NetBioDyn

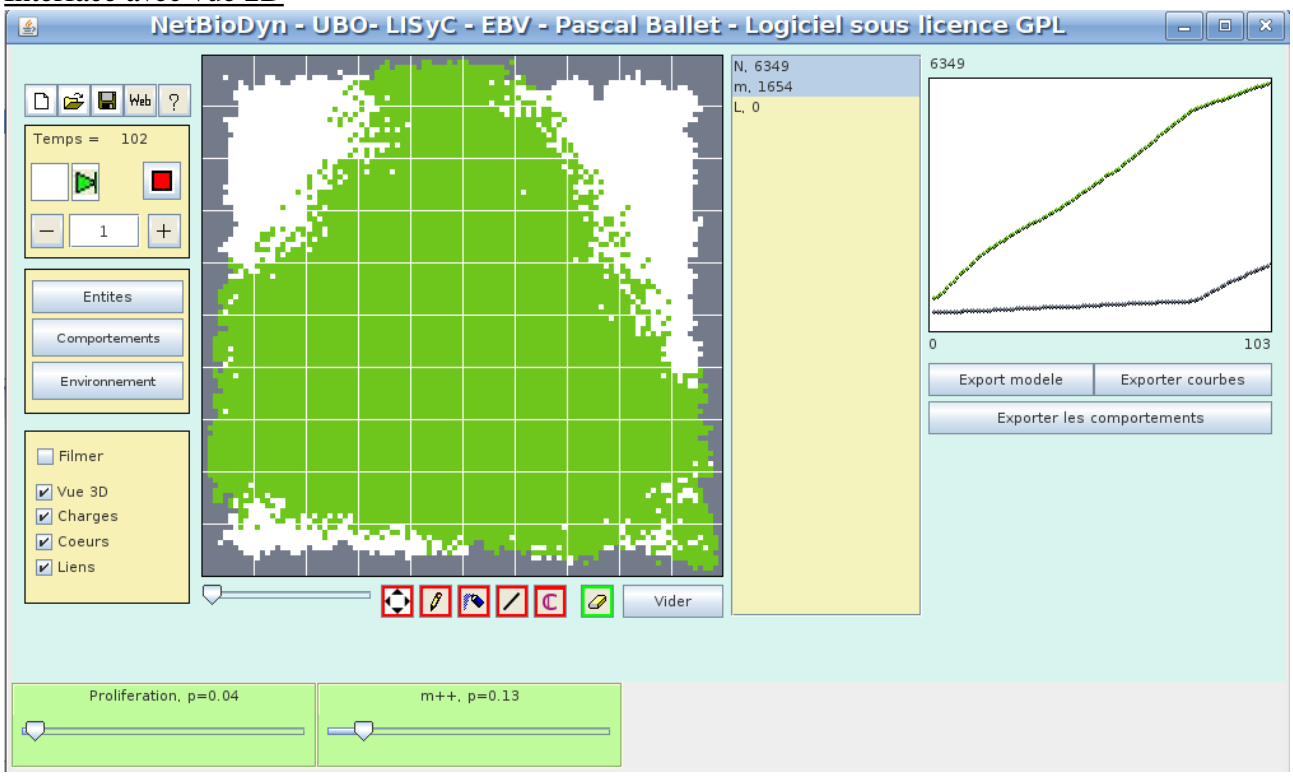
http://netbiodyn.tuxfamily.org

Logiciel simple pour le développement de systèmes multiagent complexes.

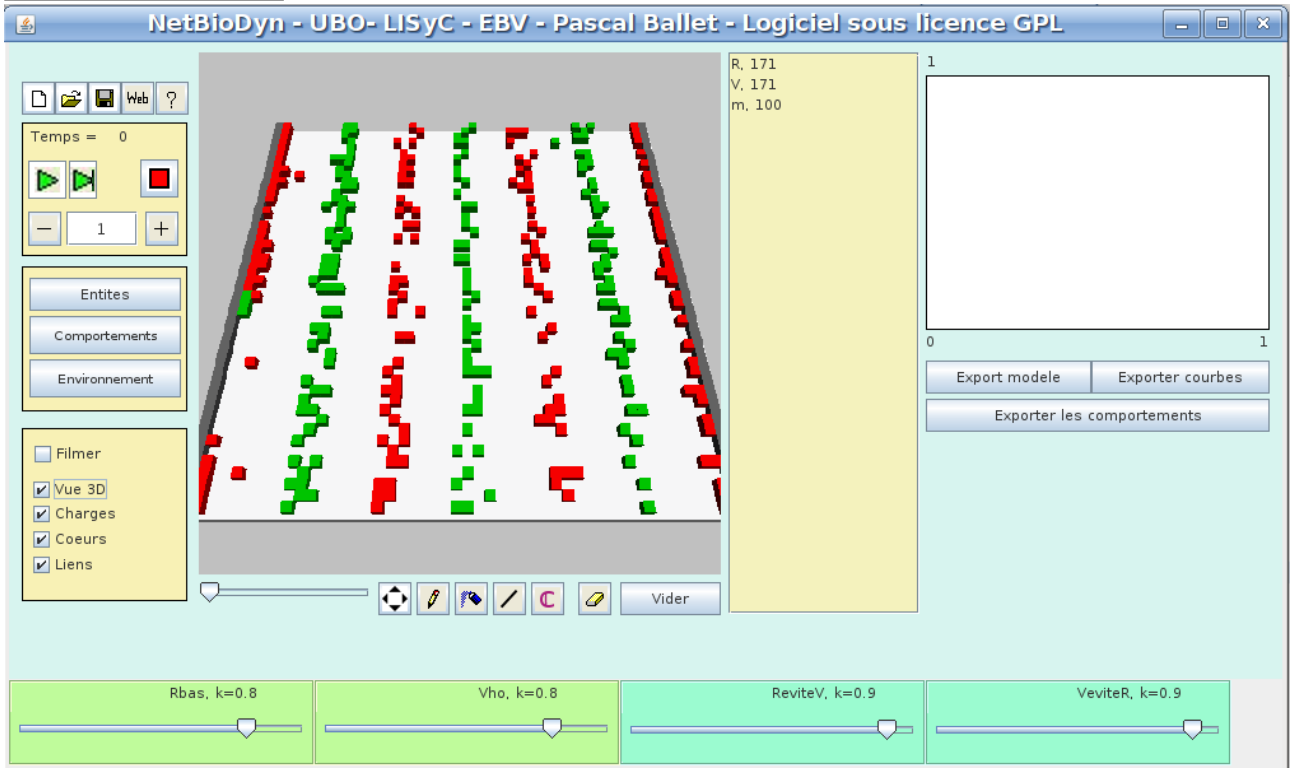
Auteur : Pascal BALLET
Laboratoire d'Informatique des Systèmes Complexes (LISyC – EA 3883) - Université de Bretagne Occidentale

Interface graphique de netBioDyn

Interface avec vue 2D



Interface avec vue 3D



Objectif

L'utilisation des systèmes multiagents est intéressante pour la mise au point de logiciels simulant des phénomènes complexes comme les systèmes biologiques.
Cependant, la difficulté de mise en oeuvre des systèmes multiagents requiert généralement des compétences poussées en informatique. De plus, pour modéliser des phénomènes naturels complexes, des connaissances poussées en physique et en mathématique sont généralement nécessaires.
C'est pourquoi, la mise au point de logiciels facilitant l'utilisation de l'approche multiagents, par le masquage d'une grande partie des difficultés peut s'avérer utile. Ainsi les connaissances ne dépendant pas du domaine d'expertise du modélisateur, comme la programmation d'architecture agent ou bien la résolution d'équations d'équation biochimiques, ne deviennent plus obligatoire pour obtenir des simulations crédibles.
L'architecture BioDyn d'une part et les logiciels associés flexBioDyn et netBioDyn simplifient le développement d'applications multiagent pour le plus grand nombre.

Agents

3 types d'agents sont utilisés dans l'environnement. Les reaxels (reaction elements), les connexels (connexion elements) et les complexels (complex elements, a composition of reaxels and connexels).

Les reaxels sont des agents ayant les propriétés suivantes :

- un nom
- une position (x,y,z)
- un volume élémentaire (cubique ou sphérique)
- un ensemble de récepteurs situés à la surface du volume
- un ensemble de charges situées à la surface du volume

Les connexels sont des agents ayant les propriétés suivantes :

- un nom
- deux récepteurs (partagés avec deux réaxels différents)
- une longueur au repos
- une fonction d'interaction (qui influence le déplacement des deux réaxels)
- une présence physique ou non (permet de bloquer ou non les réaxels)

Les complexels sont des prototypes d'agents possédants :

- un nom
- un ensemble de reaxels
- un ensemble de connexels

Les reaxels et les connexels sont les agents représentant la structure géométrique et physique des systèmes développés avec flexBioDyn.

Les complexels sont un étiquetage arbitraire de structures géométriques et physiques (composés de réaxels et de connexels) pratique pour le développement de SMA, tant du point de vue de la création des structures d'agents que de la mise au point des réactions (situations / actions). Les complexels n'ont pas d'existence propre dans la simulation mais sont essentiels : ils sont les « mots » du modélisateur pour décrire son système multiagent, les lettres étant les réaxels et les connexels et la grammaire étant les réactions.

Assemblages

Les réaxels sont reliés entre eux par les connexels. Cela permet de créer des structures géométriques et physiques déformables.
Néanmoins pour la représentation de structures géométriques et physiques solides cette approche n'est pas adaptée. C'est pourquoi les réaxels sont capables de s'assembler les uns avec les autres de manière rigide. Chaque assemblage est alors traité comme en tout en ce qui concerne les déplacements.

Il est important de noter ici qu'il n'y a pas de rapport entre un assemblage et un complexel. Un assemblage de réaxels peut être nommé par le concepteur en lui attribuant un complexel (ainsi il sera réutilisable facilement pour la conception), mais un assemblage peu également être étiqueté de manière multiples (selon par exemple les parties intéressantes). Inversement, un complexel n'est pas nécessairement un assemblage.

Comportements

Deux types de comportements sont disponibles. Les réactions (transformation des agents sous certaines conditions) et les déplacements (modifications de la position des réaxels – éventuellement assemblés).

Réactions

Les réactions permettent de transformer un groupe de réaxels et de connexels dans une situation particulière en un autre groupe de réaxels et de connexels dans une situation différente.
Cette transformation est soumise à une probabilité de réalisation. En d'autres termes, si les conditions sont réunies pour qu'une réaction se produise, elle ne se produira effectivement qu'après le tirage d'un nombre aléatoire.
Lors d'une réaction, le principe de conservation maximum est appliqué. Cela signifie qu'un minimum de changements sont appliquées dans le système au cours d'une réaction. En pratique, lors de la transformation d'un réaxel en un autre, les connexels sont conservés s'ils ne sont pas explicitement supprimés.

