

# [TER\_M1\_2023] Rapport



## Les Créateurs:

Yahnis Saint-Val (yahnis.saint-val@etu.umontpellier.fr)

Adèle Imparato (adele.imparato@etu.umontpellier.fr)

Arthur Villaroya-Palau (arthur.villaroya-palau@etu.umontpellier.fr)

Florian Thepaut (florian.thepaut@etu.umontpellier.fr)



Simulation de l'évolution d'un écosystème  
de créatures dans un environnement 3D

# Table des matières

<b>A</b>	<b>Introduction</b>	<b>4</b>
A .1	Présentation du projet	5
A .2	Objectifs	5
A .3	Inspirations et articles scientifiques	6
<b>B</b>	<b>Méthodes et implémentation</b>	<b>7</b>
B .1	Simuler l'évolution	8
B .2	Environnement 3D	9
B .2.1	Terrain	9
B .2.1.1	Création des biomes	9
B .2.2	Nourriture	12
B .2.2.1	Arbres et apparitions des fruits	12
B .2.2.2	Nutritivité	14
B .3	Créatures	15
B .3.1	Structure des créatures	15
B .3.2	Rendu visuel des créatures	17
B .3.2.1	Génération du maillage	17
B .3.2.2	Plaquage du maillage	19
B .3.3	Code génétique des créatures	20
B .3.4	Mouvement des créatures	22
B .3.4.1	Oscillation des articulations	22
B .3.4.2	Contrôle du degré de stimulation	23
B .3.4.3	Synchronisation	24
B .3.5	Physique des créatures	25
B .3.5.1	Problèmes avec la physique de base	25
B .3.5.2	Solution	25
B .4	Simulation	26
B .4.1	Apparitions des créatures	27
B .4.2	Cycle de vie d'une créature	28
B .4.3	Énergie	29
B .4.3.1	Consommation	29
B .4.3.2	Nutrition et attaque	30

---

B .4.4	Paramètres comportementaux . . . . .	31
B .4.4.1	Agressivité . . . . .	31
B .4.4.2	Altitude préférentielle de la nourriture . . . . .	31
B .4.5	Reproduction et mutations . . . . .	32
B .4.5.1	Conditions de reproduction . . . . .	32
B .4.5.2	Mutations . . . . .	33
B .4.5.3	Crossovers . . . . .	36
B .4.6	Statistiques . . . . .	37
B .4.6.1	Graphiques . . . . .	37
B .4.6.2	Coloration des créatures . . . . .	38
B .4.6.3	Informations individuelles . . . . .	39
B .5	Pré-entraînements . . . . .	40
B .5.1	Apprendre à avancer . . . . .	41
B .5.2	Apprendre à tourner . . . . .	41
B .5.3	Aller plus vite . . . . .	42
B .5.4	Optimiser l'énergie consommée . . . . .	42
<b>C</b>	<b>Résultats et analyses</b>	<b>43</b>
C .1	Création des créatures et pré-entraînements . . . . .	44
C .1.0.1	Comment créer des créatures ? . . . . .	44
C .1.0.2	Présentation de nos créatures . . . . .	44
C .1.1	Créatures après entraînements . . . . .	45
C .1.1.1	Apprendre à avancer . . . . .	45
C .1.1.2	Apprendre à tourner . . . . .	47
C .1.1.3	Aller plus vite . . . . .	49
C .1.1.4	Optimiser l'énergie consommée . . . . .	51
C .2	Simulation . . . . .	54
C .2.1	Forêt . . . . .	54
C .2.2	Banquise . . . . .	55
C .2.3	Plaine . . . . .	56
C .2.4	Désert . . . . .	58
<b>D</b>	<b>Conclusion et perspectives</b>	<b>59</b>
<b>E</b>	<b>Annexes</b>	<b>61</b>

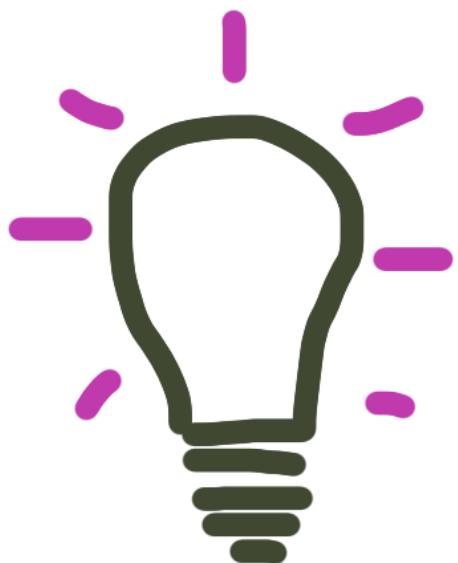


# Remerciements

Nous tenons à remercier nos encadrants, Marc Hartley et Noura Faraj, pour leur aide précieuse et leurs conseils éclairés tout au long du projet.

Ils ont su nous apporter détermination et positivité.

# Introduction





## A .1 Présentation du projet

Dans ce projet, nous visons à simuler un écosystème fictif de créatures évoluant dans un environnement 3D. On souhaite observer son évolution selon plusieurs critères, afin de mettre en lumière le phénomène de sélection naturelle.

L'idée est donc d'avoir des créatures capables d'évoluer, par mutations aléatoires, de génération en génération, et d'observer comment elles s'adaptent à différents environnements.

## A .2 Objectifs

En somme, l'objectif de ce projet est de mettre en lumière le concept de sélection naturelle et de voir s'il est possible de le reproduire dans un environnement virtuel en 3D. Nous espérons donc pouvoir observer quels comportements et structures permettent aux créatures de survivre dans quels environnements.

