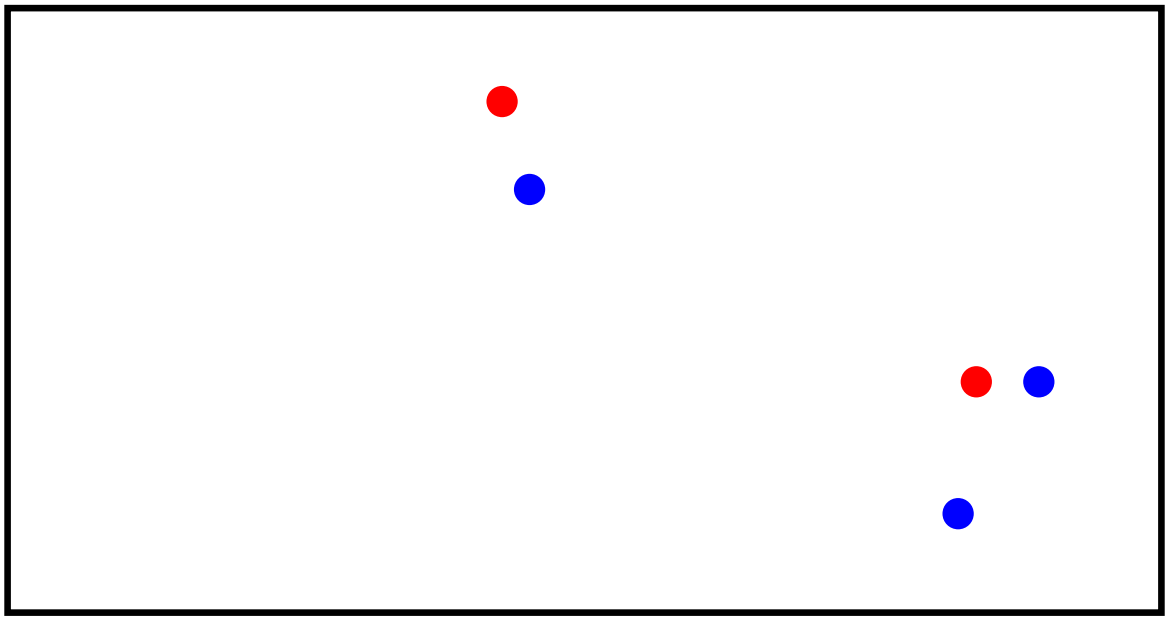
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

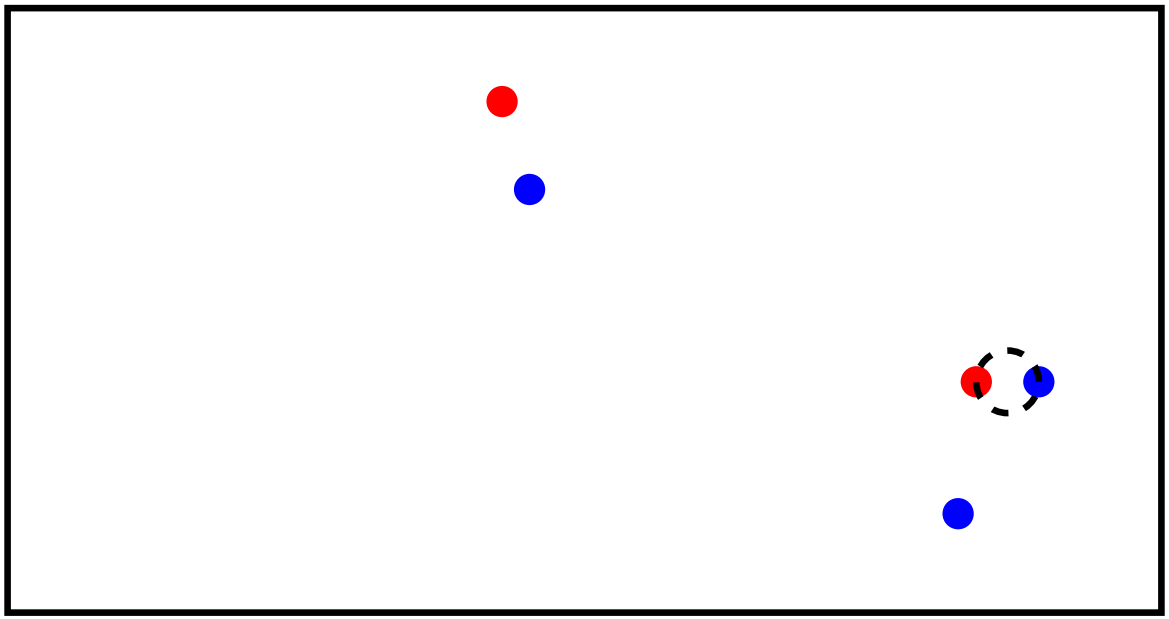
● ● → ● ●

● ● → ● ●

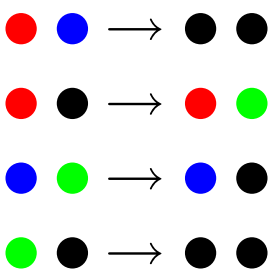
● ● → ● ●

● ● → ● ●

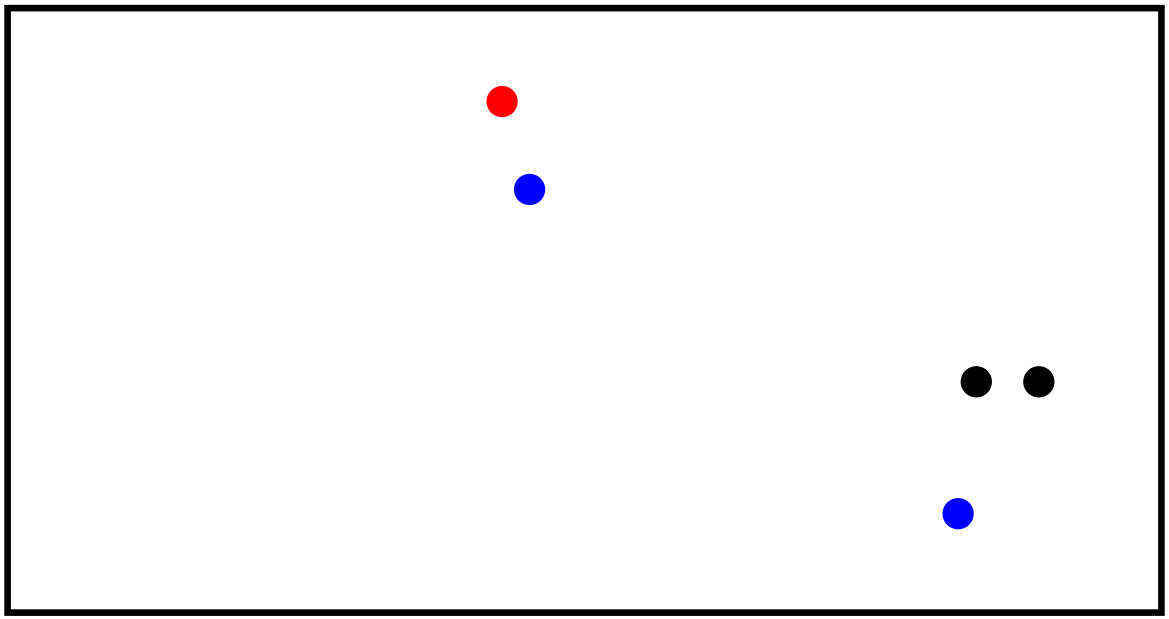
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

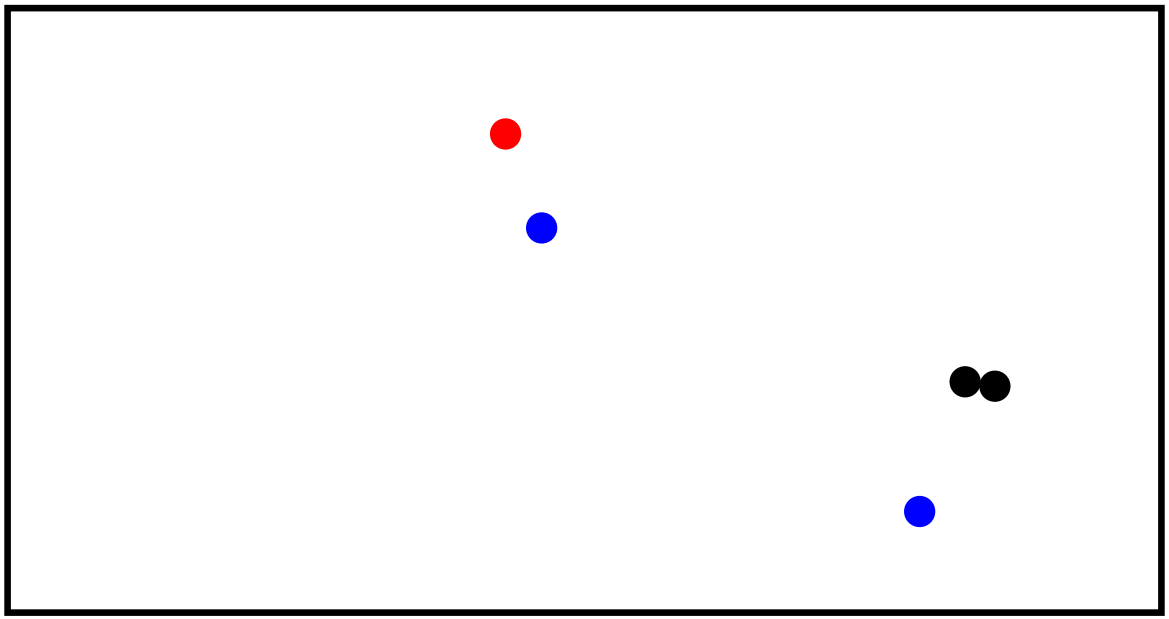
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

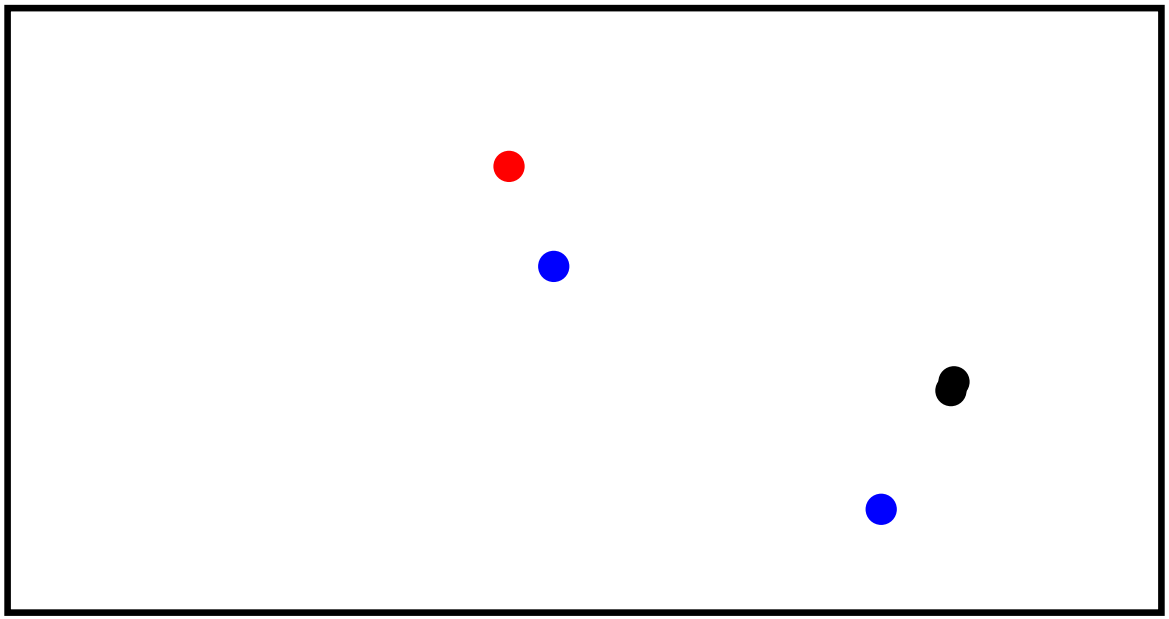
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

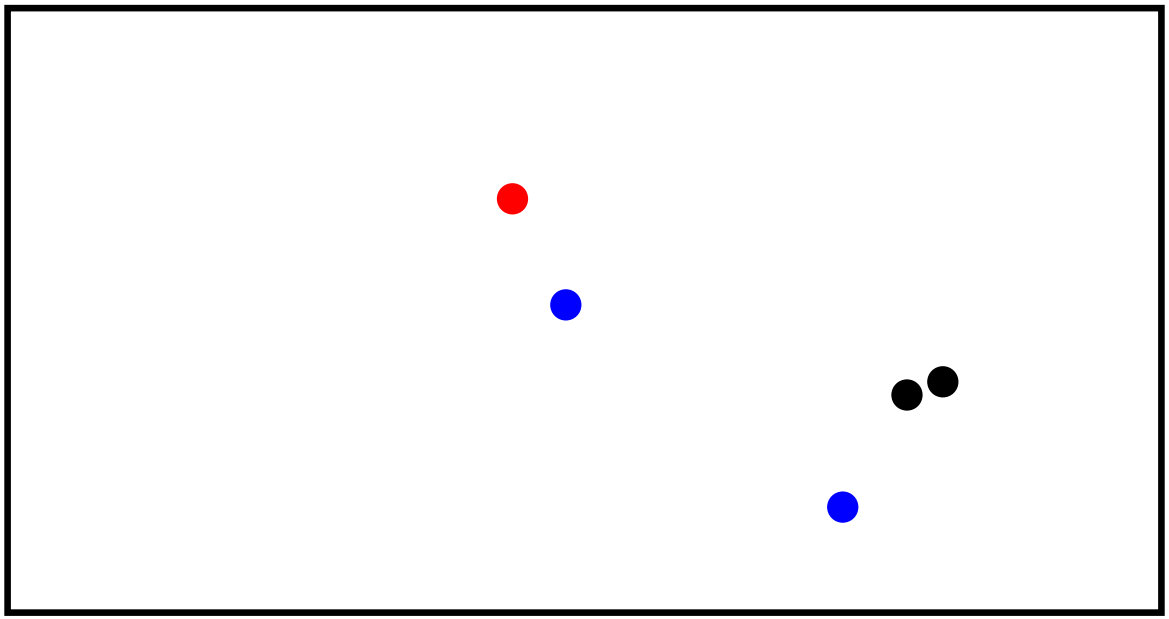
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

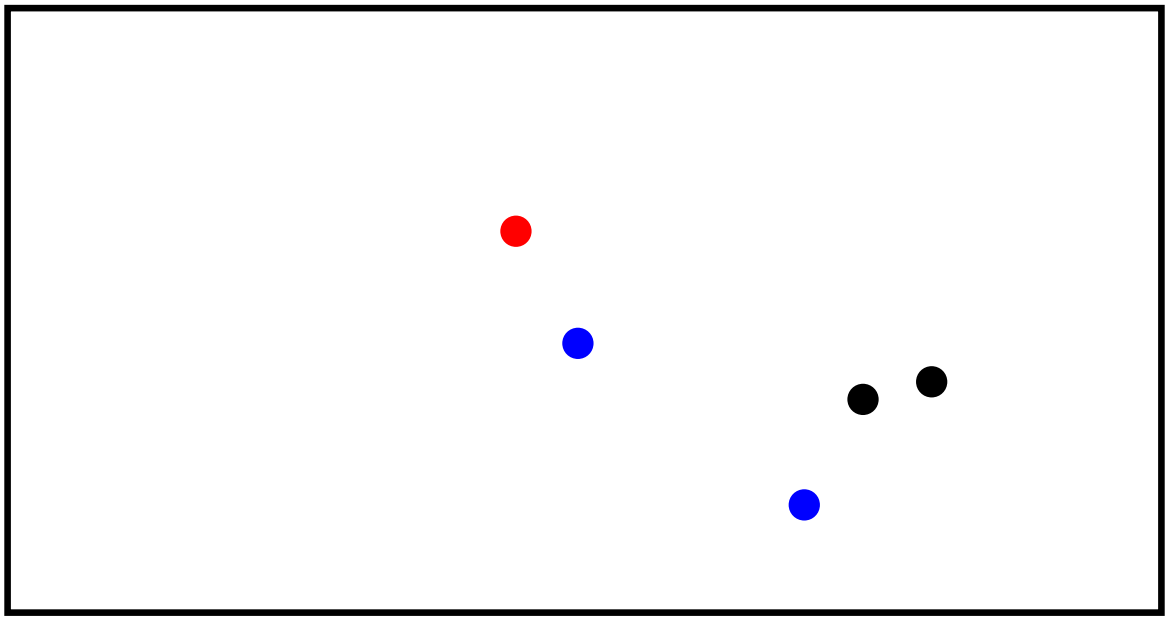
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

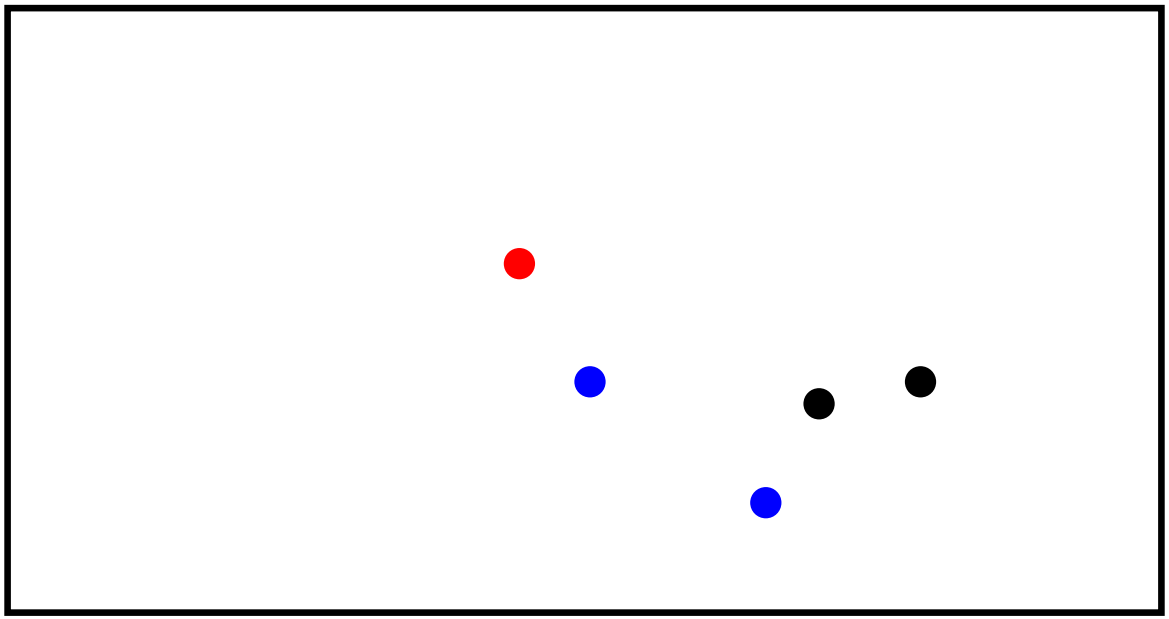
● ● → ● ●

● ● → ● ●

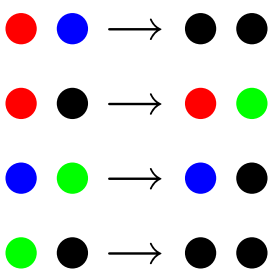
● ● → ● ●

● ● → ● ●

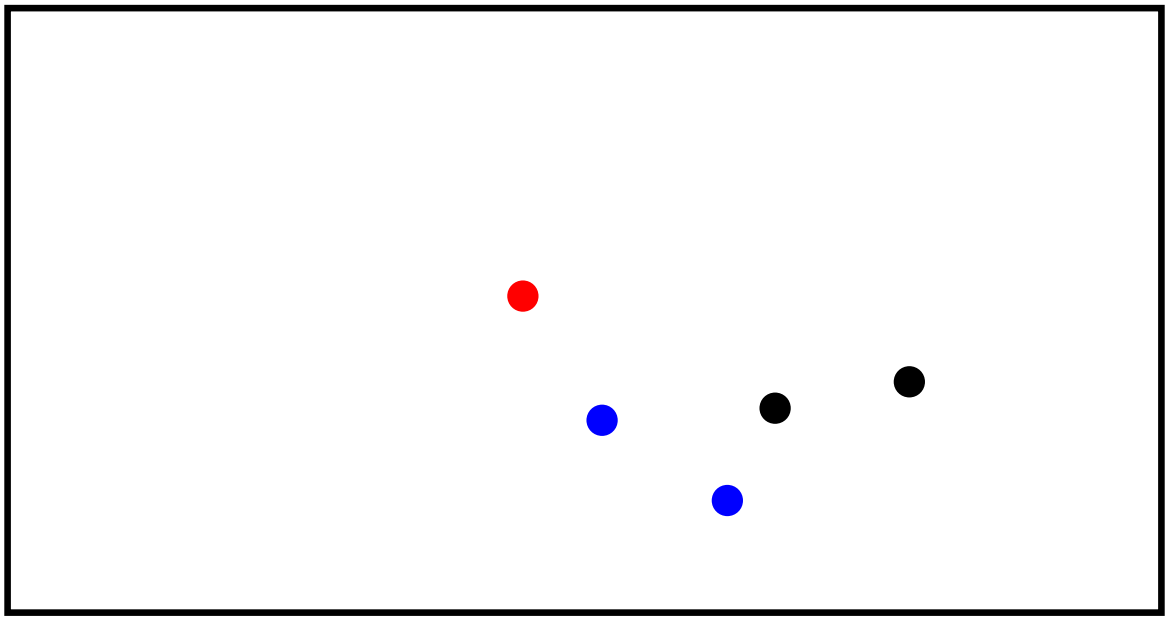
Exemple 1 : partant de 3 ●, 2 ●



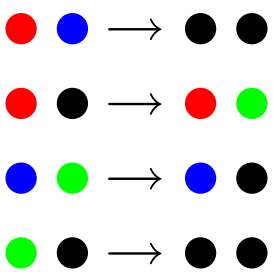
Program :



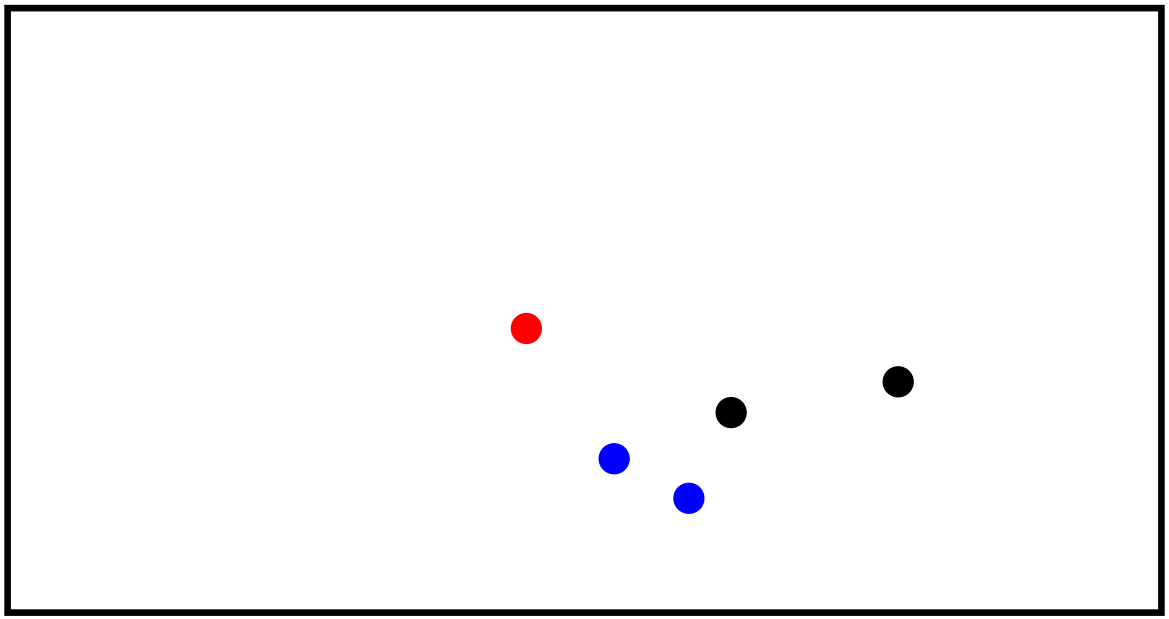
Exemple 1 : partant de 3 ●, 2 ●



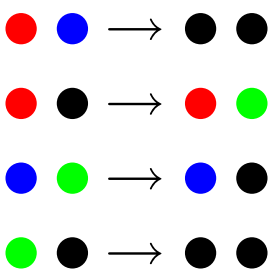
Program :



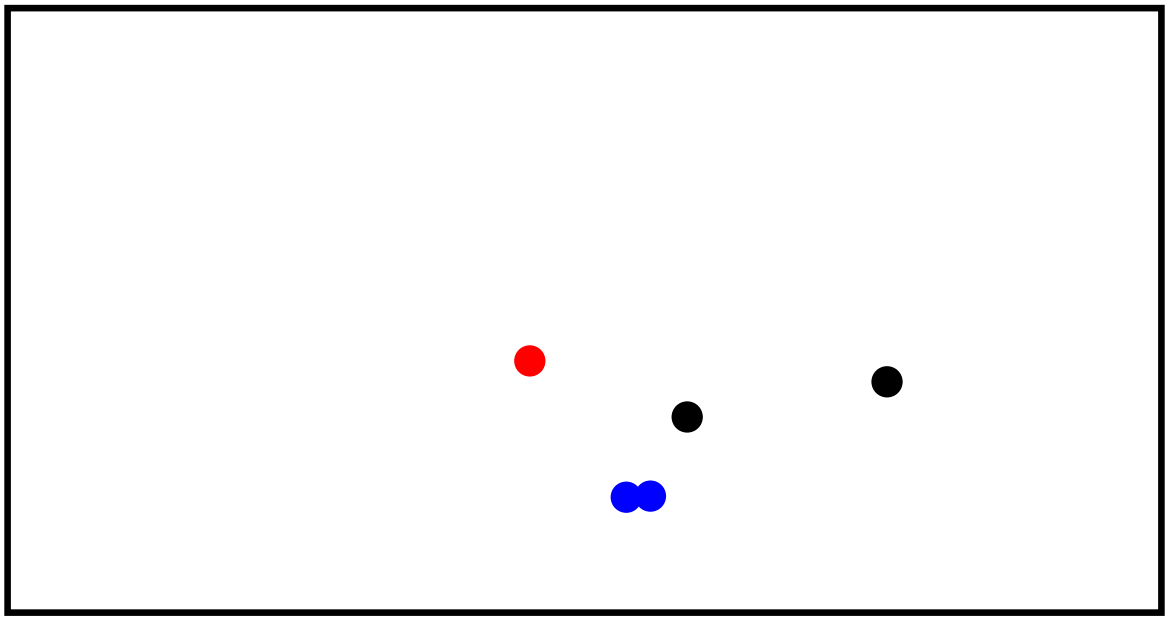
Exemple 1 : partant de 3 ●, 2 ●



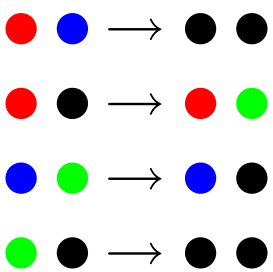
Program :



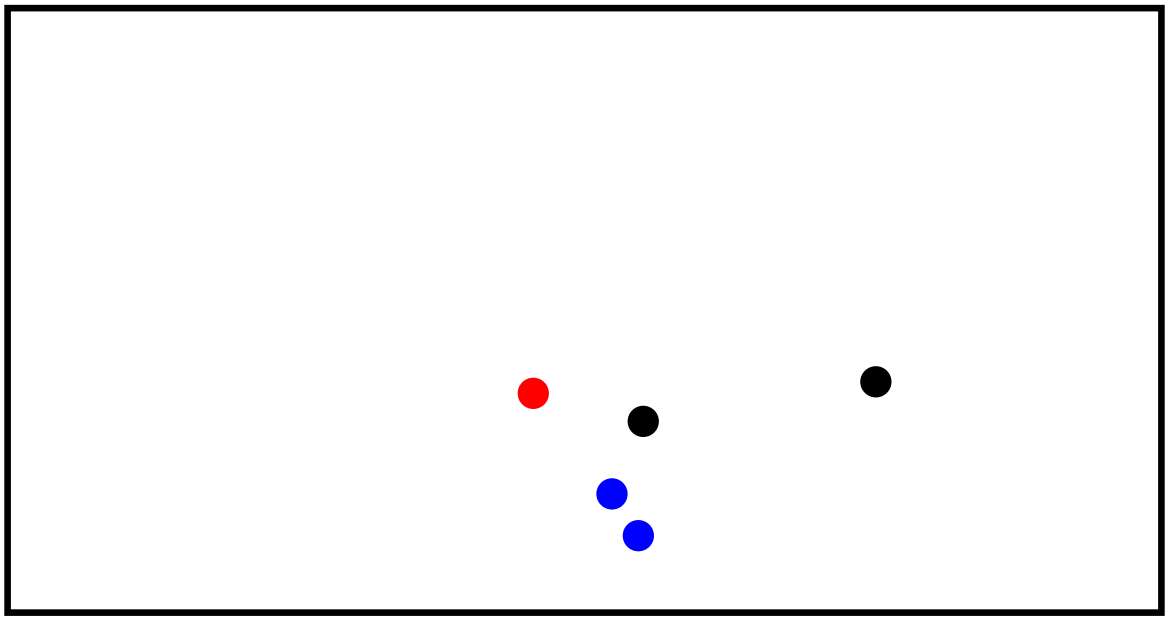
Exemple 1 : partant de 3 ●, 2 ●



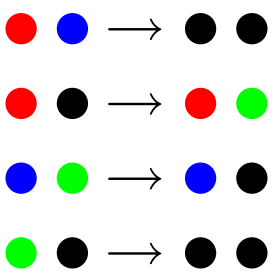
Program :



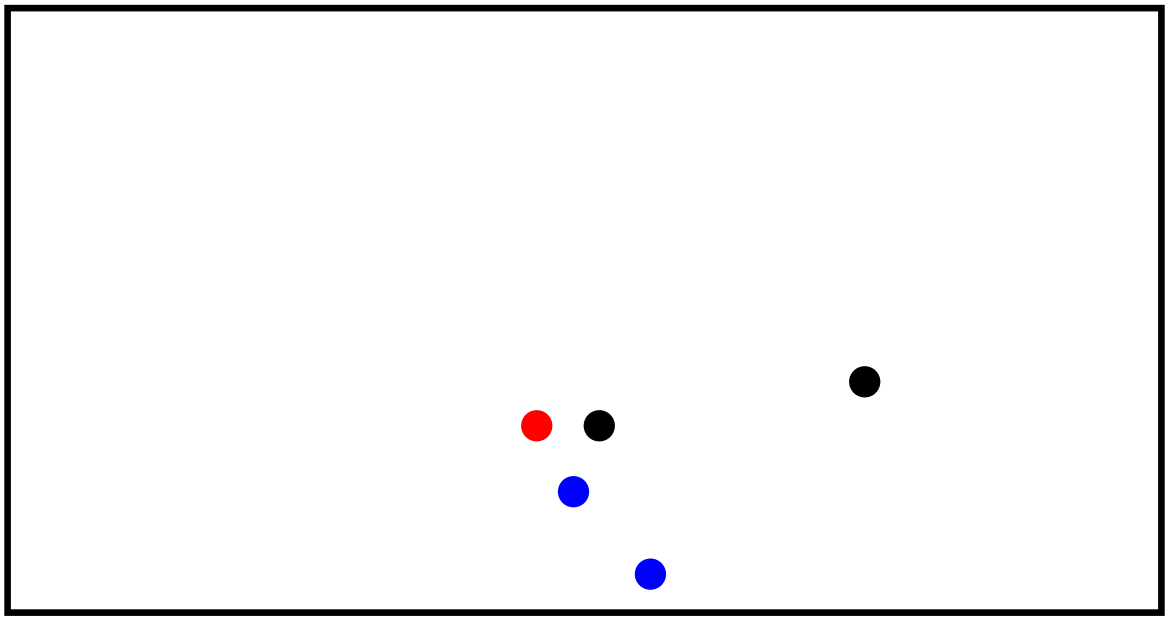
Exemple 1 : partant de 3 ●, 2 ●



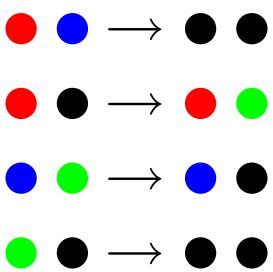
Program :



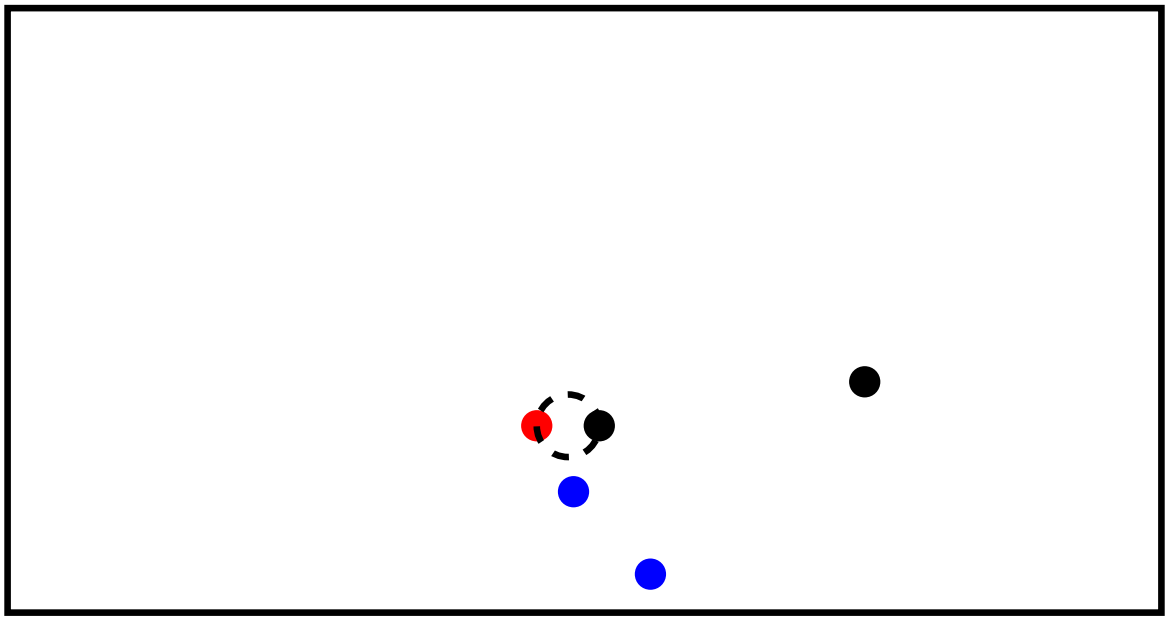
Exemple 1 : partant de 3 ●, 2 ●



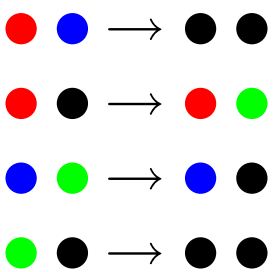
Program :



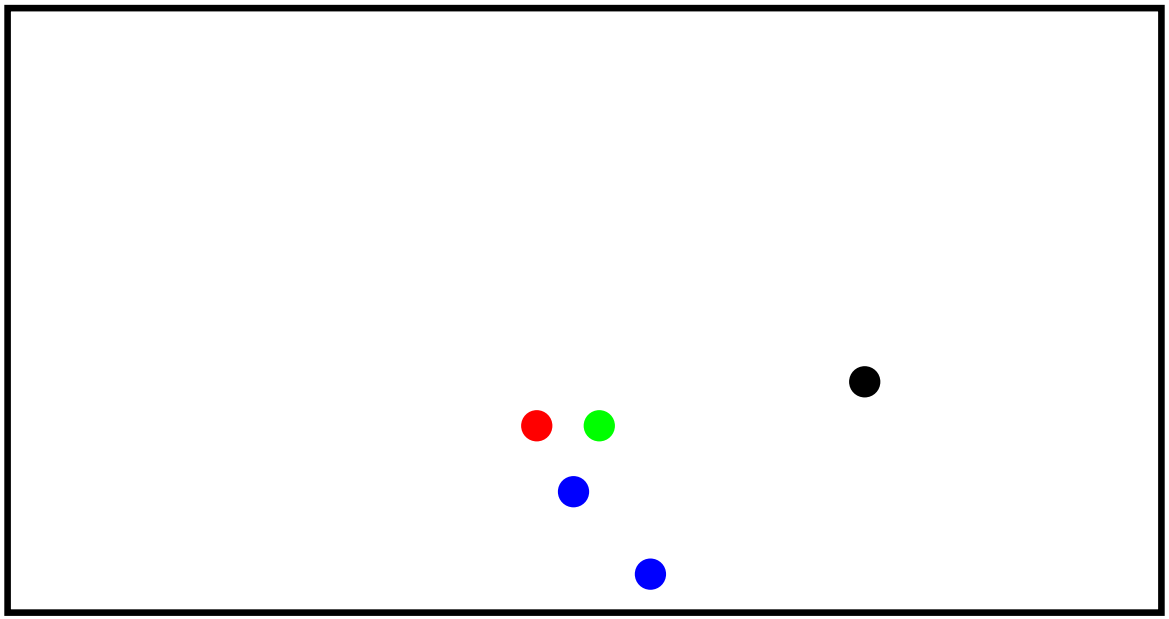
Exemple 1 : partant de 3 ●, 2 ●



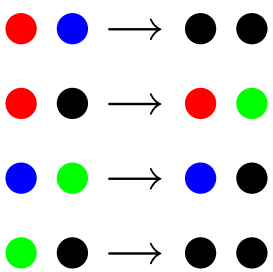
Program :