Le langage permettant d'exprimer les transformations est composé d'une structure conditionnelle qui précise dans quelle situation doivent se trouver les réaxels et le connexels pour réagir et d'une structure séquentielle d'actions permettant d'effectuer la transformation souhaitée.

La structure conditionnelle peut se décrire de la manière suivante :

Soit r l'ensemble des types de réaxels et c l'ensemble des types de connexels déclarés dans une modélisation : $r = \{r^1, r^2, .. r^{nr}\}$ et $c = \{c^1, c^2, .. c^{nc}\}$ avec nr le nombre de types de réaxels et nc le nombre de type de connexels.

Chaque type de réaxel r^i (avec i dans [0, nr]) et chaque type de connexel c^j (avec j dans [0, nc]) est pseudo-instancié dans la structure conditionnelle pour en permettre un repérage sans ambigüité. Chaque réaxel r^i sera pseudo-instancié en r^i_0, r^i_1,... r^i_p,... avec $p < pu$ où pu est le nombre d'instances différentes de r^i utilisées dans la structure conditionnelle. De même, chaque connexel c^j sera pseudo-instancié en c^j_0, c^j_1,... c^j_q,... avec $q < qu$ où qu est le nombre d'instances différentes de c^j utilisées dans la structure conditionnelle.

Les conditions possibles sont les suivantes :

- R **touche** R
- R **touche** R **à** gauche / droite / haut / bas
- R **est en** (dx,dy,dz) **relativement à** R
- C **lié à** R

Les actions possibles sont :

- **supprimer** R/C
- **ajouter** R/C
- **remplacer** R **par** R
- **remplacer** L **par** L
- **connecter** C à R et R
- **appliquer sur** R **la force** (fx,fy,fz)

Pour des raisons d'implémentation et de liberté maximum, ces structures conditionnelles sont des méthodes statiques de la classe ConditionsActions ayant les méthodes bool touche(R0,R1), bool est_en(R0,R1,dx,dy,dz), bool lie_a(R,L), void supprimer(R/L), void ajouter(R/L), etc.

Déplacements

Les déplacements sont issu d'un calcul de forces lié à la théorie de Newton en régime sur-amorti. Les forces sont composées des interactions (connexels), des charges et de l'agitation thermique d'un fluide implicite (eau, membrane, ...). Ce sont les assemblages de réaxels qui sont soumis à ces forces.

A chaque appel du comportement de déplacement, les assemblages sont re-calculés en fonction des formes portées par les réaxels. La forme est représentée par un nombre entre 0 et 100. La complémentarité entre deux formes est la distance entre ces deux formes. Cette complémentarité représente alors simplement la probabilité, par pas de simulation, que deux réaxels soient liés pour former un assemblage.

Pour des raisons de simplicité et d'efficacité de simulation, la vitesse de déplacement est plafonnée à 1case par pas de simulation. Dans les faits, si le modélisateur souhaite faire une simulation quantitative, il est contraint de donner la vitesse de 1 case par pas de simulation au réaxel pouvant se déplacer le plus rapidement.

Les réaxels ne peuvent en aucun cas se superposer les uns les autres. Ceci permet de reproduire la notion d'encombrement qui se retrouve dans de nombreux systèmes complexes.

L'impact du mouvement brownien d'un milieu liquide est représenté par une force de Langevin dont l'intensité dépendant de la température. Cette force peut éventuellement être biaisée en orientation pour reproduire un flux.

La force résultante par réaxel est calculée au fur et à mesure des calculs d'influences (connexels, langevin et charges). Ici, il est indispensable d'appliquer un sémaphore d'exclusion mutuelle par réaxel pour éviter que deux comportements d'influence ne modifient de manière erronée (quasi-simultanée avec écrasement mutuel de la variable contenant la valeur de la résultante) la résultante d'un seul et même réaxel.

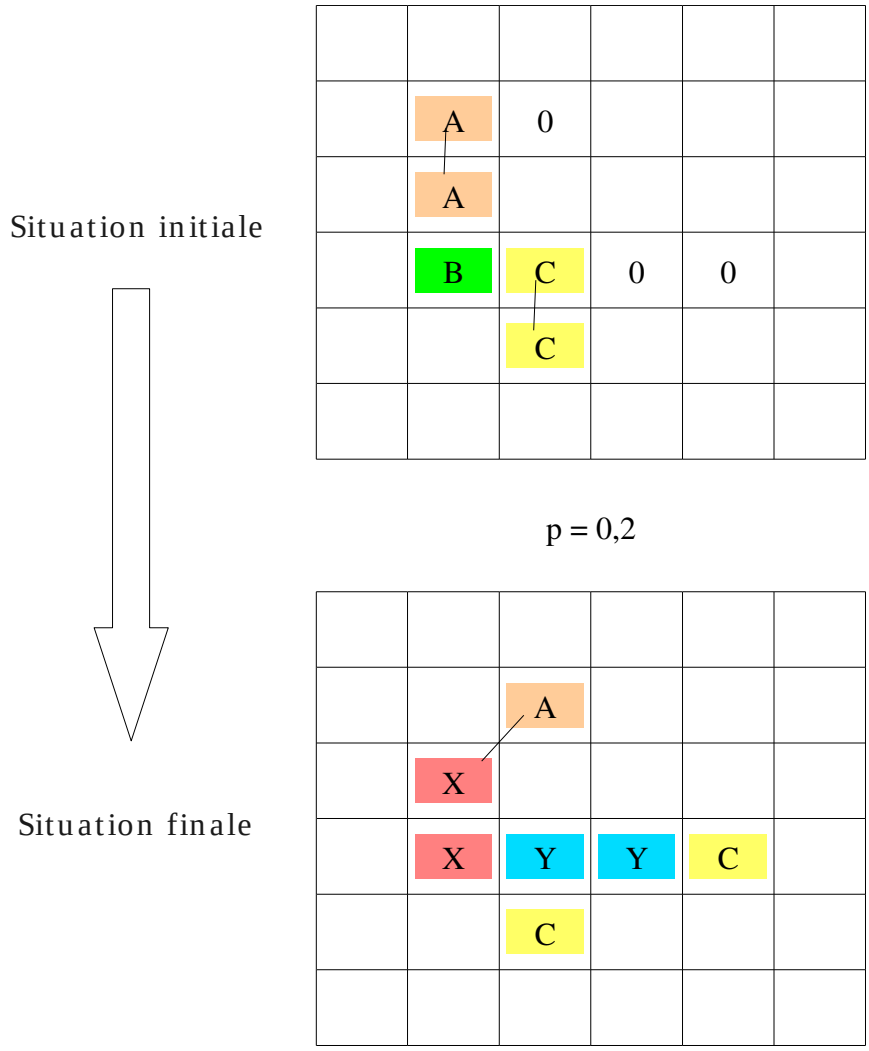
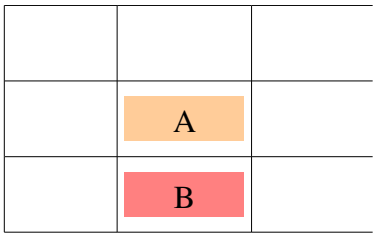


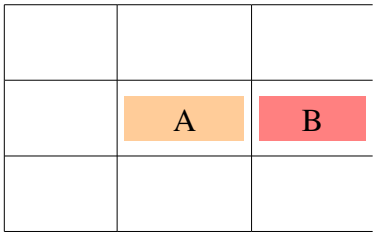
Illustration 1: Exemple d'une réaction. Lorsque la situation initiale apparaît au cours d'une simulation et que la situation finale peut s'appliquer (les cases où des réaxels sont créés sont vides), la situation finale remplace alors la situation initiale. En détail il est possible d'expliciter le déroulement de la réaction : le réaxel A du haut est supprimé (et non pas déplacé à droite car la notion de mouvement n'existe pas pour les réactions – ceci n'implique pas que pour le concepteur il s'agisse d'un déplacement), le A du bas est remplacé par un X, le B est remplacé par un X, le C du haut est remplacé par un Y, le C du bas est conservé, un Y et un C sont créés à droite du précédent Y créé, le lien entre A et A est supprimé, le lien entre C et C est supprimé et un lie entre X et A est créé. Il est à noter ici que les cases vides peuvent contenir n'importe quels réaxels. De même, il peut exister tout forme de liens entre les réaxels. En application du principe de moindre modification (ou conservation maximum), les réaxels et les liens non explicités dans les situations initiales et finales sont conservés. Si par exemple il existe un lien entre A et B dans la situation initiale, il y aura conservation de ce lien entre X et X dans la situation finale.

La description des réactions par le modélisateur peuvent se faire selon plusieurs approches complémentaires. En effet, il est difficilement envisageable dans une utilisation courante de BioDyn de décrire toutes les situations et actions possibles. Par exemple, pour indiquer que lorsque le réaxel A rencontre le réaxel B il y a production d'un réaxel C à la place de A et suppression de B, il faudrait :
1-décrire les 4 situations possibles de rencontre de A et de B (les 4 postions cardinales)
2-décrire la situation de création de C et de suppression de A et de B.

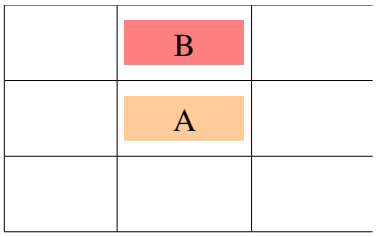
Situation 1 :



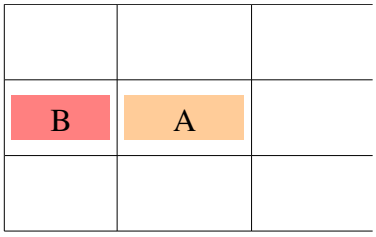
Situation 2 :



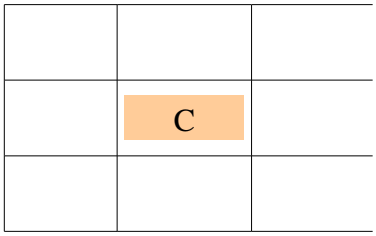
Situation 3 :



Situation 4 :



Action 1 :



Afin de limiter une description exhaustive de toutes les situations possibles, une écriture particulière des réactions, appelé réaction semi-située a été développée. Par exemple, pour décrire la réaction précédente, il suffit d'écrire en réaction semi-située :

(A)+B=>(C)+0

avec les parenthèses indiquant le réaxel se trouvant au centre de la réaction et le 0 le réaxel à supprimer.