Nos objectifs seront donc :

- Modéliser des créatures virtuelles capables d'évoluer grâce à des mutations aléatoires.
- Concevoir un environnement virtuel en 3D propice à l'évolution et à l'observation des créatures.
- Combiner les créatures et l'environnement pour créer une simulation interactive.
- Observer les comportements et les structures des créatures qui favorisent leur survie dans des environnements spécifiques.

Nous avons réalisé le projet sous Unity 2021.

## A .3 Inspirations et articles scientifiques

L'idée d'un tel projet nous est d'abord venu d'un autre projet similaire, réalisé au premier semestre, qui simule l'évolution de créatures dans un environnement 2D. Vous pouvez trouver une présentation et des vidéos exemples [ici](#).

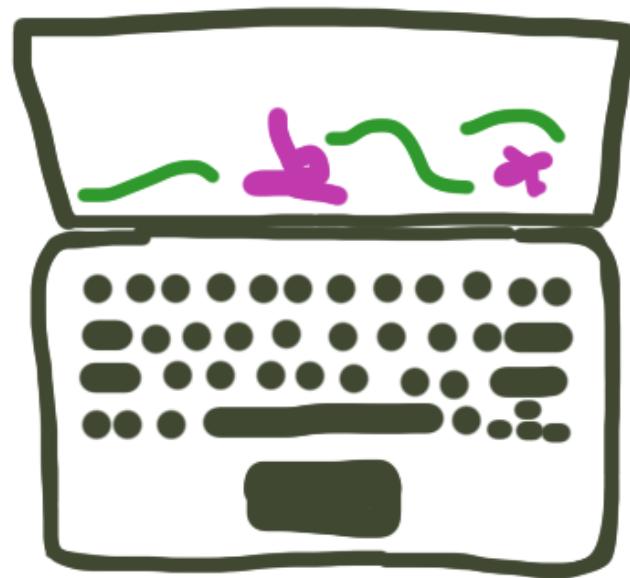
Nous avons donc souhaité reprendre l'idée de ce projet, et l'adapter à un environnement 3D.

Afin de mener à bien ce projet, il nous a fallu d'abord nous intéresser à des recherches similaires. C'est pourquoi nous avons parcouru plusieurs articles scientifiques se rapprochant de notre sujet pour en apprendre plus sur ce qui a déjà été fait :

- **"Creature Academy : A System For Virtual Creature Evolution"**, Marcin L. Pilat et Christian Jacob [\[1\]](#)  
Creature Academy présente un monde virtuel mettant en oeuvre des créatures fictives et évolutives au sein d'un environnement physique 3D. Ce rapport analyse plusieurs aspects de l'évolution comme les différentes morphologies des créatures, leurs comportements ou encore les différentes tâches d'entraînement.
- **"Evolving 3D Morphology and Behavior by Competition"**, Karl Sims [\[2\]](#)  
Article de Karl Sims, considéré comme une référence et détaillant une simulation d'évolution de créatures. Il propose une approche où l'ADN des créatures est modélisé par un graphe, et où ces dernières s'affrontent dans un environnement 3D pour le contrôle d'une ressource.
- **"Recent Developments in the Evolution of Morphologies and Controllers for Physically Simulated Creatures"**, Tim Taylor et al. [\[3\]](#)  
Cet article résume les avancées récentes en simulation évolutive 3D, en s'appuyant notamment sur une ré-implémentation moderne de la simulation de Karl Sims.

Nous comparerons les différentes parties de notre projet à ces différents papiers de recherche tout au long de ce rapport.

# Méthodes et implémentation





## B .1 Simuler l'évolution

On veut simuler l'évolution de créatures dans un environnement 3D. Il nous faut donc un environnement adapté, et des créatures capables d'évoluer.

Pour l'environnement, nous avons choisi de créer différents biomes<sup>1</sup>, avec différentes caractéristiques : cela nous permettra de tester plusieurs contextes et de voir comment cela influence l'évolution de nos créatures.

Pour les créatures, il nous faut choisir un modèle compatible avec de l'évolution, c'est à dire avoir des créatures dont les caractéristiques peuvent être modifiées, et passées à leur progénitures.

Dans un premier temps, nous allons détailler les caractéristiques de l'environnement. Ensuite, nous allons détailler comment nous avons modéliser nos créatures.

1. Dans ce projet, un biome fait référence à une zone géographique du terrain qui partage des caractéristiques physiques similaires (i.e. sol, types d'arbres, types de nourriture, conditions environnementales).

## B .2 Environnement 3D

### B .2.1 Terrain

L'environnement consiste en un large terrain 3D (Figure B .1) généré avec l'outil Terrain de Unity. Celui-ci est composé de 5 biomes, à savoir, la banquise, la forêt, la montagne, la plaine et le désert.

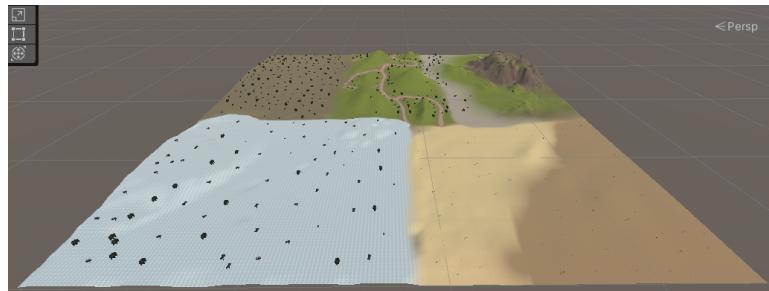


FIGURE B .1 – Terrain 3D constitué de 5 biomes : la banquise, la forêt, la montagne, la plaine et le désert.