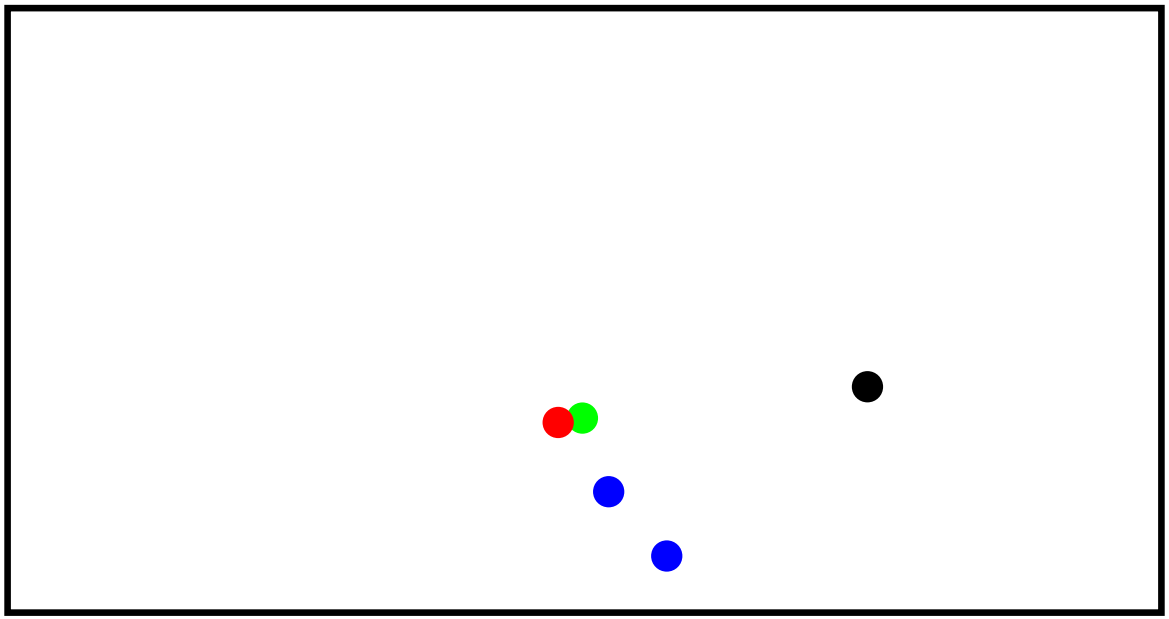
Exemple 1 : partant de 3 ●, 2 ●



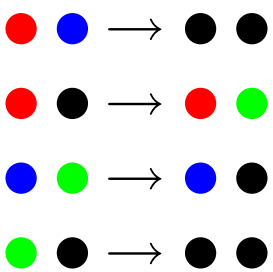
Program :



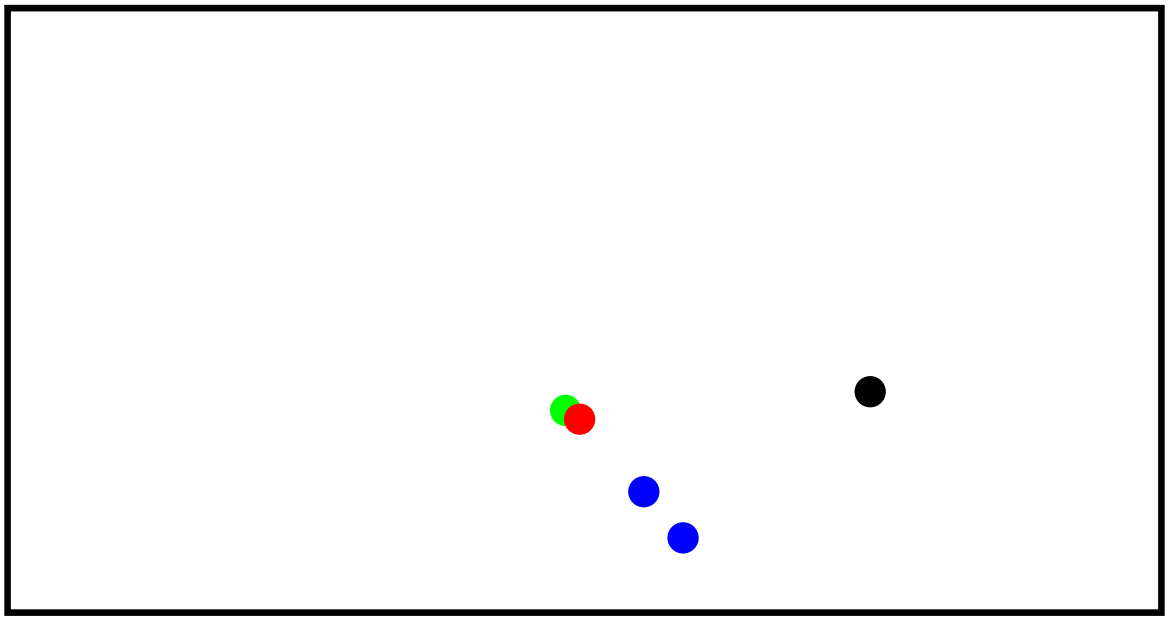
Exemple 1 : partant de 3 ●, 2 ●



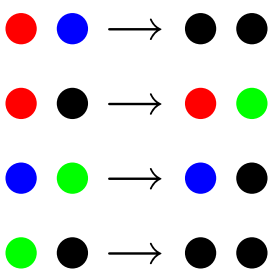
Program :



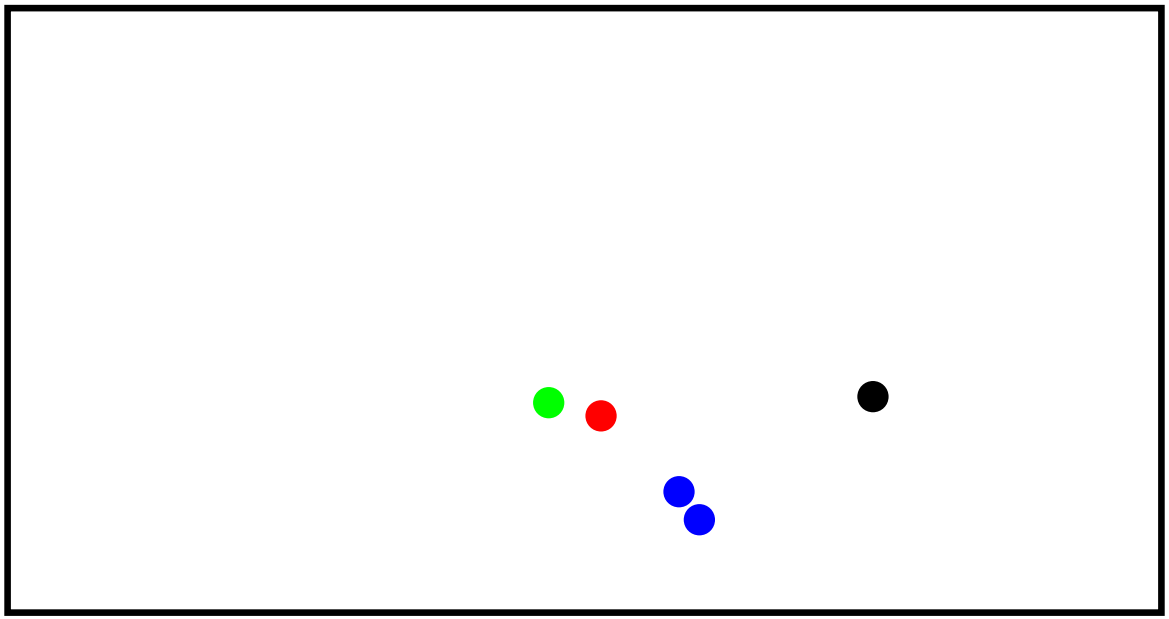
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

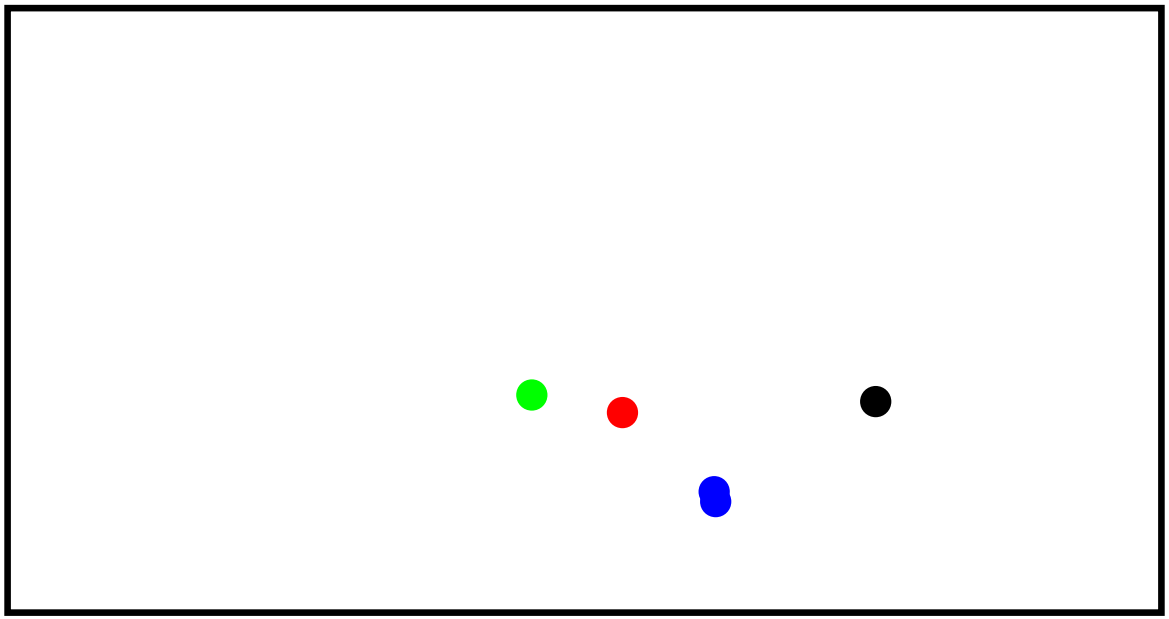
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

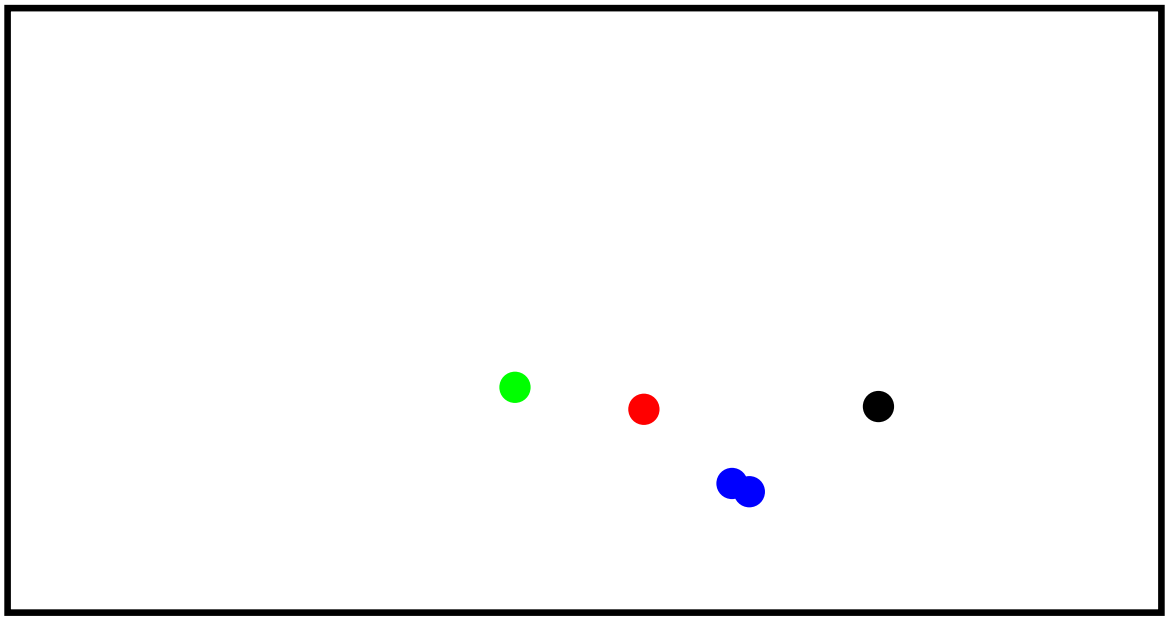
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

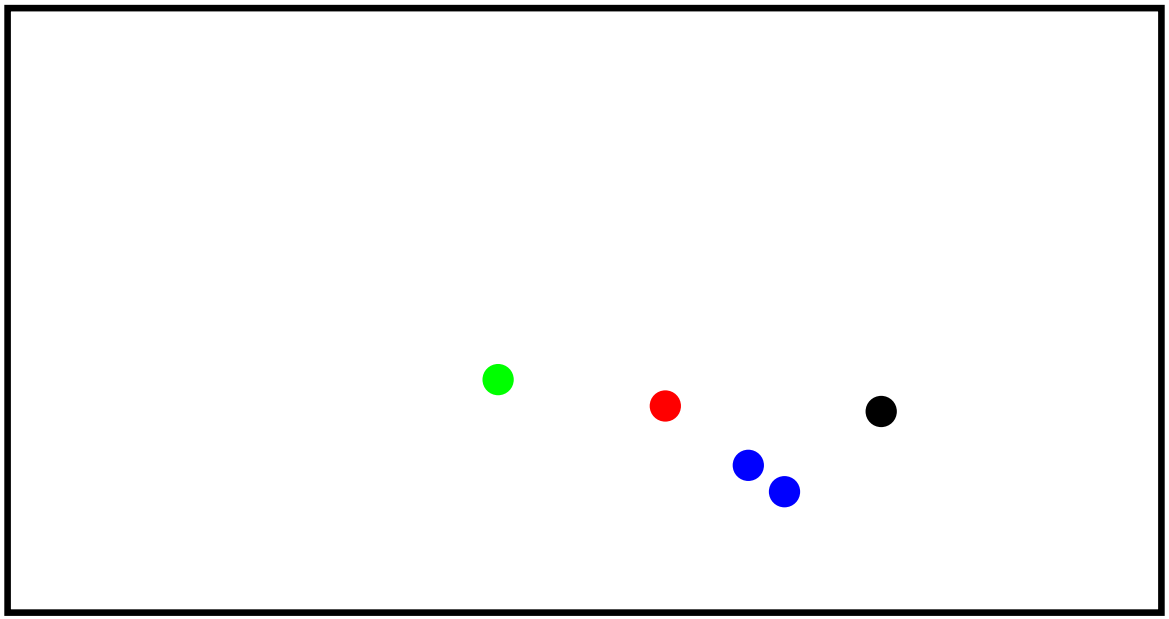
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

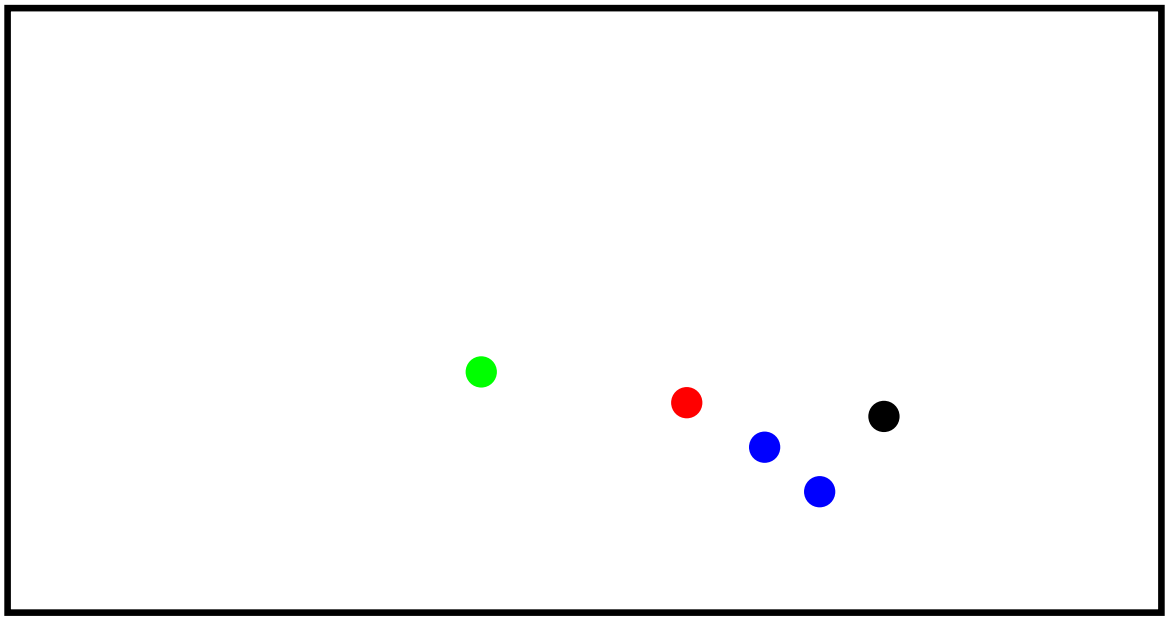
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

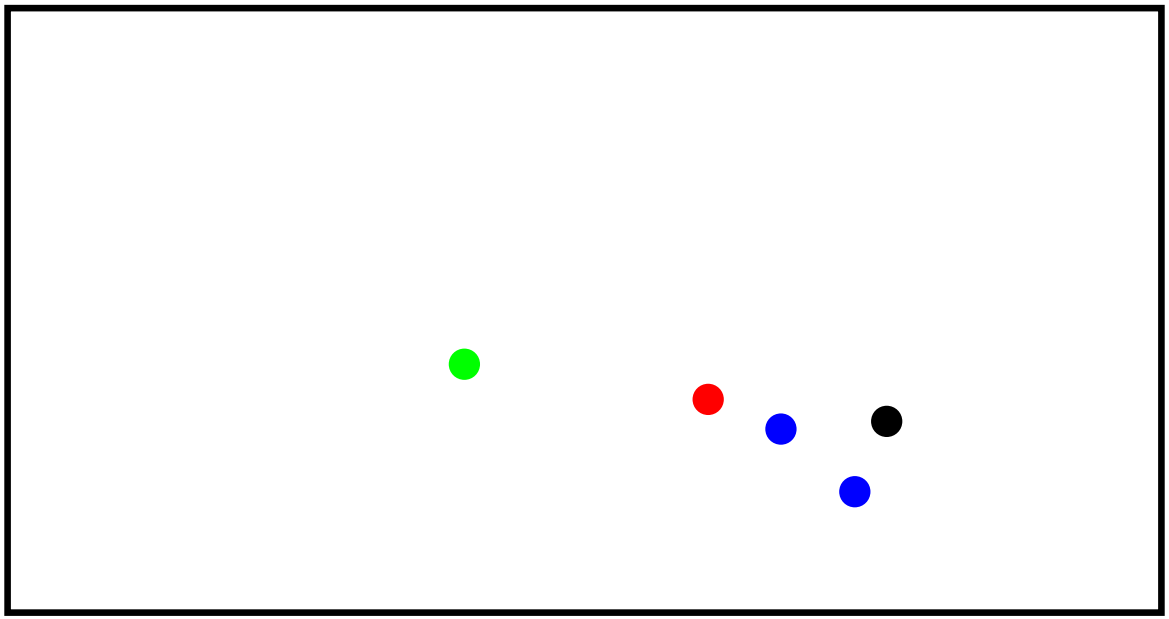
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

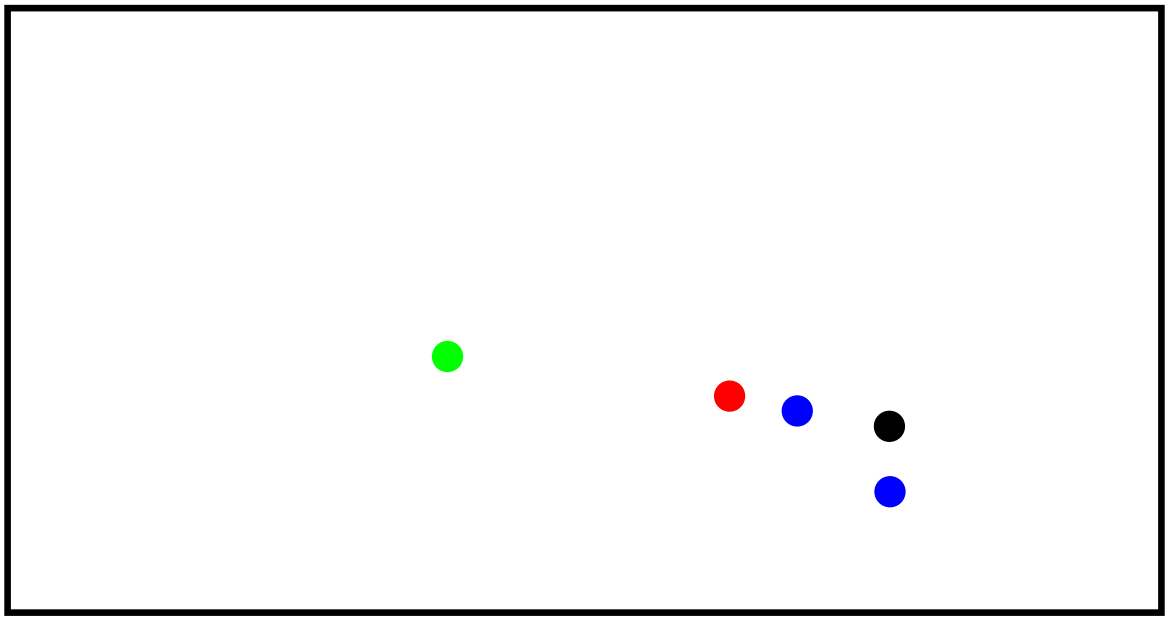
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

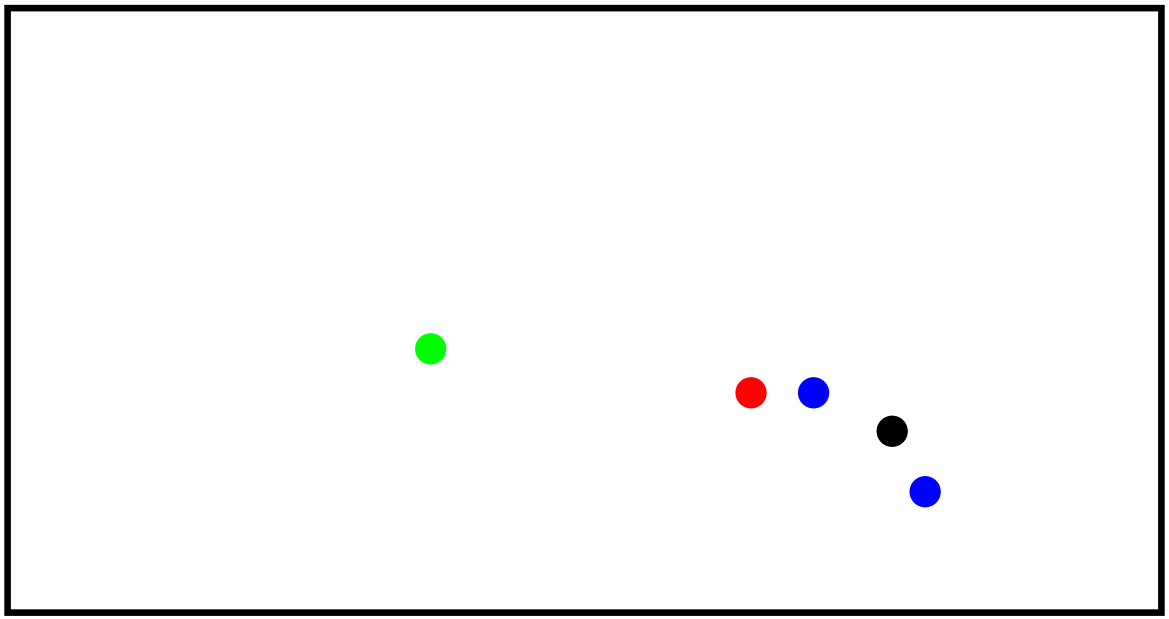
● ● → ● ●

● ● → ● ●

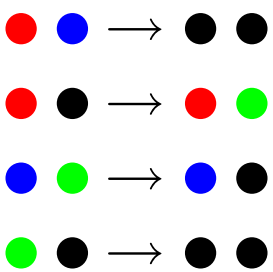
● ● → ● ●

● ● → ● ●

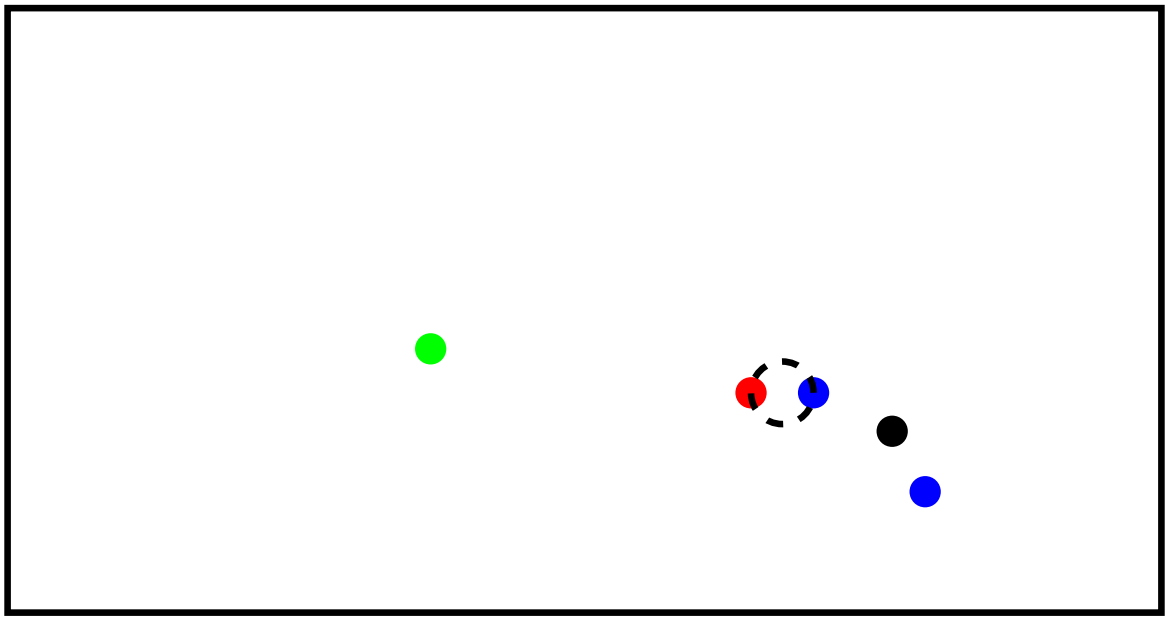
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

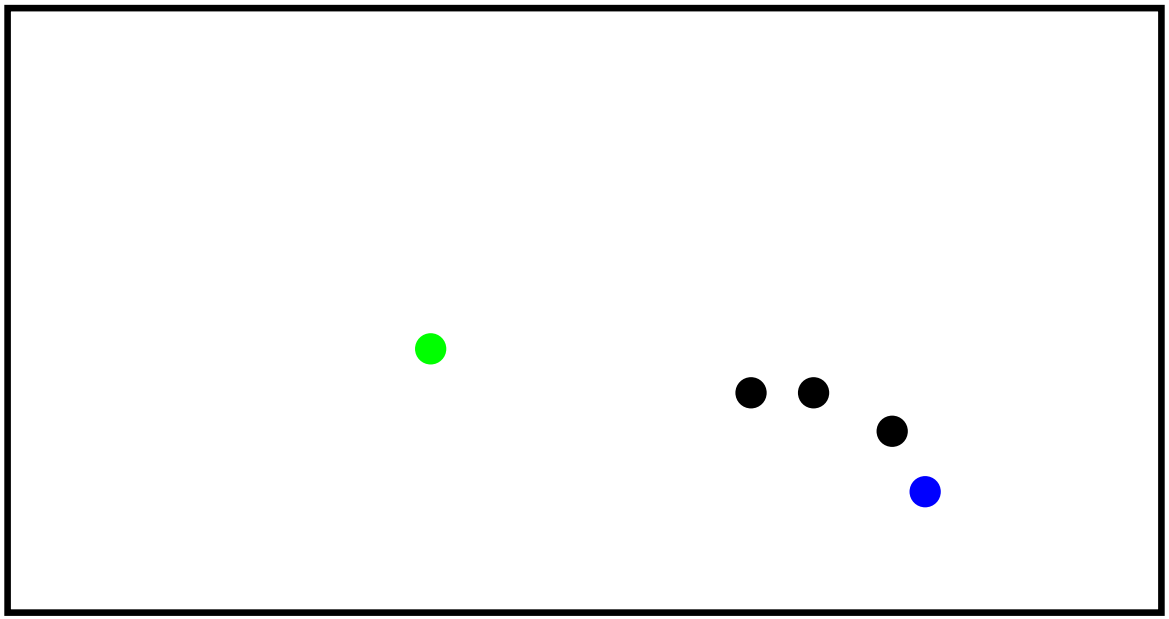
● ● → ● ●

● ● → ● ●

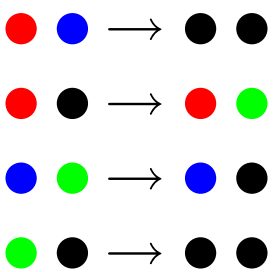
● ● → ● ●

● ● → ● ●

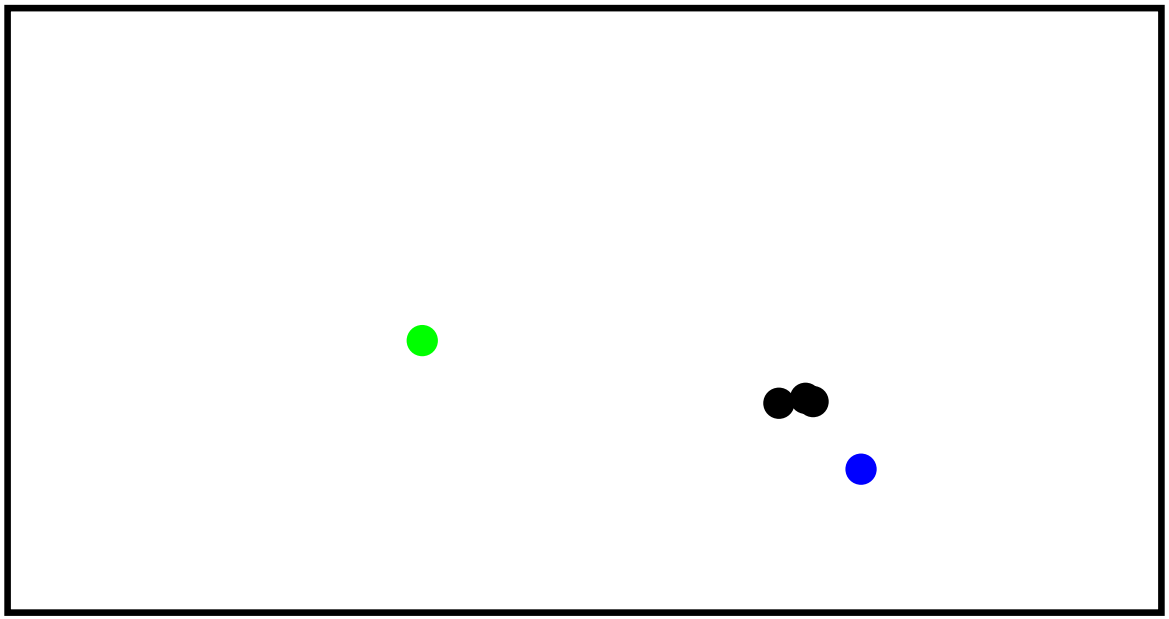
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

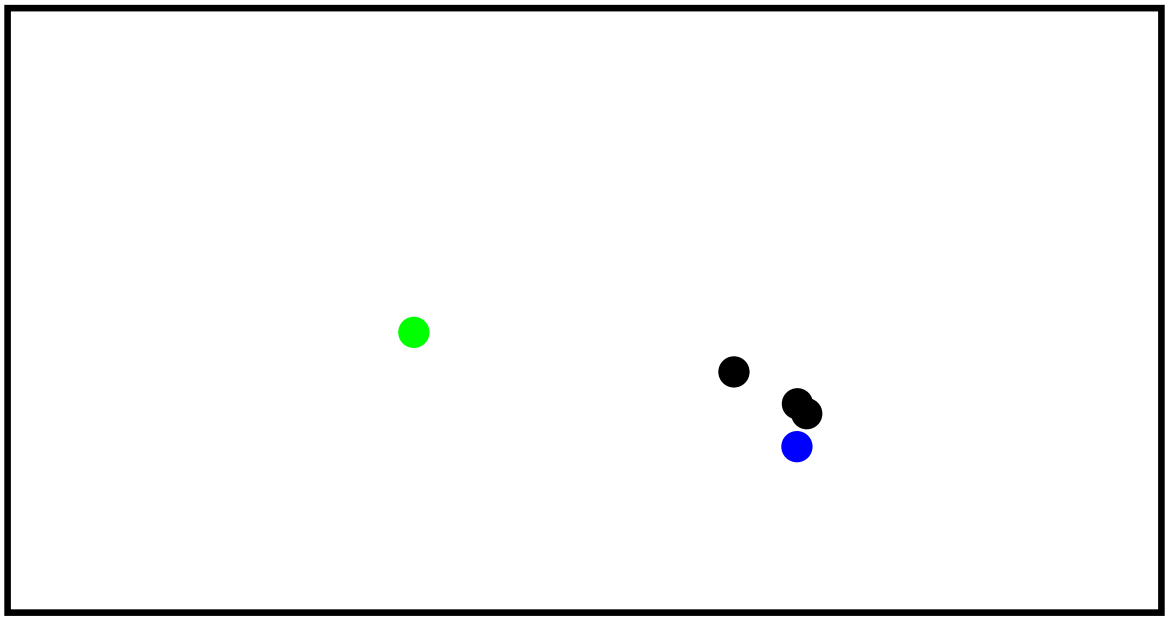
● ● → ● ●

● ● → ● ●

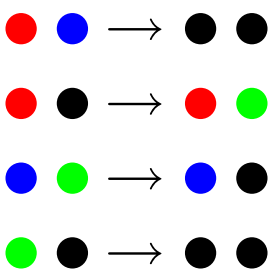
● ● → ● ●

● ● → ● ●

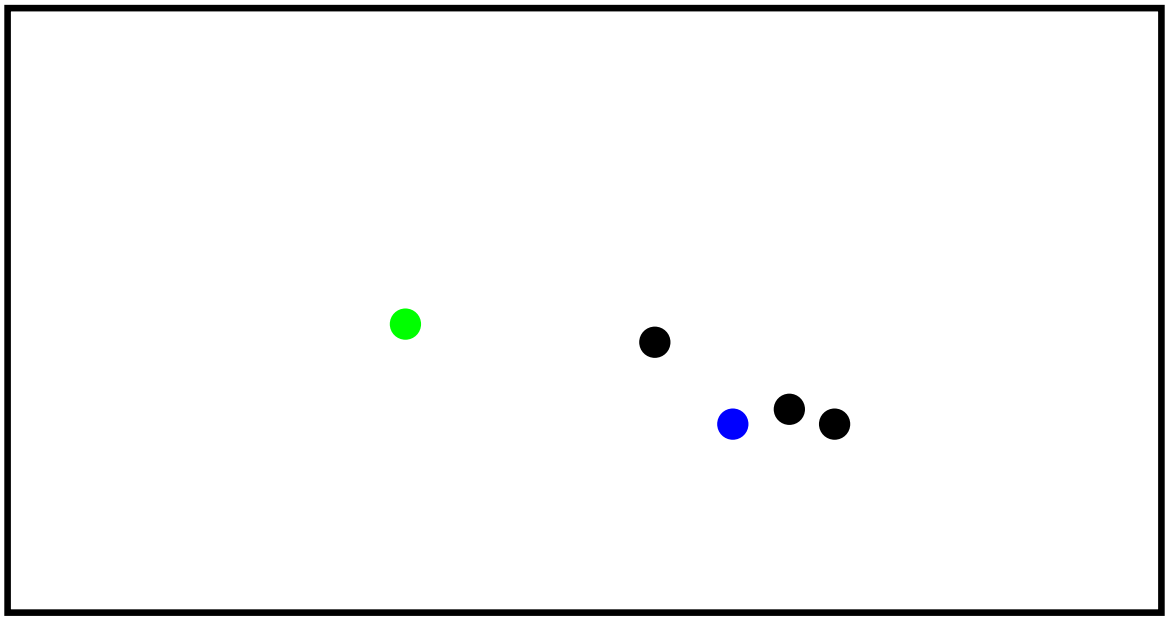
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

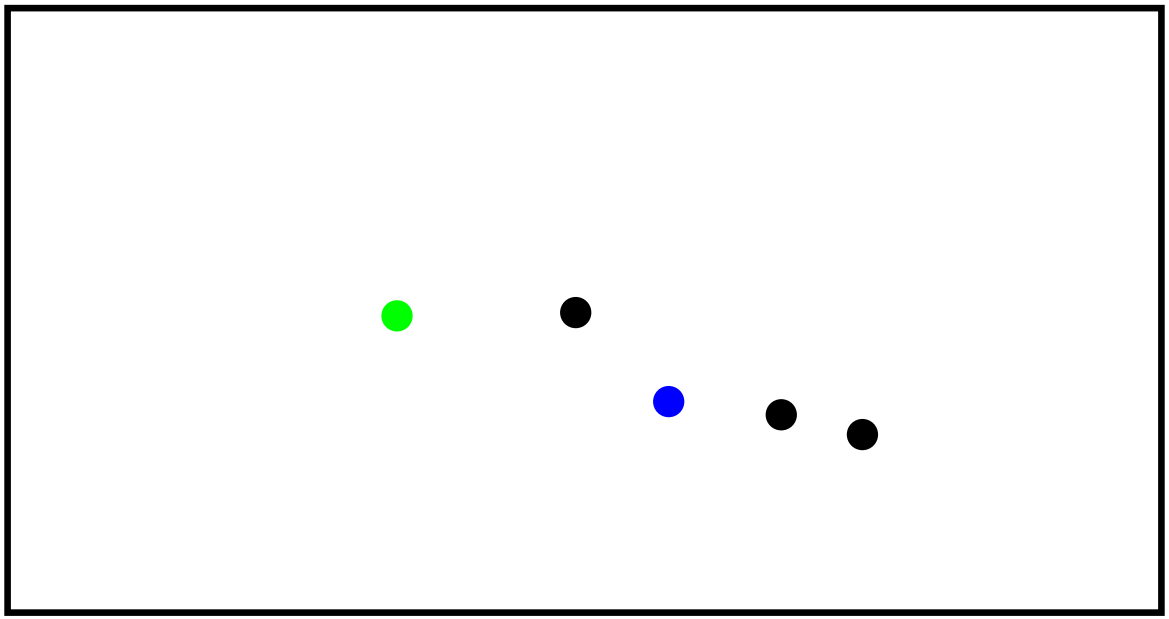
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

