Par défaut, on considère un Lsystem est un objet qui contient des règles, un contexte d’execution (namespace : containeur d’objet python, des objets graphiques, des paramètre d’exec), une tortue, des paramètres de tortue, etc.

On peut supposer qu’on a un Lsystème qui importe un sous -lsystem.

* + Lpyimport myrules
  + Lpyimport myrules as group 3

Dans le cas des règles de production :

* On suppose des contextes d’exécution différents.
* Activation des règles ou des groupes de règle a la demande ?
  + def StartEach() :
    - Activate(myrules,0)
  + Les règles viennent s’appliquer normalement. Elles peuvent être réécrites par le l-system courant.

Dans le cas des règles d’interprétation :

* Quelles ressources pour la tortue choisir ?
  + Formaliser et attacher des ressources a la tortue (couleurs, formes ?)
  + Comment choisir ou activer une ressource ?
* Activation des règles ou des groupes de règle a la demande ?
  + Maximum depth ?
  + Activation des règles par une fonction d’activation
  + Application en profondeur. Les règles doivent être combinées avec les précédentes ? Ou doit on supposer qu’on applique d’abord les règles du lsystem maitre puis sur un symbole terminal, on applique les règles du second lsystem ?

Est ce que ca aide/complique la définition d’un solveur ?

Un solveur est un lsystem (groupe de règles) pour lequel on peut avoir plusieurs iterations. Ce solveur reste en vase clot ?

maitre

Maximum depth: 2

MyFlower  -->A

A --> B

B --> AppleFlower

Souslsystem

Maximum depth: 10

AppleFlower --> A

A --> MyFlower