#### B .2.1.1 Crédation des biomes

Ces cinq biomes sont utilisés comme cinq terrains indépendants lors de la simulation. C'est à dire que les créatures d'un biome à l'autre n'interagissent pas entre elles.

Les biomes se caractérisent par leur relief, leur texture et les arbres qui y sont placés (voir section B.2.2). Il est à noter que les textures sont purement décoratives, elles n'influencent pas le comportement des créatures.

De plus, nous avons fait en sorte que les biomes aient un impact sur le déroulement de la simulation, afin de différencier les simulations d'un environnement à l'autre.

Le biome banquise (Figure B .2) est assez grand, avec peu d'arbres. Globalement plat, il possède quelques buttes sur-élevées. Le climat froid et hostile va forcer les créatures à consommer plus d'énergie (voir partie B.4.3.1), et la reproduction sera plus difficile (voir partie B.4.5.1).

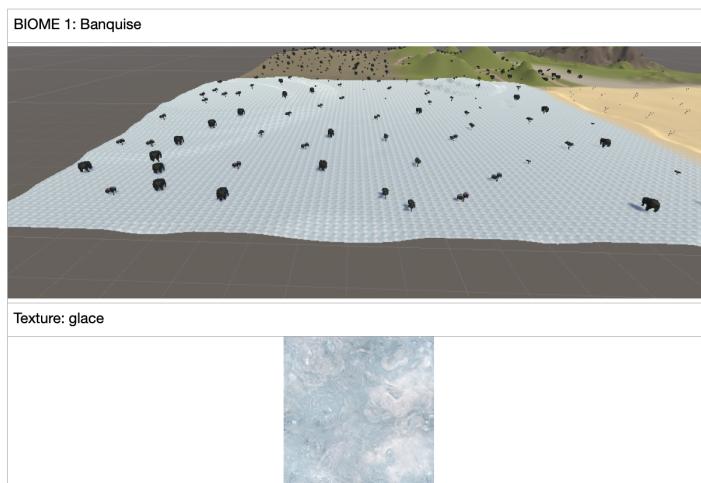


FIGURE B .2 – Aperçu du biome banquise et de sa texture.

Le biome désert (Figure B .3) est assez grand et complètement plat, avec peu d'arbres, mais leurs fruits sont très nutritifs. Le climat chaud et sec va forcer les créatures à consommer plus d'énergie, mais la reproduction sera facilitée.

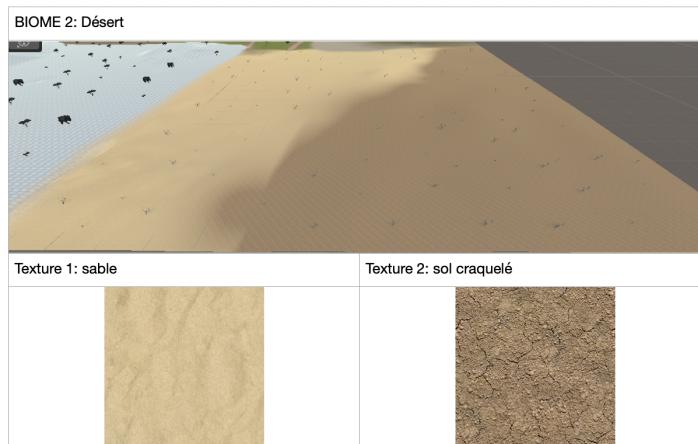


FIGURE B .3 – Aperçu du biome désert et de ses textures.

Le biome plaine (Figure B .4) est assez petit. Il composé de collines entre lesquelles passent des chemins plats, le long desquels sont concentrés les arbres.



FIGURE B .4 – Aperçu du biome plaine et de ses composantes.

La montagne (Figure B .5) est là à titre indicatif et décoratif. Sa topologie rendra bien trop dur les déplacements de nos créatures.

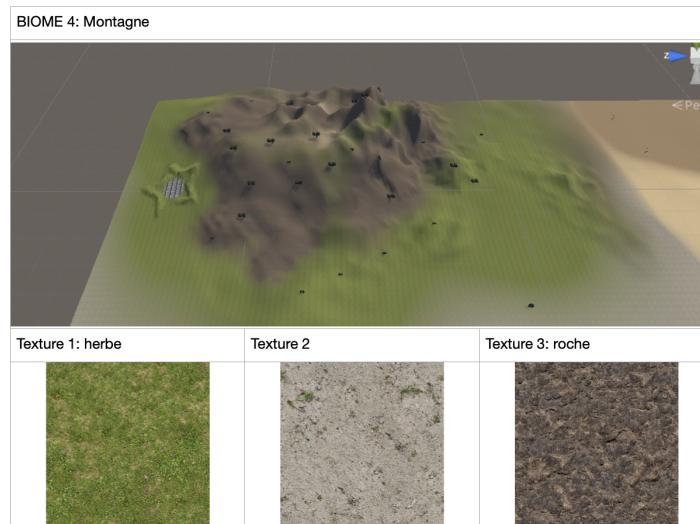


FIGURE B .5 – Aperçu du biome montagne et de ses textures.

Le biome forêt (Figure B .6) est assez petit, mais riche en arbres et donc en nourriture.

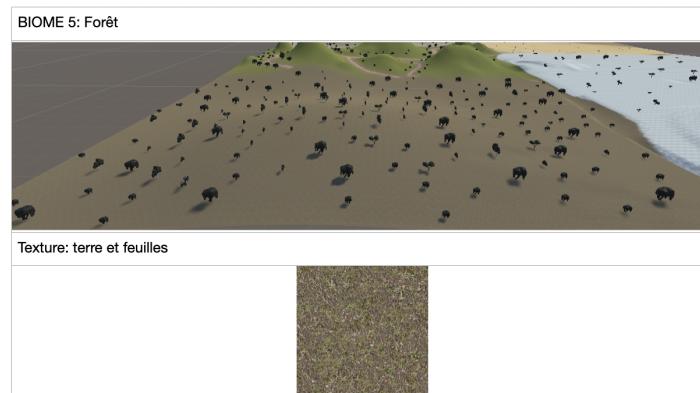


FIGURE B .6 – Aperçu du biome forêt et de ses textures.