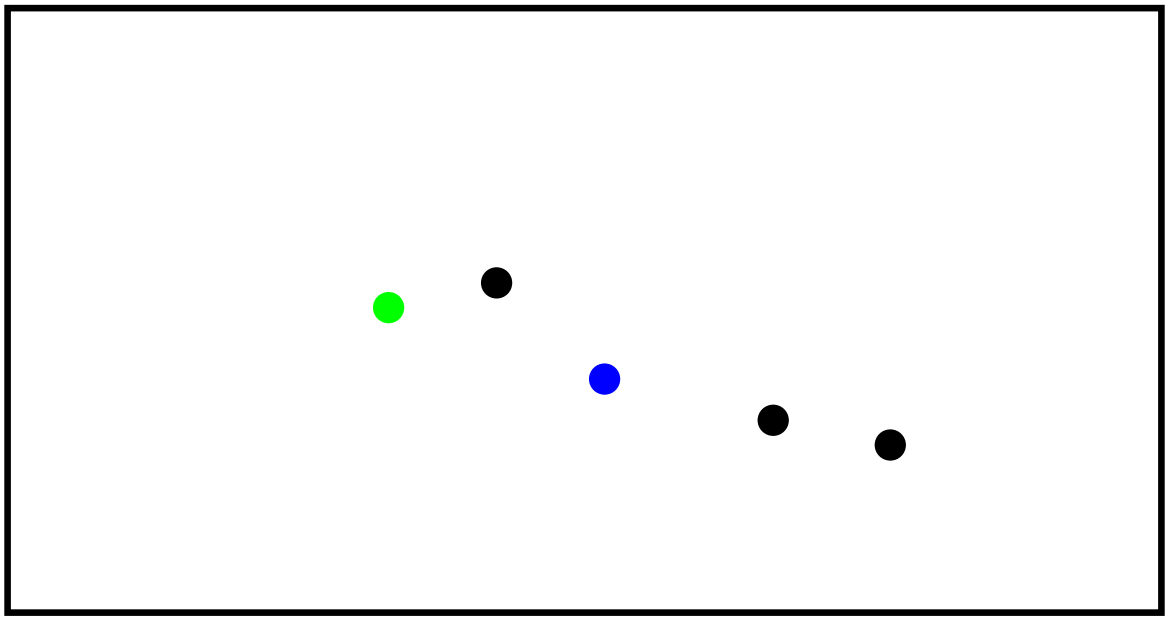
● ● → ● ●

● ● → ● ●

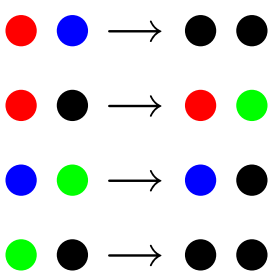
● ● → ● ●

● ● → ● ●

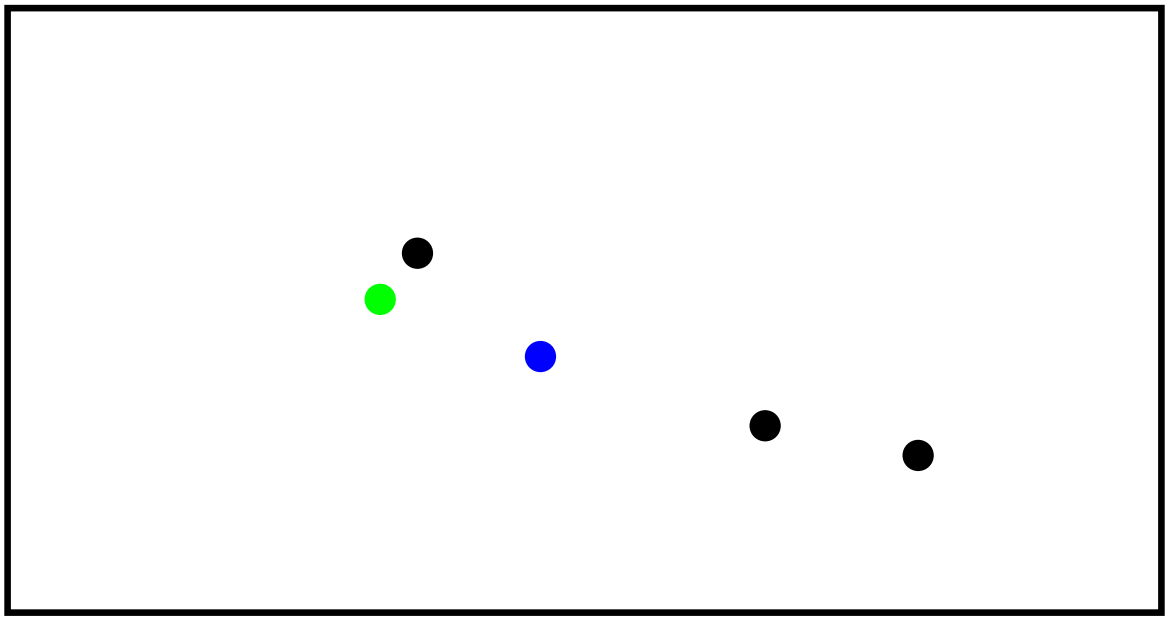
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

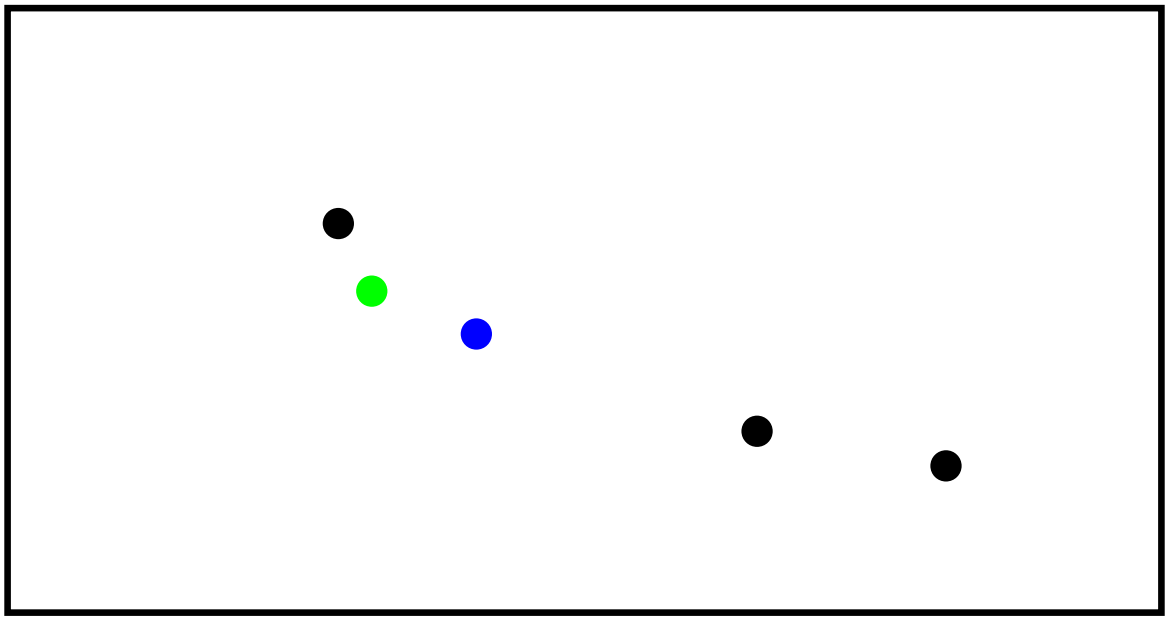
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

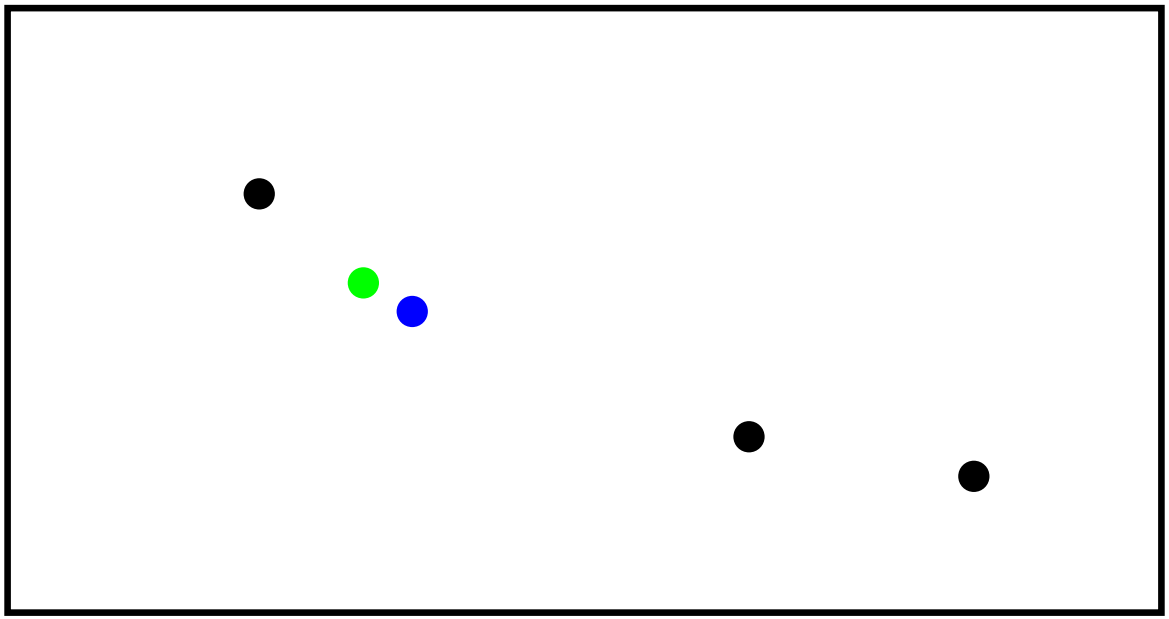
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1 : partant de 3 ●, 2 ●



Program :

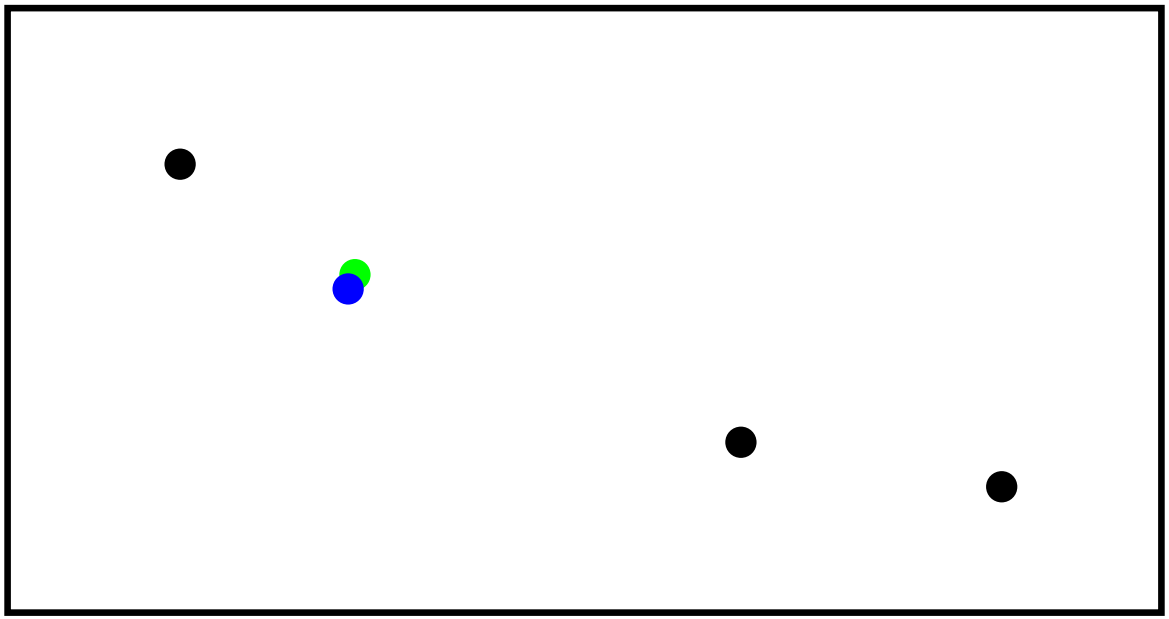
● ● → ● ●

● ● → ● ●

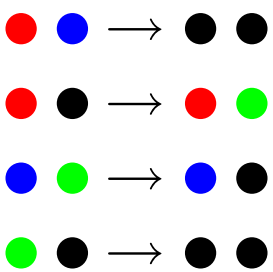
● ● → ● ●

● ● → ● ●

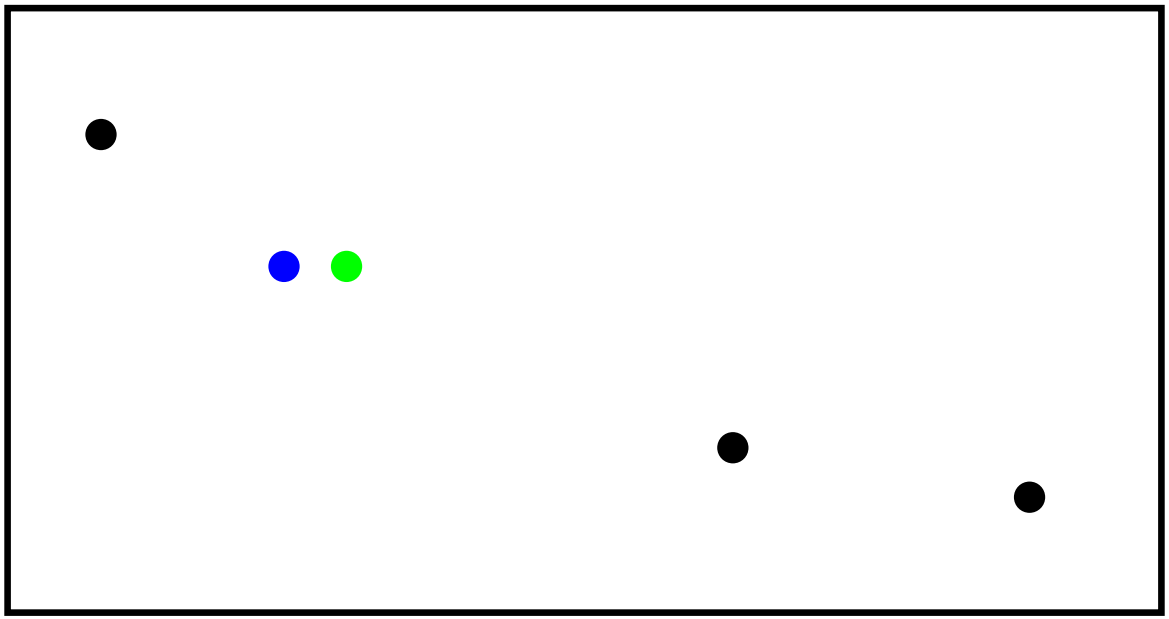
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

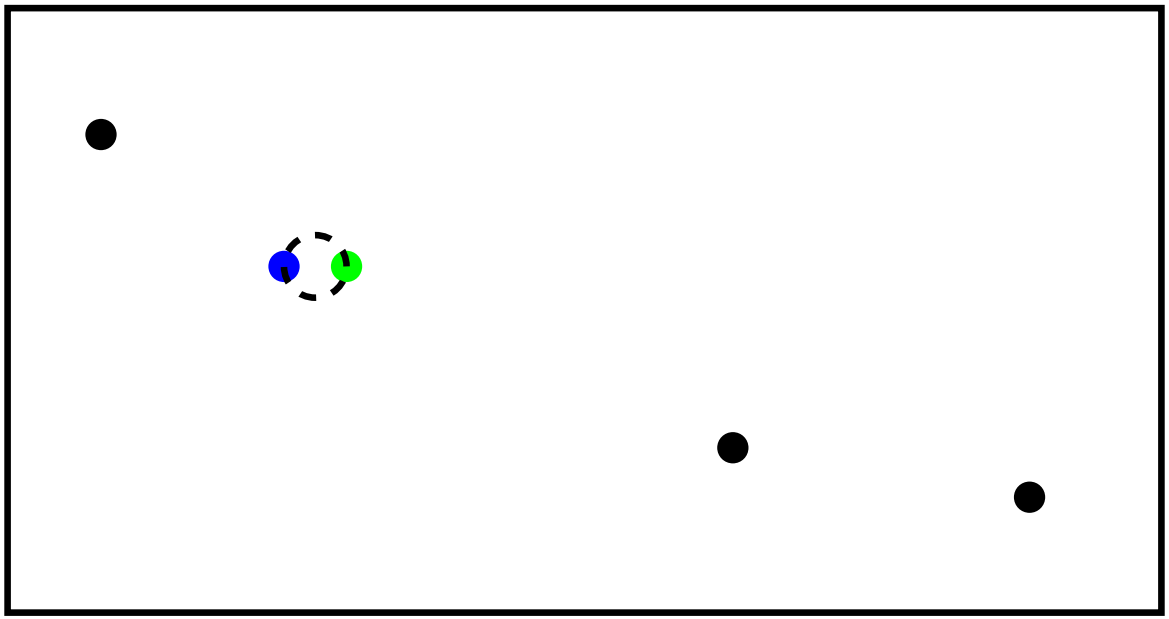
● ● → ● ●

● ● → ● ●

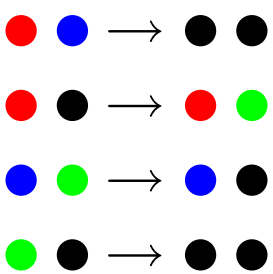
● ● → ● ●

● ● → ● ●

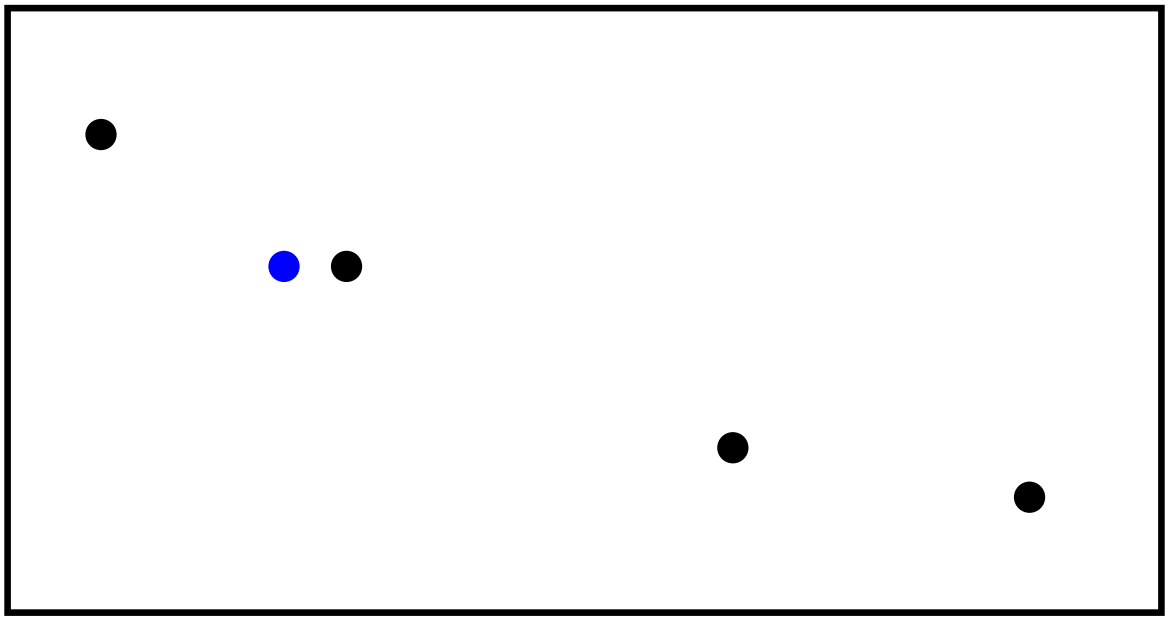
Exemple 1 : partant de 3 ●, 2 ●



Program :



Exemple 1 : partant de 3 ●, 2 ●



Program :

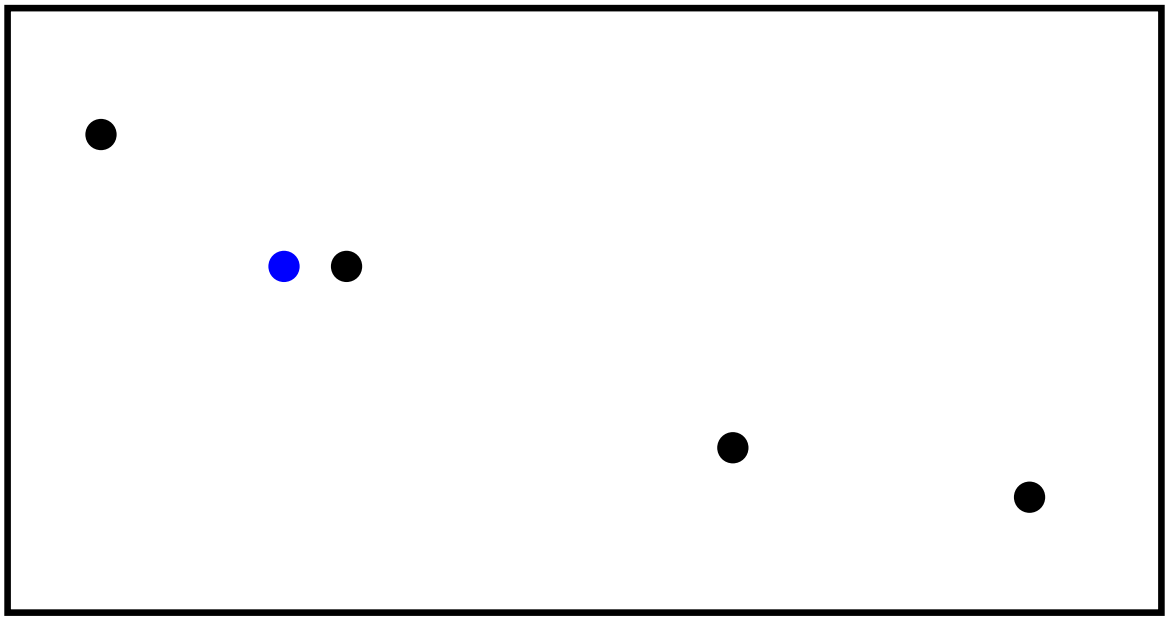
● ● → ● ●

● ● → ● ●

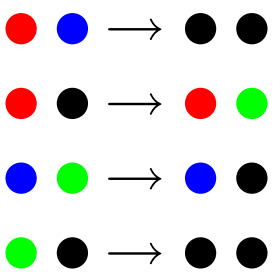
● ● → ● ●

● ● → ● ●

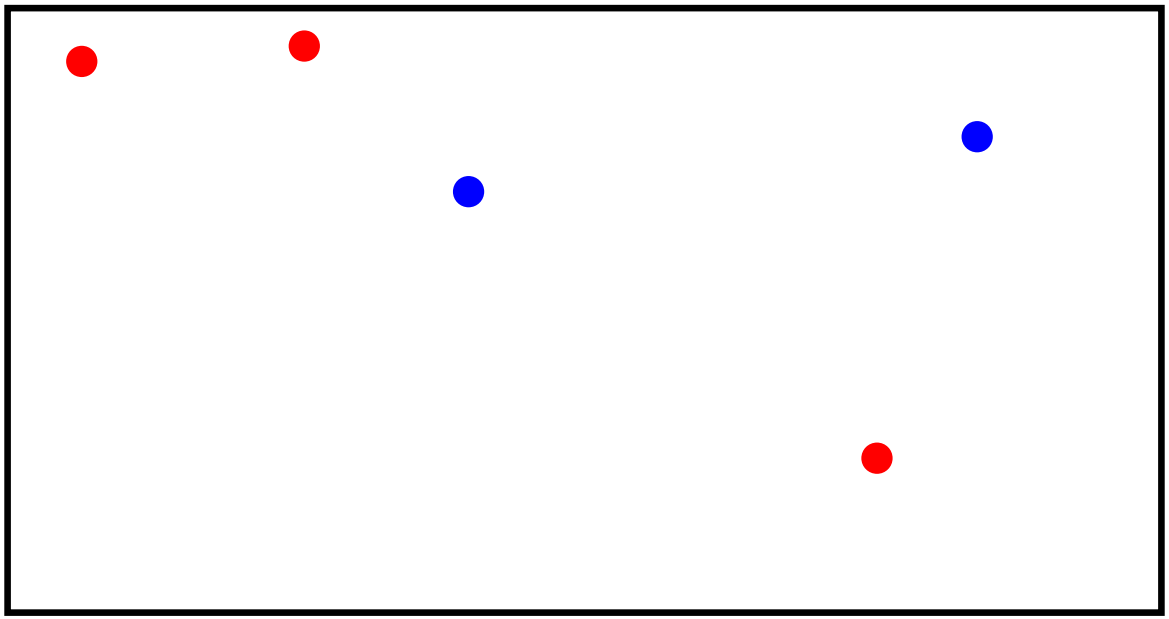
Example 1 : Résultat final



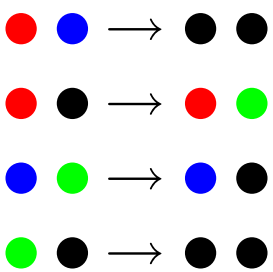
Program :



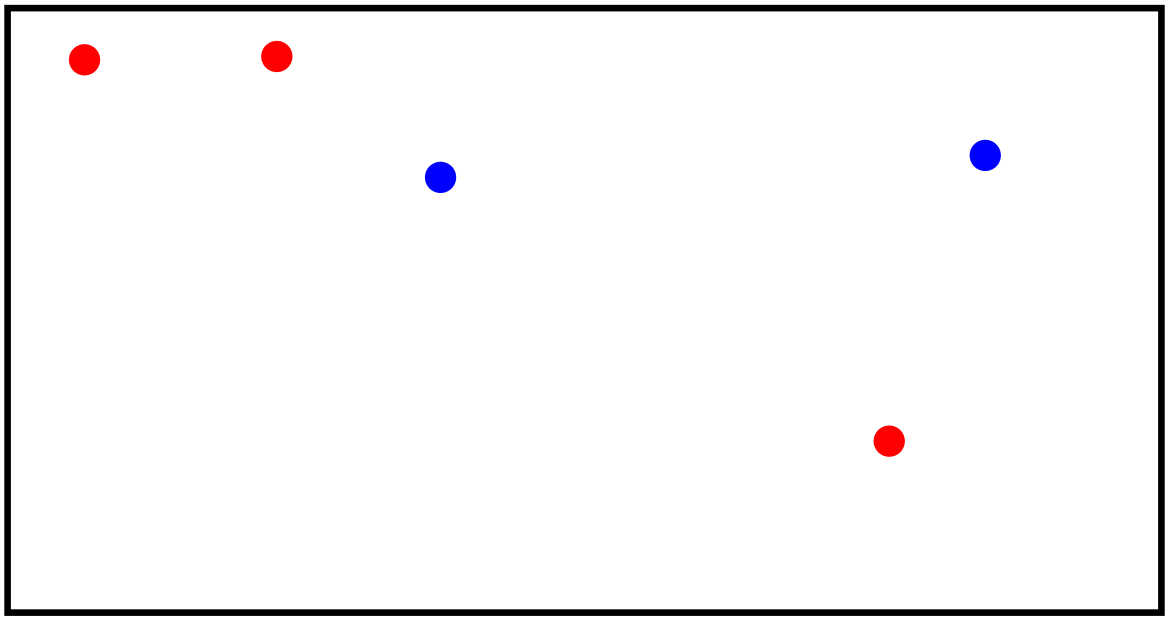
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

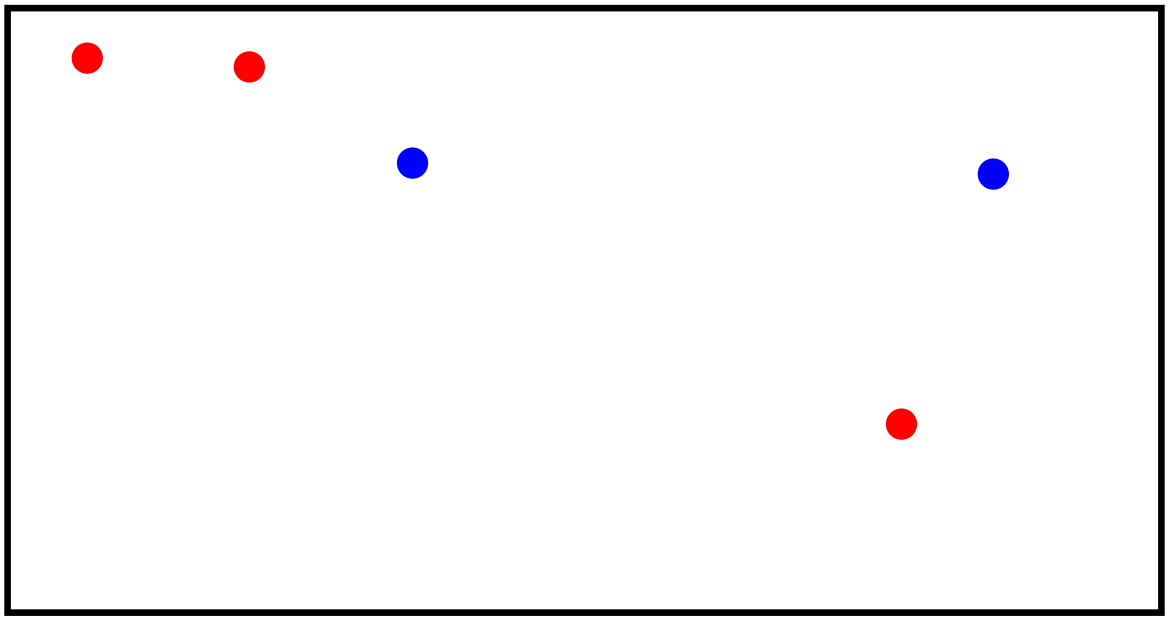
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

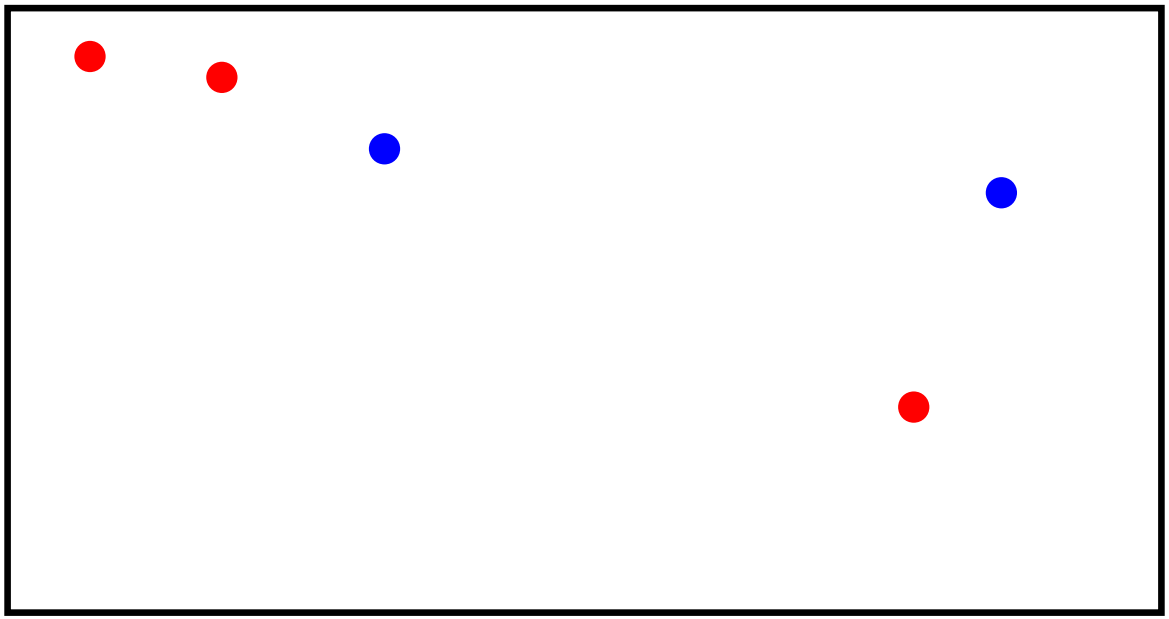
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

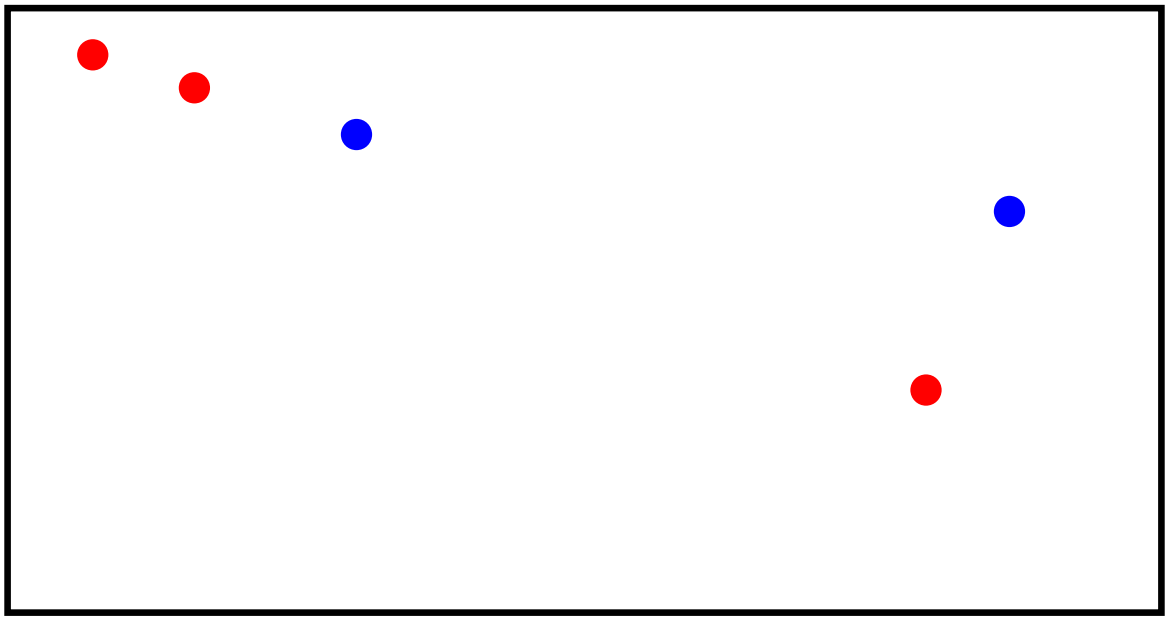
● ● → ● ●

● ● → ● ●

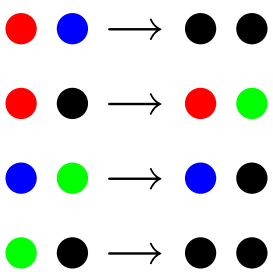
● ● → ● ●

● ● → ● ●

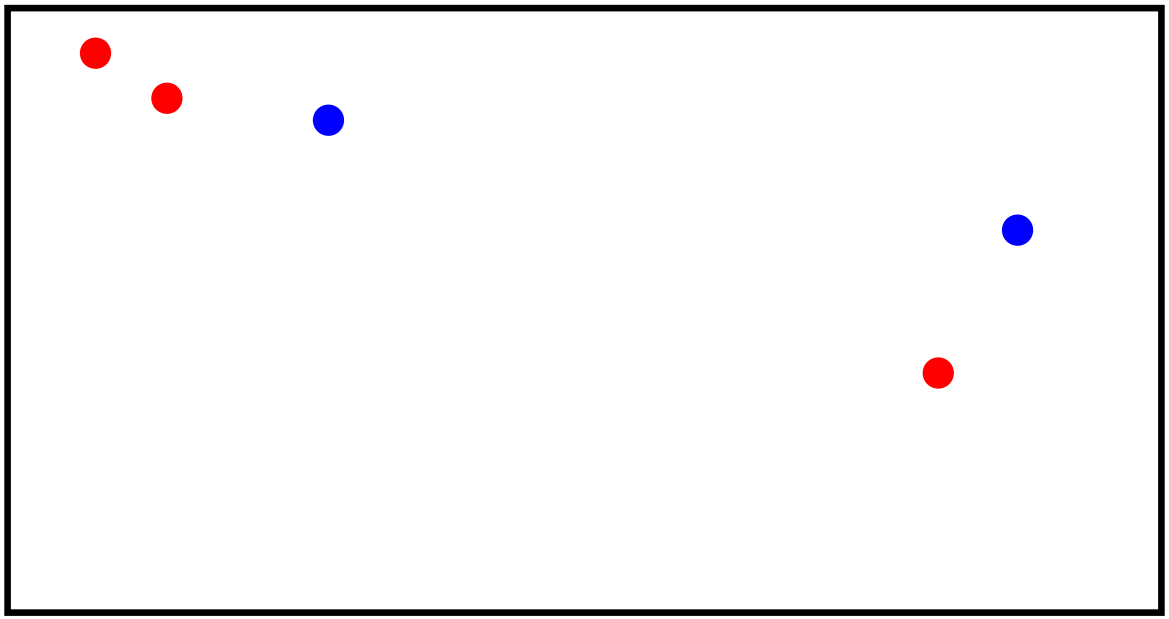
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