Exemple 1 :

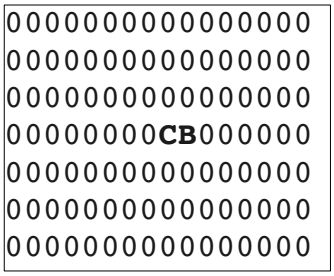
Soit la réaction semi-située :

(A)+B => (C)

et l'environnement :

Cette réaction peut s'appliquer deux fois dans l'environnement et donner au choix :

ou bien



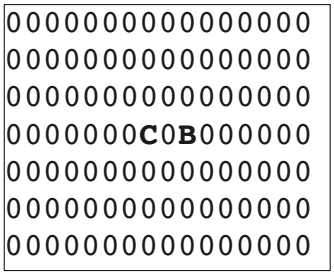
Exemple 2 :

Soit la réaction semi-située :

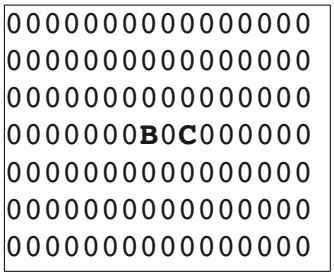
(B)+A => (C)

et l'environnement :

Cette réaction peut s'appliquer deux fois dans l'environnement et donner au choix :



ou bien



Lois et règles élémentaires dans netBioDyn

1-Un réaxel occupe une case de l'environnement et une seule.
2-Une case de l'environnement ne peut contenir qu'un seul réaxel.
3-Un connexel repose sur 2 réaxels et seulement 2.
4-Un réaxel peut porter entre 0 et une infinité de connexels.
5-Si un réaxels portant un ou plusieurs connexel est supprimé, les connexels qu'il porte sont supprimés également.
6-Si un réaxel est remplacé par un autre réaxel, les connexels du réaxel remplacé sont transmis au réaxel remplaçant.
7-Une réaction ne peut se produire que a) si tous les réactifs sont présents et b) si tous les produits peuvent être créés selon la disposition requise (choisie par le modélisateur).
8-Les réactions et les réaxels en concurrence (mêmes réactifs pour les réactions et même case pour les réaxels) sont équitablement traités pour a) garantir l'unicité de l'utilisation des réactifs et b) éviter les biais de simulation.

Gestion des connexels

Un connexel permet de relier deux reaxels. Un connexel peut être imperméable ou non, c'est à dire qu'il peut bloquer ou non les déplacements des réaxels selon le choix du modélisateur.
Comme les reaxels, les connexels sont projetés sur une matrice pour un accès rapide, selon la position dans l'espace. Cependant, et contrairement aux réaxels, il est possible d'avoir plusieurs connexels à la même position. Chaque case de la matrice est donc une liste de connexels, un même connexel étant répété dans chaque case sur toute sa longueur.

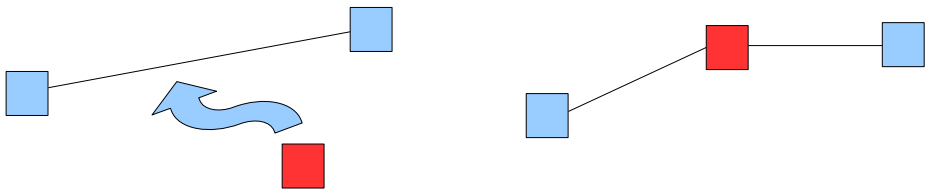
Relation entre réaxels et connexels

Les connexels permettent :

de lier de manière élastique 2 réaxels
de bloquer éventuellement le passage des réaxels (imperméables)

ainsi des compartiments et des membranes peuvent être représentés ce qui n'est pas possible avec les réaxels tels que définit précédemment.

Les réaxels peuvent s'insérer dans les connexels selon le schéma suivant :



1-Le réaxel rouge se déplace et rencontre un connexel. 2-Le réaxel est inséré dans le connexel. Cette connexion consiste à supprimer l'ancien connexel et à crée deux nouveaux connexels reliant le réaxel inséré avec les deux réaxels portant le connexel initial.

L'insertion d'un réaxel dans un connexel imperméable ne peut s'appliquer que lorsque les règles physiques élémentaires de netBioDyn sont respectées, c'est à dire lorsque :

le réaxel considéré est explicitement capable de s'insérer dans un connexel
les deux liens créés ne coupent aucun réaxel exceptés leurs supports
les deux liens créés ne coupent pas d'autres liens imperméables.

Ces deux dernières règles garantissent l'imperméabilité des connexels même lorsqu'un réaxel vient s'y insérer.

L'algorithme permettant cette opération est le suivant :

1-Le réaxel r0 tente de se déplacer sur le connexel c porté par deux réaxels r1 et r2.
2-Si r0 peut s'insérer dans c, le connexel c est supprimé.
3-Les deux nouveau connexels sont créés.
4-Si les connexels créés ne coupent aucun réaxel (hormis r1,r0 et r2), ni aucun autre connexel imperméable, l'opération est validée.

Détails de l'algorithme de traitement d'un pas de simulation

1-RAZ des matrices de connexels et de champs
2-Placement des connexels imperméables
3-Calcul des réactions possibles
4-Application équilibrée des réactions
5-Placement des champs d'influence (forces)
6-Calcul des mouvements possibles
7-Application des mouvements (incluant les insertions de réaxels dans les connexels).
8-Retour à l'étape 1.

Exemples d'applications

Exemples avec des réaxels seuls

Écoulement de grains de sable

La modélisation et la simulation simplifiée d'un écoulement de grains de sable peut se réaliser à l'aide de deux entités (grain et paroi) et de deux comportements (tomber et rouler).

Une entité Grain permet de représenter un grain de sable. Ce grain de sable est fixe et seuls les comportements associés lui permettront de se déplacer (tomber ou rouler).
Une entité Paroi permet de modéliser le sol et les murs sur lesquels les grains de sables s'accumulent ou s'écoulent. La paroi est fixe et n'est soumis à aucun comportement.

Le comportement de chute du grain de sable est une réaction située :

Comportement de chute :

Comportement	Situation			p	Action		
Chute				0,5			
		Grain					
		vide			Grain		

Ce comportement signifie que si du vide se trouve sous le grain de sable, celui-ci est déplacé vers le bas. Il est important de noter ici que le grain de sable de l'action n'est pas le grain de sable de la situation. Il s'agit d'une suppression pour le grain de la situation et d'une création pour le grain de l'action.

Le comportement de roulement est la capacité du grain de sable à se déplacer horizontalement sur d'autres grains de sables.

Comportement de roulement à gauche :

Comportement	Situation			p	Action		
Roule_gauche				0,1			
	vide	Grain			Grain		
		Grain				Grain	

Ce comportement signifie que si du vide se trouve à gauche d'un grain de sable reposant sur un autre grain de sable, celui-ci est déplacé vers la gauche.

Comportement de roulement à droite :

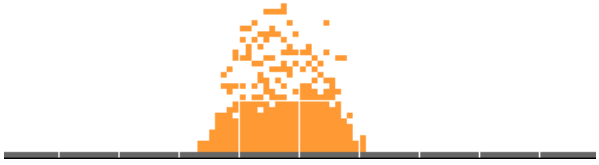
Comportement	Situation			p	Action		
Roule_droite				0,1			
		Grain	vide				Grain
		Grain				Grain	

Ce comportement signifie que si du vide se trouve à droite d'un grain de sable reposant sur un autre grain de sable, celui-ci est déplacé vers la droite.

Remarquer que la probabilité de rouler à droite ou à gauche n'est que de 0,1 car le grain de sable a d'abord tendance à rester sur place lorsqu'il repose sur un autre grain de sable.

Soit la situation de départ avec 300 grains de sable dans le vide et une paroi horizontale pour représenter le sol.

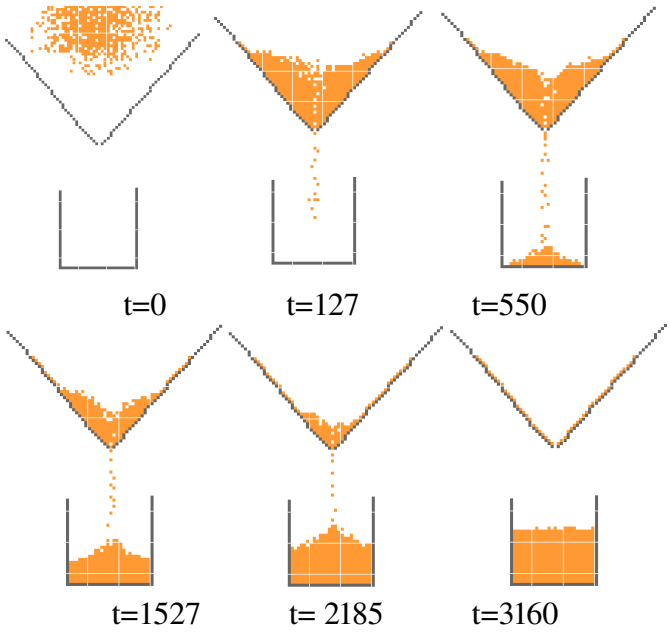
Après 157 pas de simulation, le tas de sable commence à apparaître.



Le tas s'aplatit au fur et à mesure (t=642) :

Cependant après un certain temps le tas de sable s'aplatit jusqu'à être totalement plat. Il s'agit ici de la limite de cette simulation ou les déplacement horizontaux à gauche ou à droite ne prennent pas suffisamment en compte le contexte du grain de sable pour savoir s'il peut réellement bouger ou bien s'il est coincé.

Il est possible de modifier les parois pour, par exemple, réaliser un sablier :



Switch du phage lambda

Dans la bactérie E. Coli infectée par le virus phage lambda, deux états sont possibles : lythique ou lysogénique. L'apparition d'un état plutôt qu'un autre est l'aboutissement de réactions se déroulant au niveau de l'ADN de la bactérie et plus précisément au niveau des gènes cI et cro. Le gène cI exprime la protéine CI et le gène cro la protéine CRO. Un mécanisme de rétroaction positive est observé autour des séquences or1, or2 et or3. Si la protéine CRO se fixe sur la séquence or3, alors seule la protéine CRO peut être retranscrite. De le même manière, si la protéine CI est fixée sur la séquence or1, alors seule la protéine CI peut être retranscrite.
Le mécanisme de transcription du gène cro n'est possible que si une protéine CRO est fixée sur le site or3.
Le mécanisme de transcription du gène CI n'est possible que si une protéine CI est fixée sur le site or1.

Ce système induit au bout d'un certain temps soit l'expression exclusive de CRO (état lysogénique) soit l'expression exclusive de CI (état lythique).

Pour la simulation, les entités suivantes sont créées :

- site_or321 de mobilité nulle et de 1/2 vie infinie. Cette entité regroupe les 3 sites or1, or2 et or3.
- gène_c1 de mobilité nulle et de 1/2 vie infinie.
- gène_cro de mobilité nulle et de 1/2 vie infinie.
- la protéine C1 de mobilité valant 1 et de 1/2 vie égale à 1000.
- la protéine CRO de mobilité valant 1.0 et de 1/2 vie égale à 1000.

Un complexel seq_ADN regroupant respectivement le gène_CI, le site_or321 et le gène_cro se touchant est créé.