## B .2.2 Nourriture

### B .2.2.1 Arbres et apparitions des fruits

Les arbres sont des éléments essentiels pour nos simulations. Pour chaque biome, nous aurons un type d'arbre spécifique avec différentes caractéristiques. Ces caractéristiques sont notamment le nombre de fruits que peut avoir l'arbre, la nutritivité de ces fruits ainsi que le délai pour qu'un nouveau fruit apparaisse sur l'arbre.

Les différentes espèces d'arbres sont les suivantes :

L'arbre du désert (Figure B .7) ne peut avoir qu'un fruit à la fois, mais son fruit est le seul à avoir une nutritivité plus élevée que celle de base.

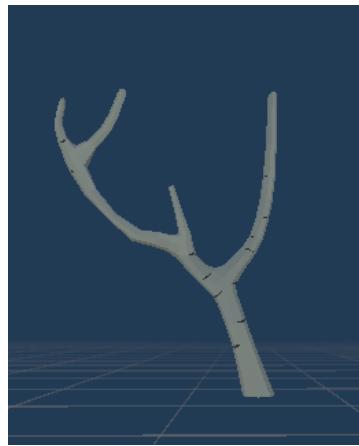


FIGURE B .7 – Arbre du désert

L'arbre enneigé (Figure B .8) peut avoir jusqu'à 2 fruits à la fois. Ses fruits ont une nutritivité de base normale.



FIGURE B .8 – Arbre enneigé

L'arbre du marais et de la forêt (Figure B .9) peut avoir jusqu'à 4 fruits à la fois. Ses fruits ont une nutritivité de base normale.



FIGURE B .9 – Arbre du marais et de la forêt

L'arbre de la plaine (Figure B .10) peut avoir jusqu'à 3 fruits à la fois. Ses fruits ont une nutritivité de base normale.

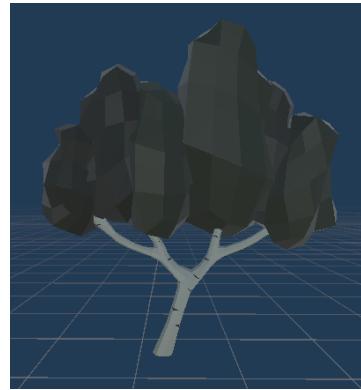


FIGURE B .10 – Arbre de la plaine

L'arbre de la montagne (Figure B .11) peut avoir jusqu'à 7 fruits à la fois. Ses fruits ont une nutritivité de base normale.



FIGURE B .11 – Arbre de la montagne

Il faut également noter que nous avons fait des versions "minis" pour avoir des fruits plus proches du sol et faciliter l'accès des fruits aux créatures.

Les fruits apparaissent sur des points fixes spécifiques à chaque arbre. Ils sont définis à la main avec Unity dans chaque prefab de chaque arbre. Après un certain délai un fruit apparaît si un point d'apparition est vide. Quand un fruit apparaît il est rajouté dans un "manager" qui aura les informations de chaque fruit dans la scène. Quand un fruit est mangé, il disparaît de l'arbre, est détruit dans le manager et le point d'apparition où il était redévenu vide.

Vous pouvez trouver des vidéos illustrant des apparitions de fruits en suivant ce [lien](#).

### B .2.2.2 Nutritivité

La nutritivité d'un fruit est un élément essentiel dans le développement d'une créature. Cette nutritivité va influer sur l'énergie que va recevoir la créature lorsqu'elle mange ce fruit (voir section B.4.3.2). La nutritivité d'un fruit est défini par deux paramètres : l'arbre sur lequel se trouve le fruit et sa hauteur par rapport au sol.

Plus un fruit est haut, plus il est nutritif. La nutritivité d'un fruit est régit par la formule suivante :

$$\text{nutritivite} = \text{nutritivite\_fruit} \times \text{nutritivite\_arbre} \times \text{coef\_hauteur}$$

$$\text{coef\_hauteur} = \text{coef\_min} + ((\text{coef\_max} - \text{coef\_min}) \times (\text{hauteur\_fruit}/\text{hauteur\_max}))$$

"nutritivite\_fruit" : nutritivité de base du fruit

"nutritivite\_arbre" : coefficient de nutritivité de l'arbre

"coef\_min" et "coef\_max" : coefficients de hauteur minimum et maximum

"hauteur\_fruit" : hauteur du fruit

"hauteur\_max" : hauteur maximale d'un fruit

La couleur de chaque fruit dépend de sa nutritivité afin d'avoir un visuel plus pratique et esthétique.



FIGURE B .12 – Échelle de couleur du moins au plus nutritif



FIGURE B .13 – Exemple de fruits sur un arbre

## B .3 Créatures

### B .3.1 Structure des créatures

Nos créatures sont définies par un arbre d'articulations. Chaque créature se compose d'une boule centrale, à laquelle sont rattachées plusieurs articulations, auxquelles d'autres articulations peuvent être rattachées à leur tour, et ainsi de suite. Cette structure nous permet d'imaginer un large éventail de créatures possibles, allant du simple ver à l'arthropode doté de multiples membres.