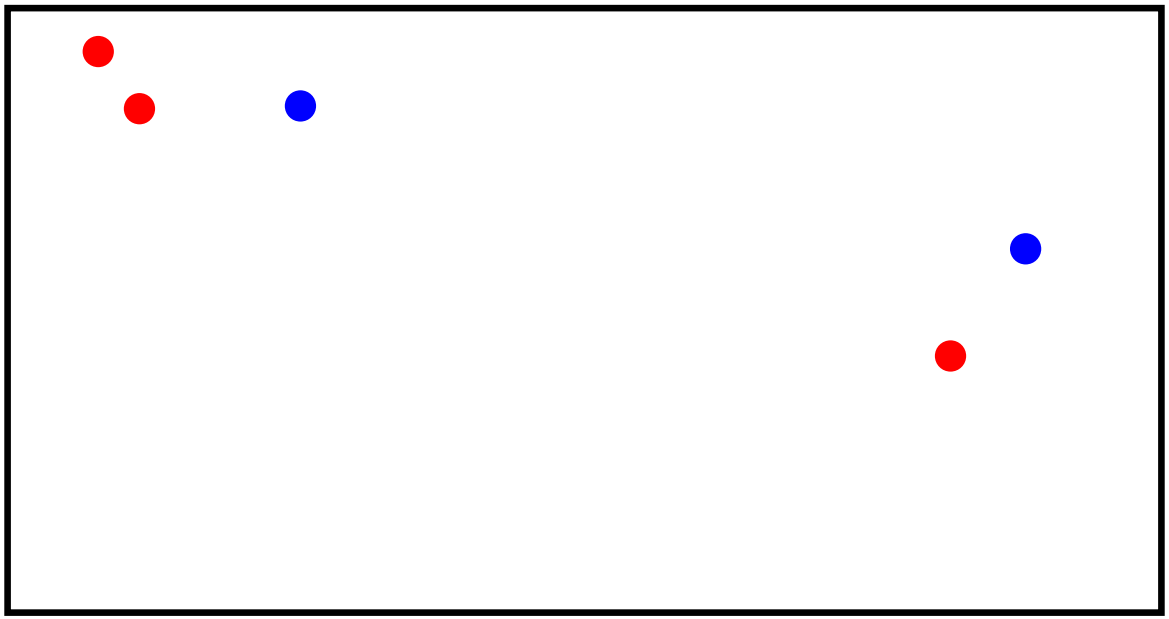
● ● → ● ●

● ● → ● ●

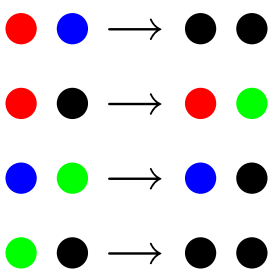
● ● → ● ●

● ● → ● ●

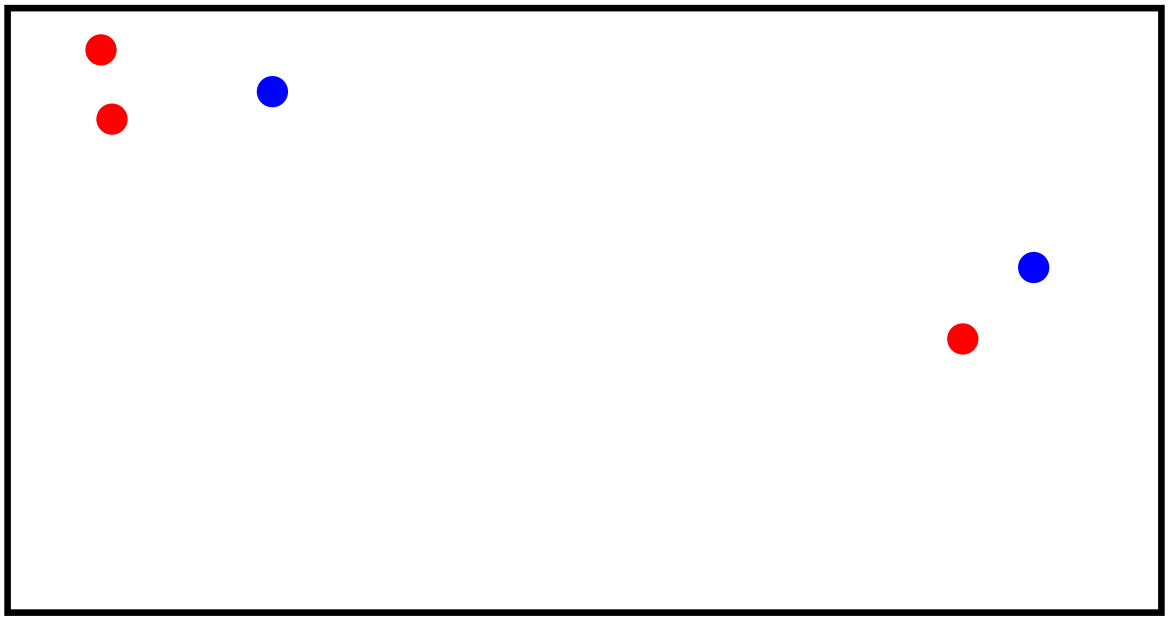
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

● ● → ● ●

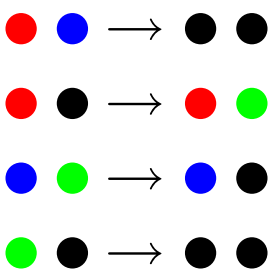
● ● → ● ●

● ● → ● ●

● ● → ● ●

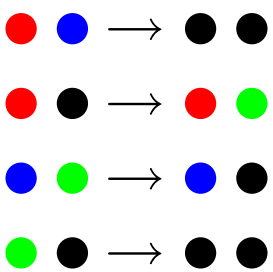
Exemple 2 : partant de 2 ●, 3●

Program :



Exemple 2 : partant de 2 ●, 3●

Program :



Exemple 2 : partant de 2 ●, 3●

Program :

● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●

Program :

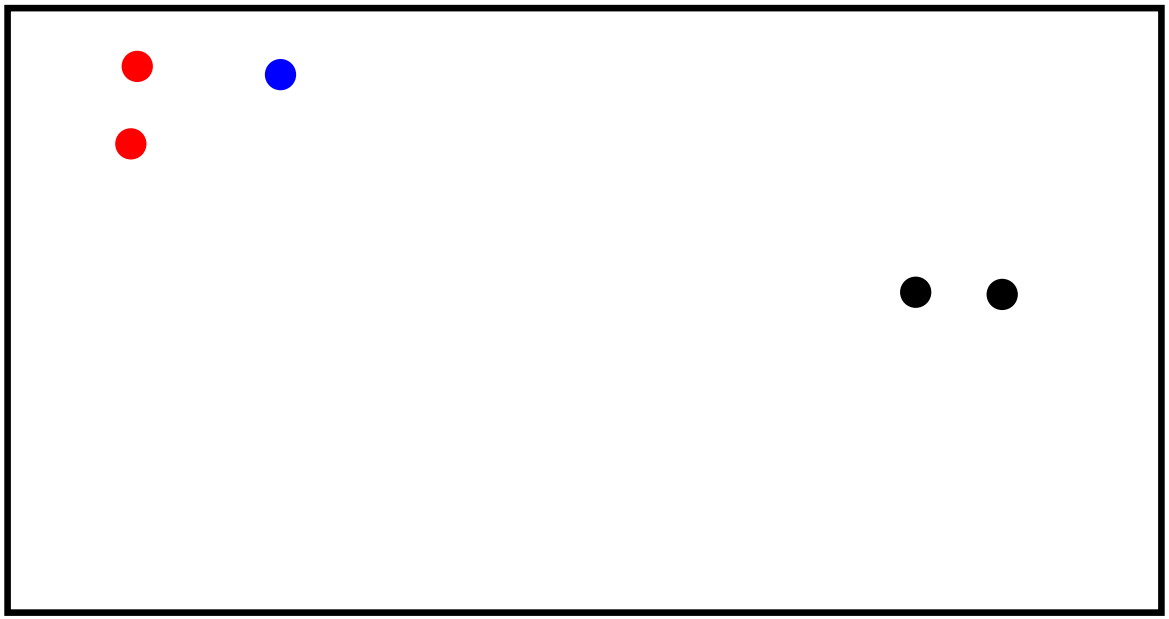
● ● → ● ●

● ● → ● ●

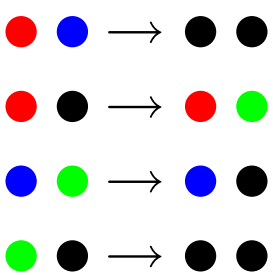
● ● → ● ●

● ● → ● ●

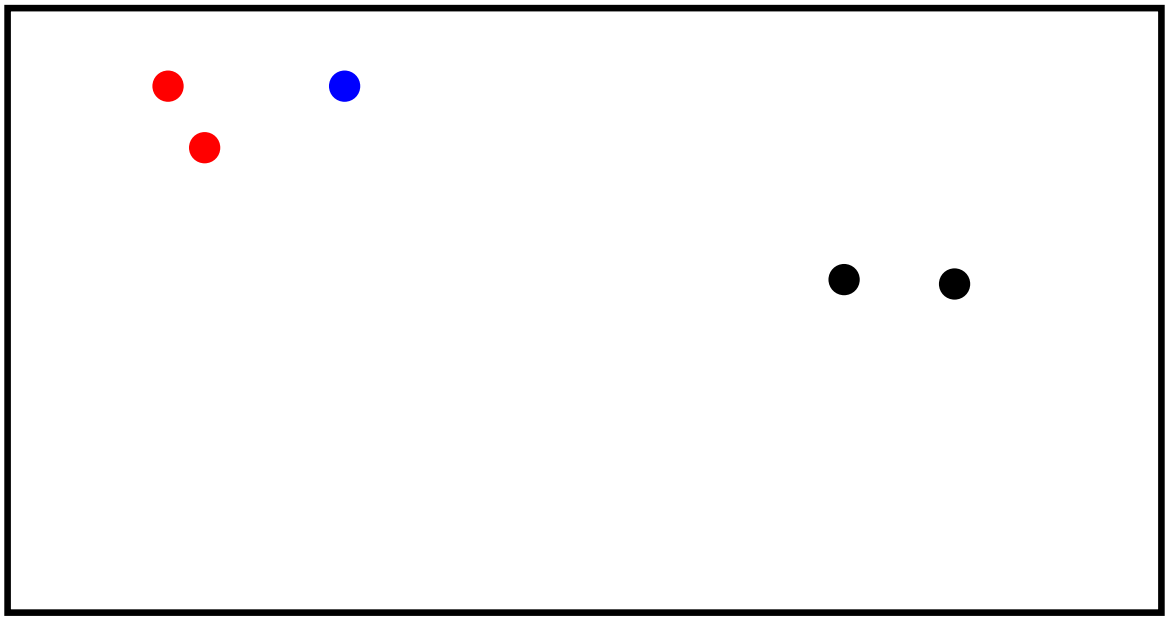
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

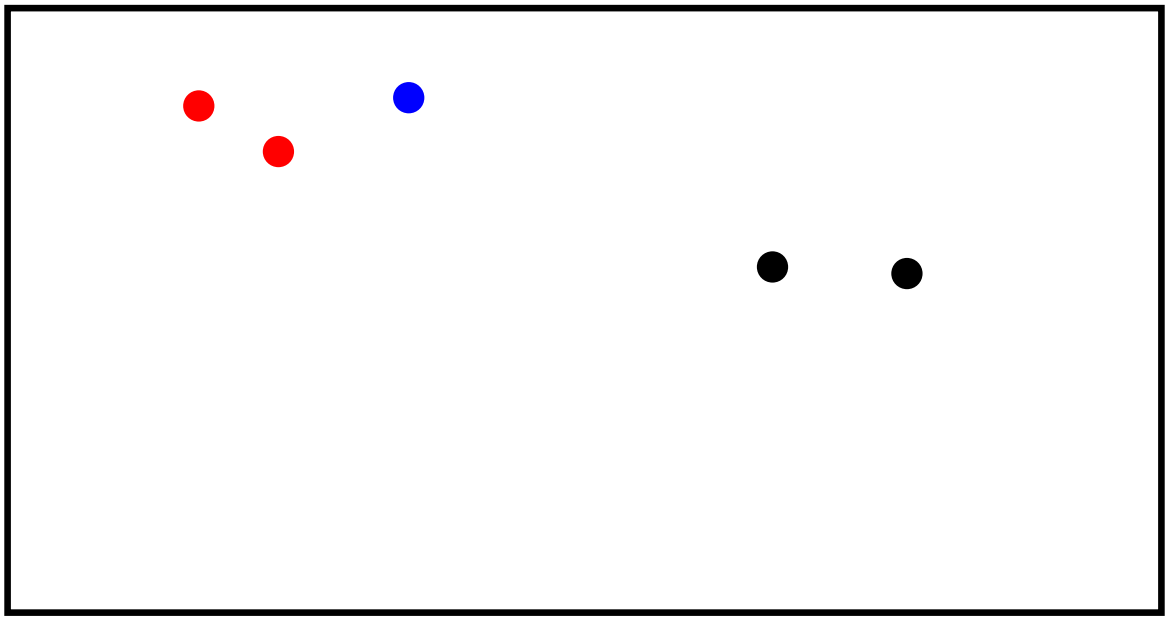
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

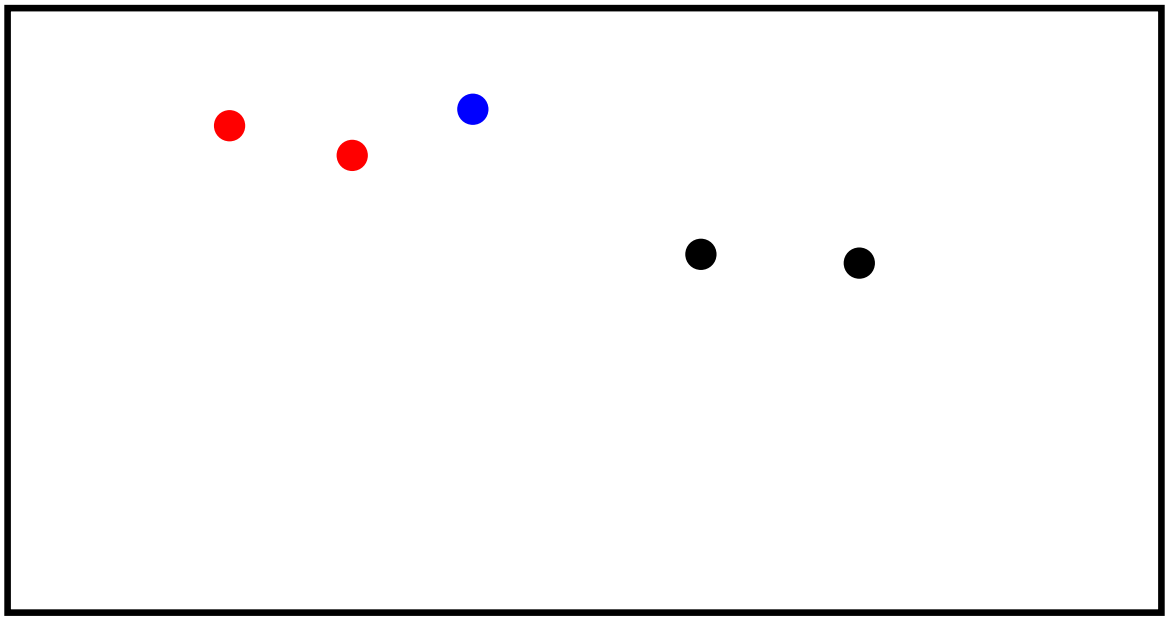
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

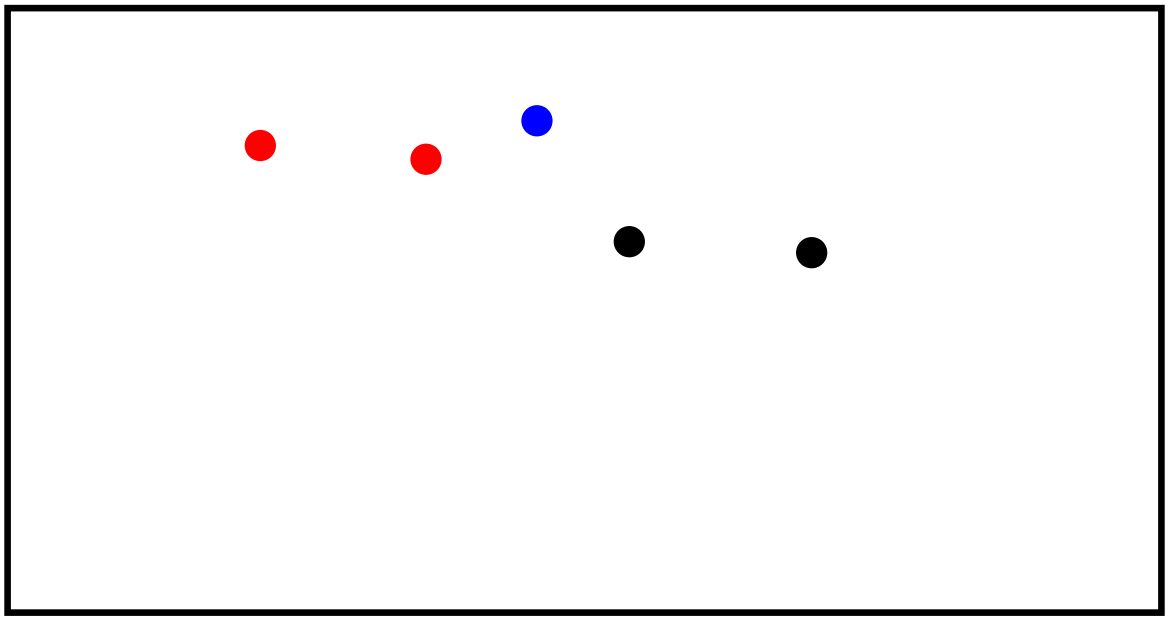
● ● → ● ●

● ● → ● ●

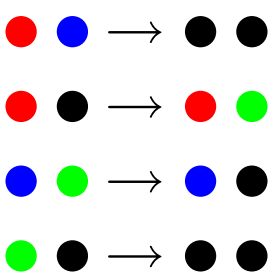
● ● → ● ●

● ● → ● ●

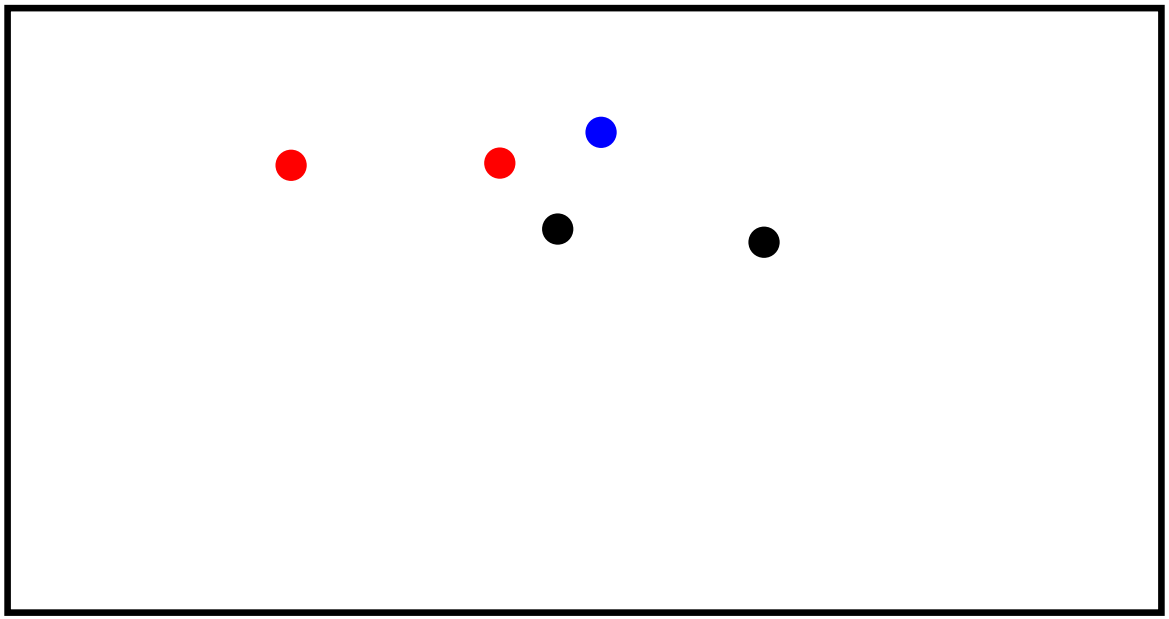
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

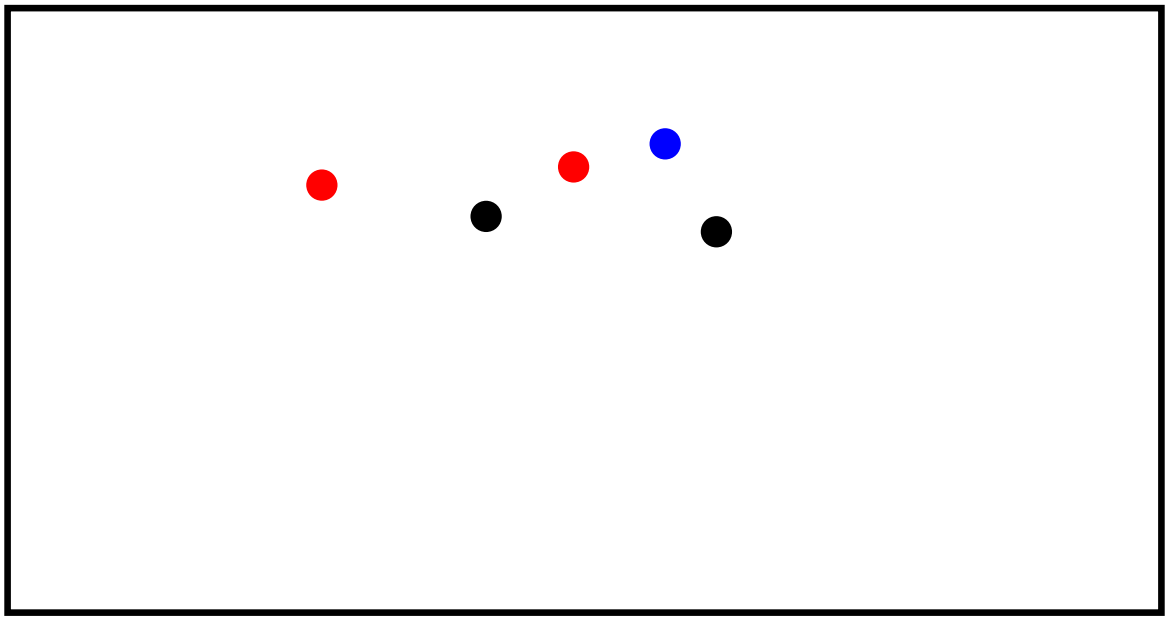
● ● → ● ●

● ● → ● ●

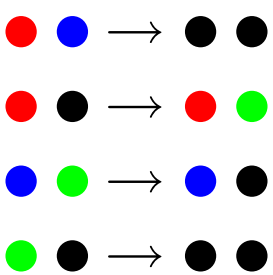
● ● → ● ●

● ● → ● ●

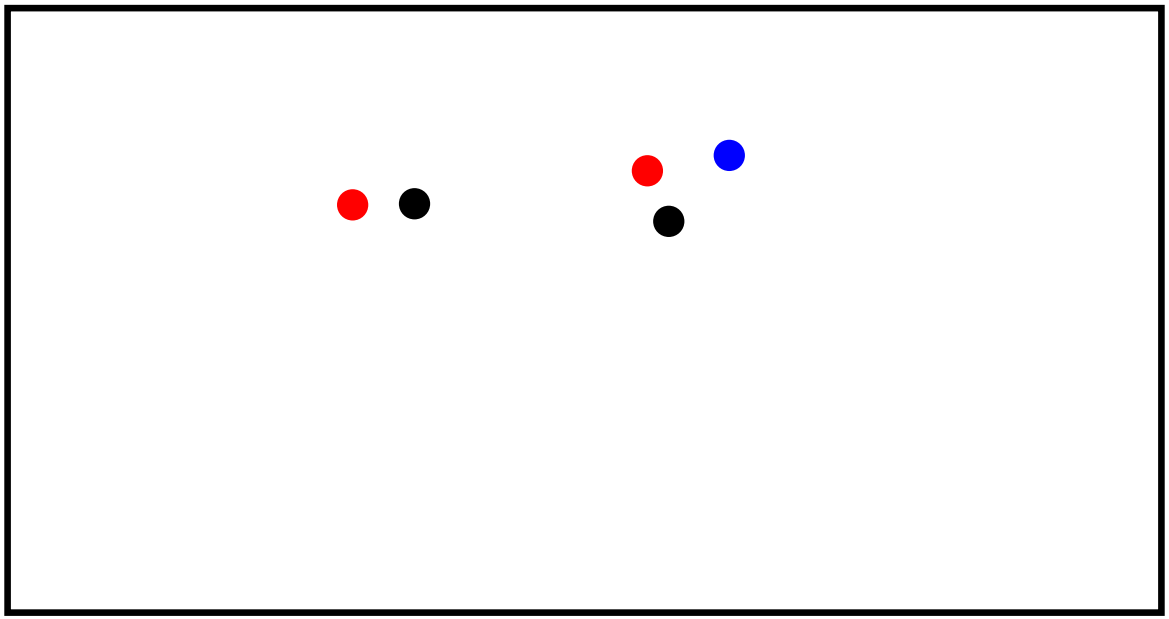
Exemple 2 : partant de 2 ●, 3●



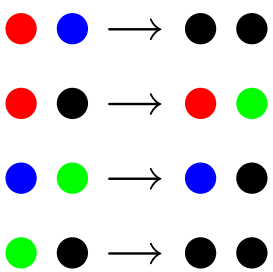
Program :



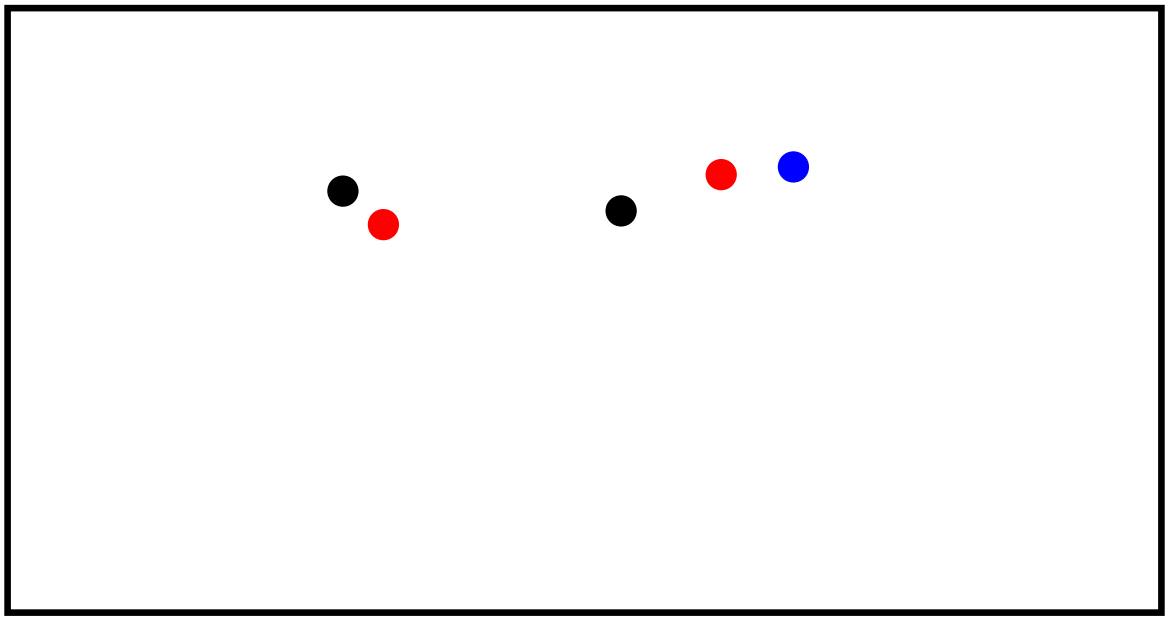
Exemple 2 : partant de 2 ●, 3●



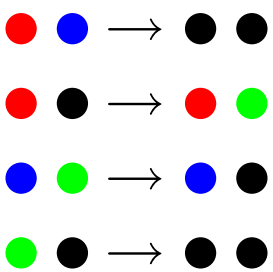
Program :



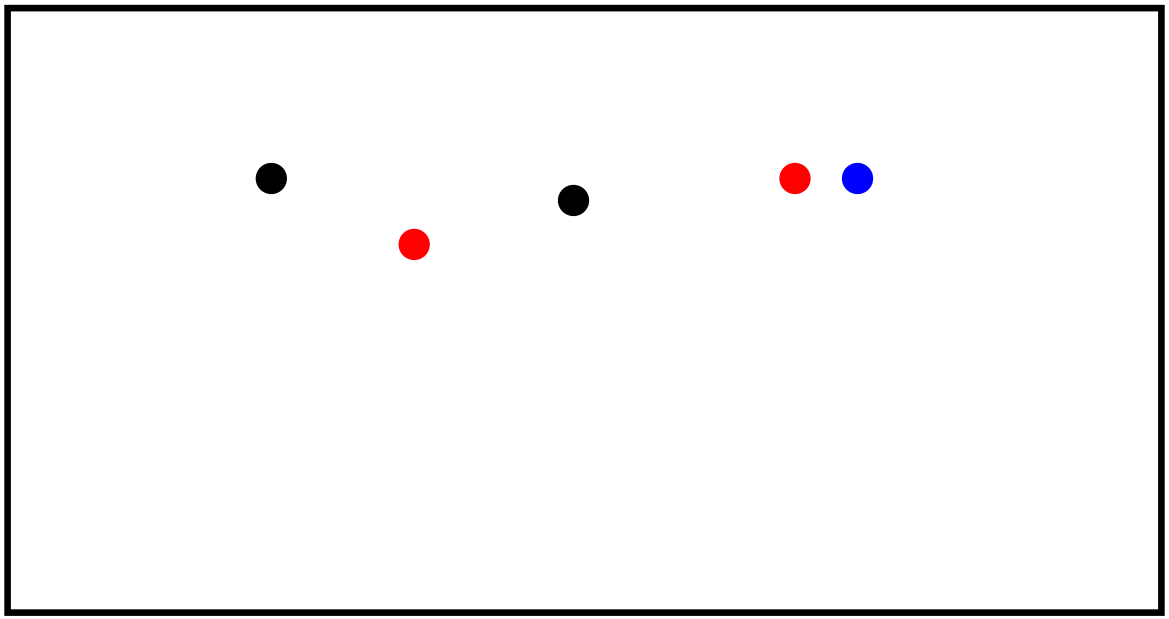
Exemple 2 : partant de 2 ●, 3●



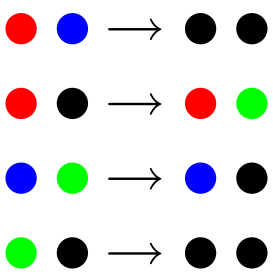
Program :



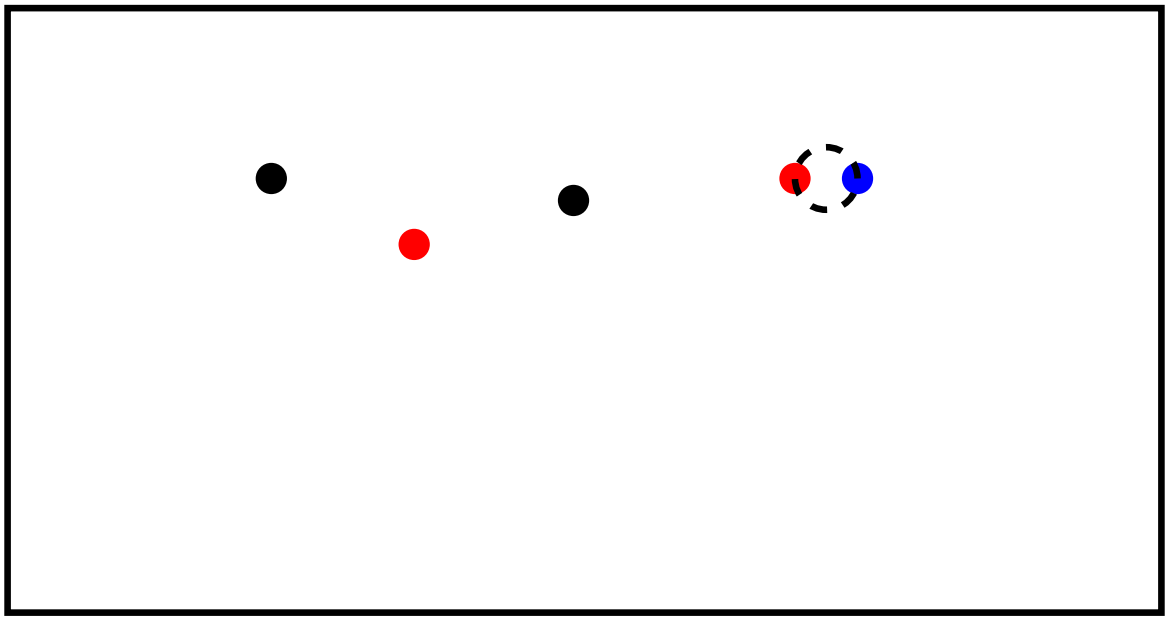
Exemple 2 : partant de 2 ●, 3●



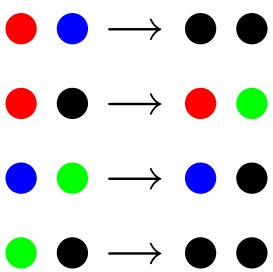
Program :



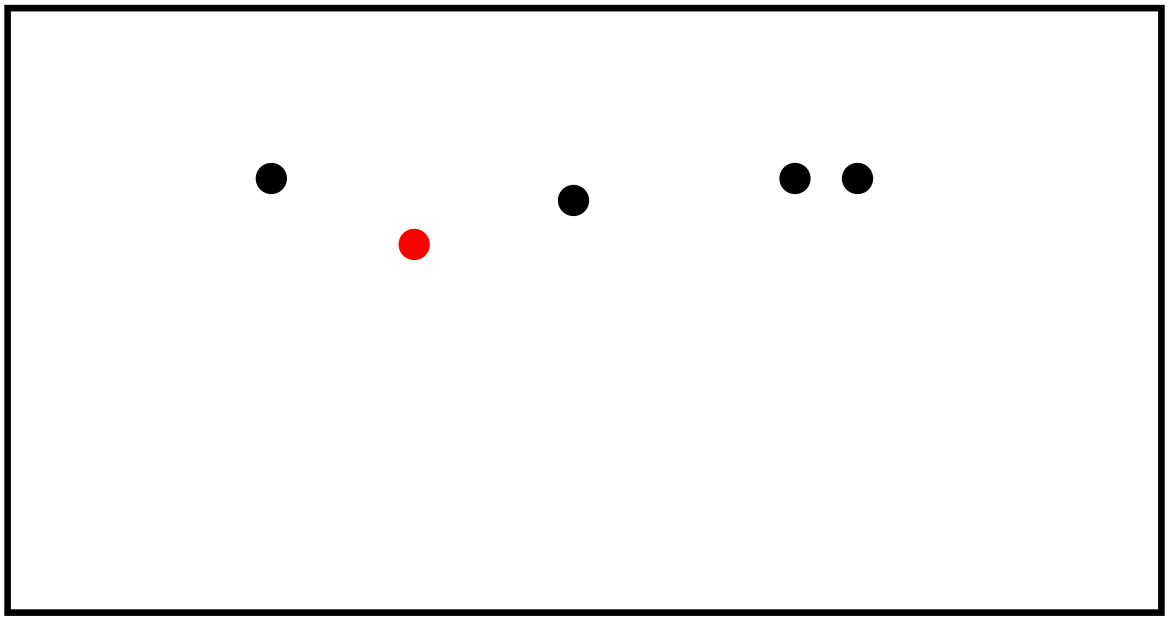
Exemple 2 : partant de 2 ●, 3●



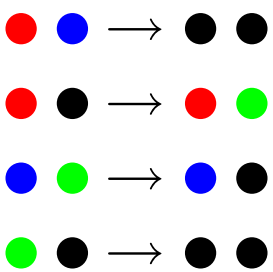
Program :



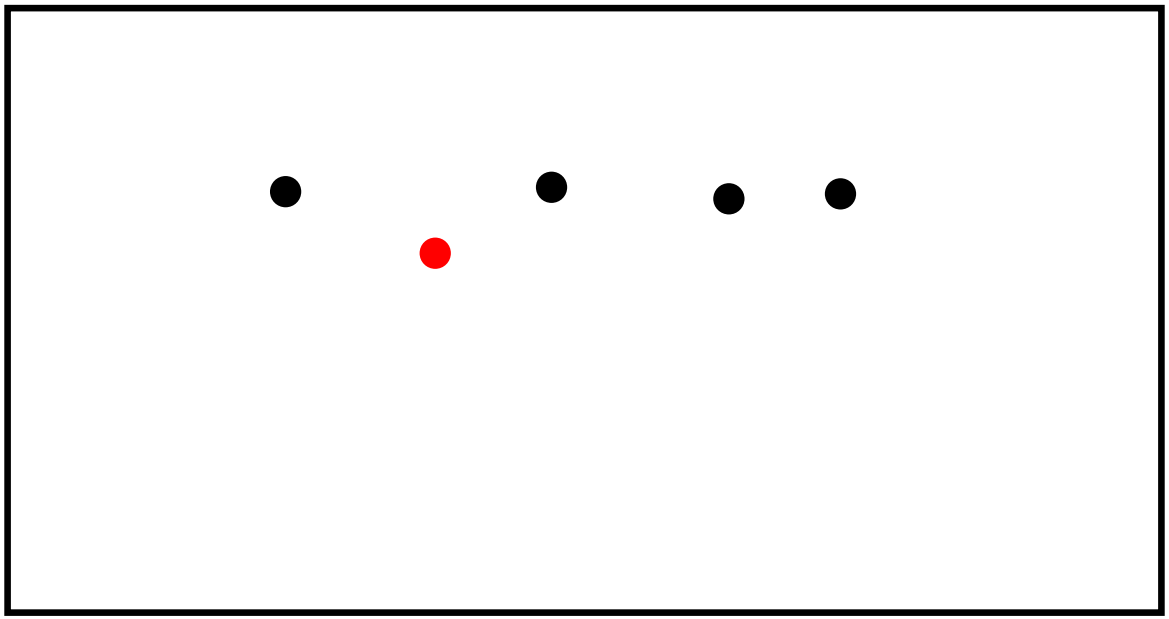
Exemple 2 : partant de 2 ●, 3●



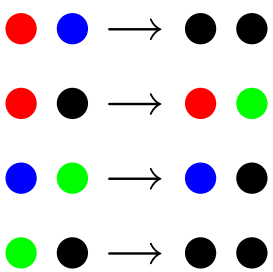
Program :