Afin de reproduire les mécanismes connus du switch du phage lambda, deux réactions semi-situées sont créés, l'un concernant la transcription de la protéine CRO et l'autre sur la transcription de la protéine CI :

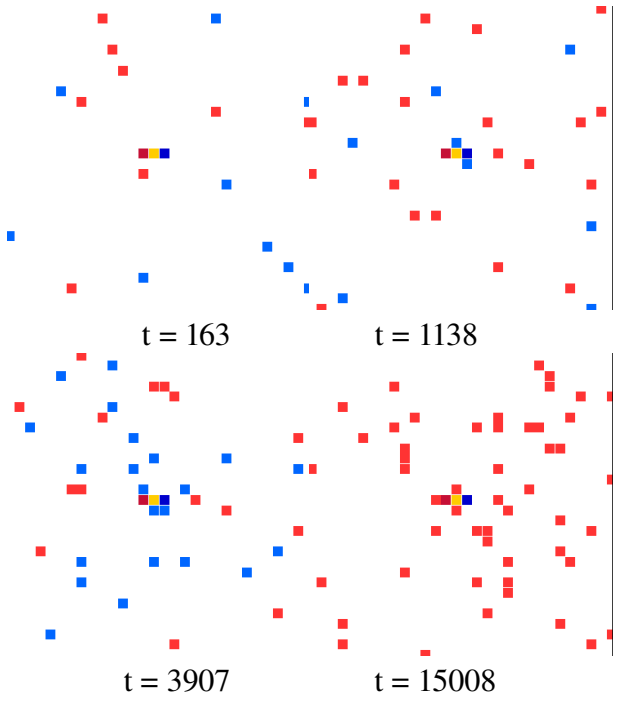
- trans_CRO : (site_or321) + gene_c1 + gene_cro + CRO + vide =0.1=> (site_or321) + gene_C1 + gene_CRO + CRO + CRO
- trans_C1 : (site_or321) + gene_c1 + gene_cro + C1 + vide =0.1=> (site_or321) + gene_c1 + gene_cro + C1 + C1

Le comportement trans_CRO indique qu'une protéine CRO est transcrite si le site_or321, entouré du gène_cI et du gène_cro, est fixé par une protéine CRO et seulement elle (vide).

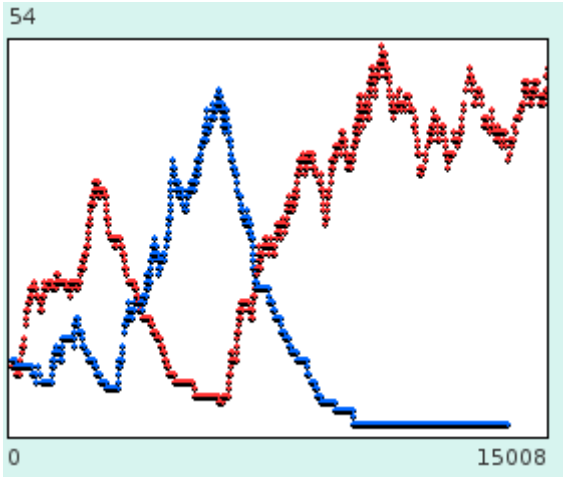
Le comportement trans_CI indique qu'une protéine CI est transcrite si le site_or321, entouré du gène_cI et du gène_cro, est fixé par une protéine CI et seulement elle (vide).

La mise en place de l'état initial nécessite le placement de quelques molécules CRO et CI pour initier les réactions semi-situées décrites précédemment. En effet, dans ce système simplifié la transcription de CRO ou de CI ne peut s'effectuer que si respectivement une protéine CRO ou une protéine CI se fixe sur le site_or321. La figure suivante montre l'état initial de la simulation dans une grille de 30x30 :

La simulation donne les résultats suivants :

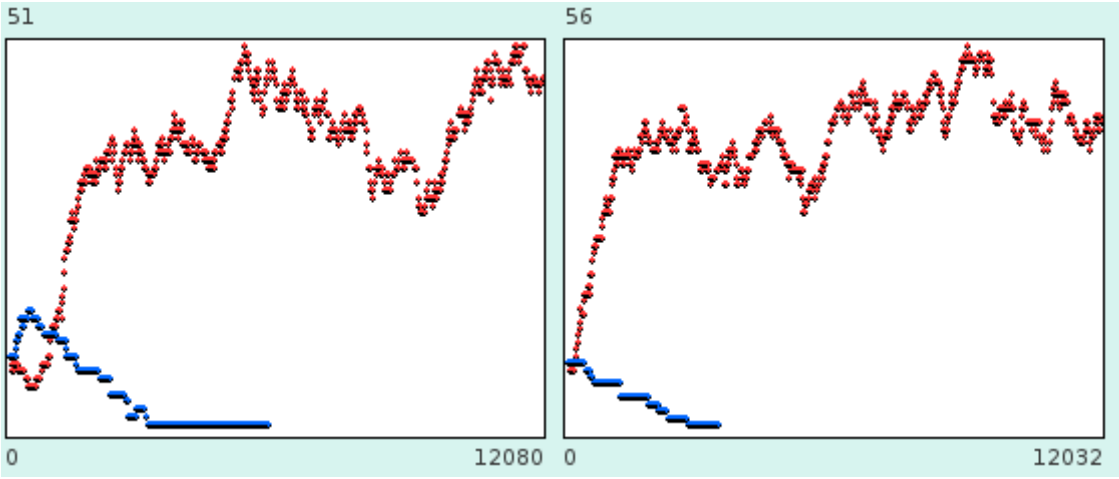


Les courbes du nombre de protéines CRO (bleu) et CI (rouge) au cours du temps pour cette simulation sont :

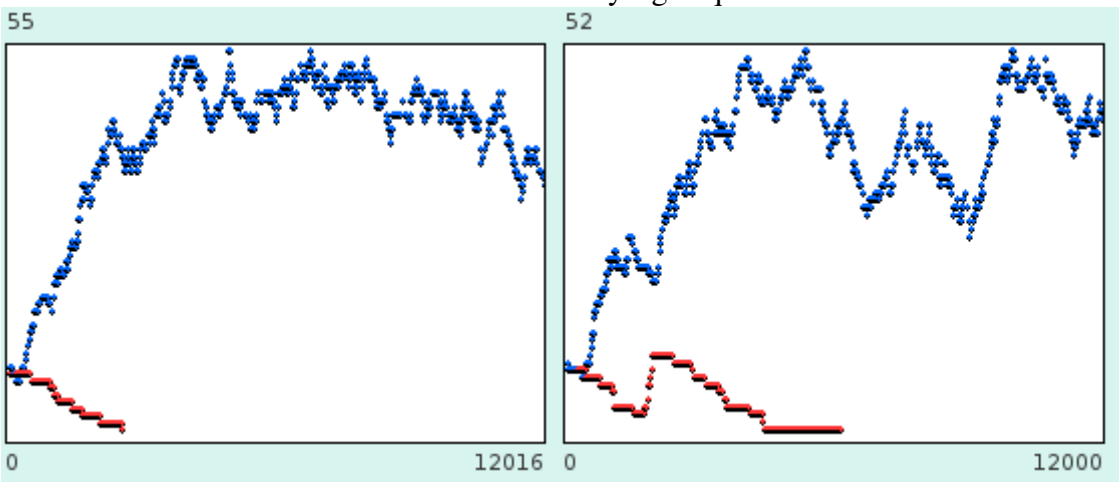


Le switch est effectif à 14000 puisqu'à cette date plus aucune protéine CRO n'est présente dans la simulation.
Une période d'instabilité est observée initialement, montrant le caractère probabiliste de la simulation.

Voici des courbes du nombre de protéines CRO (bleu) et CI (rouge) issues d'autres simulations avec le même état initial :



Switch dans l'état lysogénique



Switch dans l'état lytique

Evolution d'une population de scarabées

Cet exemple illustre le comportement de scarabées dans un environnement de 10 mètres par 10 mètres composé de haies, de prairies et de chemins. Deux haies de 10 m x 0,5 m sont disposées autours de la prairie.
1 pas de simulation = 1 heure, la grille fait 100x100 (donc 1 case = 10 cm x 10 cm).

Chaque état des entités correspond à un agent particulier. Par exemple, le scarabée dans la haie est représenté par l'entité *scarab_haie* et le scarabé dans la prairie est représenté par l'entité *scarab_prairie*.

Les haies, la prairie et le chemin sont représentés à l'aide d'entités fixes qui sont disséminées dans l'environnement.

Les comportements sont représentés sous la forme de réactions. Par exemple pour indiquer qu'un scarabée change s'il passe de la prairie à la haie, cela s'écrit : *scarab_prairie* + *haie* => *scarab_haie* + *haie*

Pour simplifier le problème les scarabées mâles et les scarabées femelles ne sont pas differenciés .

Dans les haies, les scarabées :
- se déplacent lentement : 5 cm / h en moyenne, soit une probabilité de déplacement égal à 0,5.
- se reproduisent fréquemment : 1 chance sur 10 par rencontre entre deux scarabées. Le comportement est alors :

(*scarab_haie)+ scarab_haie =0,1=> (scarab_haie) + scarab_haie + scarab_haie*

- et vivent longtemps : 720 heures = 30 jours en moyenne, soit une demi-vie de 720.

Dans les prairies, les scarabées :
- se déplacent rapidement : 10 cm / h en moyenne, soit une probabilité de déplacement égal à 1.
- ne se reproduisent pas
- et vive moins longtemps que dans les haies (300 h à cause des prédateurs).

Dans les chemins, les scarabées :
- se déplacent rapidement (10 cm / h en moyenne),
- ne se reproduisent pas
- et vive peu de temps à cause des nombreux piétons et prédateurs (50 h).

Auto-répartition de récepteurs membranaires

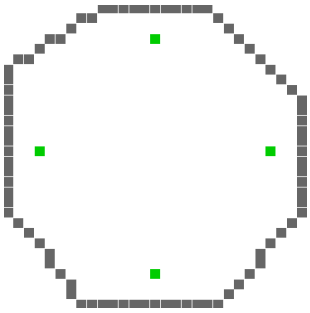
L'étude de la migration cellulaire peut s'effectuer grâce à deux SMA l'un fonctionnant en relation avec l'autre. Le premier SMA est celui qui régule la répartition des récepteurs membranaires en fonction des signaux d'adhésion reçu par la cellule virtuelle. L'autre, correspond à un modèle de cellule virtuelle déformable dont les capacité d'adhésion dépendent de la répartition des récepteurs du premier SMA.
Le SMA de répartition des récepteurs en fonction des signaux d'adhésion provenant de l'environnement est composé de 2 types d'entités :
- R un récepteur membranaire mobile capable de se polymériser,
- N une molécule membranaire fixe et sensible au substrat adhésif. N a deux états : activée si elle est en contact avec du substrat adhésif (Na), ou inactivée sinon (N).

Les comportements permettant d'observer certains mécanismes de migration cellulaire comme la bascule et la variation de vitesse de migration en fonction de la densité de substrat peut se représenter avec les 3 comportements semi-situés qui suivent.

