INTRODUCTION : Pourquoi La programmation fonctionnelle?

- Vous avez étudié jusqu'à présent la programmation impérative :
  - Conception d'un programme comme une suite d'instructions.
  - L'ordinateur doit impérativement suivre cette suite d'instructions pour que le programme s'exécute correctement.

Cette conception date des débuts de l'informatique et semble au premier abord assez naturelle.

INTRODUCTION : Pourquoi La programmation fonctionnelle?

Il existe en programmation comme dans la plupart des autres disciplines scientifiques, différentes manières de « voir le monde ».

La programmation fonctionnelle propose un autre style, plus proche de la pensée mathématique pure.

Pour le mathématicien, le concept d'instruction n'existe pas, celui d'affectation encore moins

INTRODUCTION : Pourquoi La programmation fonctionnelle?

Qu'emprunte-t-on au mathématicien ?

Deux concepts fondamentaux :

- Les **fonctions** et la possibilité de les composer, la loi o jouant le rôle du début…fin impératif.
- Le **principe de récurrence**, dont l'itération (boucle tant_que) formera un cas particulier.

Vous devrez donc avoir la même rigueur de pensée qu'en algèbre, ni plus ni moins. N

ous n'allons pas pour autant manipuler systématiquement des objets mathématiques ! Mais nos schémas de pensée seront au fond les mêmes qu'en mathématiques.

INTRODUCTION : Pourquoi La programmation fonctionnelle?

AUCUNE CONNAISSANCE PREALABLE DE PROGRAMMATION N'EST NECESSAIRE A CE COURS !

Ne raisonnez pas en C ou en algorithmique :

- nous abordons une autre façon de pensée
- Ayez un esprit de débutant.

INTRODUCTION : historique LISP

Inventé en 1958 au MIT par J. McCarthy : un des langages de programmation les plus anciens.

- Utilisation longtemps limitée à cause de la faible puissance des ordinateurs.
- Aujourd'hui, utilisé non seulement en IA, mais aussi dans d'autres domaines :

- Il existe plusieurs dialectes LISP, dans ce cours : scheme (racket)

INTRODUCTION

- 3 grands domaines informatiques sont apparus dans les années 60 :
  - Génie Logiciel
  - Simulation de Systèmes
  - Intelligence Artificielle
- Génie Logiciel : développement de méthodes de conception de logiciel ou de langages de programmation
- Simulation : développement d'algorithmes ou de langages de simulation du comportement de systèmes
- I.A. : développement de concepts et méthodes visant à permettre à l'ordinateur de « raisonner ».