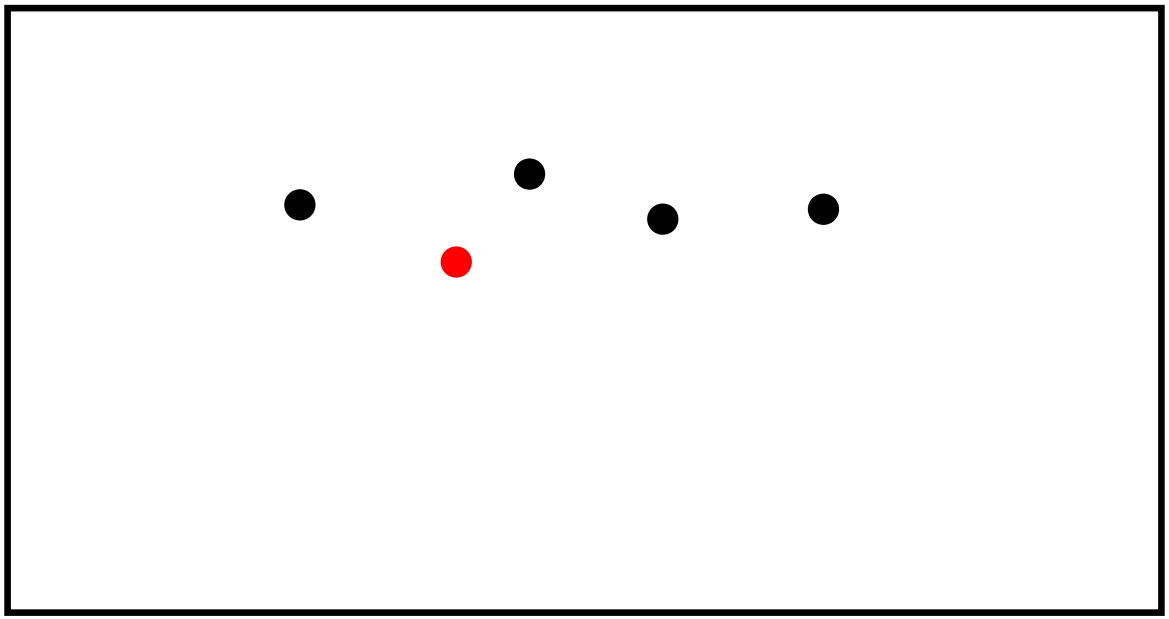
Exemple 2 : partant de 2 ●, 3●



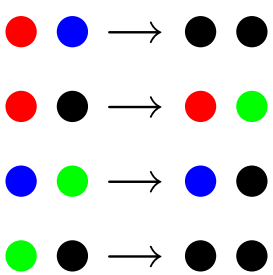
Program :



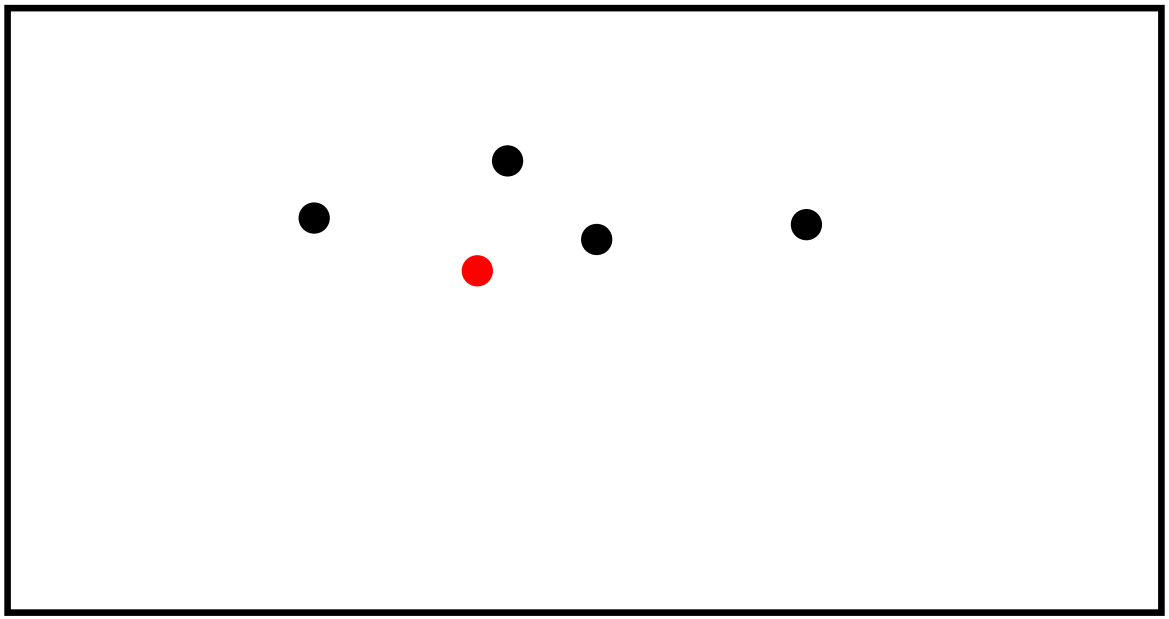
Exemple 2 : partant de 2 ●, 3●



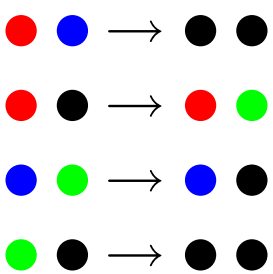
Program :



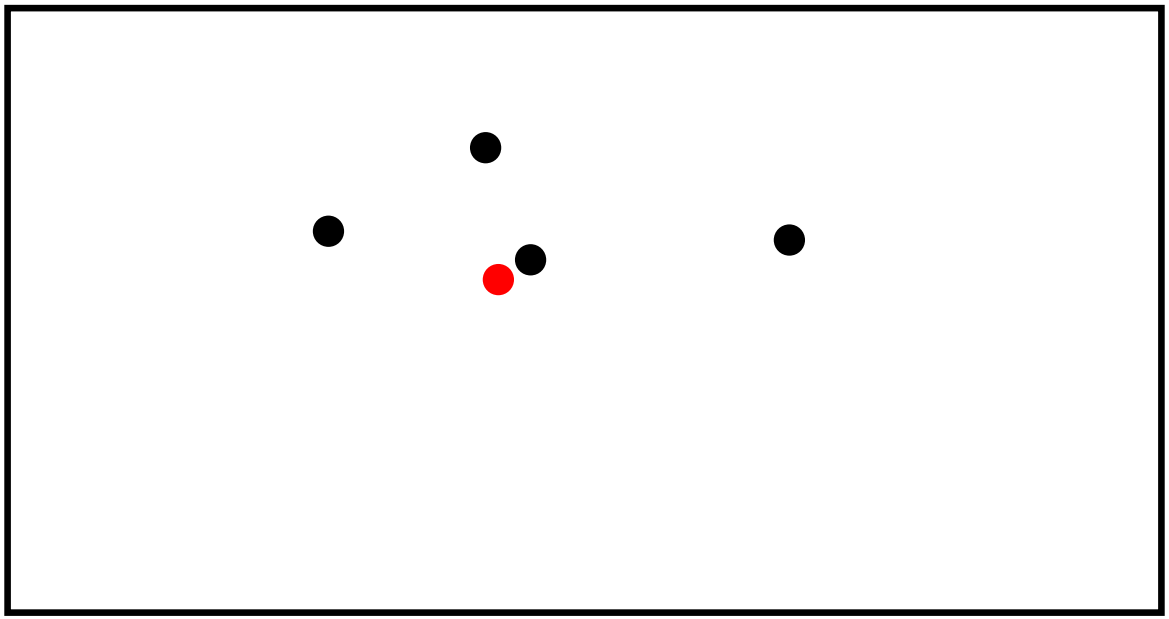
Exemple 2 : partant de 2 ●, 3●



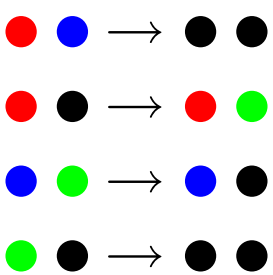
Program :



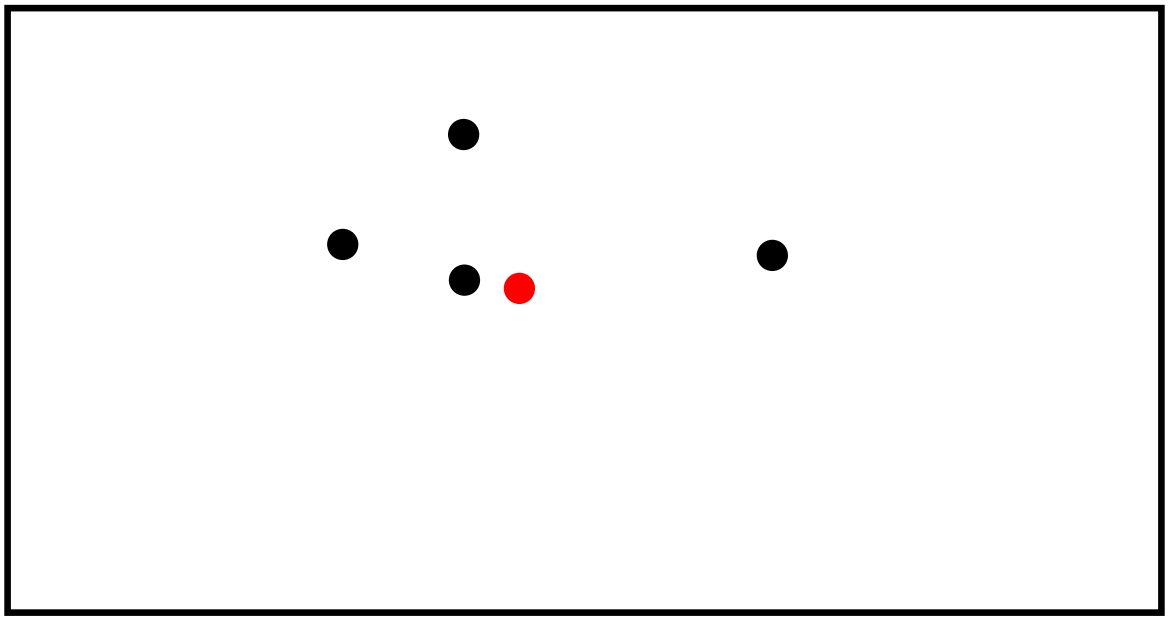
Exemple 2 : partant de 2 ●, 3●



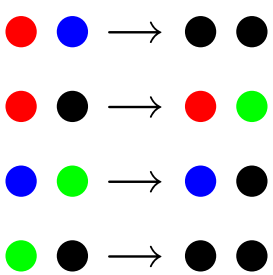
Program :



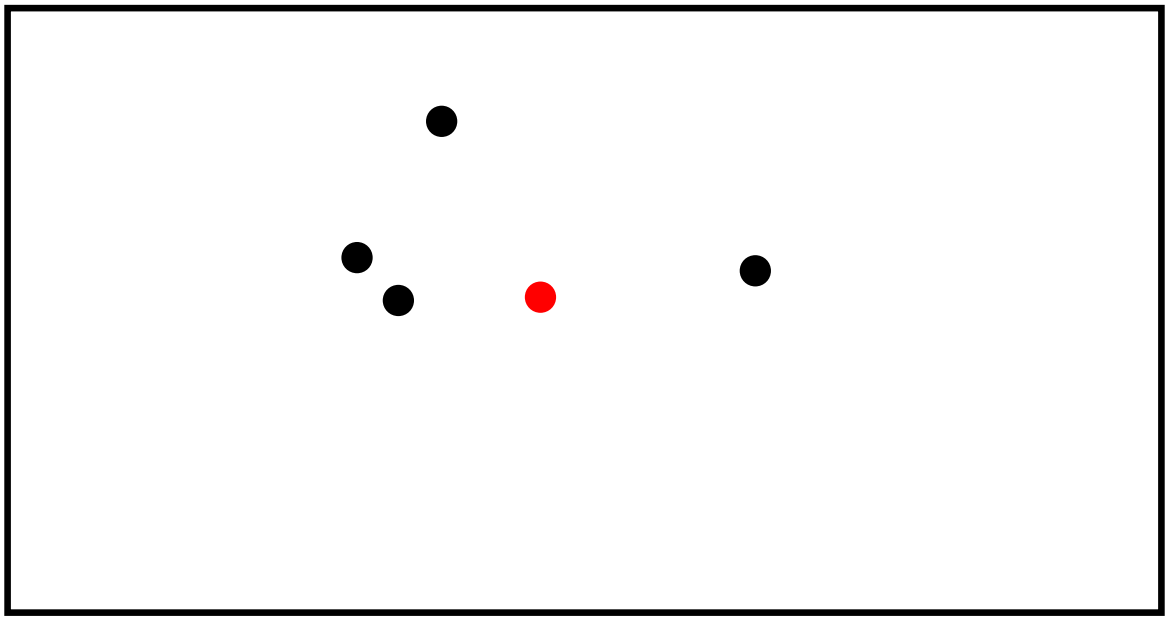
Exemple 2 : partant de 2 ●, 3●



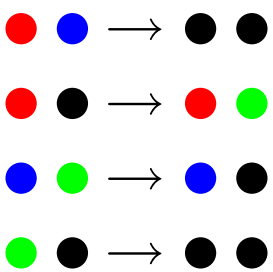
Program :



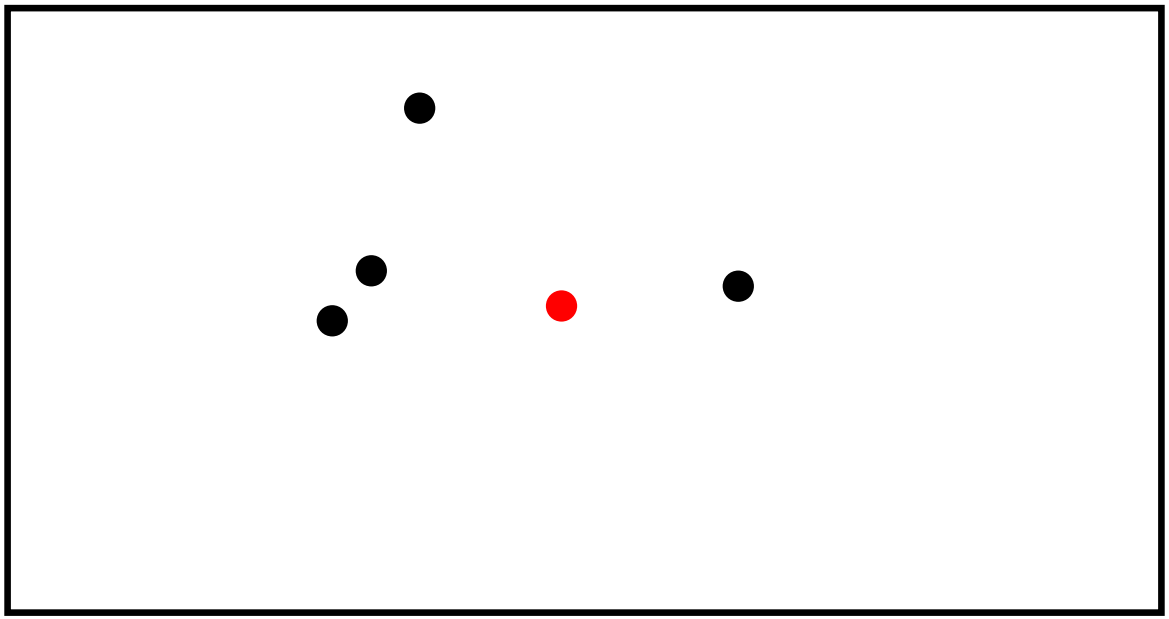
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

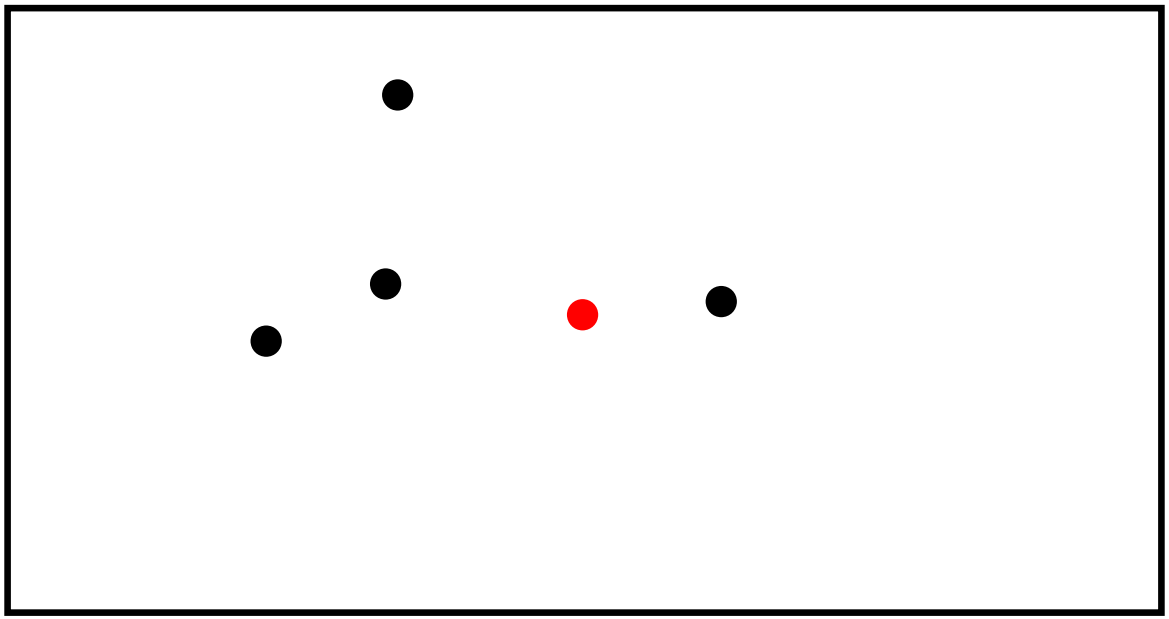
● ● → ● ●

● ● → ● ●

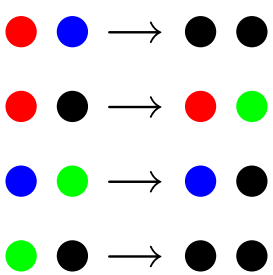
● ● → ● ●

● ● → ● ●

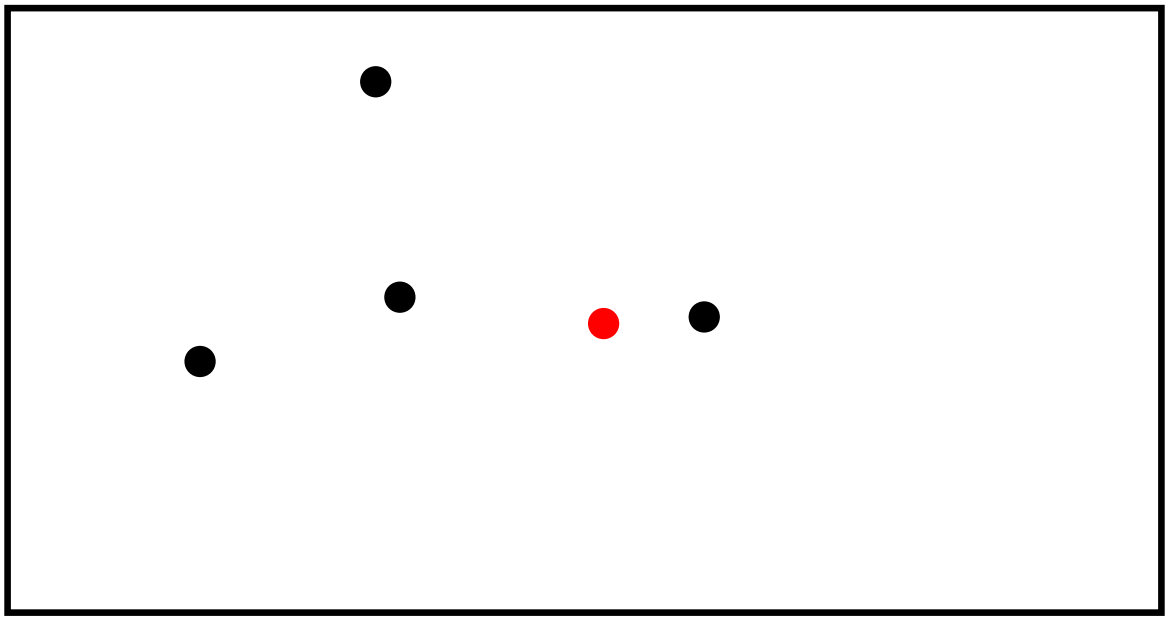
Exemple 2 : partant de 2 ●, 3●



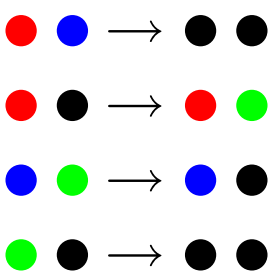
Program :



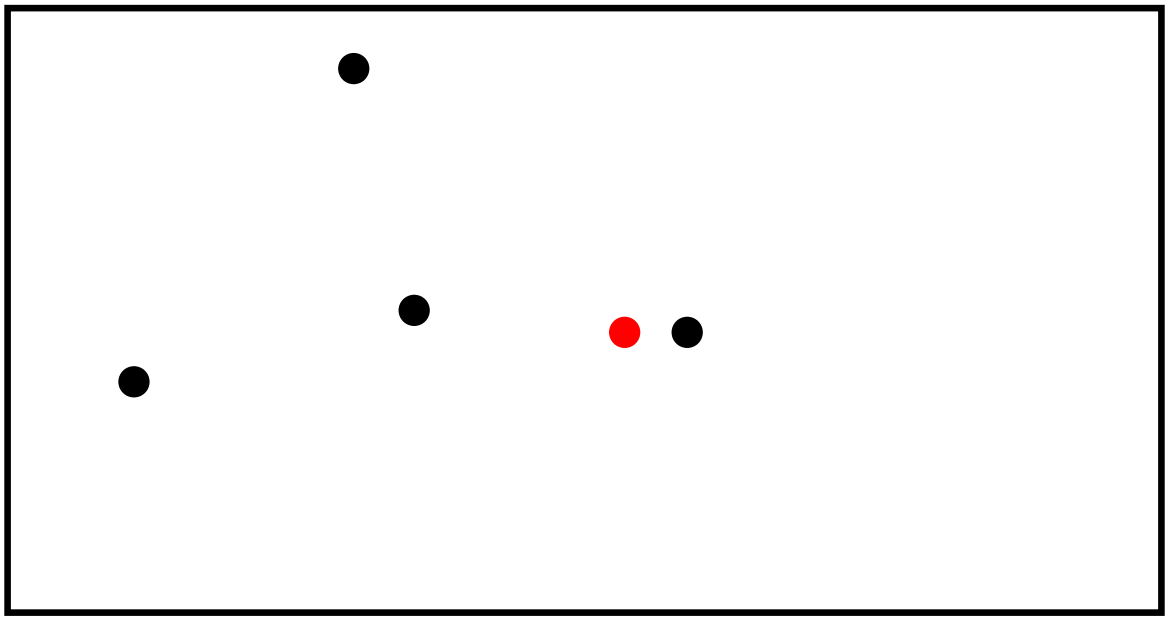
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

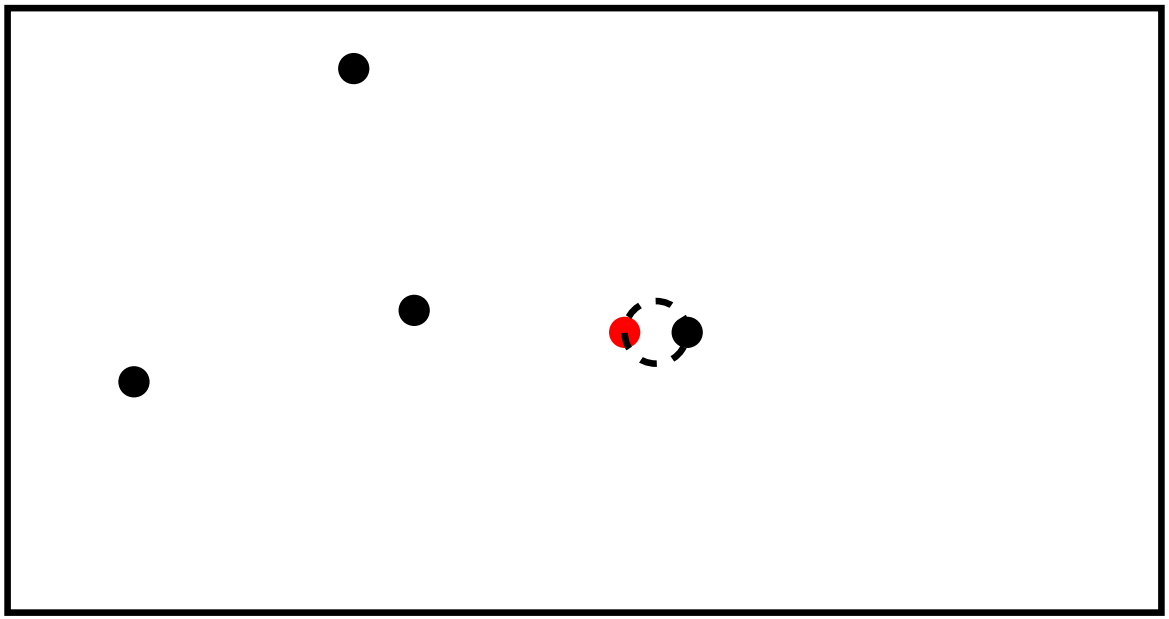
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

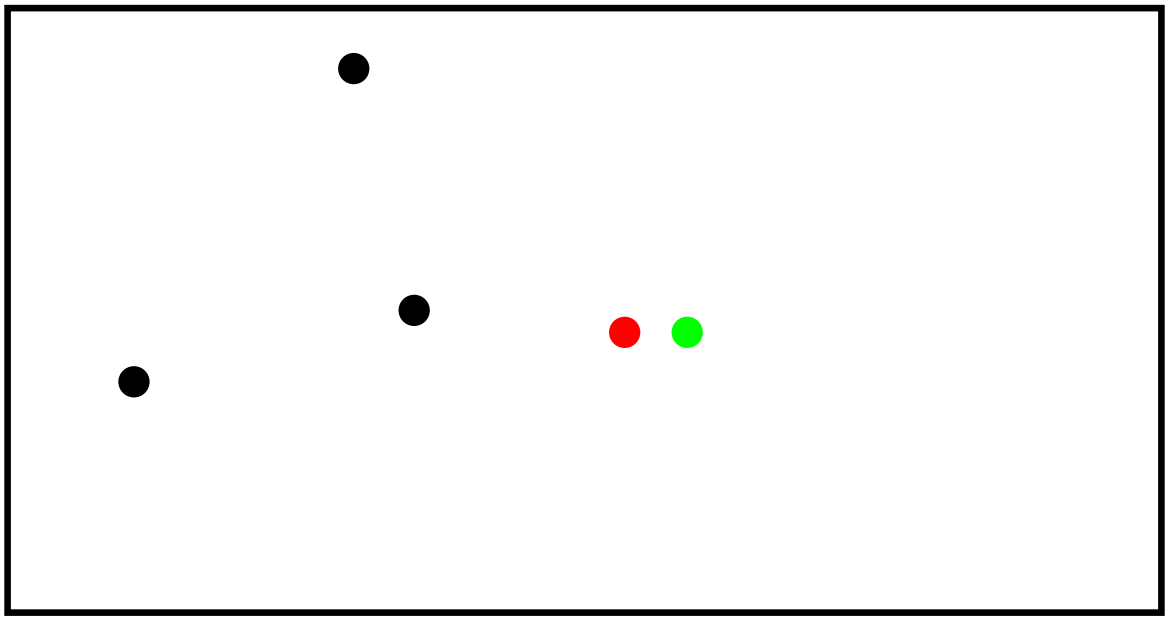
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

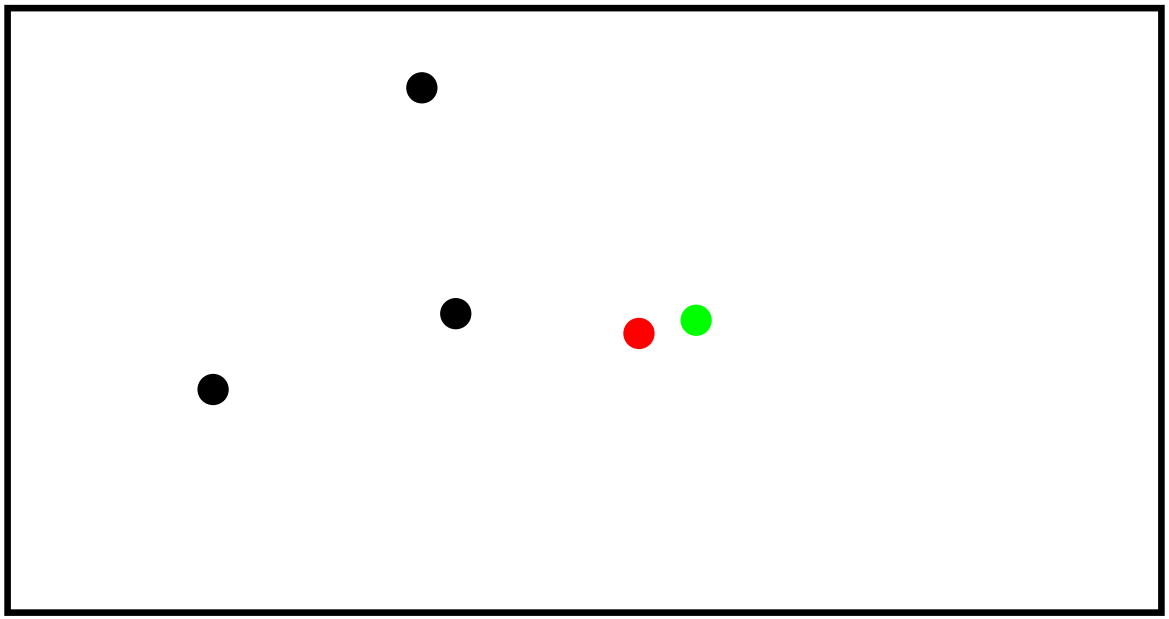
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

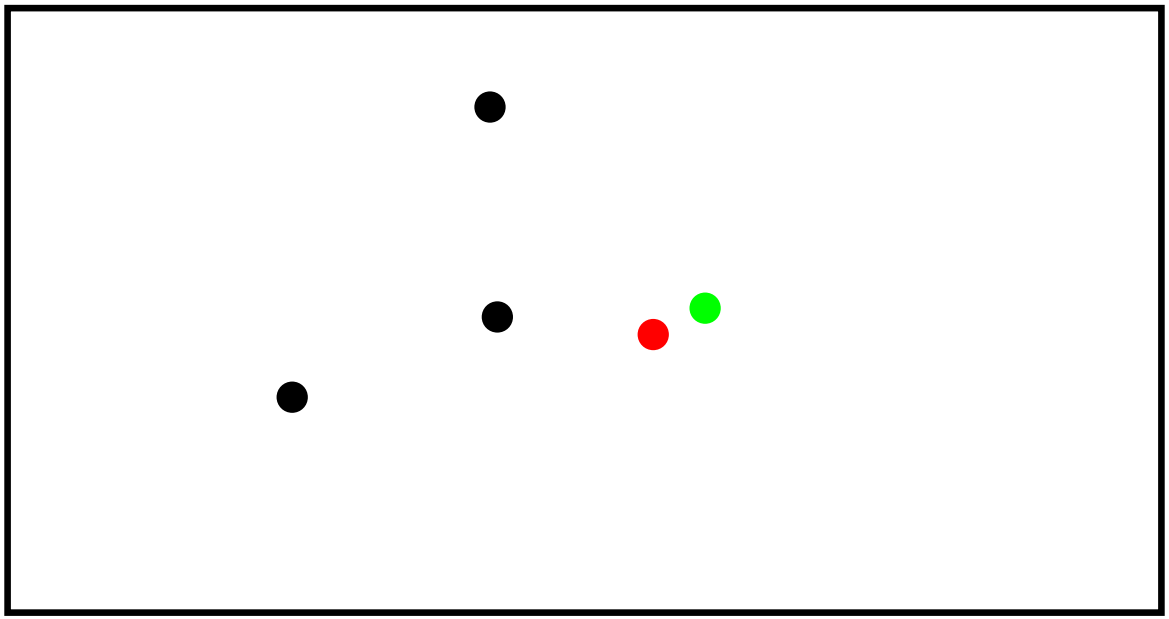
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

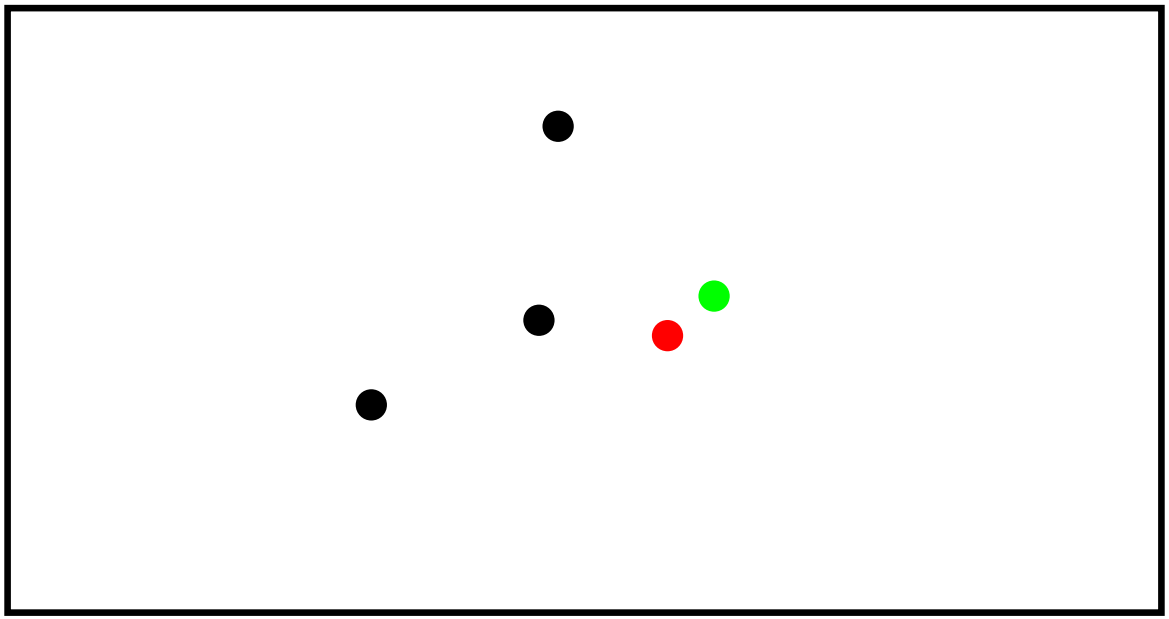
● ● → ● ●

● ● → ● ●

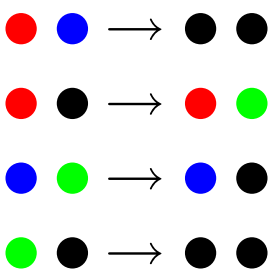
● ● → ● ●

● ● → ● ●

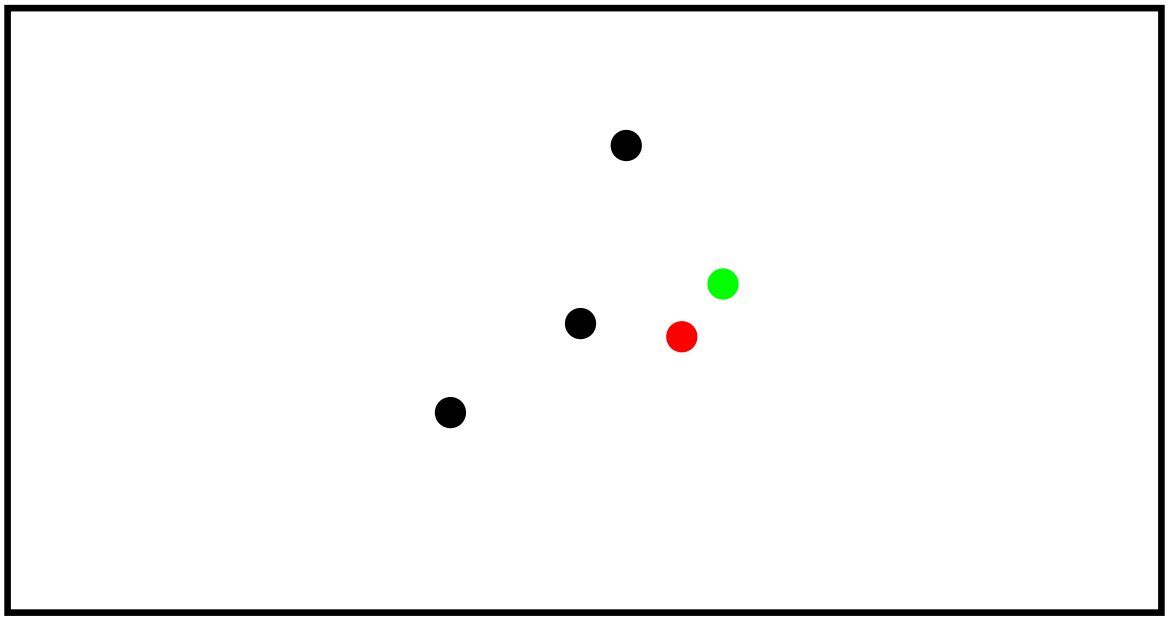
Exemple 2 : partant de 2 ●, 3●



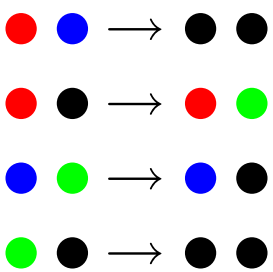
Program :