NA=NA+R : Na =d=>Na + R avec d la densité de substrat compris entre 0 et 1.
RvNA : R Migre vers Na avec l'intensité 0.4
RvR : R Migre vers R avec l'intensité 0.25

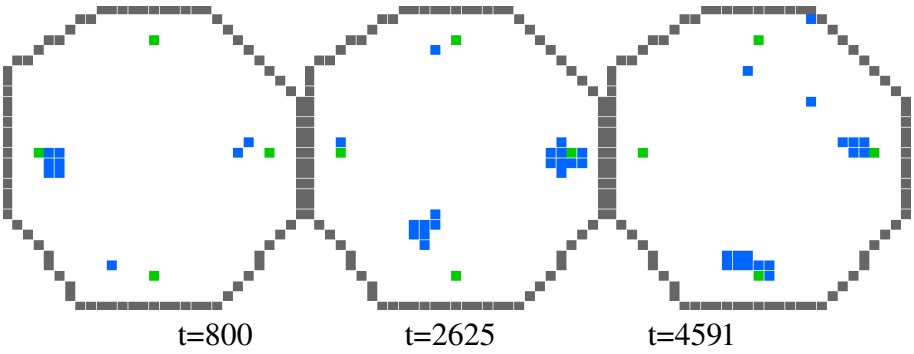
Simulation de la culbute :
On suppose ici une densité constante de substrat tout autour de la cellule.

d = 0 :
Aucune adhésion possible donc pas de migration ni de culbute.



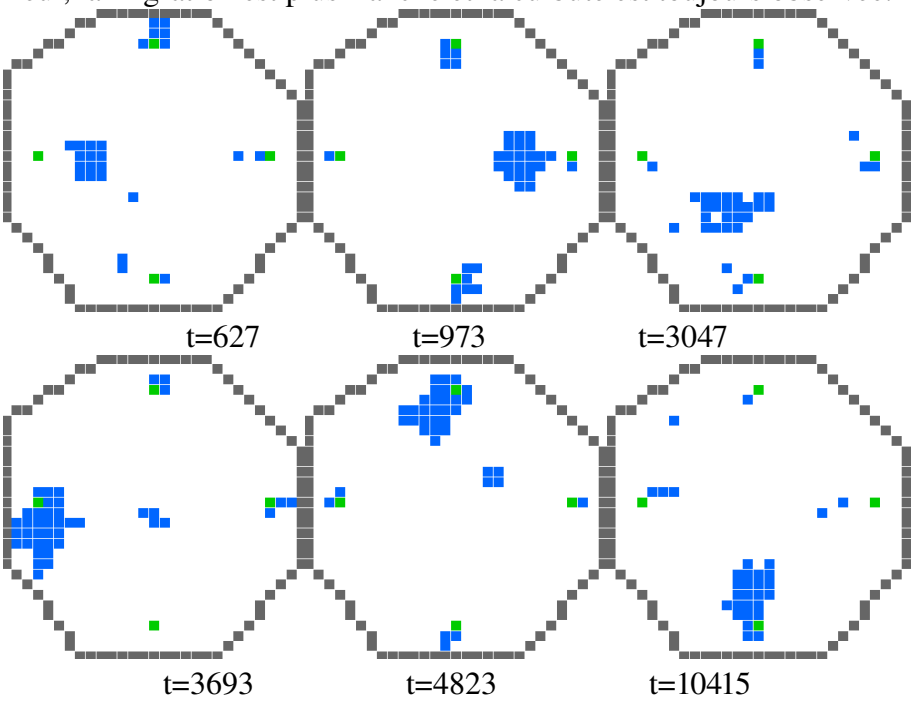
d = 0,01 :

La migration est faible et la culbute est observée.



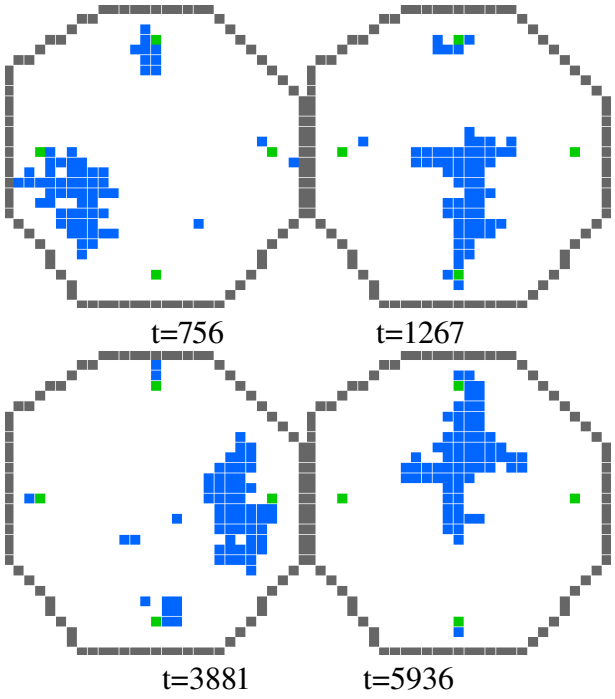
d = 0,02 :

L'adhésion est meilleur, la migration est plus franche et la culbute est toujours observée.

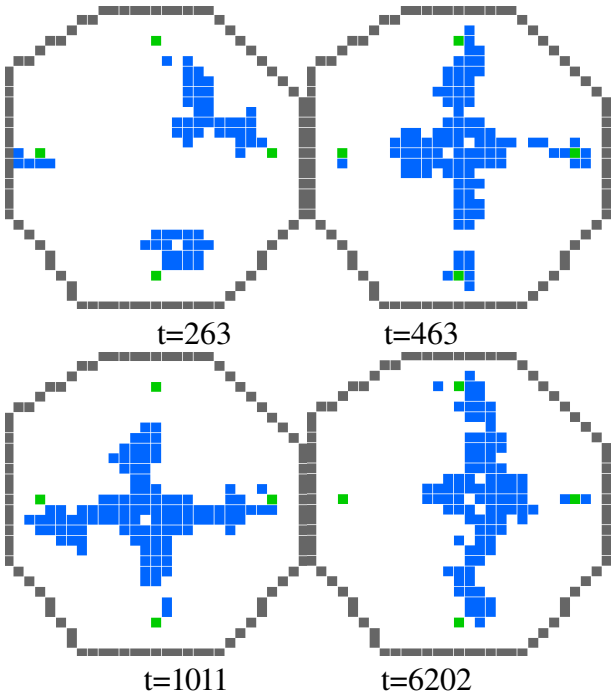


d = 0,04 :

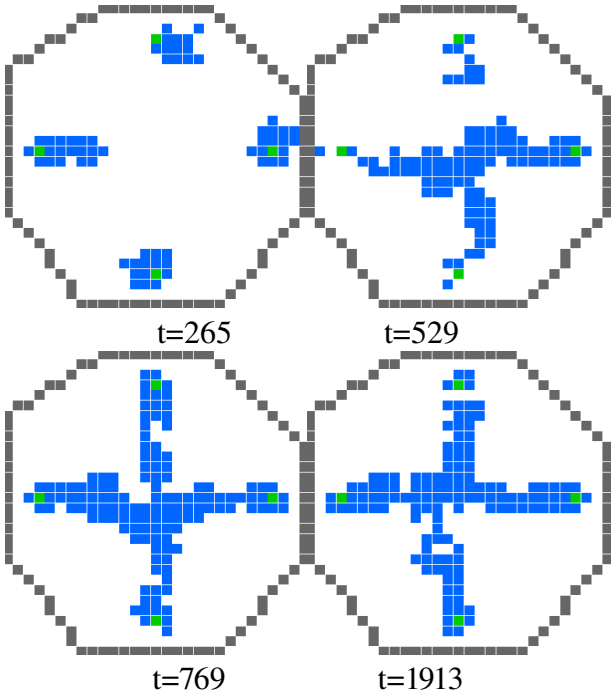
La vitesse de migration est maximale et la culbute est encore observée.



d = 0,08 :
La vitesse de migration chute et une certaine saturation des récepteurs membranaire est observée.



d = 0,16 :
La vitesse de migration est quasi nulle car la cellule virtuelle est saturée de récepteurs membranaires. La cellule a tendance à s'étaller sur le substrat.



Cette modélisation permet de reproduire simplement plusieurs phénomènes observés lors de la migration de cellules sur un substrat plus ou moins adhésif :

- la migration de la cellule virtuelle avec le phénomène de bascule
- la courbe en cloche de la vitesse de migration en fonction de la densité de substrat avec notamment la saturation lorsque trop de récepteurs membranaires sont recrutés
- l'étalement de la cellule virtuelle dans les zones de forte densité de substrat

Circuit logique

Pour réaliser un circuit logique avec BioDyn, il est possible, mais ce n'est sans doute pas l'unique solution, d'utiliser :
a) un fil ayant 2 états 0 ou 1 (absence ou présence de courant)
b) 2 crête de signaux + ou – (fronts de signaux)
c) une masse absorbant les crêtes de signaux
d) les trois portes logiques élémentaires ET, OU et NON

En terme d'entité BioDyn, cela donne les entités suivantes :

0 :	Mobilite =0.0,	1/2 Vie =infinie
1 :	Mobilite =0.0,	1/2 Vie =infinie
+ :	Mobilite =0.0,	1/2 Vie =infinie
- :	Mobilite =0.0,	1/2 Vie =infinie
ET :	Mobilite =0.0,	1/2 Vie =infinie
OU :	Mobilite =0.0,	1/2 Vie =infinie
NON :	Mobilite =0.0,	1/2 Vie =infinie
Masse :	Mobilite =0.0,	1/2 Vie =infinie

Ensuite il faut donner les comportements associés aux entités.

Le fil change d'état en fonction des crêtes de signaux qu'il reçoit. Une crête positive (+) fait passer le fil de

l'état 0 à l'état 1 (comportement Monter) et inversement une crête négative (-) fait passer le fil de l'état 1 à l'état 0 (comportement Baisser) :

Monter : (+) ⊕ 0 =1.0=> (1) ⊕ +
Baisser : (-) ⊕ 1 =1.0=> (0) ⊕ -

Une pseudo-masse[1] permet d'absorber les crêtes :

Masse_absorbe_+ : (Masse) ⊕ + =1.0=>Masse ⊕ 1
Masse_absorbe_- : (Masse) ⊕ - =1.0=>Masse ⊕ 0

Les comportements reproduisant les portes logiques sont descriptibles à l'aide des réactions situées. Il est par exemple possible de définir une porte logique à la position (xp,yp) et par convention de fixer les entrées e à gauche (en xp-1) et la sortie s à droite (en xp+1). Voici le détail pour chacune des trois portes.