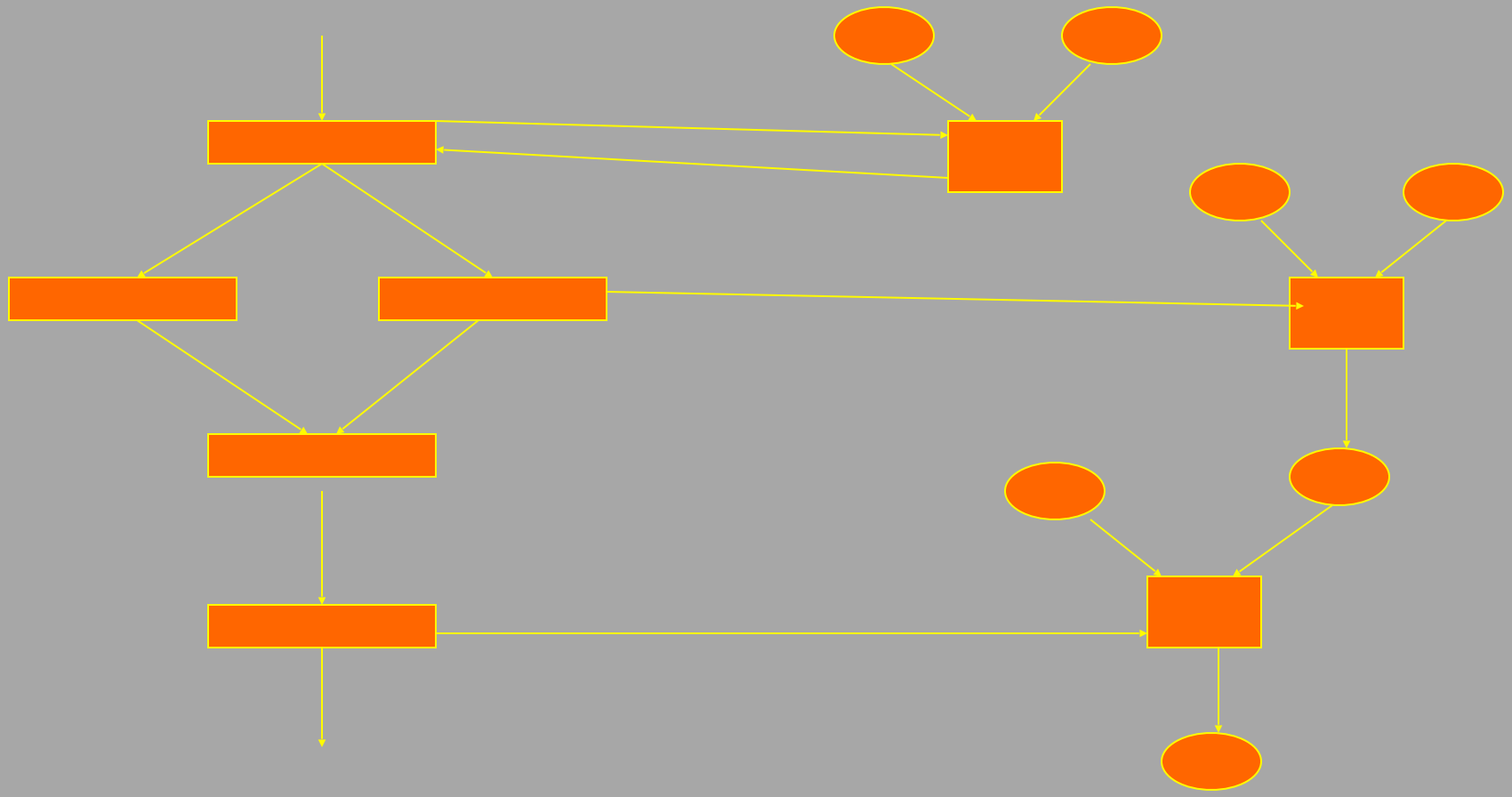
Introduction

6 paragdimes de programmation

- Procédurale : données passives + procédures actives
- Fonctionnelle : fondée sur la récursivité et la définition de fonctions
- Orientée Objets : Objets + attributs (données)
  - comportements (procédures)
- Orientée Données : données actives (actions réflexes)
- Orientée règles : règles + faits.
- Programmation neuromimétique : fondée sur les réseaux de neurones.

Introduction

Programmation procédurale



Introduction

Programmation fonctionnelle

Definition (Langage fonctionnel)