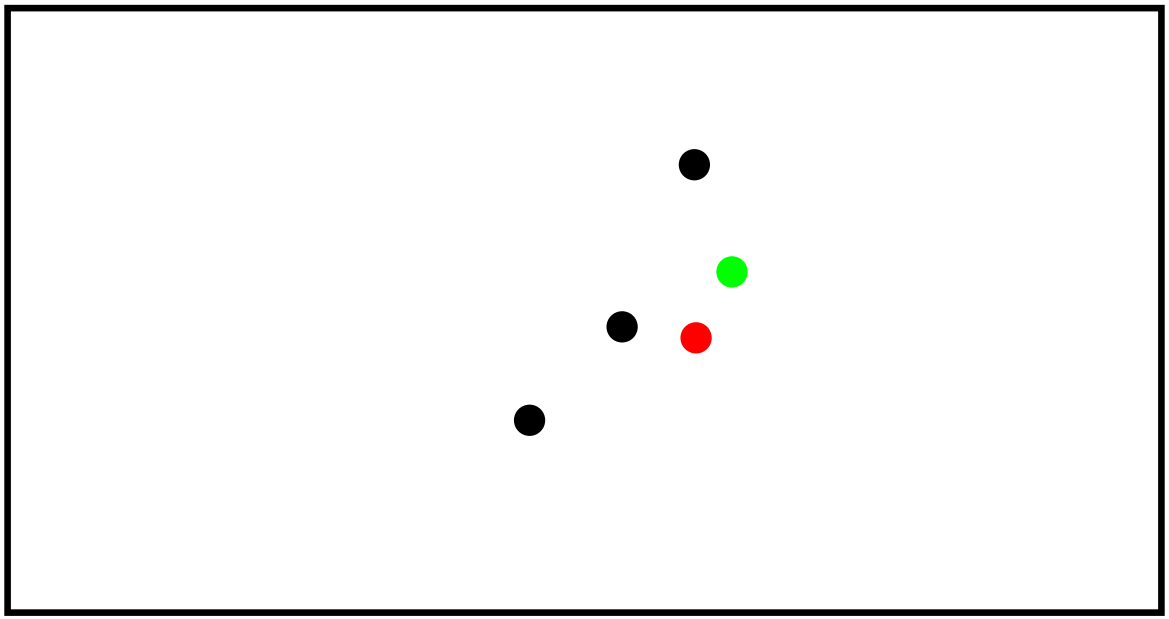
Exemple 2 : partant de 2 ●, 3●



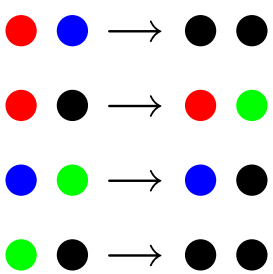
Program :



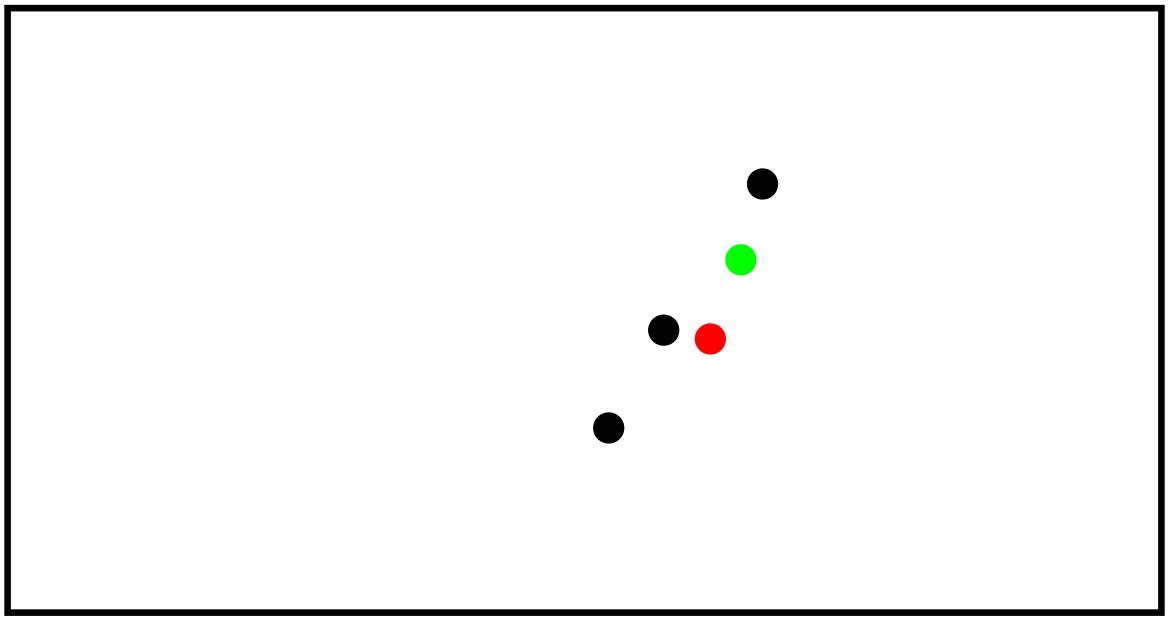
Exemple 2 : partant de 2 ●, 3●



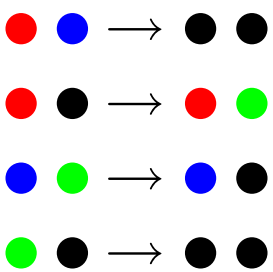
Program :



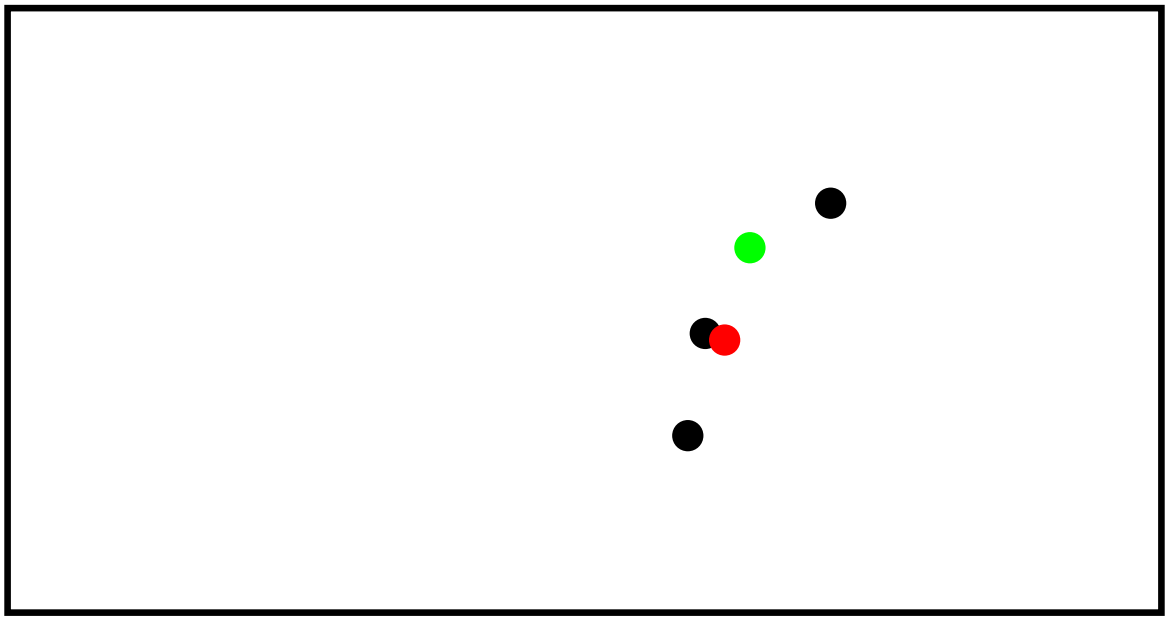
Exemple 2 : partant de 2 ●, 3●



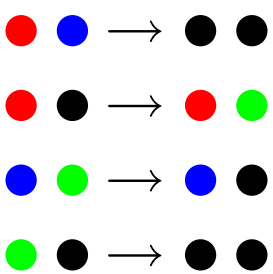
Program :



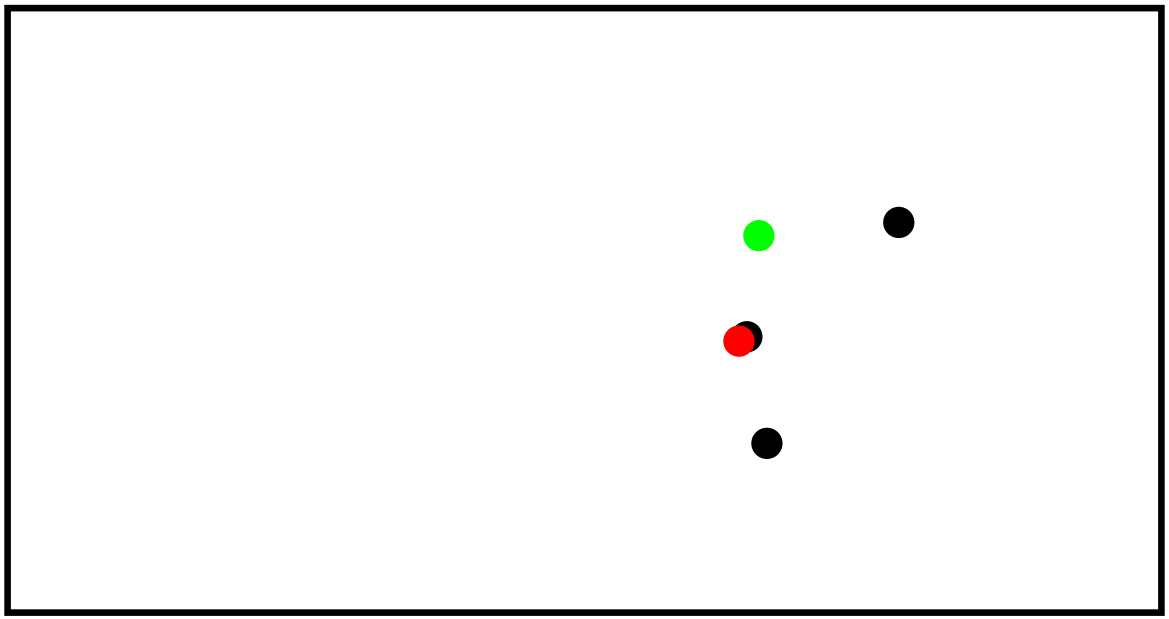
Exemple 2 : partant de 2 ●, 3●



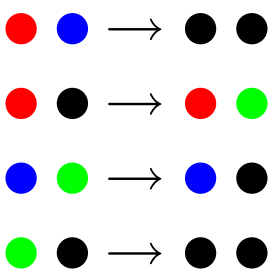
Program :



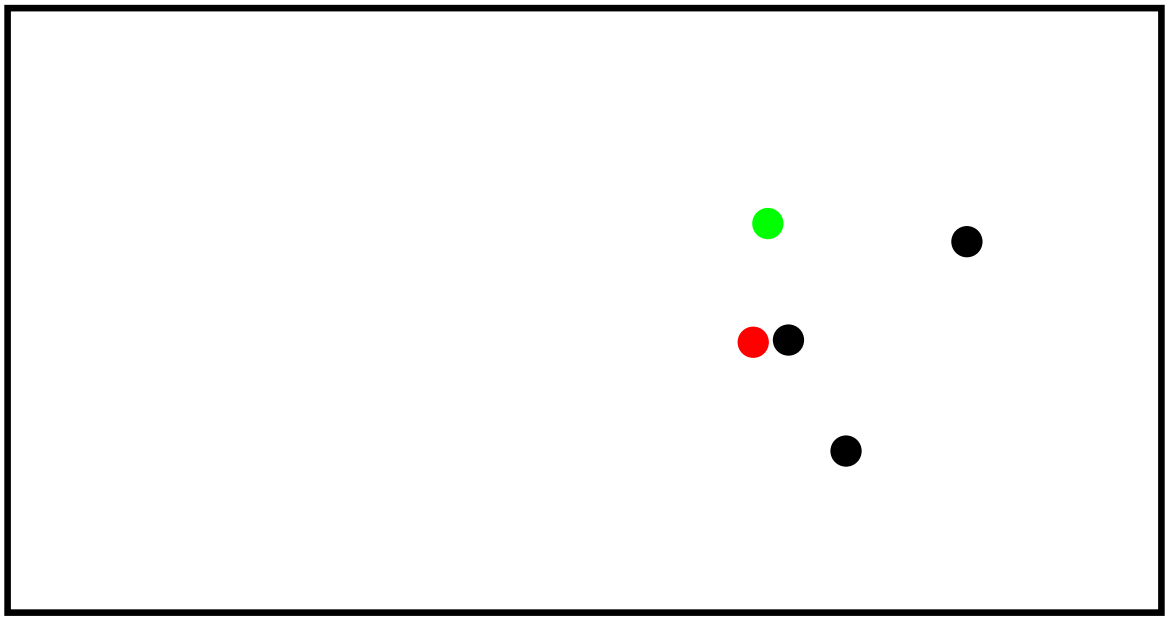
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

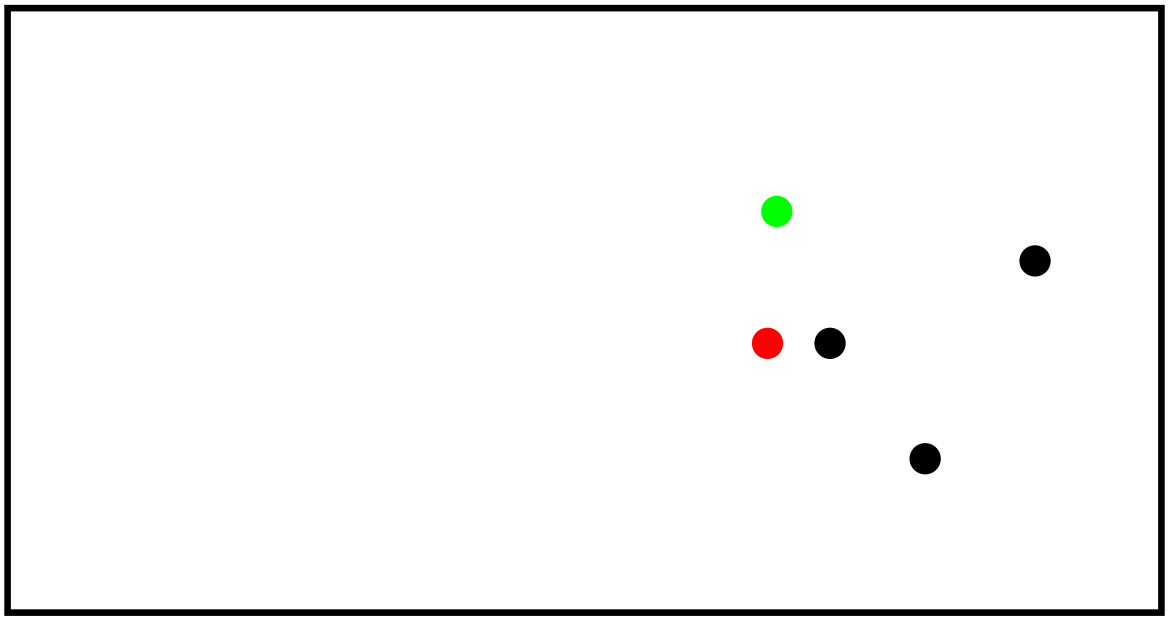
● ● → ● ●

● ● → ● ●

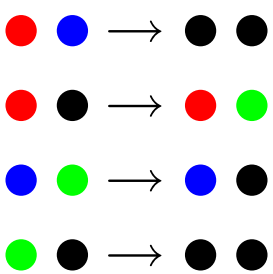
● ● → ● ●

● ● → ● ●

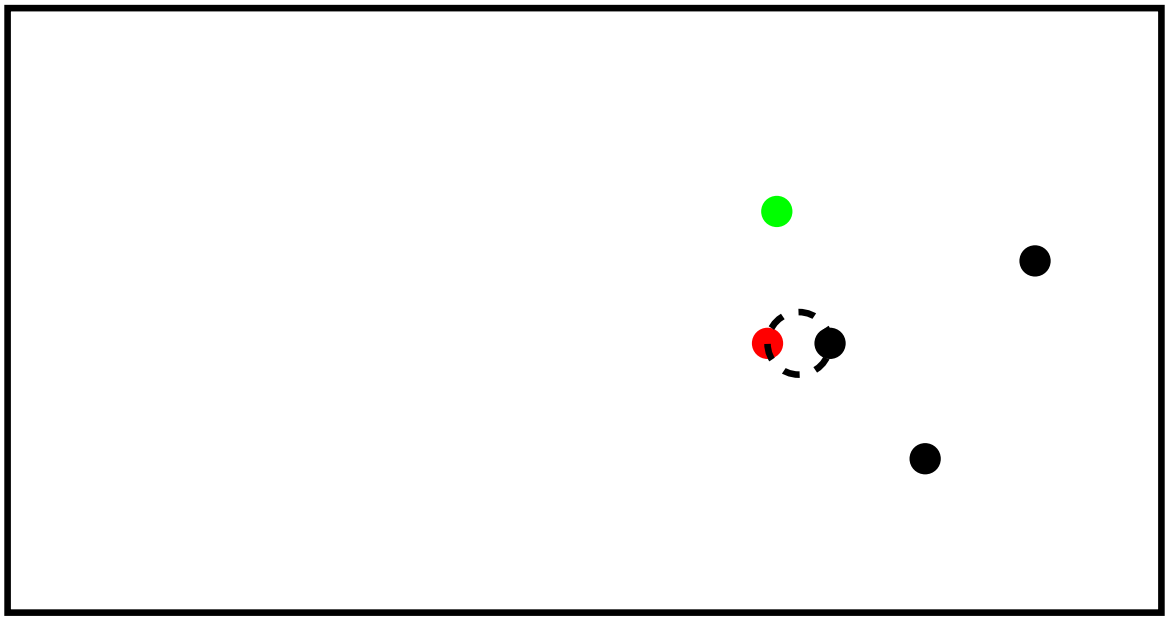
Exemple 2 : partant de 2 ●, 3●



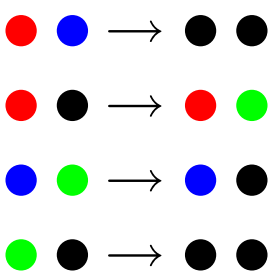
Program :



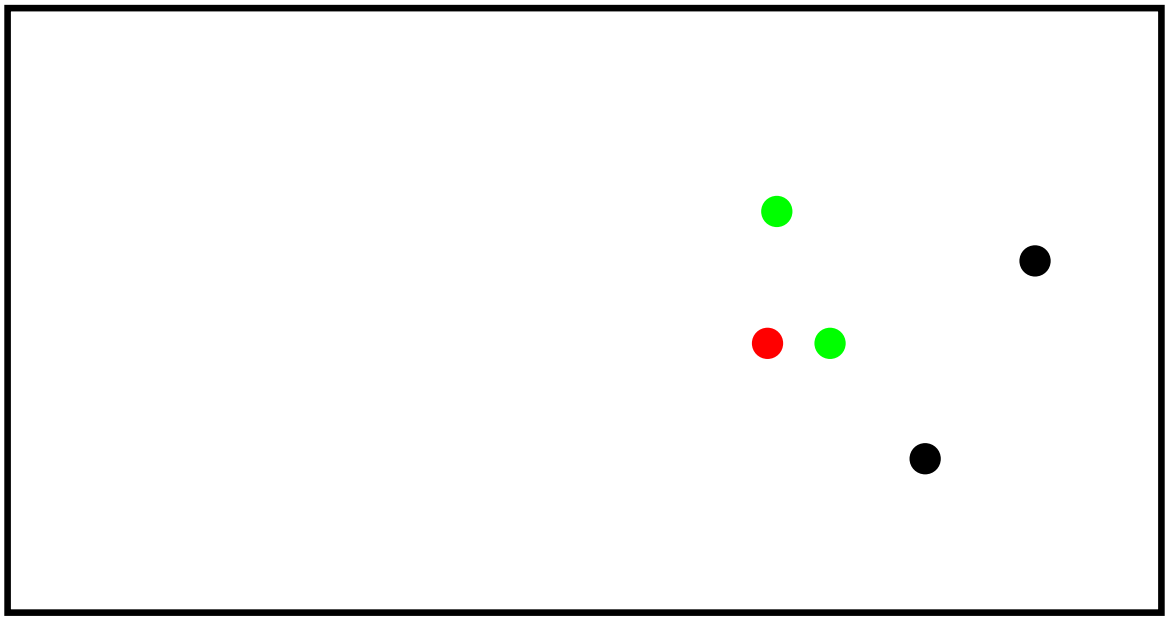
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

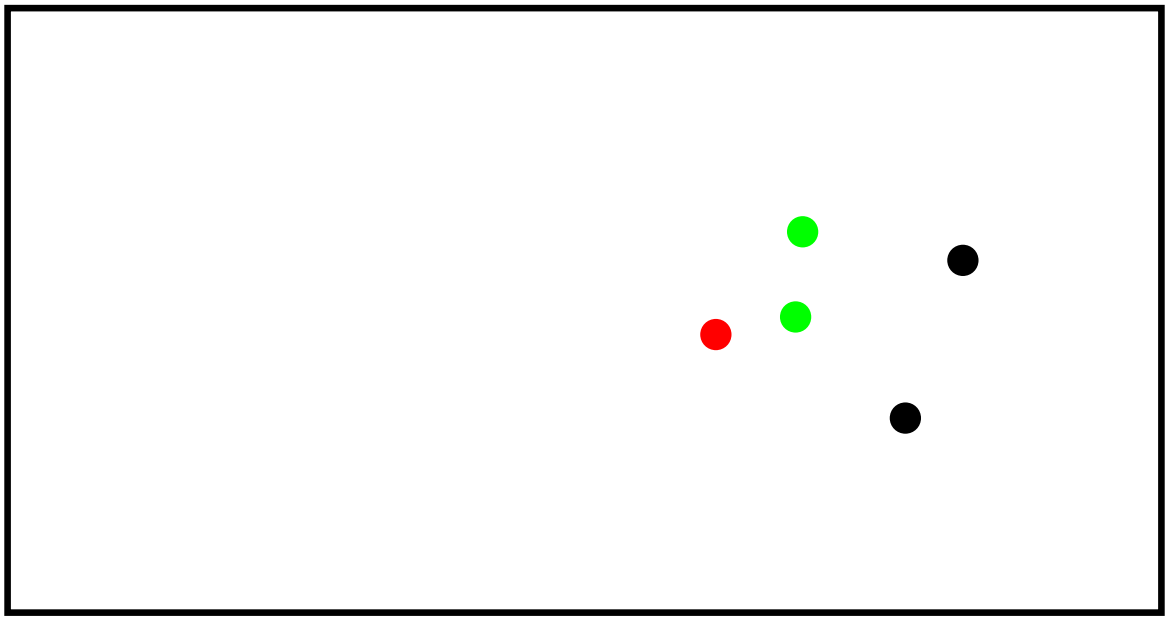
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

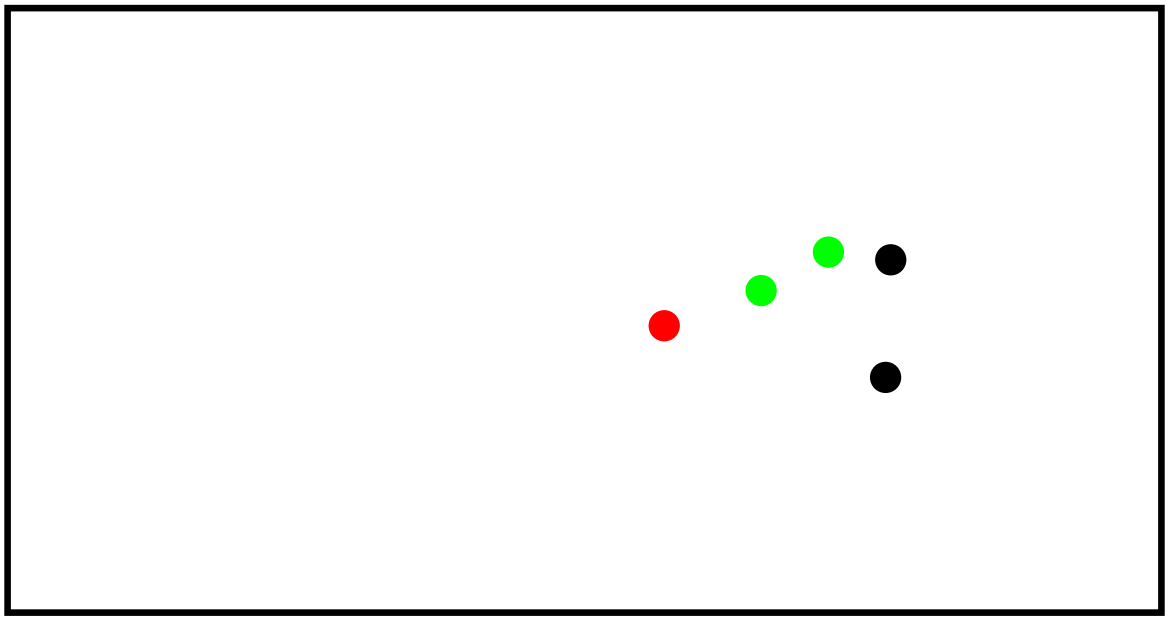
● ● → ● ●

● ● → ● ●

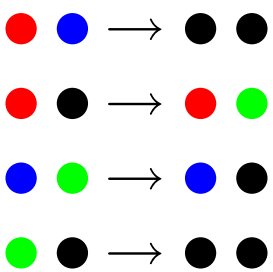
● ● → ● ●

● ● → ● ●

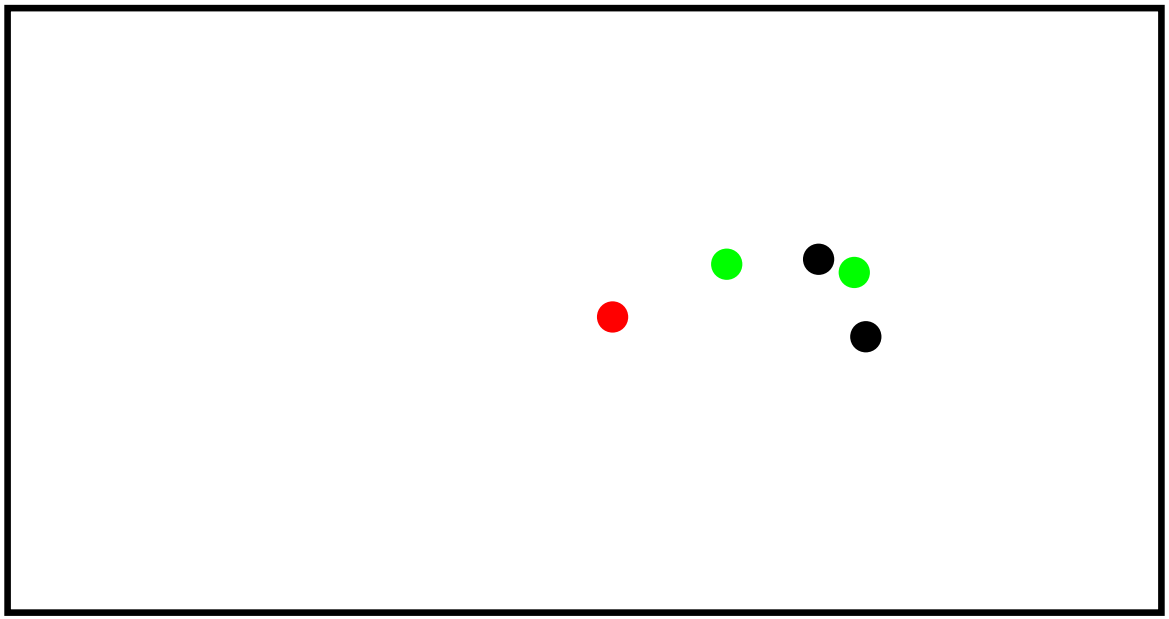
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

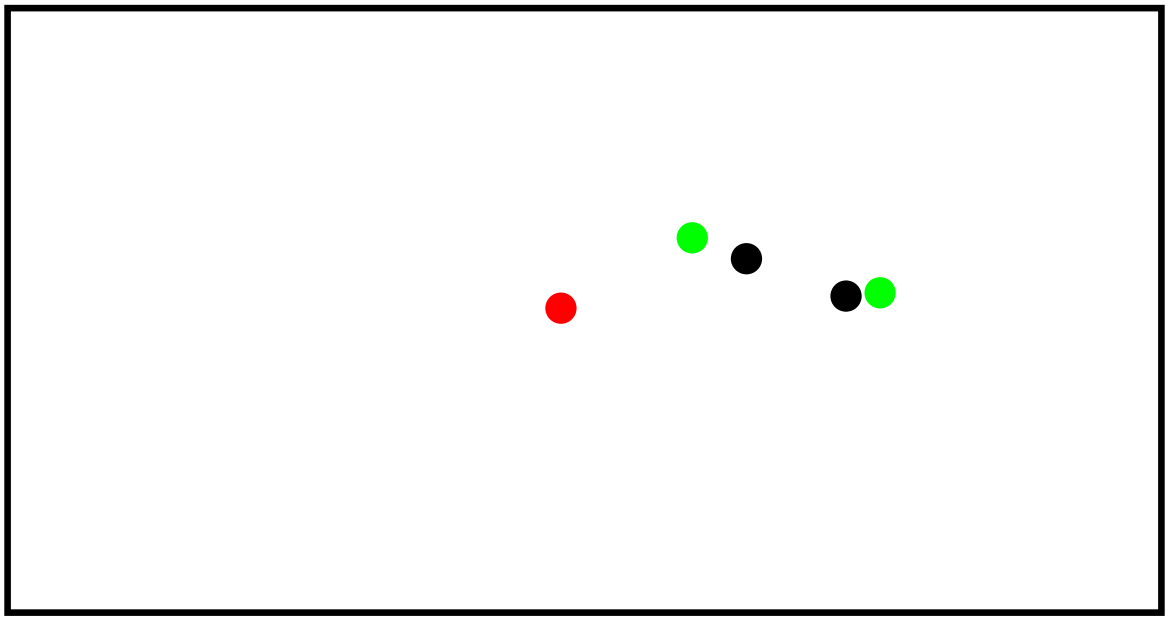
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

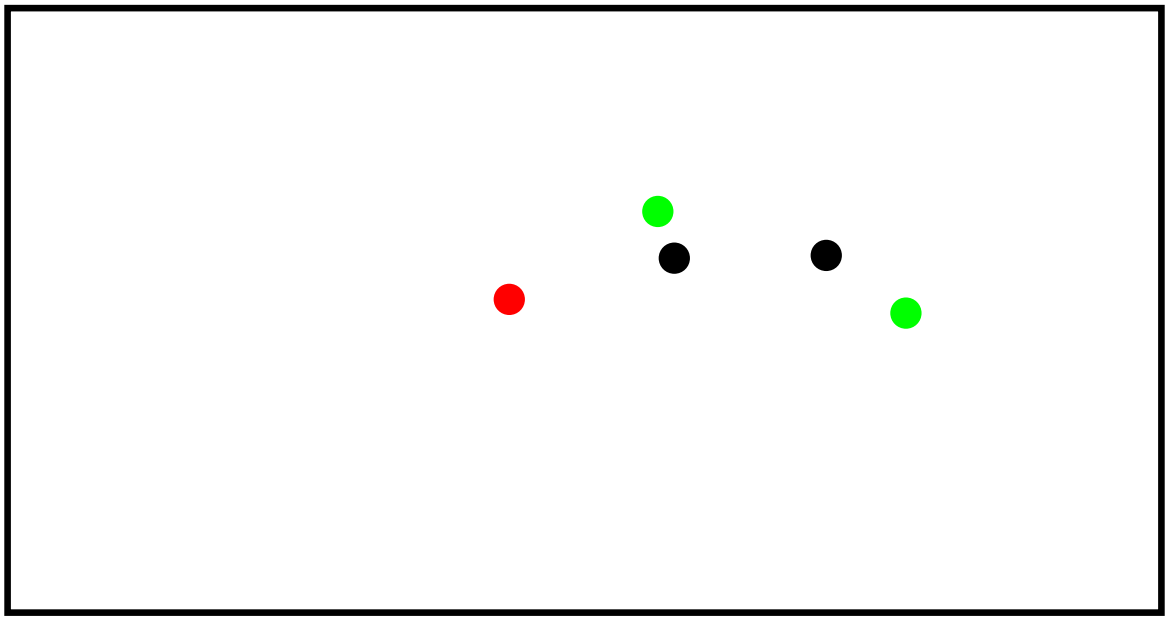
● ● → ● ●

● ● → ● ●

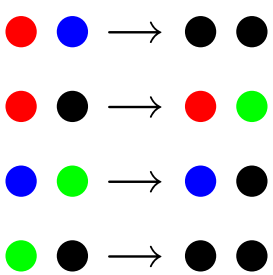
● ● → ● ●

● ● → ● ●

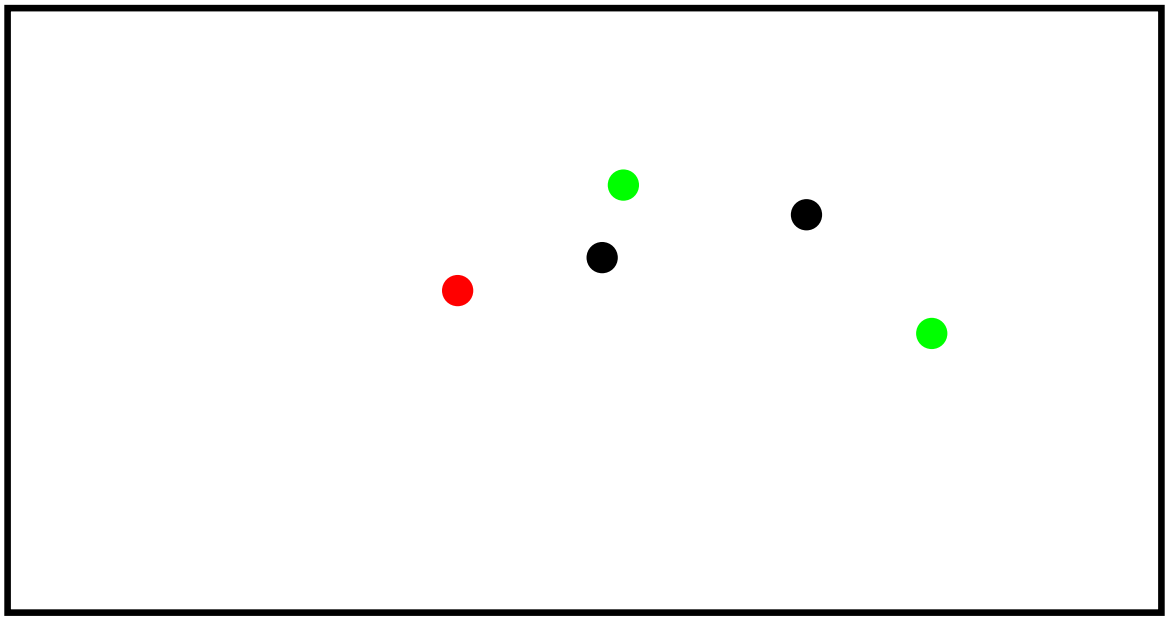
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

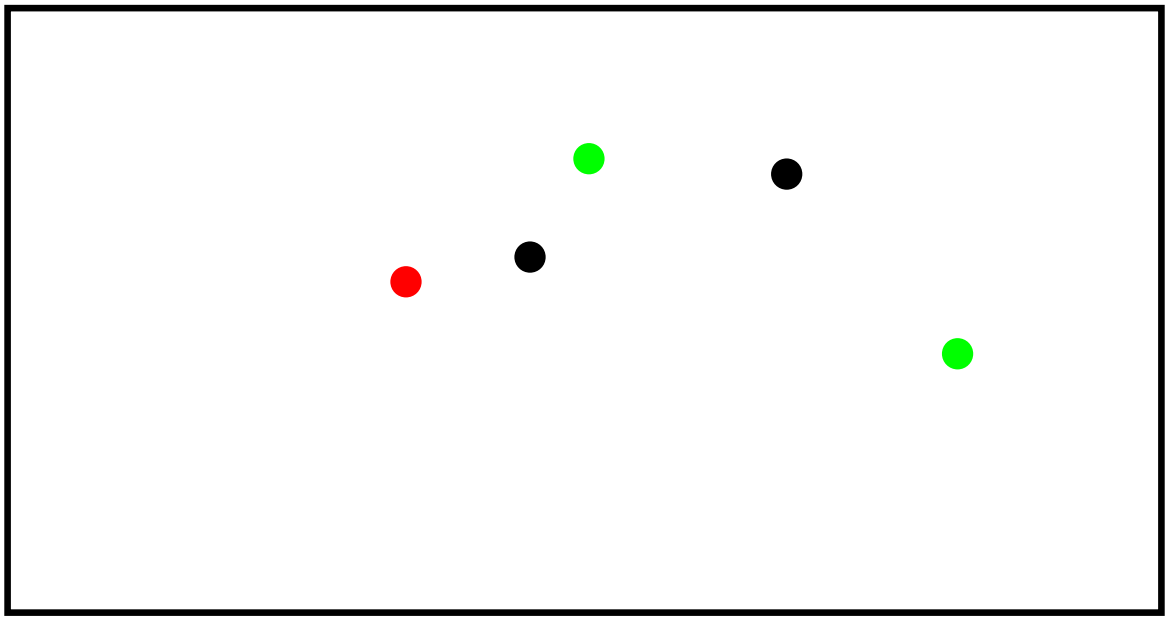
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