Il est possible de représenter chaque créature par un arbre. Par exemple, pour une créature ressemblant à un ver (Figure B .14), l'arbre correspondant pourrait être le suivant (deux membres attachés l'un à l'autre) :

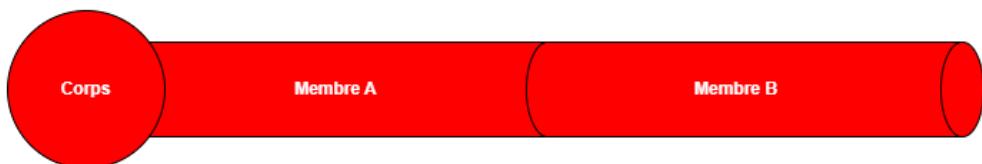
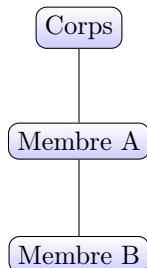
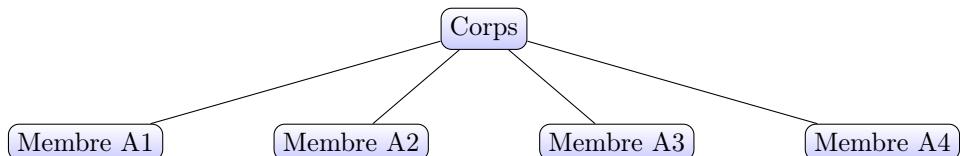


FIGURE B .14 – Schéma d'une créature ayant la forme d'un ver

Pour une créature ressemblant à une araignée (Figure B .15) et dotée de 4 membres attachés à son corps, on pourrait plutôt avoir l'arbre suivant :



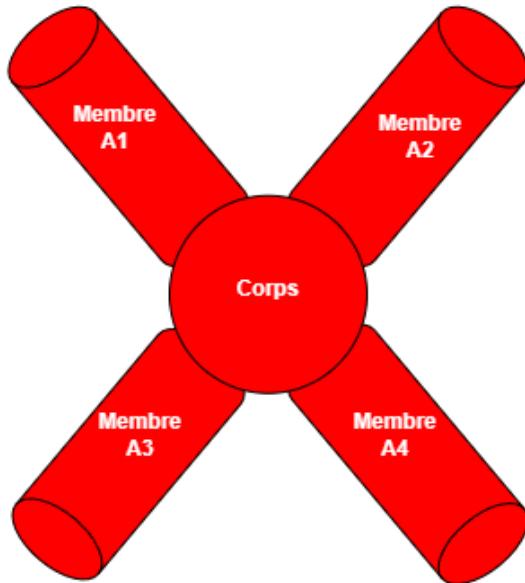


FIGURE B .15 – Schéma d'une créature ayant la forme d'une araignée

Dans son article "Evolving 3D Morphology and Behavior by Competition"[\[2\]](#), Karl Sims utilise un graphe orienté pour modéliser la structure de ses créatures. En général, ces graphes commencent par un nœud "racine" qui ressemble à ce que pourrait être le corps de nos créatures. Les nœuds peuvent se connecter à eux-mêmes et être connectés plusieurs fois à un noeud enfant pour créer une duplication de cette connexion.

## B .3.2 Rendu visuel des créatures

### B .3.2.1 Génération du maillage

Pour "afficher" nos créatures, il faut pouvoir générer un visuel procéduralement, afin que n'importe quel arrangement d'articulations puisse être représenté.

Une solution simple aurait été de placer une sphère ou une autre forme simple au niveau de chaque articulation ; mais une telle représentation permet difficilement d'afficher des articulations ayant des longueurs variables, sans générer de "trous" entre les sphères trop éloignées, ou de superpositions entre les sphères trop proches.

Nous avons opté pour une autre solution, plus complexe mais mieux adaptée : on va représenter les "membres", et non les articulations. Un membre est simplement un lien entre deux articulations.

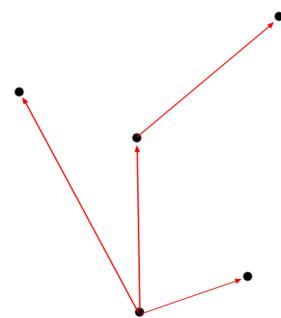
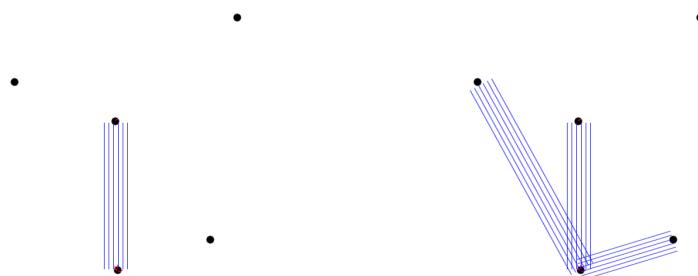


FIGURE B .16 – Exemple d'arborescence d'articulations

Chaque membre est affiché par un cylindre, plus ou moins long, reliant une articulation parent à une articulation enfant.

Pour chaque articulation, on génère donc un cylindre en direction de chacune de ses articulations enfants ; le cylindre est "extrudé" à partir de l'articulation parent, en direction de l'articulation enfant.



(a) Génération d'un cylindre entre l'articulation racine et son premier enfant

(b) Génération des cylindres pour les autres enfants

FIGURE B .17

Ensuite, pour chaque cylindre, on relie son extrémité 'début' à l'extrémité 'fin' du cylindre de l'articulation parent (qui aura préalablement été généré avec son parent), afin de conserver une continuité dans le maillage.

Une articulation qui n'a pas d'enfant (feuille de l'arbre) n'engendre pas de membre : on ne va donc

pas utiliser de cylindre pour la représenter. On va simplement "fermer" le maillage, et fermant l'extrémité 'fin' du cylindre de l'articulation parent.