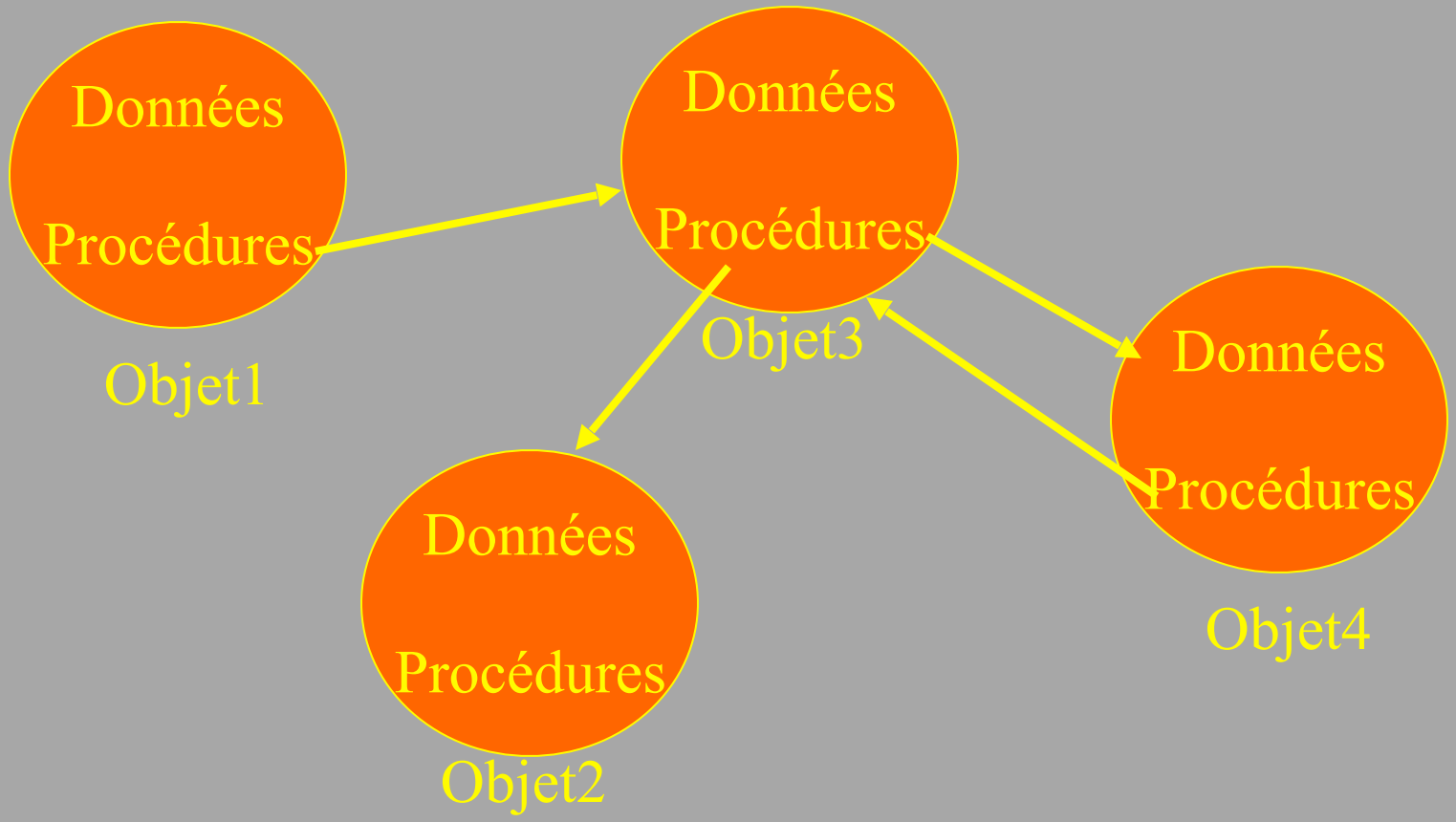
Langage qui permet de manipuler des fonctions comme des valeurs de première classe (entiers, chaînes de caractères).

En particulier, on peut écrire :

- des fonctions qui prennent des fonctions en argument ;
- des fonctions qui renvoient des fonctions créées dynamiquement.