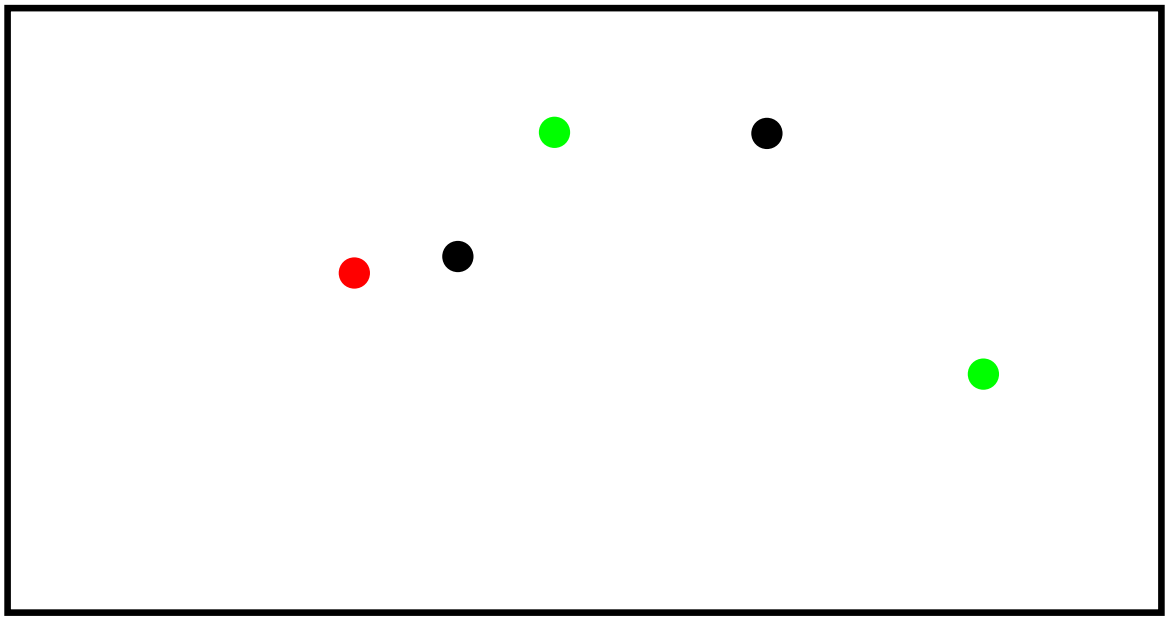
● ● → ● ●

● ● → ● ●

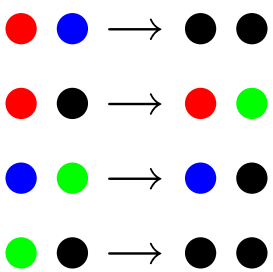
● ● → ● ●

● ● → ● ●

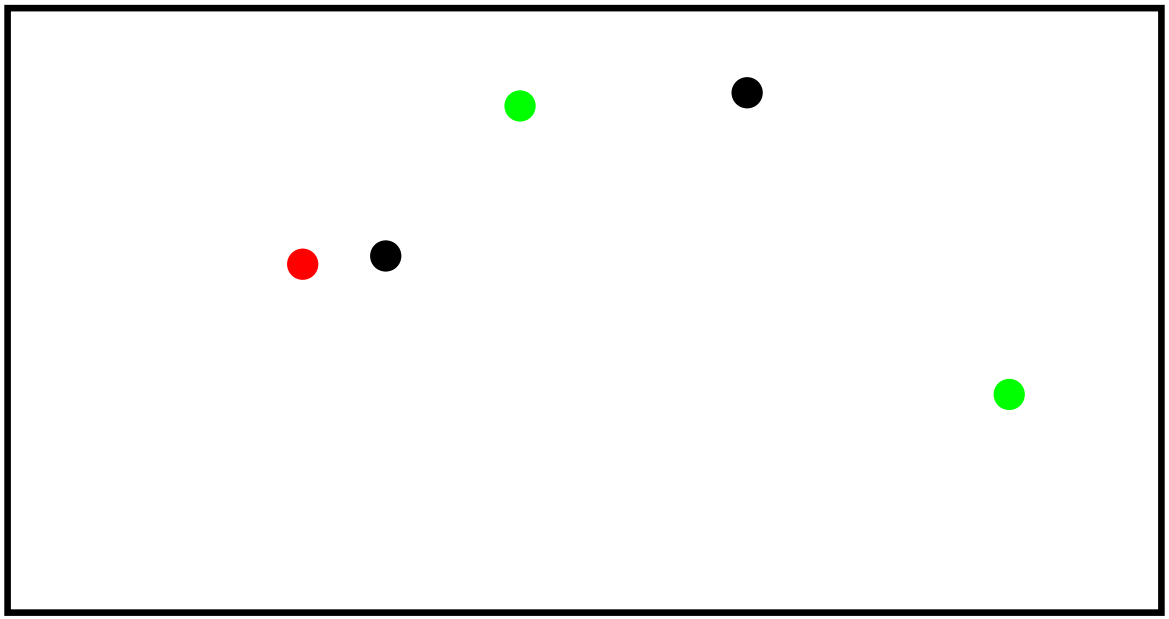
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

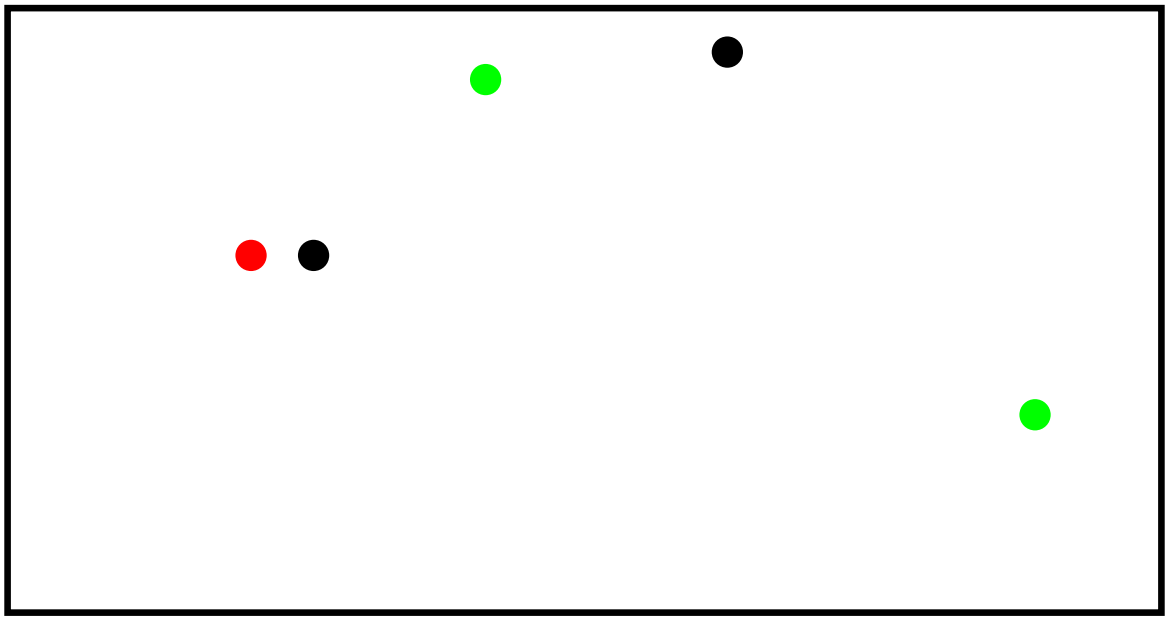
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

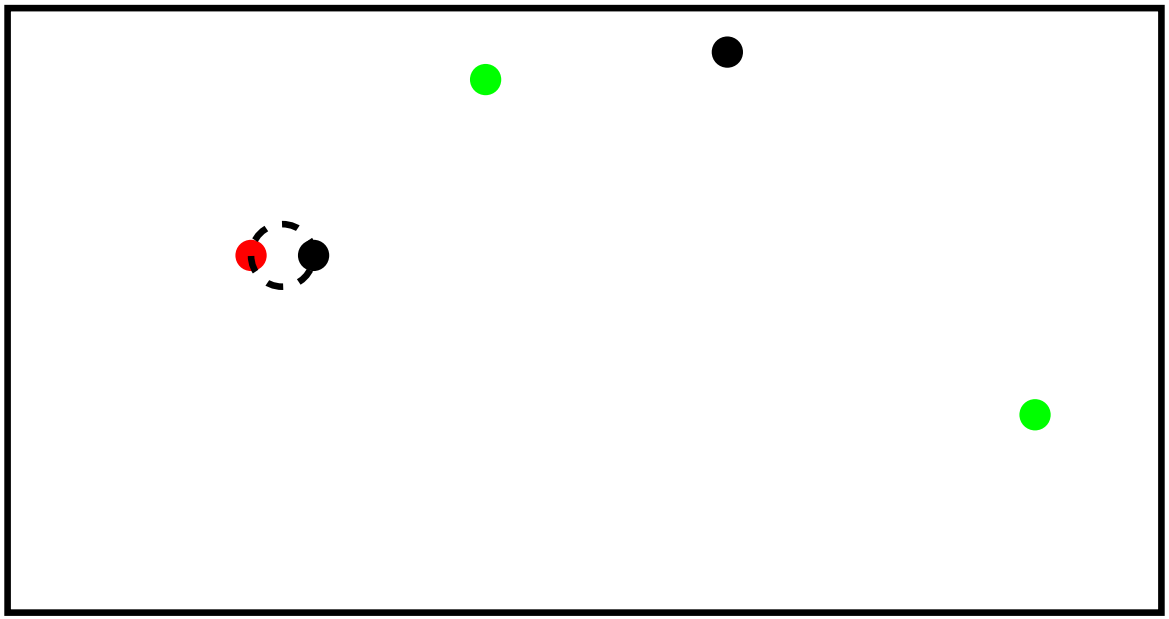
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

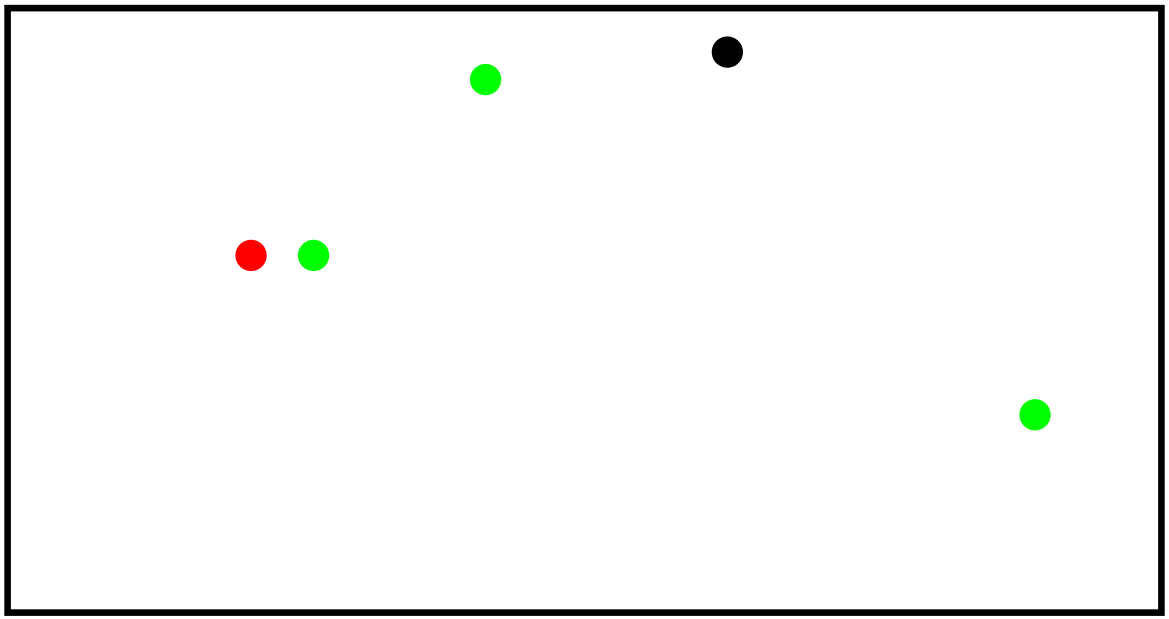
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

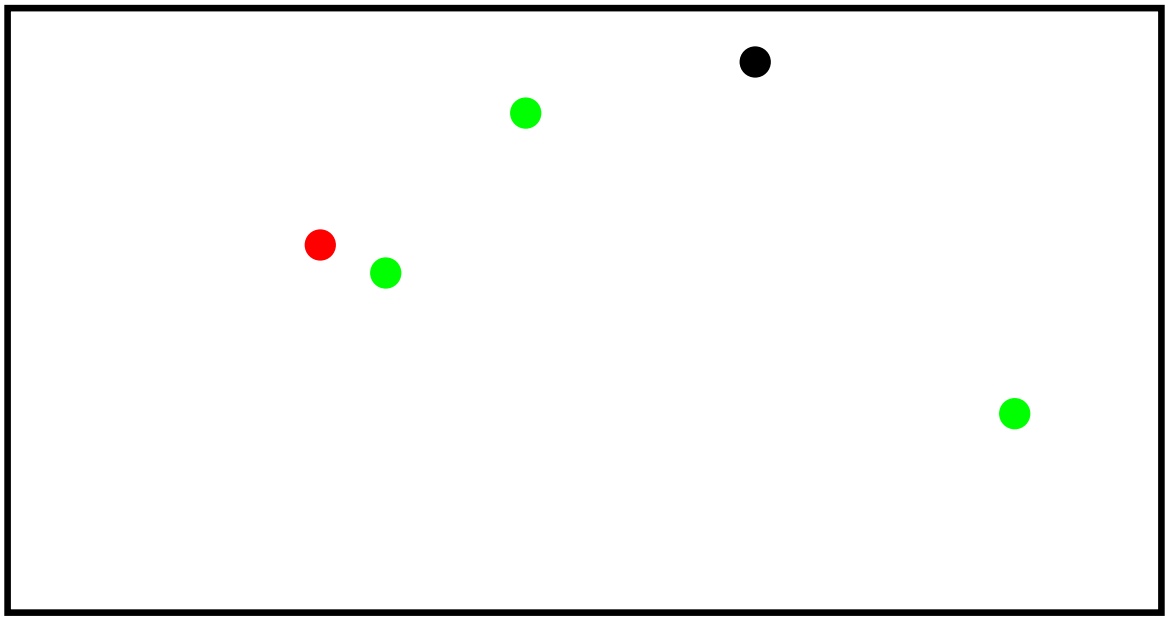
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

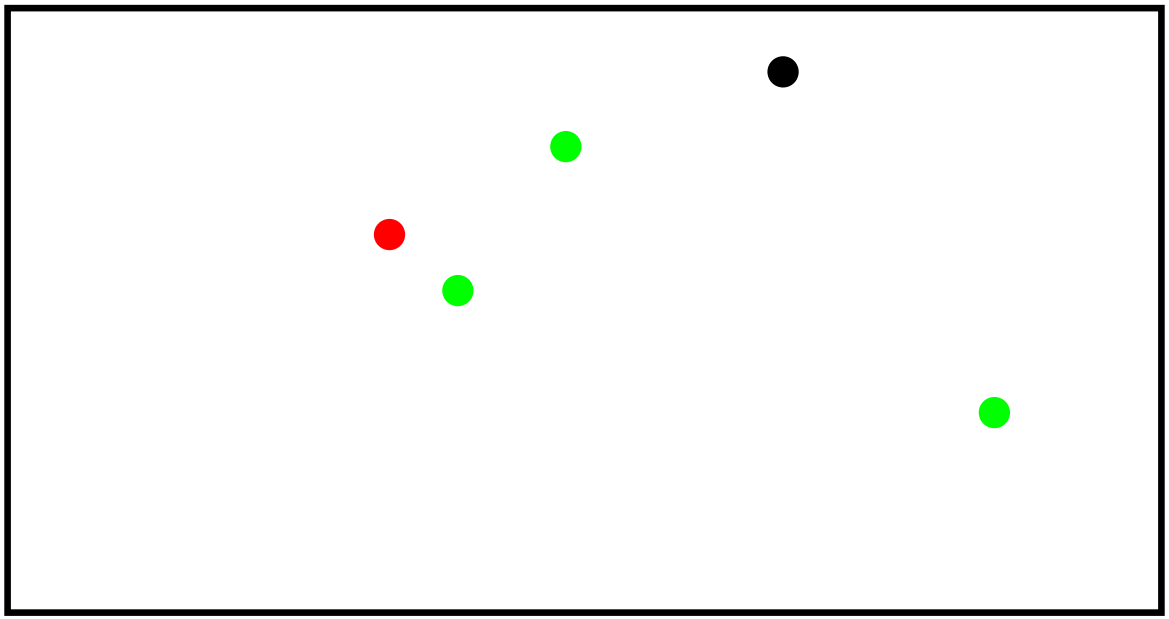
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

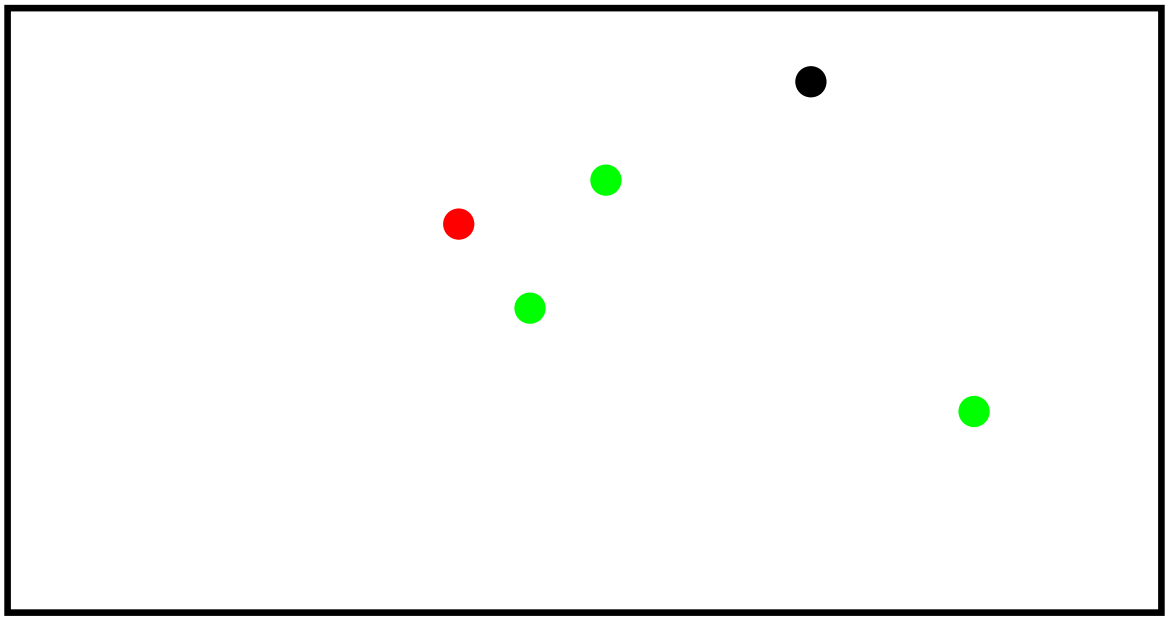
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

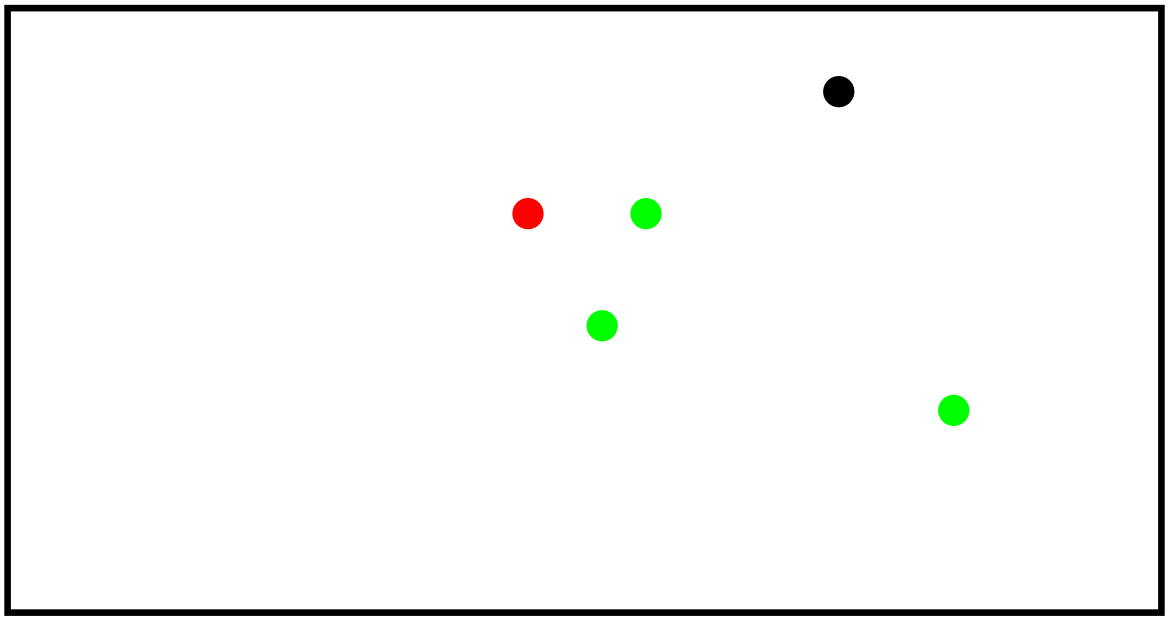
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

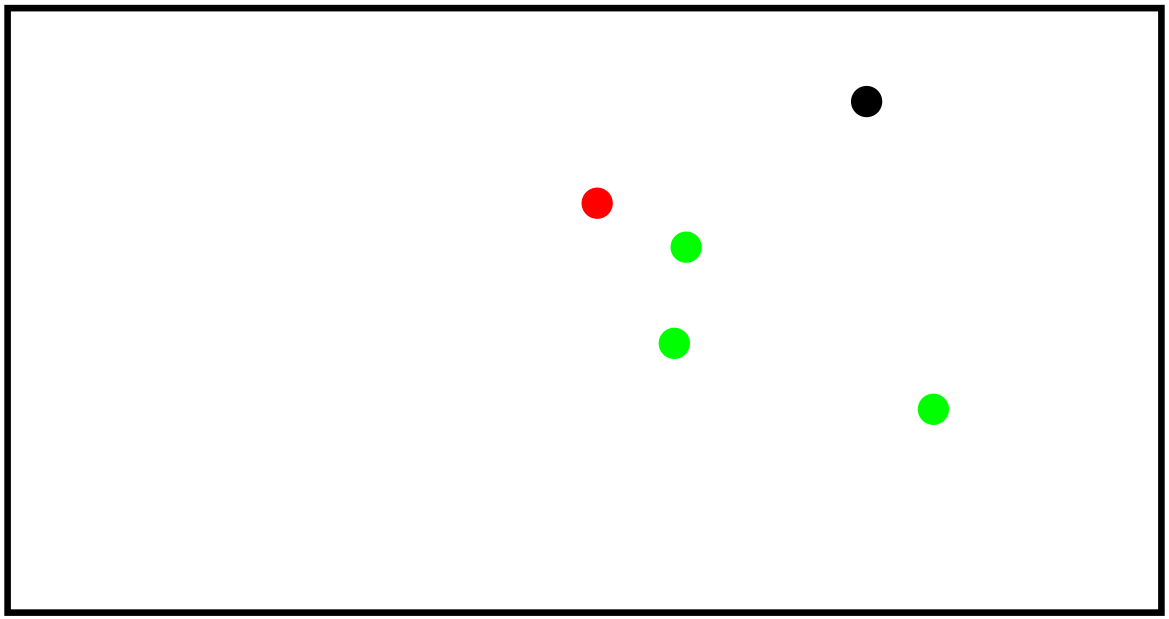
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

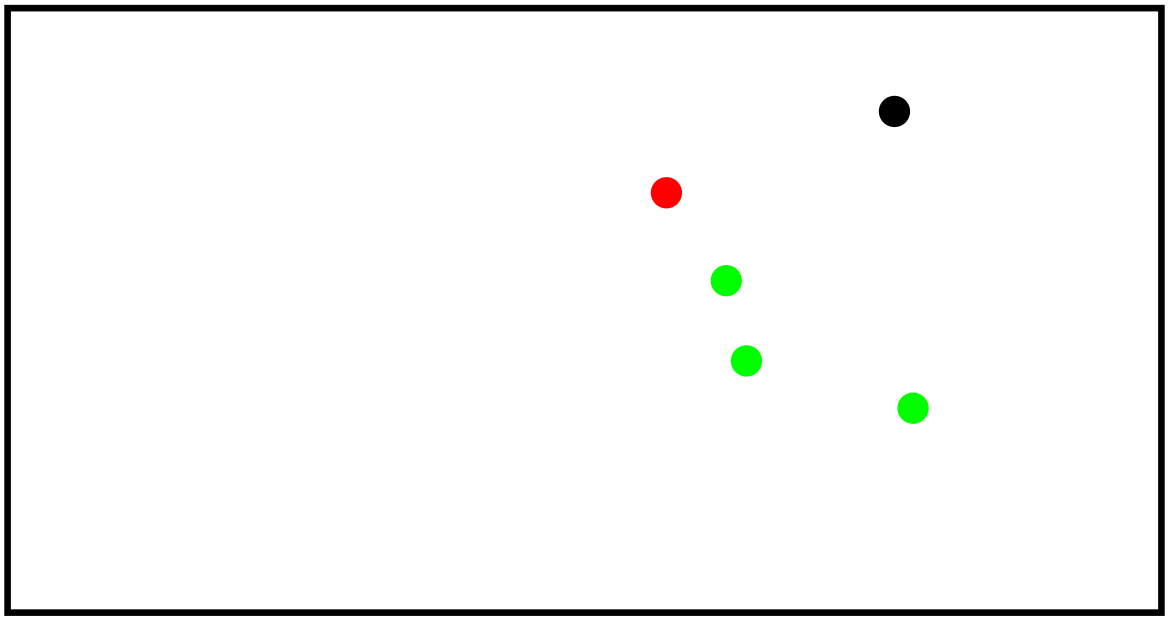
● ● → ● ●

● ● → ● ●

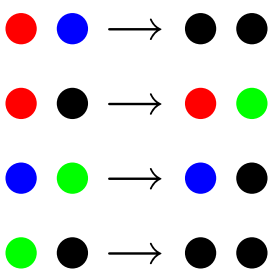
● ● → ● ●

● ● → ● ●

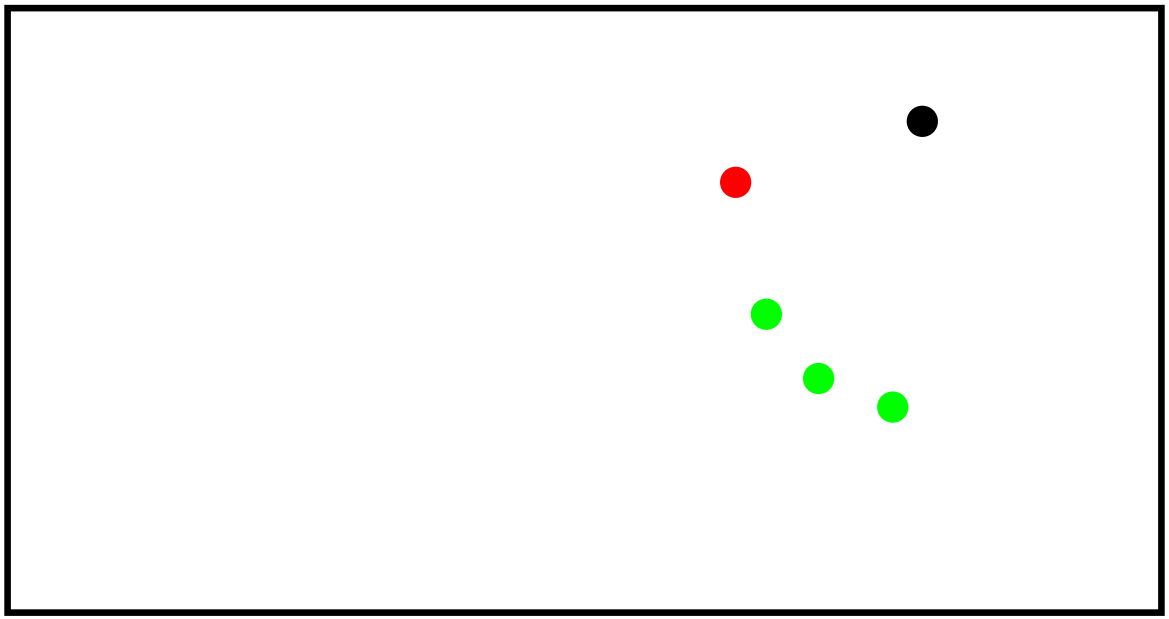
Exemple 2 : partant de 2 ●, 3●



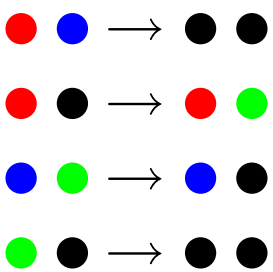
Program :



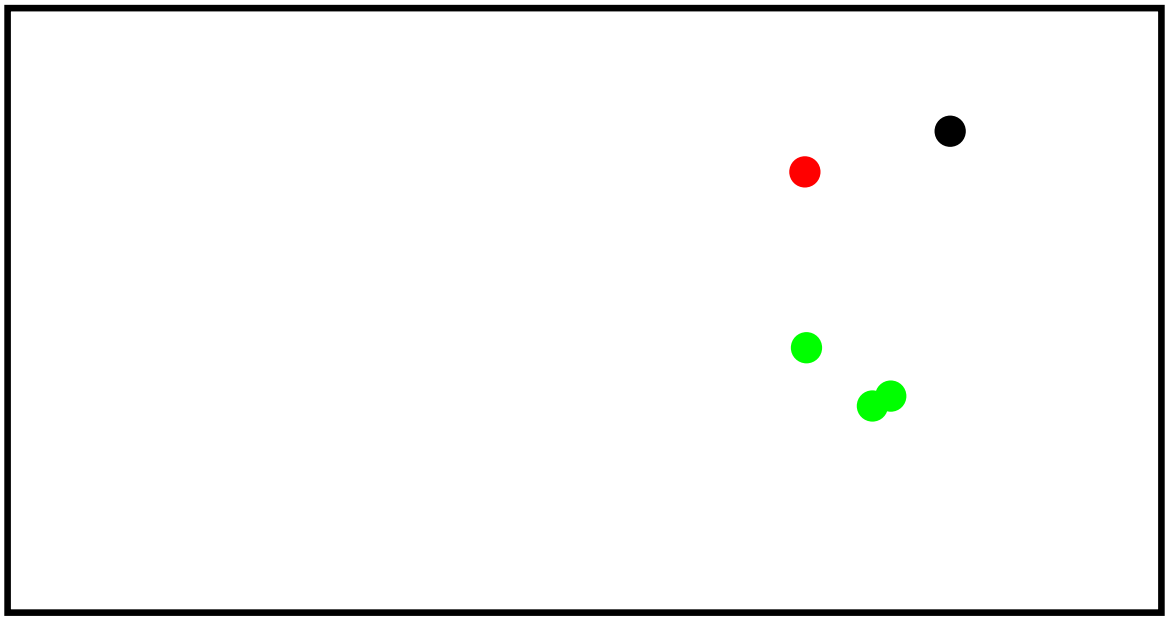
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

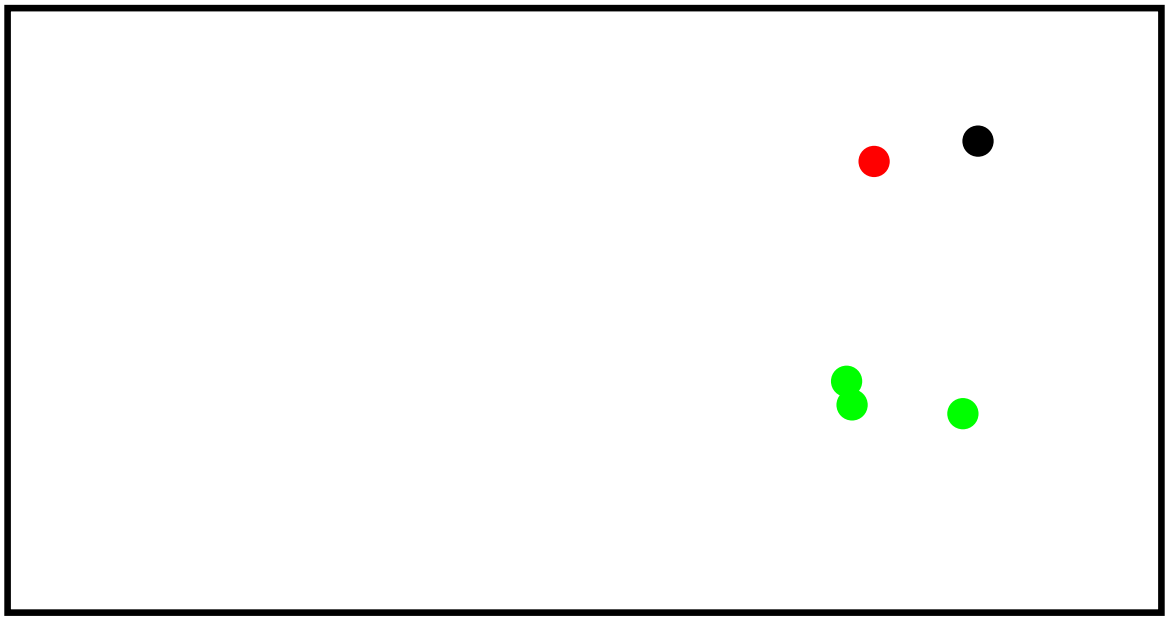
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

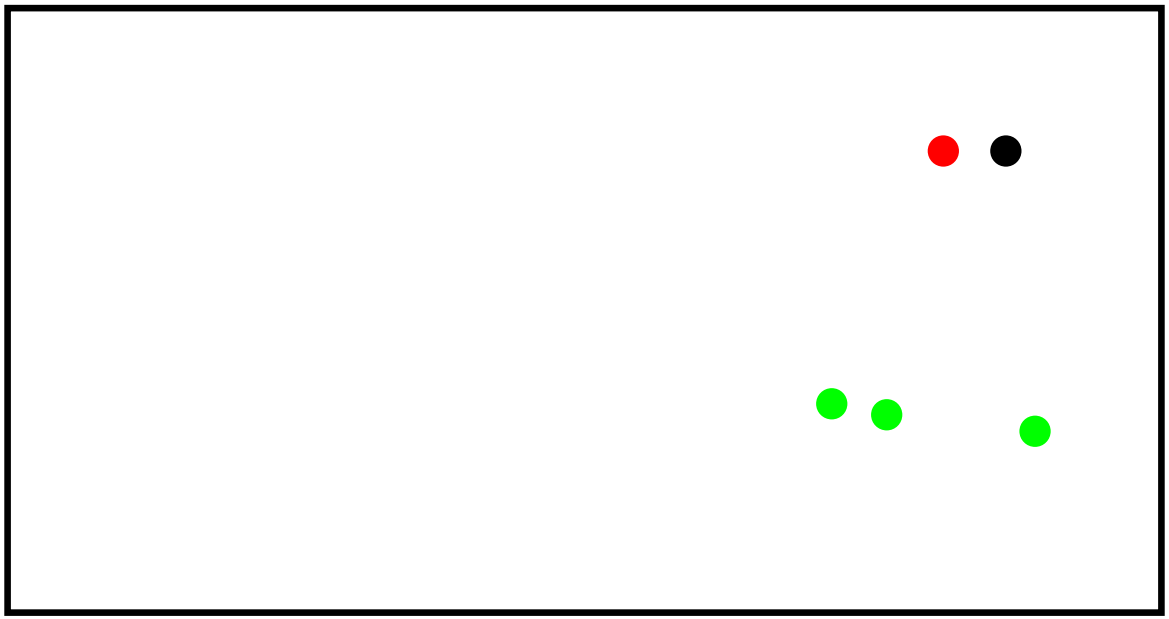
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 2 : partant de 2 ●, 3●



Program :