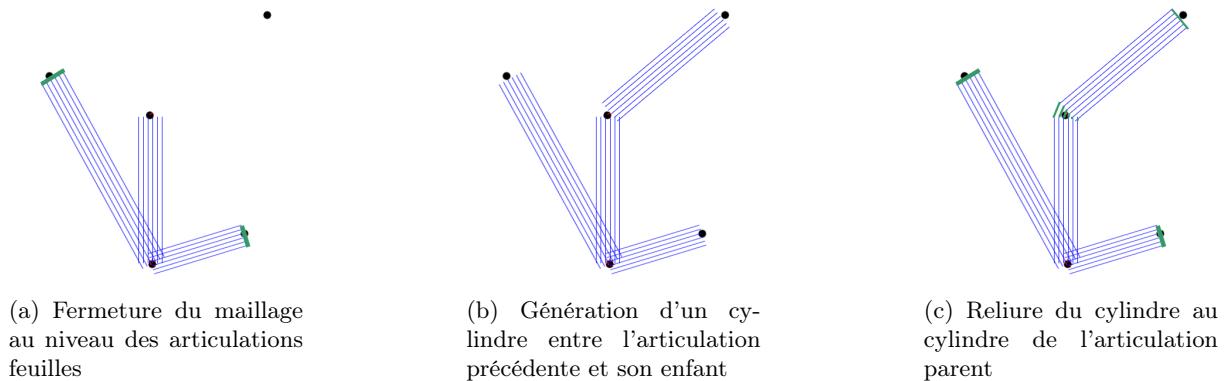


FIGURE B .18

Pour générer un tel maillage, nous avons écrit un algorithme récursif, associé à un algorithme permettant de générer un cylindre (voir l'implémentation C# en annexe, script `CreatureBody`, méthode `GenerateLimbsMesh`)

---

**Algorithm 1** GenererMaillage

---

```

Require: parent(articulation), extremite precedente(listedepoints)
    if parent.nombre_enfant > 0 then
        for i < parent.nombre_enfant do
            nouvelle _extremite ← GenererCylindre(parent, parent.enfants[i], extremite precedente)
            GenererMaillage(parent.enfants[i], nouvelle _extremite)
        end for
    else
        Fermer le cylindre précédent en reliant les points de <extremite precedente>
    end if

```

---

De son côté, Karl Sims a modélisé ses créatures avec des "boîtes" plus ou moins grosses qui relient les différentes articulations.

Dans la ré-implémentation de l'article "Recent Developments in the Evolution of Morphologies and Controllers for Physically Simulated Creature" par Tim Taylor et al.[3], les créatures sont modélisées par des capsules et des cylindres qui relient les articulations.



### B .3.2.2 Plaquage du maillage

Nos articulations sont mobiles, il faut donc que le maillage puisse les suivre et se déformer !

Pour ce faire, nous avons utilisé le composant **SkinnedMeshRenderer** de Unity, qui permet justement de plaquer un maillage sur un arbre d'articulations (représenté par une hiérarchie de GameObject). Ce composant est généralement utilisé pour permettre d'animer des personnages dans un jeu ; mais il est aussi totalement adapté à notre cas !

Pour l'utiliser, il faut d'abord indiquer à chaque sommet du maillage à quelle articulation il est rattaché. Pour cela, la classe Mesh (que l'on utilise jusque là pour générer et stocker notre maillage) contient deux tableaux : **bindPoses** et **boneWeights**.

Le premier permet de mémoriser la matrice de transformation initiale de chaque articulation.

Le second permet d'associer à chaque sommet une ou plusieurs articulations, avec différents poids.

Pour pouvoir utiliser ces structures de données, on va d'abord devoir modifier notre algorithme de génération afin de mémoriser les informations nécessaires (articulation associée à chaque sommet) :

---

#### Algorithm 2 GenererMaillage

---

```
Require: parent(articulation), extremite _precedente(listedepoints)
points _articulation ← []
points _par _articulations.add(points _articulation)
if parent.nombre_enfant > 0 then
    for i < parent.nombre_enfant do
        nouvelle _extremite ← GenererCylindre(parent, parent.enfants[i], points _articulation, extremite _precedente)
        GenererMaillage(parent.enfants[i], nouvelle _extremite)
    end for
else
    Fermer le cylindre précédent en reliant les points de <extremite _precedente>
end if
```

---

La variable "points \_par \_articulations" est une liste de listes. Elle associe à chaque articulation la liste des sommets du maillage qui lui sont associés.

Lors de la construction du cylindre, l'algorithme remplit une liste avec chacun des points créés ; liste qui est ensuite mémorisée dans "points \_par \_articulations".

On peut maintenant plaquer notre maillage sur nos articulations :

---

#### Algorithm 3 PlaquerMaillage

---

```
Require: mesh(maillage), points _par _articulations(listedelistesdepoints)
poses ← [nombre _articulation]
poids ← [nombre _points]
for bone < nombre _articulation do
    poses ← articulations[bone].matrice _transformation
    for i < taille(points _par _articulations [bone]) do
        poids[points _par _articulations [bone][i]].index _articulation ← bone
        poids[points _par _articulations [bone][i]].poids ← 1
    end for
end for
mesh.poses ← poses
mesh.poids ← poids
mesh.articulations ← articulations
```

---



(a) Affichage de la créature "ver" vue plus haut  
(b) Affichage de la créature "araignée" vue plus haut