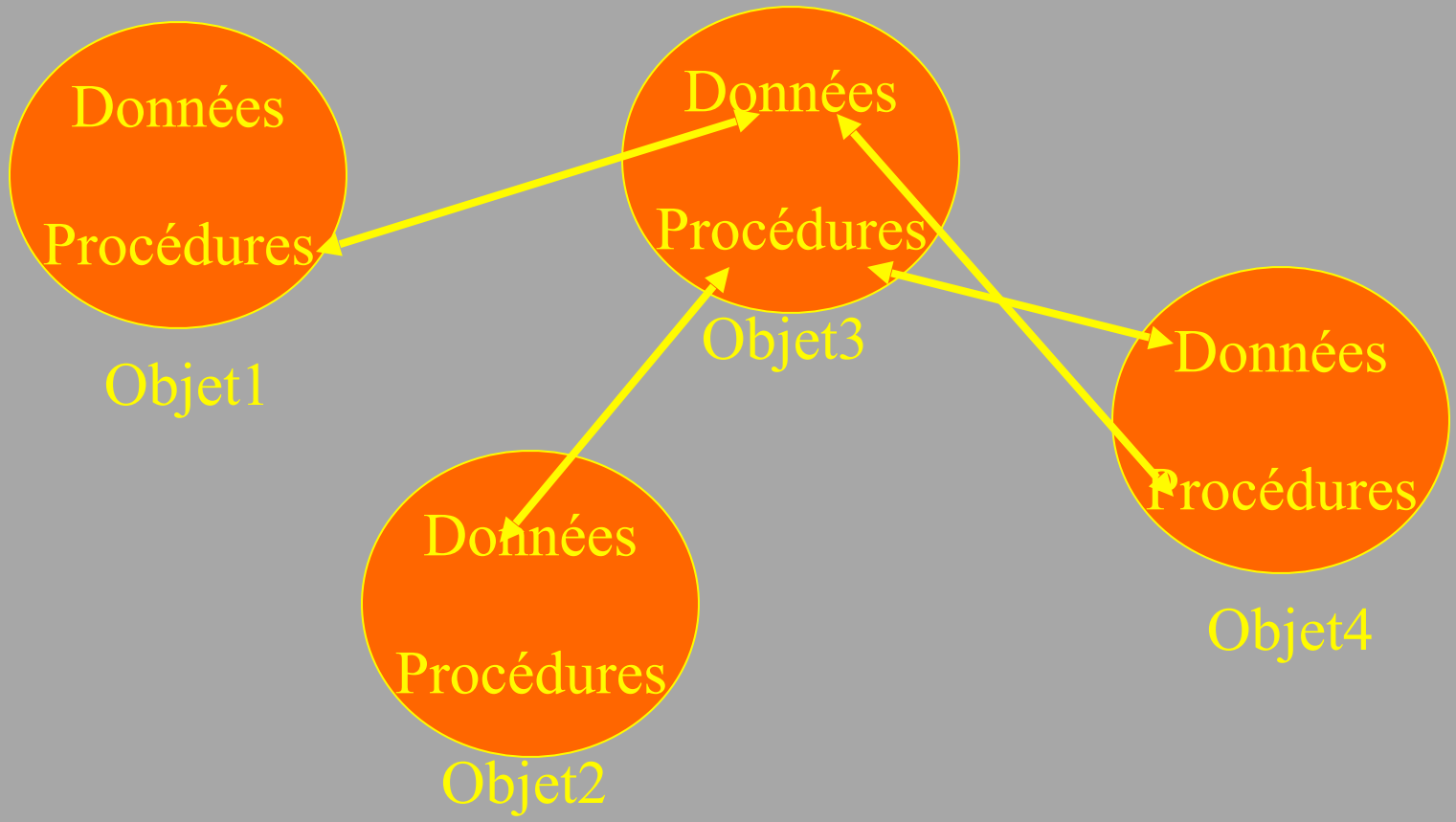
Introduction

Programmation Orientée Objets



Introduction

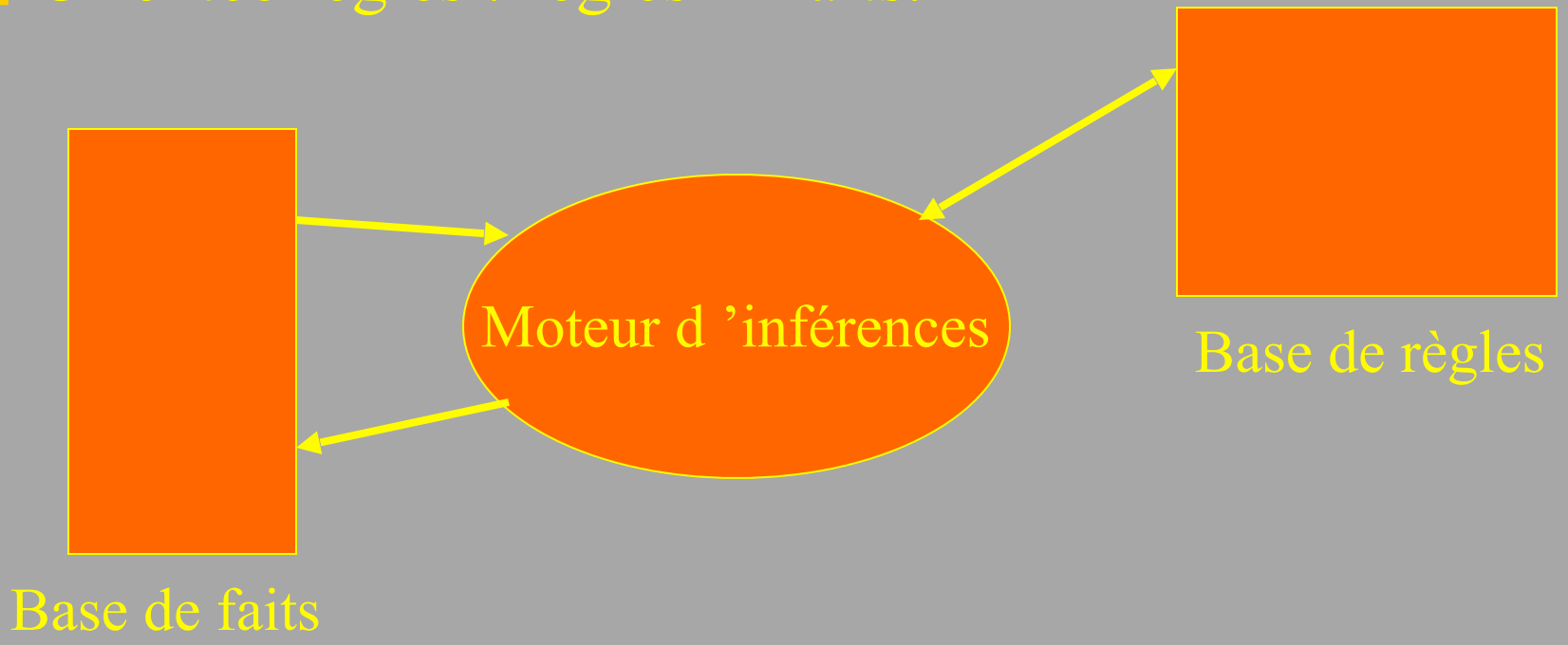
Programmation Orientée Données



Introduction

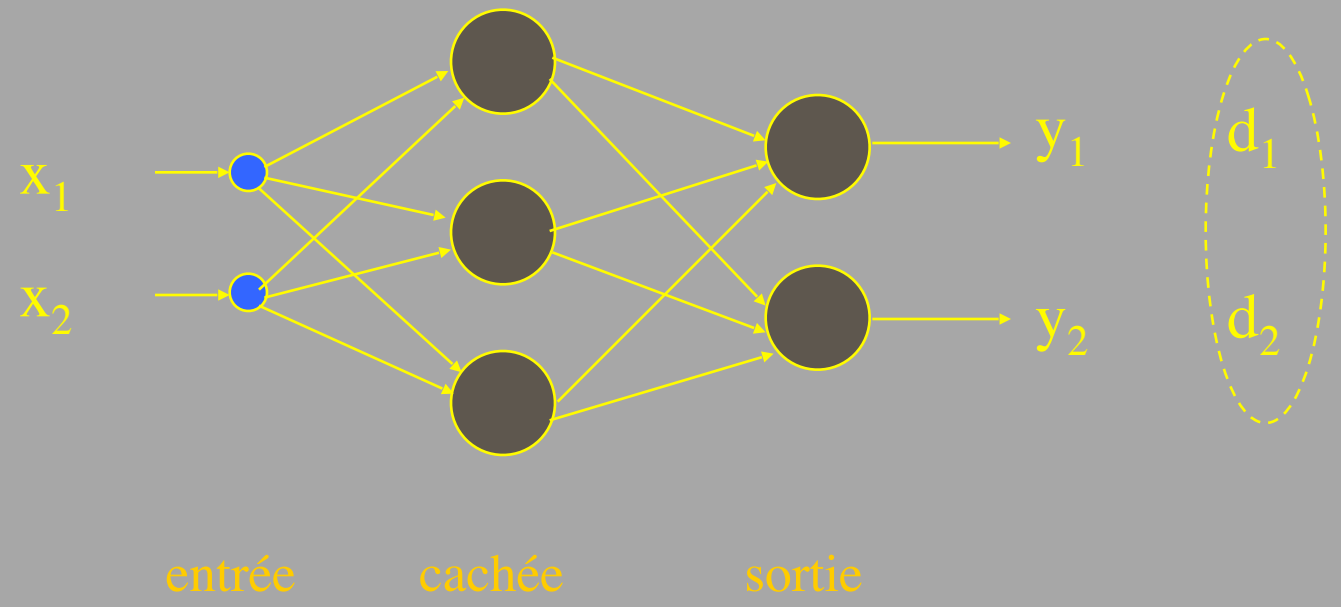
Programmation orientée règles

2 Orientée règles : règles + faits.



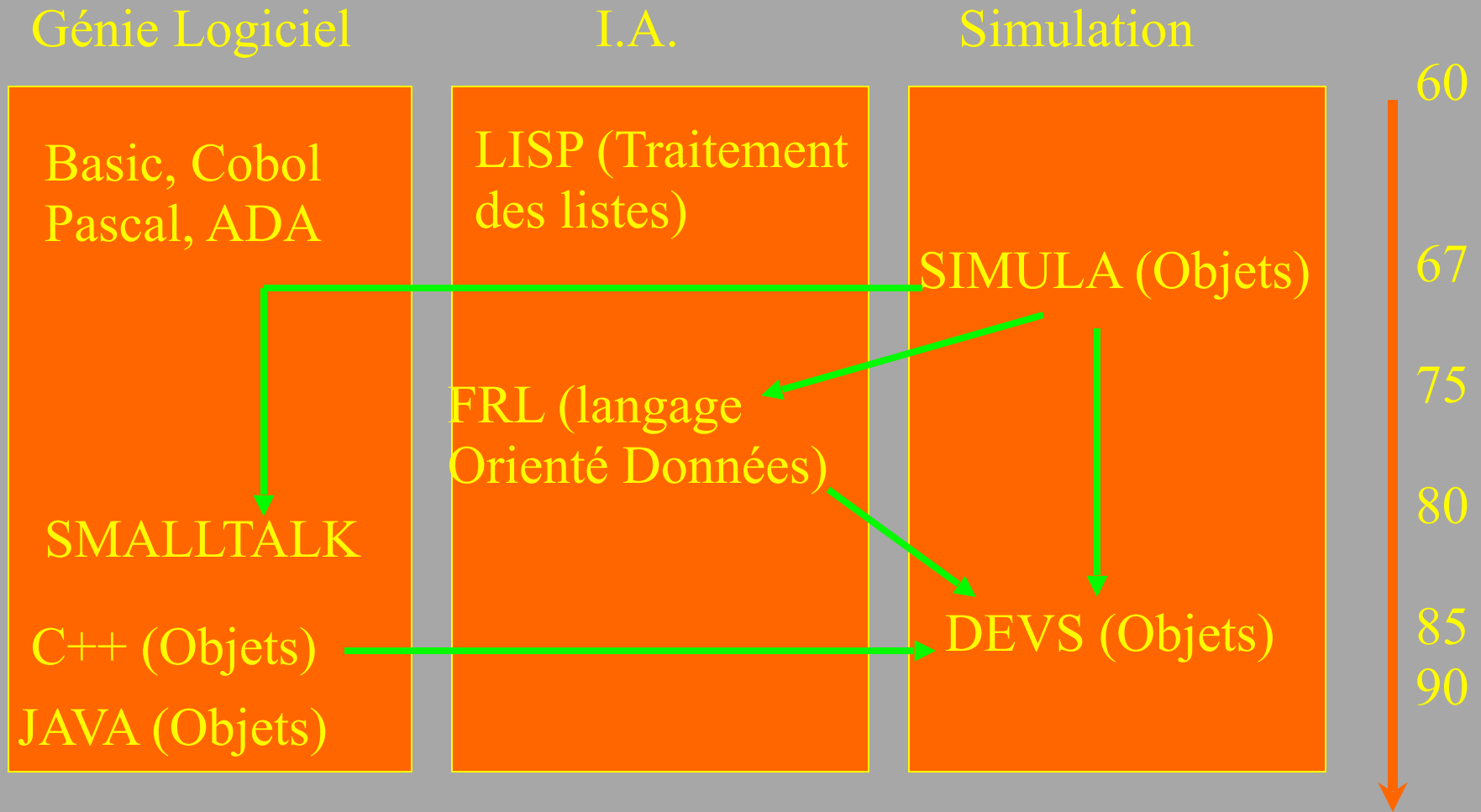
Introduction

Programmation Neuromimétique



INTRODUCTION : historique

Notion d 'objets



INTRODUCTION :Une syntaxe étrange venue d'ailleurs !