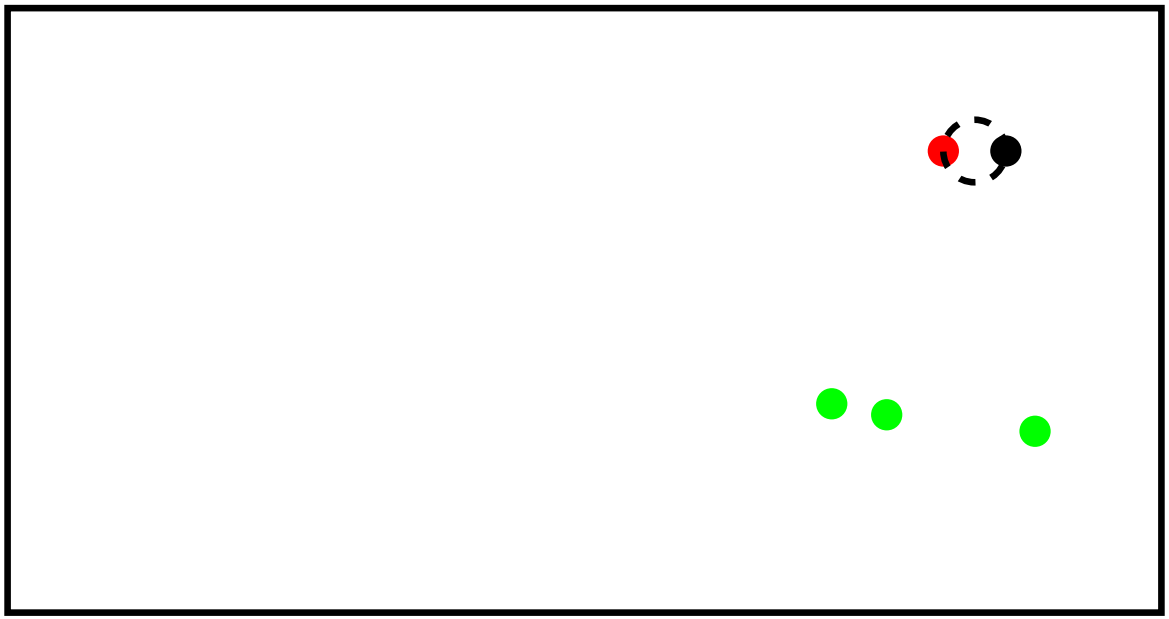
● ● → ● ●

● ● → ● ●

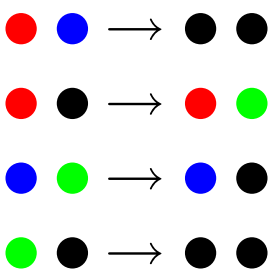
● ● → ● ●

● ● → ● ●

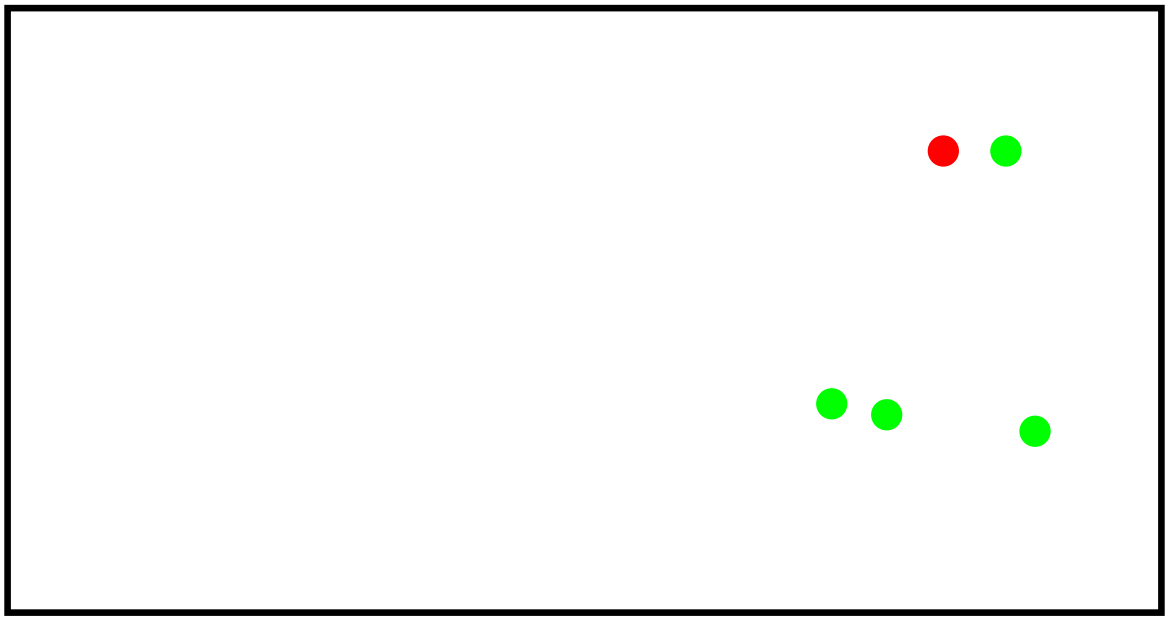
Exemple 2 : partant de 2 ●, 3●



Program :



Exemple 2 : partant de 2 ●, 3●



Program :

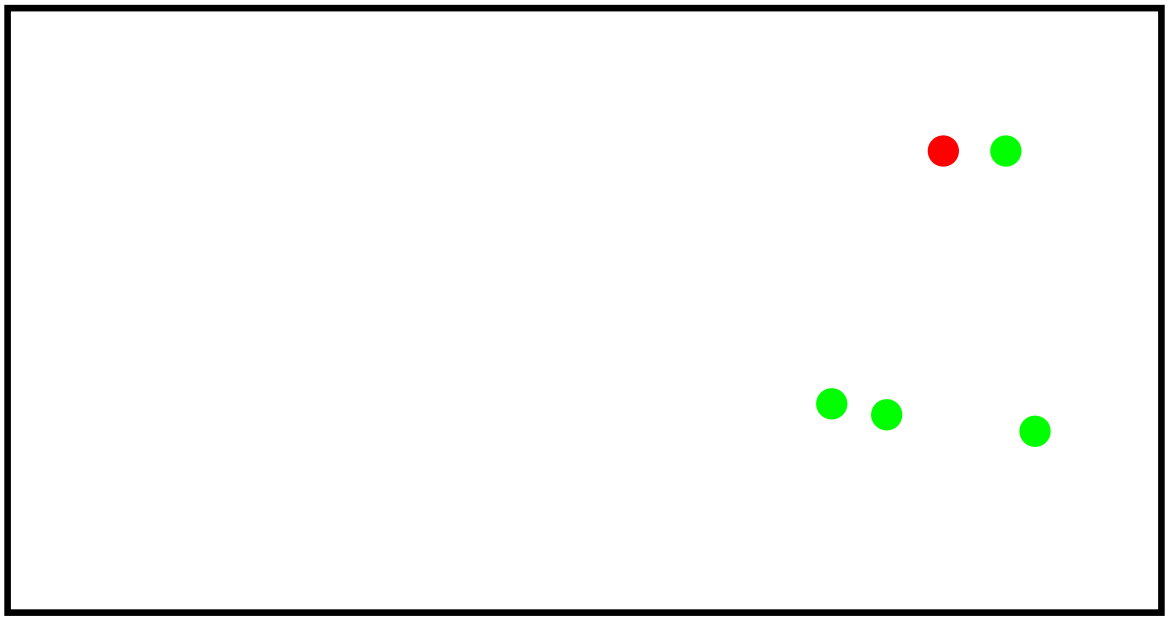
● ● → ● ●

● ● → ● ●

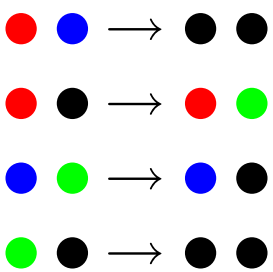
● ● → ● ●

● ● → ● ●

Example 2 : Résultat final



Program :



Qu'est ce qui est calculé ?

Interprétons

- ● et ● par *Oui*.
- ● et ● par *Non*.
- Quel que soit l'état initial, ultimement, tous les agents seront d'accord sur une valeur.
- Cette valeur sera *Oui* ssi la population initiale de ● est strictement plus grande que la population initiale de ●.

Formulation alternative

- Une configuration peut être vue comme un élément de $\mathbb{N}^4$.
  - (n_b, n_r, n_g, n_{bl}) s'il y a n_b ●, n_r ●, n_g ● et n_{bl} ●
  - $n_b + n_r + n_g + n_{bl}$ est un invariant.
- Une configuration initiale est de type $(n_b, n_r, 0, 0)$.
- Ce protocole calcule le prédicat $n_r > n_b$, c.à.d.

MAJORITE.

Cas général

- Un algorithme consiste en :
  - un ensemble fini d'états $Q = \{q_1, q_2, \cdots, q_k\}$.
  - des règles de transitions $\delta : Q \times Q \to Q \times Q$.
  - une interprétation des sorties $O : Q \to \{Oui, Non\}$.
- Une configuration instantanée est un multiensemble d'états de Q (un élément de $\mathbb{N}^k$).
- Exécutions :
  - Une exécution est une suite de configurations instantanées

    $$C_0 \vdash C_1 \vdash \ldots \vdash C_i \vdash C_{i+1} \vdash \ldots,$$

    où $C_i \vdash C_{i+1}$ signifie que deux agents sont mis à jour selon les règles de l'algorithme.
  - Une configuration instantanée C est acceptante lorsque tous les agents sont d'accord sur *Oui* ou sur *Non* : $O(q) = O(q')$ pour tout $q, q' \in C$.

Remarque : Les algorithmes sont indépendants de la taille de la population ! !

L'exemple le plus simple : Le OU logique des bits d'entrée

États : ●,●

Une règle de transition : ● ● → ● ●

La sortie d'un agent est son état.

Si tous les agents sont ●, tous les agents resteront ●
Si un agent est ●, ultimement tous les agents seront ●

Une remarque : il faut être équitable

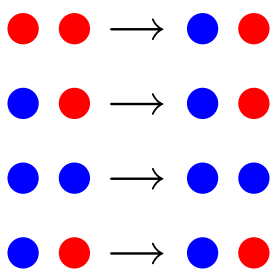
- Il faut garantir que toutes les interactions finissent par se produire.
- Par exemple, on peut supposer que l'on tire les paires d'agents selon une loi de probabilité i.i.d. uniforme.
- Approche de l'article initial :
  - une exécution est dite équitable si pour toute configuration C qui apparaît infiniment souvent dans l'exécution, si $C \vdash C'$ pour une configuration C', alors C' apparaît infiniment souvent dans l'exécution.
  - la définition d'un calcul : pour toute entrée I, pour toute exécution équitable partant de I, les agents seront ultimement dans une configuration acceptante.
- Cela ne change rien dans tout ce qui suit.

Élire un chef est facile

Au départ, tous les agents sont ●

Ultimement, un unique agent sera dans l'état ●

Programme :



Élire un chef est facile

Au départ, tous les agents sont ●

Ultimement, un unique agent sera dans l'état ●

Programme :

●● → ●●

Compter

- Exercice 1 : Compter jusqu'à 5.
  - Chaque agent est initialement ● ou ●
  - Déterminer s'il y a au moins 5 ●.

Devoir à la maison

- Exercice 2 : 5%
  - Chaque agent est initialement ● ou ●
  - Déterminer si au moins 5% des agents sont ●

Devoir à la maison

- Exercice 2 : 5%
  - Chaque agent est initialement 🔵 ou 🔴
  - Déterminer si au moins 5% des agents sont 🔴
    - Comme la majorité, sauf que chaque 🔴 annule 19 🔵.

Devoir à la maison

- Exercice 2 : 5%
  - Chaque agent est initialement 🔵 ou 🔴
  - Déterminer si au moins 5% des agents sont 🔴
    - Comme la majorité, sauf que chaque 🔴 annule 19 🔵.
- Exercice 3 : 40% ?

Devoir à la maison

- Exercice 2 : 5%
  - Chaque agent est initialement ● ou ●
  - Déterminer si au moins 5% des agents sont ●
    - Comme la majorité, sauf que chaque ● annule 19 ●.
- Exercice 3 : 40% ?
- Exercice 4 : Déterminer si $\sum_{i=1}^{k} c_i x_i \geq a$?

Déterminer si $\sum_{i=1}^{k} c_i x_i \geq a$

- Convention d'entrée : chaque agent avec son symbole d'entrée part dans l'état c_i.
- Chaque agent à un **bit de chef** qui évolue selon ● ● → ● ●.
- Chaque agent stocke une valeur dans
  $-m, -m+1, ..., m-1, m$, avec

$$m = \max(|a|+1, |c_1|, ..., |c_k|).$$

- Si un chef rencontre un non-chef, leurs valeurs changent comme suit :

$$\begin{array}{ll} x, y \rightarrow x+y, 0 & \text{if } 0 \leq x+y \leq m \\ x, y \rightarrow m, x+y-m & \text{if } x+y > m \\ x, y \rightarrow -m, x+y+m & \text{if } x+y < -m \end{array}$$

(Dans chaque cas, le premier agent dans le membre droit est le chef)
Chaque agent se rappelle la sortie du dernier chef qu'il a vu.

Pourquoi cela marche ?

- La somme des valeurs des agents est un invariant.
- La somme est ultimement concentrée dans l'unique chef (jusqu'à la valeur absolue de m).
- Si
  - $somme > m$, l'unique chef aura la valeur m $\Rightarrow$ Répondre *Oui*.
  - $somme < -m$, l'unique chef aura la valeur -m $\Rightarrow$ Répondre *Non*.
  - $-m \leq somme \leq m$, l'unique chef aura la vraie valeur $\Rightarrow$ en fonction de celle-ci.
- Dans chaque cas, le chef connaît la réponse et la diffuse à tout le monde.

Prédicats calculables

Theorem (Angluin,Aspnes,Eisenstat,Ruppert 2006)

Un prédicat est calculable ssi il est dans la liste suivante

- $\sum_{i=1}^{k} c_i x_i \geq a$, *où* a *et les* c_i *sont des constantes entières.*
- $\sum_{i=1}^{k} c_i x_i \equiv a \pmod{b}$ *où* a, b *et les* c_i *sont des constantes entières.*
- *Les combinaisons booléennes des prédicats précédents.*

Caractérisation alternative : Arithmétique de Presburger

Un prédicat est calculable ssi il peut se définir en logique du premier ordre en arithmétique de Presburger[2].

(plus vulgairement : avec les symboles
$+, 0, 1, \vee, \wedge, \neg, \forall, \exists, =, <, (,)$ et des variables)
(pas de multiplication).

Exemples :

- majorité : $x_0 < x_1$
- divisible par 3 : $\exists y : y + y + y = x_1$
- au moins 40% :$x_0 + x_0 < x_1 + x_1 + x_1$

[2] [Presburger 1929]

Variantes du modèle

Le modèle de base est maintenant assez bien compris.

Variantes considérées dans la littérature :

- Des graphes d'interactions limitées.
- Communications "One-way".
- Pannes.

Quelques noms : Dana Angluin, James Aspnes, Melody Chan, Carole Delporte-Gallet, Zoë Diamadi, David Eisenstat, Michael J. Fischer, Hugues Fauconnier, Rachid Guerraoui, Hong Jiang, René Peralta, Eric Ruppert ...

Survol : [Aspnes,Ruppert 2007]

Autres remarques

- Tout protocole de population peut être simulé par un protocole de population symétrique[3], dès que la population est de taille au moins trois.
- Autoriser du non-déterminisme n'augmente pas la puissance des protocoles de population.
- Autoriser des membres droits probabilistes n'augmente pas pas la puissance des protocoles de population.

[Angluin et al. 2004,2006].

[3] symétrique = si l'on a la règle $q_1, q_2 \longrightarrow q_1', q_2'$, alors on a la règle $q_2, q_1 \longrightarrow q_2', q_1'$.

Plan

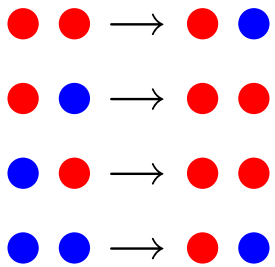
Les protocoles de population

Protocoles de grande population : modèle & convergence

Protocoles de grande population : puissance ?

Mon exemple préféré

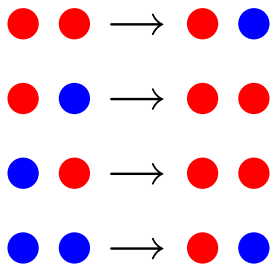
Le protocole suivant[4]



[4] [B., Cohen, Koegler 06]

Mon exemple préféré

Le protocole suivant[4]

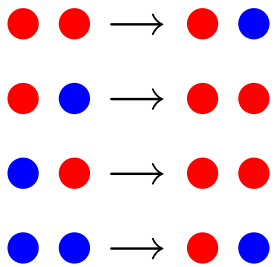


- ne calcule rien au sens de ce qui précède :

[4] [B., Cohen, Koegler 06]

Mon exemple préféré

Le protocole suivant[4]

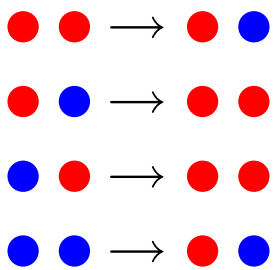


- ne calcule rien au sens de ce qui précède :
  - la configuration ● ● $\cdots$ ● est nécessairement quittée dès la première étape.
  - tout autre configuration est atteignable à partir de toute autre configuration.

[4] [B., Cohen, Koegler 06]

Mon exemple préféré : suite

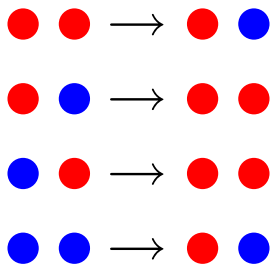
En supposant le nombre d'agents grand,



que peut-on dire de

$$p = \frac{\textit{nombre de } \bullet}{\textit{nombre de } \bullet + \textit{nombre de } \bullet} ?$$

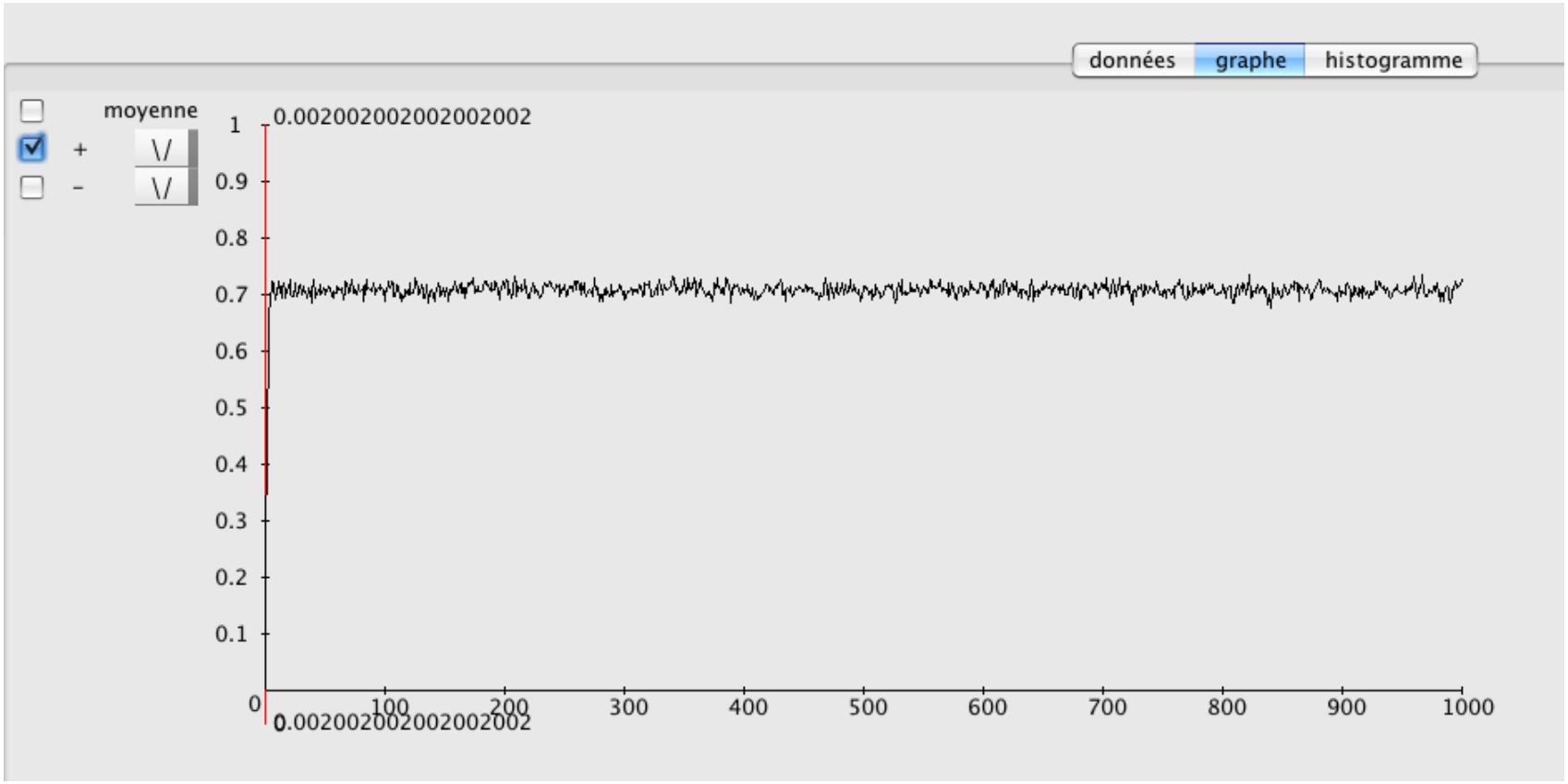
Mon exemple préféré : les hypothèses



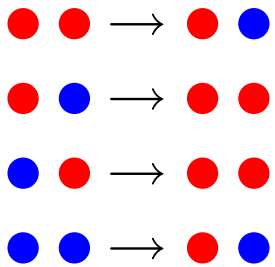
1. le nombre d'agents est grand[5].
2. on suppose les paires d'agents tirées selon une loi i.i.d. uniforme.
3. on parle de proportions d'agents dans un état donné plutôt que de nombres.
4. on voit les protocoles comme calculant des fonctions plutôt que des prédicats.

[5] Protocoles de population & grandes populations : voir aussi [Chatzigiannakis, Spirakis 08]

Approche 1 : Simulation

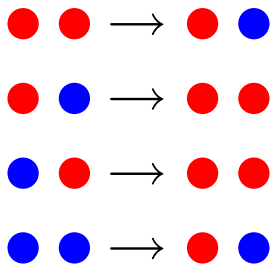


Approche 2 : Raisonnement heuristique



- Le nombre moyen de ● crée,

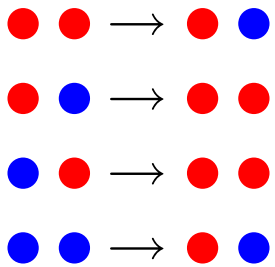
Approche 2 : Raisonnement heuristique



- Le nombre moyen de ● crée,

$$-1 * p^2 + 1 * p(1-p) + 1 * p(1-p) + 1 * (1-p)^2 = 1 - 2p^2$$

Approche 2 : Raisonnement heuristique

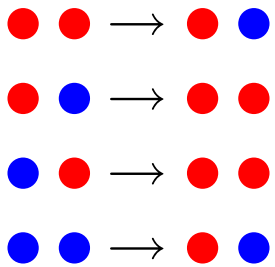


- Le nombre moyen de ● crée,

$$-1 * p^2 + 1 * p(1-p) + 1 * p(1-p) + 1 * (1-p)^2 = 1 - 2p^2$$

doit, à la limite si elle existe, être égal à 0.

Approche 2 : Raisonnement heuristique



- Le nombre moyen de ● crée,

$$-1 * p^2 + 1 * p(1-p) + 1 * p(1-p) + 1 * (1-p)^2 = 1 - 2p^2$$

doit, à la limite si elle existe, être égal à 0.

- Donc

$$p = \sqrt{\frac{1}{2}}.$$

Approche 3 : vers une vraie preuve.

Lorsque le nombre d'agents n est fixé,

- par le théorème ergodique, quelle que soit la distribution initiale, la distribution de ● va converger vers l'unique distribution stationnaire μ_n de la chaîne de Markov.
- L'espérance E_n de p sur μ_n est un nombre rationnel (calculable).

On souhaite parler de la limite de E_n ...

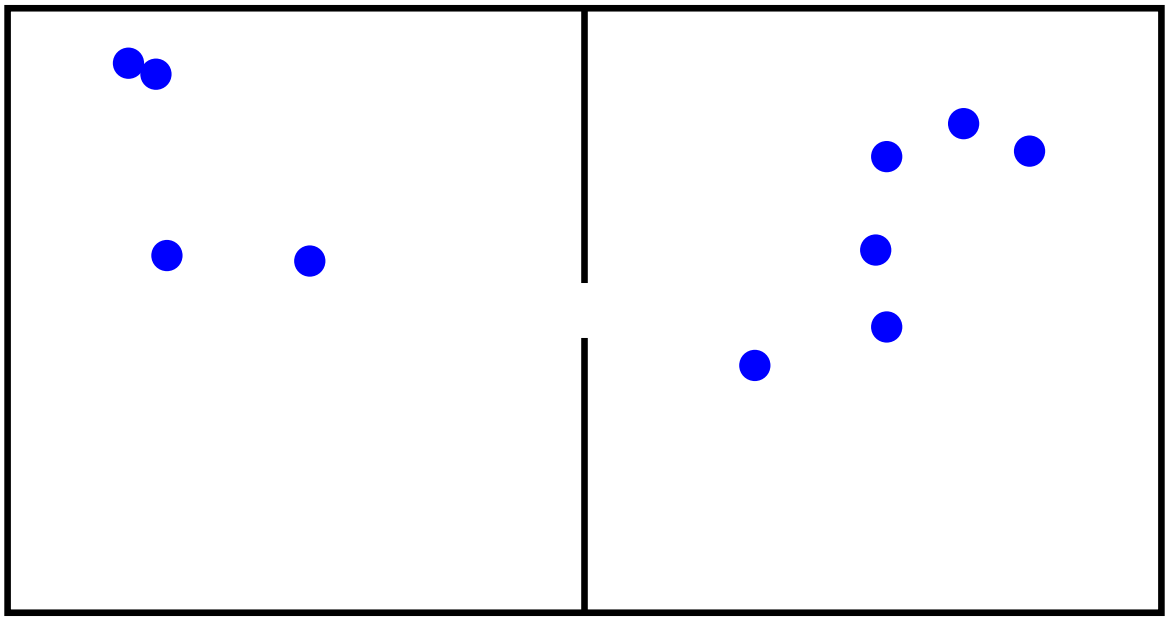
Approche 3 : Un théorème

Theorem (B., Chassaing, Cohen, Gerin, Koegler 08)

La distribution de la proportion de ● *converge vers* $\frac{\sqrt{2}}{2}$ *lorsque* n *tends vers* $+\infty$.

Le terme d'erreur est en $\mathcal{O}(\sqrt{n})$.

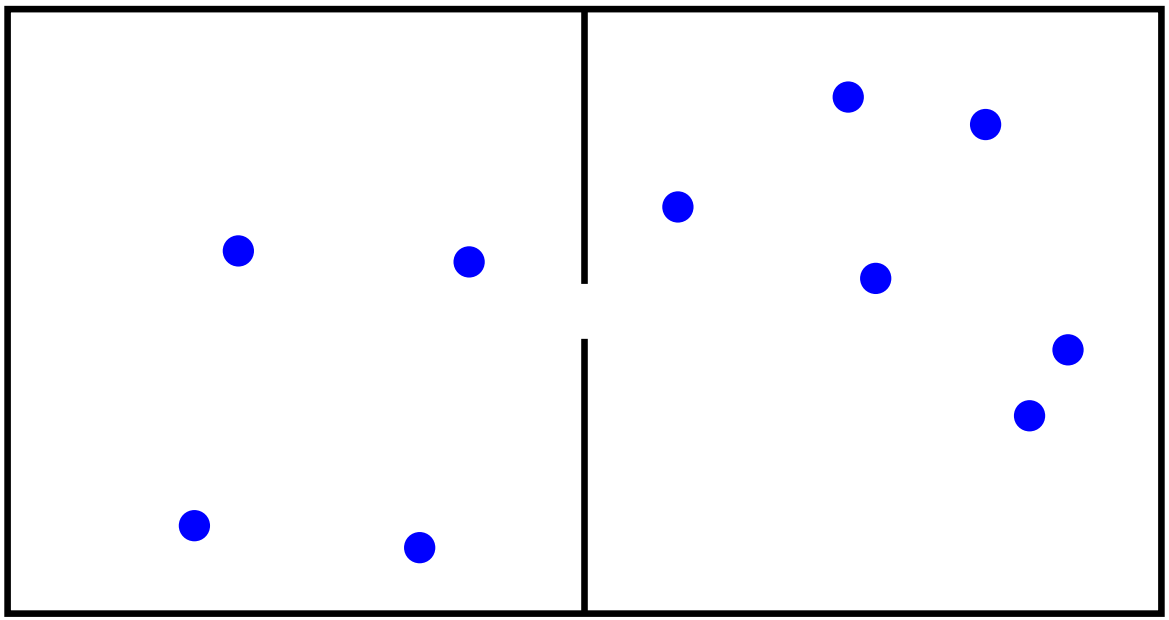
Analogie : Le modèle des urnes d'Ehrenfest[6] 1/4



- On a n boules dans 2 urnes.
- A chaque temps discret,
  - on choisit au hasard selon une loi i.i.d. uniforme une boule,
  - et on la change d'urne.

[6] [Ehrenfest 1907]

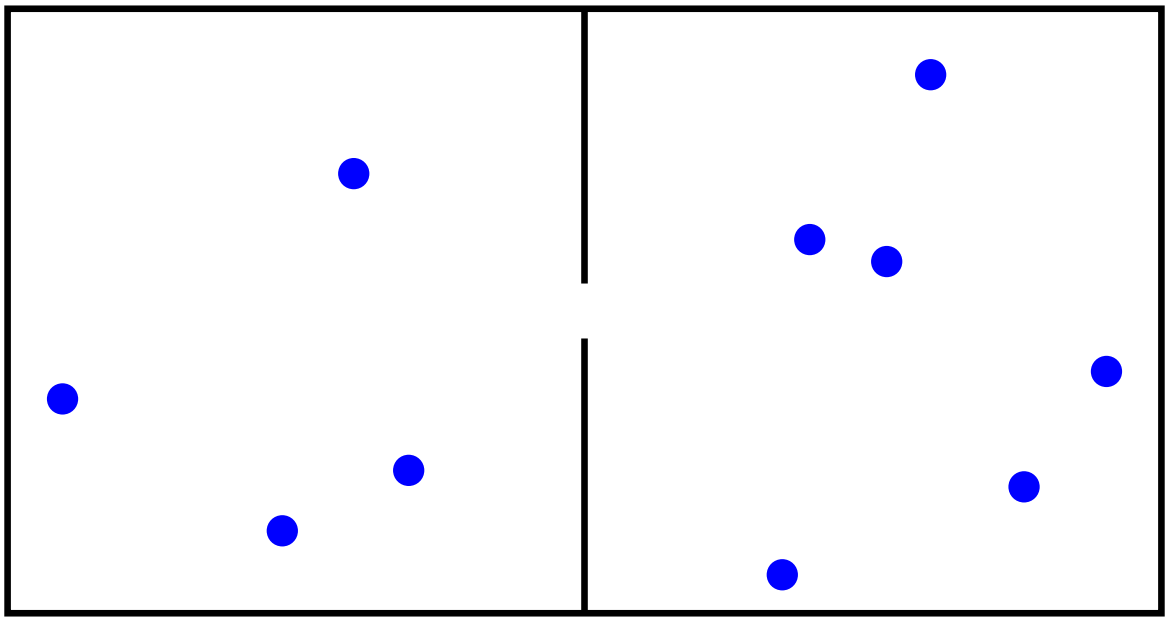
Analogie : Le modèle des urnes d'Ehrenfest 2/4



- La distribution de la proportion de boules dans chaque urne converge vers $\frac{1}{2}$ lorsque n tends vers $+\infty$.

  Le terme d'erreur est en $\mathcal{O}(\sqrt{n})$.

Analogie : Le modèle des urnes d'Ehrenfest 3/4



- Si X_n désigne le nombre de boules dans l'urne de droite, alors $(X_n)_{n\geq 0}$ est une chaîne de Markov dont les transitions sont définies par $p_{k,k+1} = 1 - p_{k,k-1} = \frac{n-k}{n}$.

- La distribution binomiale $B(n, \frac{1}{2})$ est stationnaire pour cette chaîne de Markov.

Analogie : Le modèle des urnes d'Ehrenfest 4/4

Renormalisons

- Si on pose

$$Y_t = \frac{2X_{\lfloor nt \rfloor} - n}{2\sqrt{n}}$$

$$\Delta Y_t = Y_{t+dt} - Y_t,$$

pour t de la forme $t = \frac{k}{n}$ et $dt = \frac{1}{n}$, on a

$$\begin{aligned} E[\Delta Y_t | Y_t = x] &= -2Y_t dt \\ Var(\Delta Y_t | Y_t = x) &= \frac{1}{n} + \mathcal{O}(\frac{1}{n^2}) \\ &\sim dt, \end{aligned}$$

pour $n \longrightarrow \infty$,

- Y_t converge (en loi) lorsque $n \longrightarrow \infty$ vers la solution de l'équation différentielle stochastique[7].

$$dY_t = -2Y_t dt + dB_t.$$

[7] [Stroock Varadhan 79]

Processus d'Ornstein-Uhlenbeck

- Toute solution Y_t de l'équation différentielle stochastique

$$dY_t = -bY_t dt + \sigma dB_t$$

est un processus d'Ornstein-Uhlenbeck[8].

- La distribution Gaussienne $\mathcal{N}(0, \frac{\sigma^2}{2b})$ en est une distribution stationnaire.

- En fait un tel processus Y_t d'Ornstein-Uhlenbeck s'écrit en terme du mouvement Brownien $B(t)$ sous la forme :

$$Y_t = Y_0 e^{-bt} + \frac{\sigma}{\sqrt{2b}} B(e^{2bt} - 1)e^{-bt}.$$

[8] [Ornstein-Uhlenbeck 1930]

Le cas des protocoles de population généraux 1/2

- On suppose des règles sous la forme

$$q\ q' \to \delta_1(q, q')\ \delta_2(q, q')$$

for all $(q, q') \in Q^2$.

- Le système évolue dans $[0, 1]^k$, avec $k = |Q|$.

- Soit $b : \mathbb{R}^k \to \mathbb{R}^k$ la fonction définie par

$$b(x) = \sum_{(q,q')\in Q} x_q x_{q'}(-e_q - e_{q'} + e_{\delta_1(q,q')} + e_{\delta_2(q,q')})$$

où $(e_q)_{q\in Q}$ est la base canonique de $\mathbb{R}^k$.

Le cas des protocoles de population généraux 2/2

Theorem (B., Chassaing, Cohen, Gerin, Koegler 09)

La suite de proportions $(X_n)_{n\geq 0} \in \mathbb{R}^k$ *converge (en loi) vers la solution de l'équation différentielle ordinaire*

$$dX_t = b(X_t)dt$$

avec $X_0 = x_0$.

Pas de terme d'erreur dans le cas général.

Le cas des protocoles de population généraux : Interprétation 1. Lotka-Volterra.

- Autrement dit, dans le cas général, le système converge (en loi) vers une dynamique de population dans $\mathbb{R}^k$ de la forme

$$\frac{dx_i}{dt} = x_i(a_{i,0} + \sum_{j=1}^{k} a_{ij}x_j),$$

c'est-à-dire une équation de Lotka-Volterra[9].

[9] [Lotka 1925] [Volterra 1926]

Le cas des protocoles de population généraux : Interprétation 2. Théorie des jeux.

- En utilisant le fait que $\sum_i \frac{dx_i}{dt} = 0$ cela peut aussi s'écrire[10] comme la dynamique

$$\frac{dx_i}{dt} = x_i(\sum_j a_{i,j}x_j - \sum_{j,l} a_{j,l}x_j x_l),$$

c'est-à-dire comme une dynamique de réplication.

- ou encore

$$\frac{dx_i}{dt} = x_i((Ax)_i - x^t Ax),$$

c'est-à-dire comme la dynamique d'un jeux au sens de la théorie (évolutionnaire) des jeux.

[10] en suivant [Hofbauer 81]

Plan

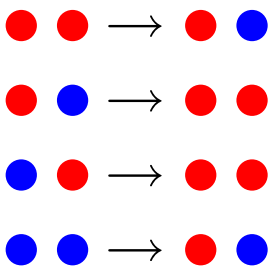
Les protocoles de population

Protocoles de grande population : modèle & convergence

Protocoles de grande population : puissance ?

Calculer avec de grandes populations 1/3

- En d'autres mots, le protocole



calcule $\sqrt{\frac{1}{2}}$.

- Quels autres nombres réels peuvent être calculés dans ce sens ?

Calculer avec de grandes populations 2/3

Faits simples :

- Les nombres calculables appartiennent à $[0, 1]$.
- Les nombres calculables sont algébriques.

- Est-ce que tous les nombres algébriques de $[0, 1]$ sont calculables ?

Calculer avec de grandes populations 3/3

Theorem

A partir d'un protocole calculant p, on peut construire un protocole calculant $\sqrt{p}$.

Theorem

A partir d'un protocole calculant p et d'un protocole calculant q, on peut construire un protocole calculant pq.

Theorem

A partir d'un protocole calculant p et d'un protocole calculant q, on peut construire un protocole calculant $\frac{p+q}{2}$.

Calculer d'autres nombres ?

- Scoop : Maple prétenderait que l'on peut calculer la racine cubique de $\frac{1}{2}$ en utilisant des appariements.

- Si l'on se limite aux protocoles symétriques sur $Q = \{●, ●\}$, il y a 27 règles, qui se décrivent chacune par un triplet $(\alpha, \beta, \gamma) \in \{-1, 0, 1\}^3$.

  - Par exemple, $\sqrt{\frac{1}{2}}$ est calculé par le protocole $(-1, +1, +1)$.

- Parmi les 27 règles possibles,
  - une est triviale,
  - dix convergent vers 0 ou 1,
  - pour les 16 autres la distribution de la proportion de ● converge vers p^* où p^* est l'unique racine du polynome

$$P(X) = (\alpha - 2\beta + \gamma)X^2 + (2\beta - 2\gamma)X + \gamma,$$

lorsque n va vers l'infini. sur [0,1].
Le terme d'erreur est en $\mathcal{O}(\sqrt{n})$.

Discussions

- Une formalisation des protocoles de population en grande population.
- Une étude de la convergence de ces protocoles vers une dynamique de population dans le cas général.
- Un développement asymptotique de la convergence dans certains cas.

Discussions

A l'instant t, nous n'avons pas de caractérisation des nombres réels calculables. Nous savons

- qu'ils sont algébriques
- qu'ils contiennent les rationnels
- qu'ils sont clos par racine carrée
- qu'ils sont clos par produit
- qu'ils sont clos par demi-somme

Travail courant et futur

- Être plus fin sur la convergence.
- Caractériser précisément les nombres calculables.
- Variantes du modèles.
- Aller plus loin :
  - Peut-on caractériser les prédicats calculables ?
  - Peut-on les caractériser par une logique ?