FIGURE B .19

### B .3.3 Code génétique des créatures

Chacune de nos créatures est définie par un génome, qui décrit chacune de ses caractéristiques (Figure B .20). Il se présente sous la forme d'une chaîne de caractères formée à partir de la concaténation de chaque articulation dans l'ordre de l'arborescence. Chaque sous-chaîne d'une articulation a la structure suivante :

Structure et Mouvement												
Noms	ORDRE	ORIENTATION X	ORIENTATION Y	LONGUEUR	AMPLITUDE X	AMPLITUDE Y	VITESSE X	VITESSE Y	OFFSET X	OFFSET Y	ANGLE X	ANGLE Y
Utilité	Position du membre dans le graphe	Inclinaison par rapport au membre parent			Longueur du membre	Amplitude du mouvement du membre, selon l'axe vertical et horizontal du parent.			Vitesse de mouvement du membre		Décalage du mouvement	
Valeurs possibles	A..Z	00 <99	00 <99	00 <99	00 <99	00 <99	00 <99	00 <99	00 <99	00 <99	00 <99	00 <99
Valeurs réelles	0..26	-180° <+180°	-180° <+180°	0.1 < 0.5	-60° <+60°	-60° <+60°	0 < 1	0 < 1	-π < +π	-π < +π	-180° <+180°	-180° <+180°
Exemple	B	50	75	50	00	50	99	50	50	25	50	25
	S'accroche au précédent membre 'A'	0°	60°	0.3	-60°	0°	1	0.5	0	-π/2	0°	-90°
Code brut :	B5075500050995050255025											
Code "propre" :	B 50 75 50 00 50 99 50 50 25 50 25											

FIGURE B .20 – Tableau descriptif de chaque caractéristique contenue dans le génome.

- **Ordre** : représenté par une lettre majuscule, et indiquant la position du membre dans le graphe. L'ordre alphabétique indique le degré de profondeur dans l'arbre.
- **Orientation** : représente le degré d'inclinaison du membre en question par rapport à son membre parent, selon son l'axe vertical et horizontal
- **Longueur** : représente la longueur du membre
- **Amplitude** : représente l'amplitude du mouvement du membre, selon l'axe vertical et horizontal du membre parent
- **Vitesse** : représente les vitesses de mouvement du membre
- **Offset** : représente les décalages des mouvements : le mouvement est une oscillation décrite par la fonction sinus, le décalage permet donc de choisir à quelle niveau on démarre l'oscillation, et donc de décaler les mouvements des membres entre eux
- **Angle** : représente les angles préférentiels : l'angle entre le vecteur 'avant' de la créature, et le vecteur qui va vers sa cible. Plus cet angle est proche de la valeur donnée ici, plus le membre va bouger rapidement (voir section B.3.4.2).

Chaque mesure est représentée par un entier compris entre 0 et 99, nous notons "0" par un double zéro "00" afin de faciliter la manipulation du génome dans nos programmes. L'orientation, l'amplitude, la vitesse, l'offset et l'angle sont représenté par deux entiers X et Y. Pour un membre, on aura donc une chaîne de caractères de la forme suivante : **B5075500050995050255025** ou bien en plus propre : **B 50 75 50 00 50 99 50 50 25 50 25**. Le génome final d'une créature est donc la concaténation des sous-chaînes de chacune de ses articulations, dans l'ordre de l'arborescence selon un parcours en profondeur.

L'algorithme de construction utilise une structure de type "tas" pour mémoriser les parents par ordre alphabétique, et y rattacher correctement les enfants. Exemple : Pour le génome ABCBCD (on a enlevé les mesures pour cet exemple), le membre racine est A, suivi de ses fils, les deux B, le premier B a un enfant qui est C, tandis que le second B a un enfant, C, qui possède lui-même un enfant, D. Pour faire plus simple, on obtient l'arbre suivant :

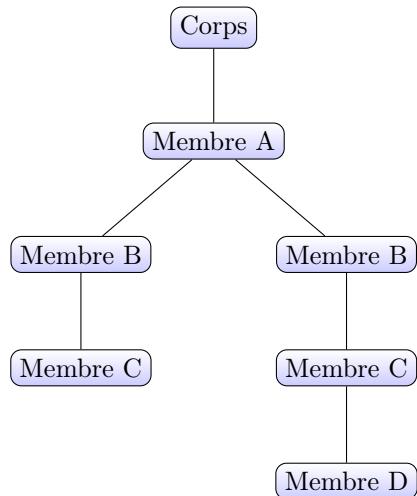


FIGURE B .21 – Schéma d'arbre pour un exemple de génome ABCBCD.

Voici quelques exemples afin d'illustrer un peu mieux à quoi ressemble un génome complet : Par exemple pour la créature ressemblant à un ver que nous prenions comme exemple plus haut, nous aurions le génome suivant (les mesures n'ont pas de sens ici, on s'intéresse à la forme surtout) :

**A50455067055905055855025B5075500050995050255025** Pour la créature ressemblant à une araignée, nous aurions un génome de cette forme :

**A50455067055905055855025A5075500050995050255025A50455067055905055855025  
A5075500050995050255025**

De cette manière nous avons un large panel de caractéristiques pour une créature, et la structure choisie pour modéliser le génome va nous aider quand nous devrons faire des modifications sur celui-ci, notamment dans le cadre des mutations/croisements que l'on va effectuer, et que nous développerons plus loin.

### B .3.4 Mouvement des créatures

#### B .3.4.1 Oscillation des articulations

Chaque articulation peut donc osciller selon deux axes, orthogonaux à la direction de l'articulation, simulant l'action d'un muscle qui se contracterait.