Porte ET :

e1		
	ET	s
e2		

avec s = e1 ∧ e2

Porte OU :

e1		
	OU	s
e2		

avec s = e1 ∨ e2

Porte NON :

e	NON	s

avec s = Ø e

Pour la porte ET :

1 Il ne s'agit pas d'une masse au sens électrique du terme puisqu'ici il s'agit ici d'absorber les crêtes. Cette pseudo-masse n'est donc pas le niveau électrique 0.

Comportement	Situation			p	Action		
ET_11_0=11_+	1			1	1		
		ET	0			ET	+
	1				1		

Ce comportement signifie que si les 2 entrées sont à 1 et que la sortie est à 0, un signal de crête montante + est envoyé à la sortie.

Comportement	Situation			p	Action		
ET_0x_1=0x_-	0			1	0		
		ET	1			ET	-

Ce comportement signifie que si l'entrée e1 est à 0 et que la sortie est à 1, un signal de crête descendante - est envoyé à la sortie.

Comportement	Situation			p	Action		
ET_x0_1=x0_-				1			
		ET	1			ET	-
	0				0		

Ce comportement signifie que si l'entrée e2 est à 0 et que la sortie est à 1, un signal de crête descendante - est envoyé à la sortie.

Pour la porte OU :

Comportement	Situation			p	Action		
OU_00_1=00_-	0			1	0		
		OU	1			OU	-
	0				0		

Ce comportement signifie que si les deux entrées sont à 0 et que la sortie est à 1, un signal de crête descendante - est envoyé à la sortie.

Comportement	Situation			p	Action		
OU_1x_0=1x_+	1			1	1		
		OU	0			OU	+

Ce comportement signifie que si e1 est à 1 et que la sortie est à 0, un signal de crête montante + est envoyé à la sortie.

Comportement	Situation			p	Action		
OU_x1_0=x1_+				1			
		OU	0			OU	+
	1				1		

Ce comportement signifie que si l'entrée e2 est à 1 et que la sortie est à 0, un signal de crête montante + est envoyé à la sortie.

Pour la porte NON :

Comportement	Situation			p	Action		
NON_0_0=0_+				1			
	0	NON	0		0	NON	+

Ce comportement signifie que si l'entrée est à 0 et que la sortie est à 0, un signal de crête montante + est envoyé à la sortie.

Comportement	Situation			p	Action		
NON_1_1=1_-				1			
	1	NON	1		1	NON	-

Ce comportement signifie que si l'entrée est à 1 et que la sortie est à 1, un signal de crête descendante - est envoyé à la sortie.

Pour qu'une porte fonctionne correctement, il est nécessaire de lui adjoindre une masse afin d'absorber les crêtes. Une crête descendante est transformée en 0 et une crête montante en 1. Trois complexels permettent de terminer la mise au point des portes logiques.

Complexel porte_ET :

0		
Masse	ET	0
0		

Complexel porte_OU :

0		
Masse	OU	0
0		

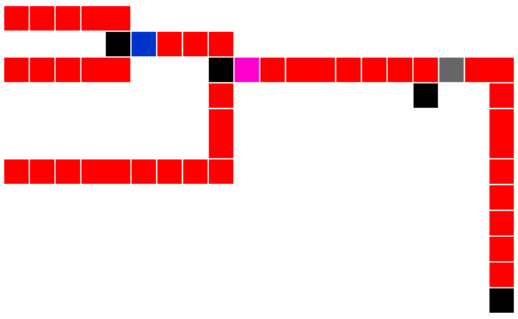
Complexel Porte_NON :

0	NON	0
Masse		

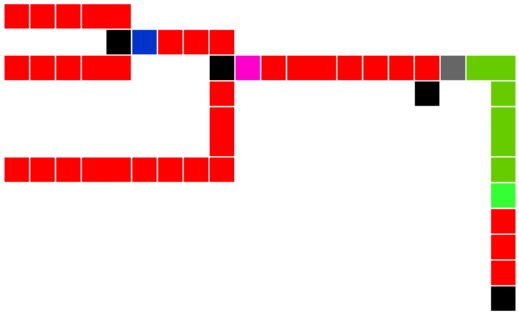
Soit le circuit logique à 3 entrées e1,e2 et e3 dont la sortie s est telle que :

$$s = Ø\ (\ (e1 \wedge e2) \vee e3).$$

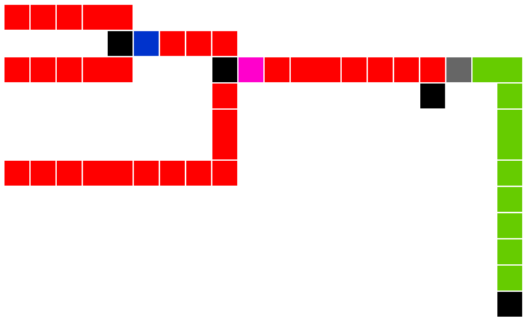
L'environnement à l'instant t=0 est composé d'une porte ET (bleu), d'une porte OU (rose), d'une porte NON (gris), de quatre pseudo-masse (noir) et de fils à l'état 0 (rouge) :



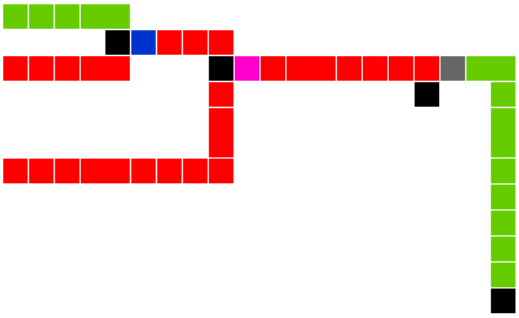
A l'instant t=7, la porte NON est en train de transformer le signal 0 (rouge) de son entrée en signal 1 (vert) à sa sortie. Noter la crête montante (vert clair) permettant de diffuser le fil 1 (vert) au détriment du fil 0 (rouge).



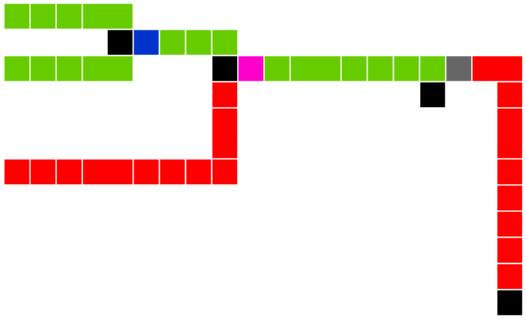
A partir de t=11, le circuit est stable et la crête a été absorbée par la pseudo-masse (noir) en bas à droite :



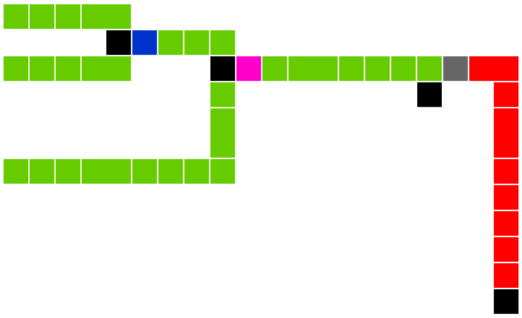
A t=12, une crête montante + est placé sur e1, entrainant à t=16 le circuit dans l'état suivant :



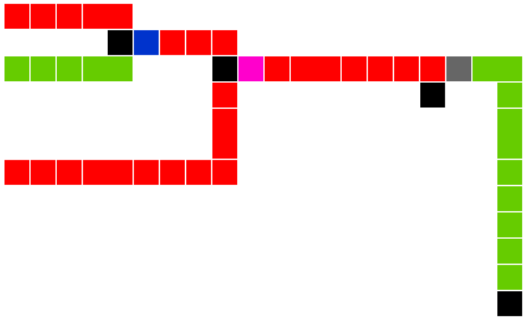
A t=17, une crête montante + est placé en e2, entrainant à t=44 le circuit dans l'état suivant :



A t=45, une crête montante est mise en e3, plaçant l'environnement à t=56 dans l'état suivant :



A t=57, deux crêtes descendantes sont placées en e1 et e3, entraînant le circuit dans l'état suivant à t= 87 :



Avec ce système, il est envisageable de créer toute sorte de circuits logiques. Grâce à l'utilisation des complexels il est possible de créer divers composants logiques de haut niveau que l'on peut à loisir agencer les uns avec les autres. Voici comme exemple le circuit logique représentant un additionneur 4 bits avec retenue.

La limite essentielle de ce système est l'impossibilité croiser des fils sans qu'ils se touchent. Ceci pourrait néanmoins être résolu en créant des entités Pont qui permettraient de transférer les crêtes en différenciant celles qui viennent horizontalement de celles qui viennent verticalement.

Robotique mobile multiagent

L'objectif est de modéliser et simuler un groupe de robots collecteurs de minerai sur un terrain accidenté. Les robots n'ont qu'une vision locale de leur environnement et sont incapable de détecter le minerai à plus d'un mètre. Ils connaissent néanmoins l'emplacement de l'usine de stockage de minerai et peuvent se diriger vers des balises radio qu'ils déposent ou reprennent eux-même.

Un robot possède deux état : *vide*, c'est à dire sans cargaison de minerai et *chargé*.
Ces deux états s'excluent mutuellement.

L'idée est de tester différents comportements de collecte afin de trouver les plus efficaces.

Un robot vide se déplace aléatoirement mais son déplacement est influencé par les balises radio vers lesquelles il tente de s'approcher.

Lorsqu'un robot vide rencontre du minerai, il s'en saisi, devient un robot chargé et dépose une balise radio pour informer et recruter d'autres robots.

Un robot plein se déplace vers l'usine de stockage en plus d'un mouvement aléatoire lui permettant de ne pas être trop facilement bloqué par un obstacle.

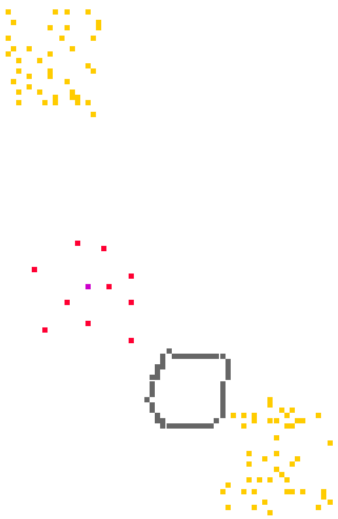
Si une balise est entourée d'au moins trois robots vides c'est qu'elle a jouée sont rôle de recruteur et elle est donc ramassé. C'est aussi un bon moyen pour relancer les robots vides dans leur recherche de minerai.

Les lignes qui suivent décrivent en détail les entités et leurs comportements associés.