Des premières années de l'informatique avec le langage LISP(1958)
Scheme date lui de 1980.

LISP fut conçu à l'origine pour les besoins des théoriciens de la programmation et surtout pour des programmeurs qui souhaitaient un langage suffisamment malléable pour les
besoins de l'Intelligence Artificielle naissante (1956).

La syntaxe de Scheme et LISP est donc très proche.

Basée sur l'idée de représenter une expression arithmétique sous une forme préfixée totalement parenthésée

INTRODUCTION :Une syntaxe étrange venue d'ailleurs !

En C, on écrit sqrt(x+2) pour exprimer la racine carrée de x+2

En Scheme, nous écrirons de manière « préfixée » :

l'opérateur sera toujours en tête de l'expression, et pour é***viter toute ambigüité, nous placerons des***

parenthèses comme ceci : (sqrt (+ x 2))

Au lieu d'écrire f(x,y,z), nous écrirons (f x y z) sans virgules

Le même processus de traduction s'opérant à tous les niveaux, c'est-à-dire dans les sous-expressions x, y et z.

Dans une expression bien formée, il y aura donc autant de parenthèses ouvrantes que de parenthèses fermantes.

Une syntaxe étrange venue d'ailleurs !

Maths	**Scheme**
1+2+3	(+ 1 2 3)
1-2+3	(+ 1 -2 3) ou (+ (- 1 2) 3)
sin(omega * t+phi)	(sin (+ (* omega t) phi)
x+2y=0	(= (+ x (* 2 y)) 0) ou (zero? (+ x (* 2 y)))
x -> x+2	(lambda (x) (+ x 2)) la fonction « x a pour image x+2 »
f o g o h	(compose f g h) la composition de fonction
si x=2 alors 3 sinon y+1	(if (= x 2) 3 (+ y 1))