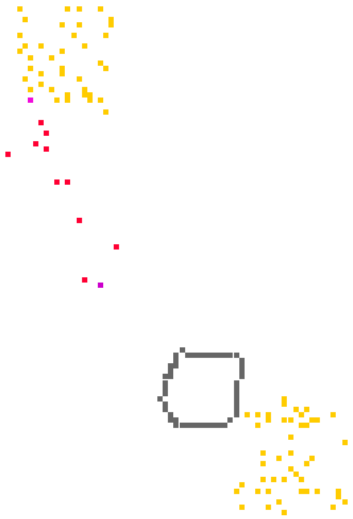
Entités:
R_vide:Mobilite =1.0, 1/2 Vie =infinie
R_charge:Mobilite =1.0, 1/2 Vie =infinie
Balise:Mobilite =0.0, 1/2 Vie =500.0
Obstacle:Mobilite =0.0, 1/2 Vie =infinie
Usine:Mobilite =0.0, 1/2 Vie =infinie
Minerai:Mobilite =0.0, 1/2 Vie =infinie

Comportements:
R_prend_Minerai : (R_vide) + Minerai =1.0=> (R_charge) + Balise
depot_usine : (R_charge) + Usine =1.0=>(R_vide) + Usine
enleve_balise : (Balise) + R_vide + R_vide + R_vide =1.0=>( ) + R_vide + R_vide + R_vide
rentrer_usine : R_charge Migre vers Usine avec la vitesse 0.5
R_vers_balise : R_vide Migre vers Balise avec la vitesse 0.5

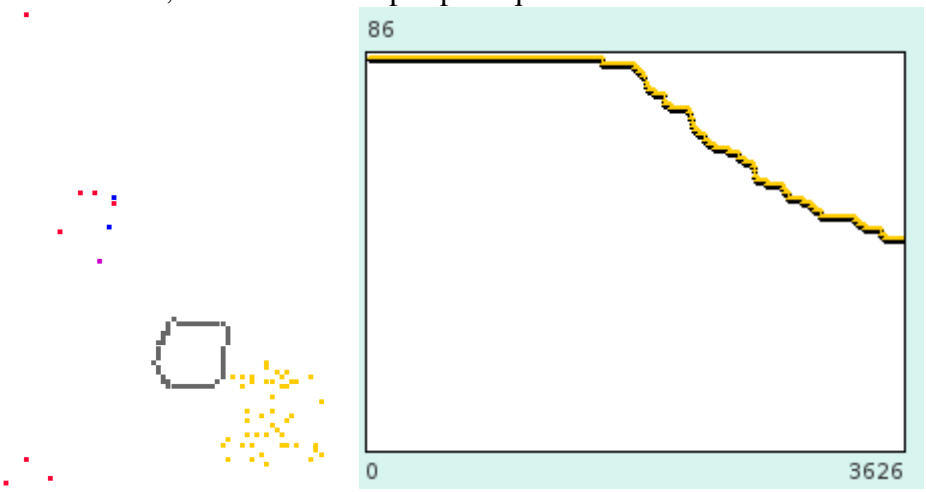
L'environnement initial est composé de l'usine de stockage du minerai (au centre en violet), des robots autour (rouge), du minerai (jaune-or) et d'obstacles (gris) :



Après un temps de 1685 pas de simulation, un robot trouve du minerai et dépose une balise. Le recrutement des robots est alors bien visible dans la simulation :

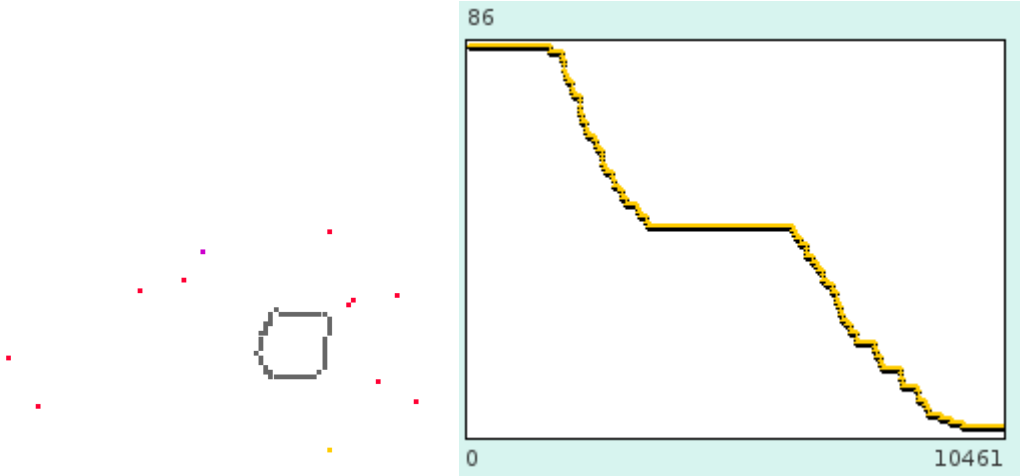


Grâce aux balises, la collecte du minerai est fortement accélérée. Alors qu'il a fallu attendre 1685 pas de simulation pour ramasser 1 minerai, il suffit de 1941 pas pour que les robots ramassent 41 minerai :



La courbe de droite indique la quantité de minerai subsistant dans l'environnement au cours du temps. La collecte est nulle le temps que l'un des robots trouve par hasard un minerai. Il dépose alors une balise qui permet de recruter d'autres robots. La collecte s'accélère alors fortement, montrant l'efficacité du procédé.

La collecte du second gisement est presque tout aussi rapide en dépit de la présence d'un obstacle :



Le dernier minerai a été ramassé à 14878 pas de simulation ce qui montre l'inadéquation de ce système pour la recherche de minerai isolé. Des améliorations comme des balises de type « zone déjà fouillées » pourraient permettre d'améliorer cette recherche en évitant aux robots de balayer plusieurs fois de suite la même zone géographique.

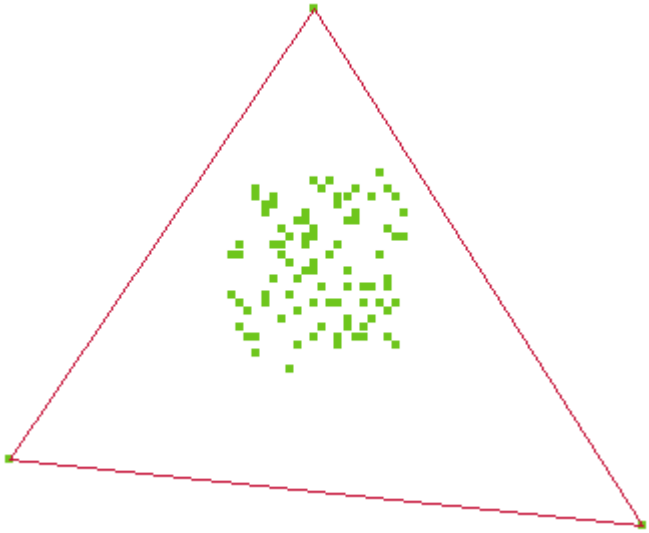
Exemples avec des réaxels et des connexels

Pattern fractals de cotes

Pour réaliser une courbe fermée fractale, il suffit :

- de créer un lien L imperméable et de longueur au repos fixée.
- de créer une entité E mobile et capable de s'insérer dans le lien L.
- de placer de nombreuses entités E à l'intérieur d'un polygone fermé de sommets E et de cotés L.

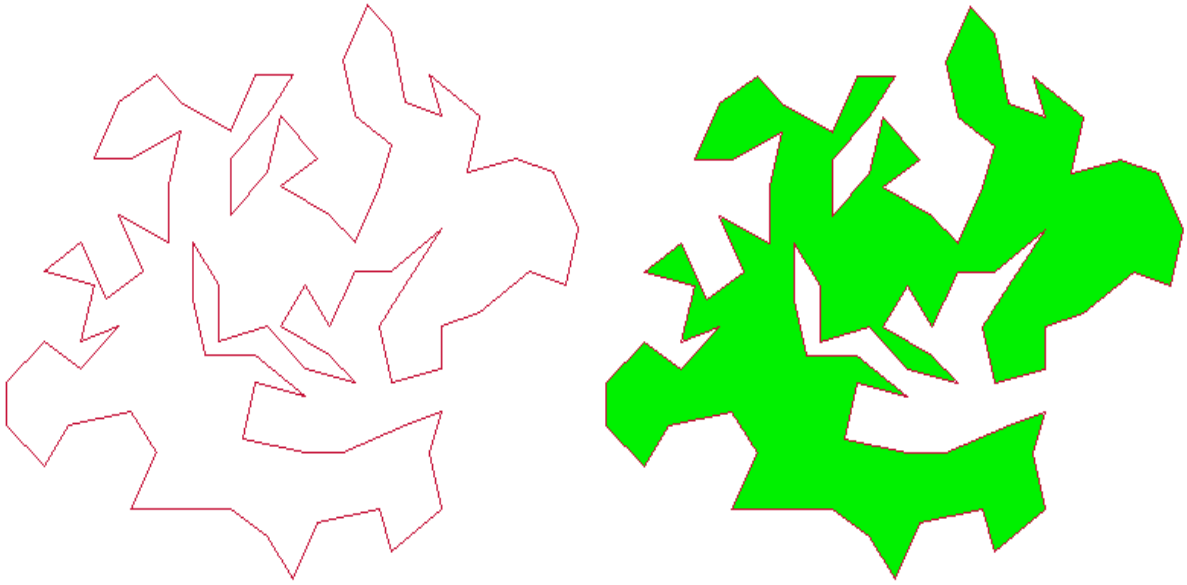
Dans l'exemple suivant, l'entité E a pour mobilité 1 et le lien L a pour longueur au repos 4. L'état initial comporte 90 E et 3 L disposés comme suit :



t = 0

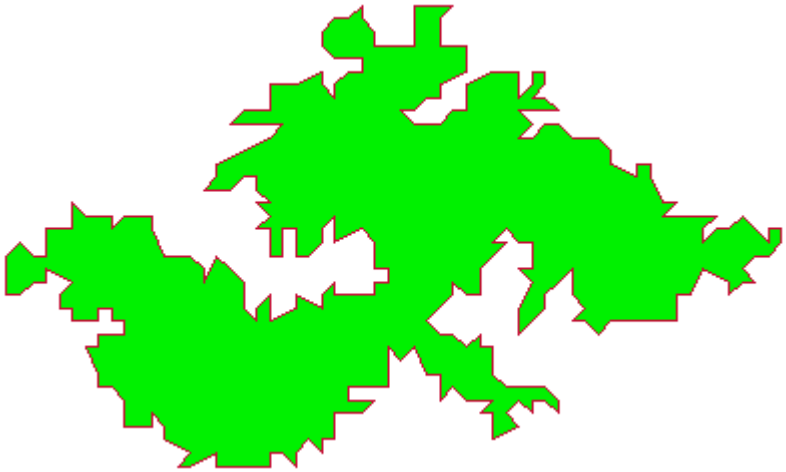
La simulation de ce système donne les évolutions suivantes :

t = 55 t = 120 t = 306 t = 454



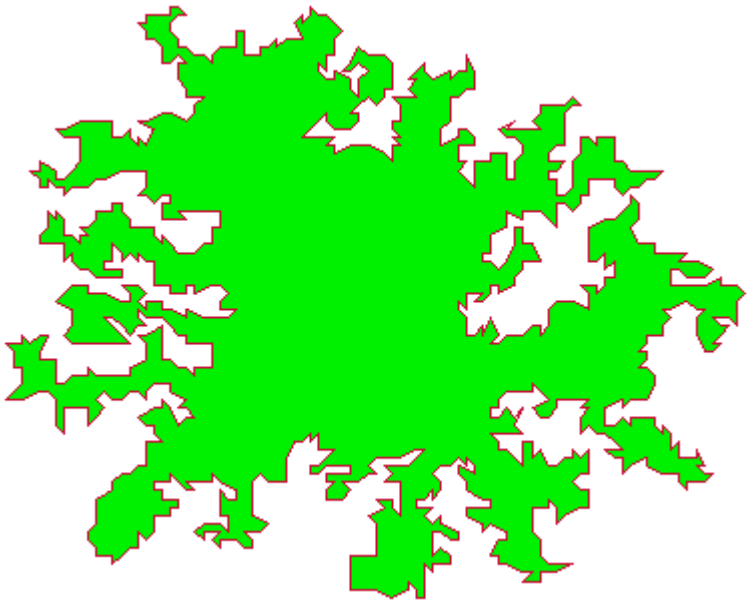
Vue vectorielle du résultat final (à gauche) et colorié pour distingué l'intérieur de l'extérieur (à droite). t = 454

Résultat avec 300 E, et une longueur au repos de 1 pour L :



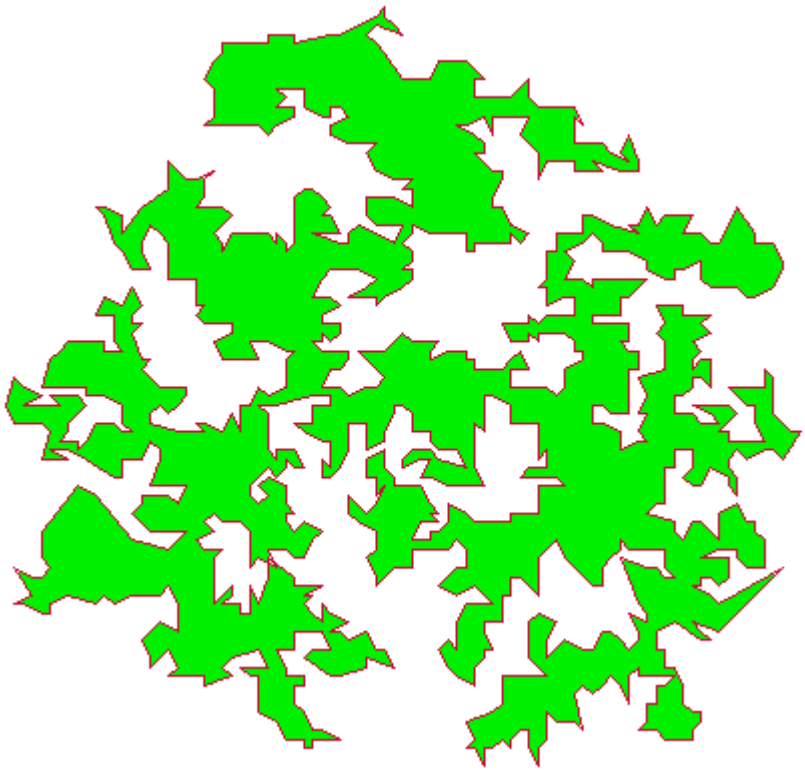
Résultat obtenu après 6269 pas de simulation.

Autre résultat avec 1000 E, et une longueur au repos de 1 pour L :



Résultat obtenu après 4752 pas de simulation.

Résultat avec 1000 E, et une longueur au repos automatique[2] pour L :



Résultat obtenu après 4756 pas de simulation. La figure obtenue est plus hétérogène que les précédentes car les longueurs au repos des liens n'est pas la même pour tous.

Formation d'un membrane mitohondriale

La membrane mitochondriale est composée d'une membrane externe et d'une membrane interne. La membrane interne de la mitochondrie exprime de nombreux replis. Dans cette simulation, une représentation simplifiée de ces deux membranes est effectuée selon les approximations suivantes :

2 Lors de l'insertion de E dans L, L est supprimé au bénéfice de L1 et de L2 avec L = L1+L2.

- La membrane externe est stable dans ses composants et donc ni ne croît ni ne décroît en terme de phospholipides.
- En revanche la membrane interne intègre de nouveaux phospholipides.
- Les nouveaux phospholipides sont produits de manière simplifiée par une entité ADN.
- Une entité H2O permet de représenter de manière explicite le milieu intra-mitochondrial.

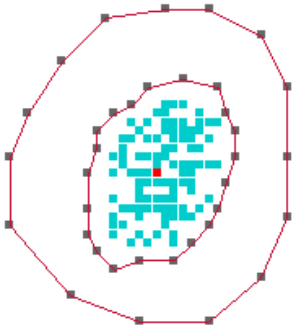
Pour résumer, les entités utilisées sont :

Réaxels
phospho : Mobilite =1.0, 1/2 Vie =infinie, intégrable dans L
ADN : Mobilite =0.1, 1/2 Vie =infinie
H2O : Mobilite =1.0, 1/2 Vie =infinie
Connexels
L : Raideur =1.0 L0 =4.0, 1/2 Vie =infinie, imperméable.

Le seul comportement utile pour cette simulation est la production de phospholipides par l'ADN :

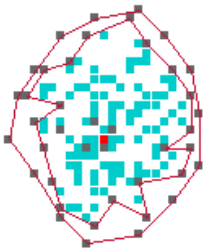
prod_phospho : ADN =0.1=>ADN + phospho

L'état initial est représenté sur le schéma suivant :



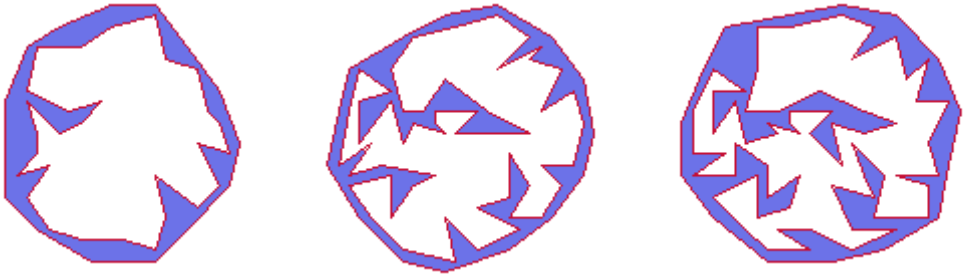
Sur ce schéma (t=0) nous distinguons les membranes externe et internes composées de reaxel *phospho* et de connexels *L.* A l'intérieur de la membrane interne se trouvent les entités H2O (bleu) et l'entité ADN (rouge). L'entité ADN produit des entités phospho de manière aléatoire (une chance sur 10 par pas de simulation).

Après 70 pas de simulation, le système équilibre les efforts liés aux entités L :

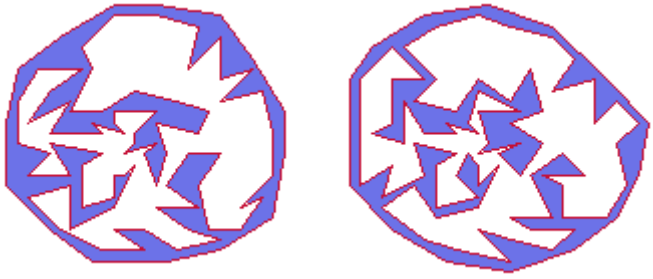


L'ADN produit des phospholipides (entité phospho) qui s'intègrent à la membrane interne lorsqu'ils entrent en contact.

Le système évolue alors selon les schémas suivants :



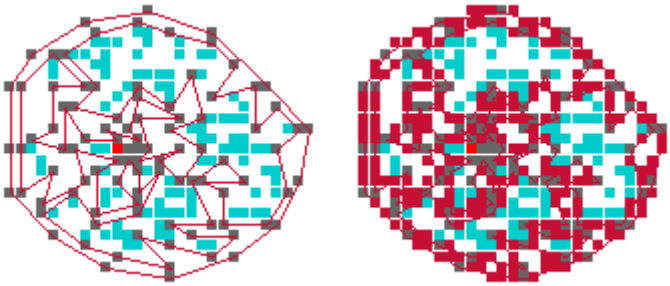
t = 320 t = 640 t = 947



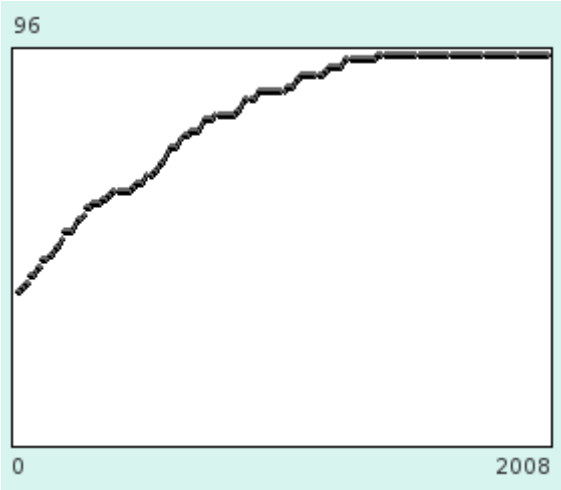
t = 1232 t = 2007

Plus le temps passe, plus la longueur de la membrane interne augmente avec l'intégration de nouveaux phospholipides. Cette longueur augmentant, la membrane interne se replie et crée ainsi de nombreux compartiments communicants de manière similaire à la membrane mitochondriale réellement observée.

L'état final (t=2007) est stable en terme de phospholipides. En effet, la matrice intra-mitochondriale est encombrée et ne permet plus la production de phospholipides :



En fin de simulation (t=2007), nous observons l'occupation de l'espace par les reaxels (à gauche) et par l'ensemble des réaxels et des connexels (à droite). Cette occupation empêche l'ADN (en rouge au centre) de produire de nouveaux phospholipides (courbe ci-dessous).



Cette courbe représente le nombre de phospholipides produits par l'ADN au court du temps. La courbe commence à la valeur 36 (à t=0). Ce nombre correspond aux phospholipides placés à l'état initial pour modéliser les membranes externe et interne. La population de phospholipides augmente mais de moins en moins vite car l'espace disponible est de plus en plus faible.

Il serait envisageable d'améliorer le système par une régulation génétique de la production des phospholipides au lieu de laisser l'espace s'en occuper. Il serait également possible d'ajouter des phénomènes biochimiques divers comme la chaîne respiratoire avec une distinction pertinente dans le positionnement des molécules qui pourrait être :

- interne à la membrane interne
- intra-membranaire, entre les deux membranes
- extra-mitochondrie, dans le milieu exterieur
- sur la membrane interne
- ou sur la membrane externe.