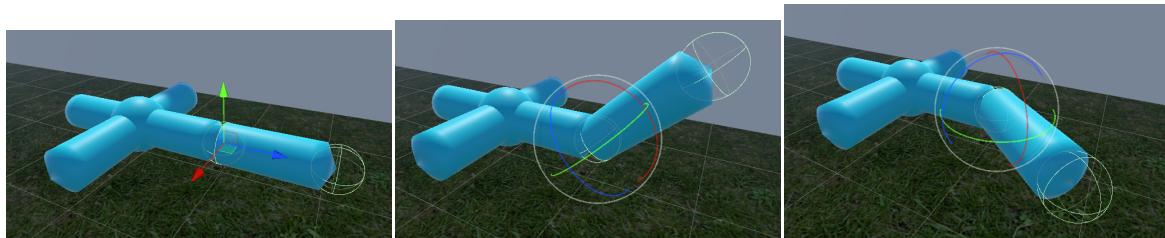
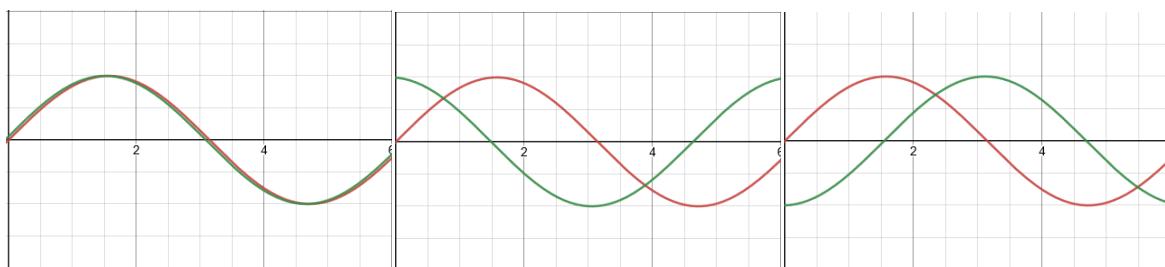


FIGURE B .22 – Possibilités d'oscillation pour une articulation.

L'oscillation est modélisée par une fonction sinus dont le paramètre augmente plus ou moins vite avec le temps, engendrant un mouvement plus ou moins rapide; sa valeur est ensuite multipliée à l'amplitude, pour obtenir l'angle courant de l'articulation. Le paramètre de la fonction sinus est décalé plus ou moins en fonction de la valeur de décalage associée à l'articulation.

Les amplitudes et vitesses des oscillations sont donc contrôlées par le code génétique associé : les valeurs comprises entre 0 et 1, permettent d'interpoler les vitesses et les amplitudes entre des valeurs minimales et maximales.

Il en va de même pour le décalage, qui est simplement interpolé entre  $-\pi$  et  $+\pi$



(a) Décalage faible

(b) Décalage positif

(c) Décalage négatif

FIGURE B .23

### B .3.4.2 Contrôle du degré de stimulation

Pour contrôler leurs articulations, les créatures possèdent un cerveau (script `CreatureBrain`). Le cerveau contrôle à chaque instant le degré de stimulation de chacune des articulations. Pour ce faire, il a besoin d'avoir une cible : un objet quelconque dans la scène, qui in-fine sera soit un fruit, soit une autre créature. Le cerveau calcule l'angle entre le vecteur "vers la cible" et le vecteur "vers l'avant" de la créature. Il va ensuite comparer cet angle aux valeurs d'angles préférentiels de chaque articulation, et les stimuler plus ou moins. L'idée est donc qu'une articulation va être davantage stimulée si la cible se trouve dans une direction proche à celle de son angle préférentiel.

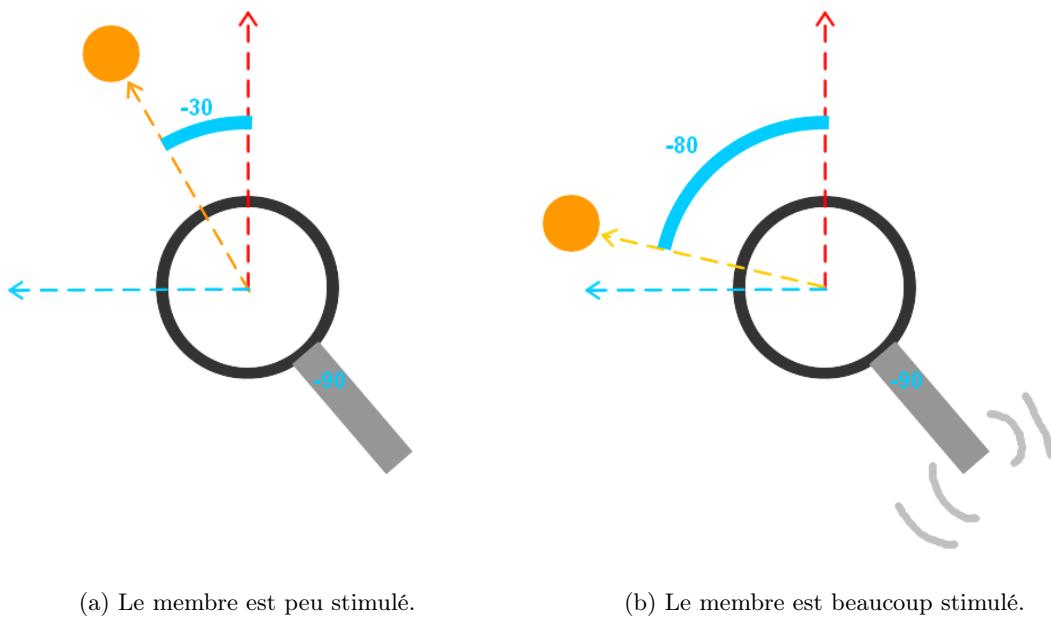


FIGURE B .24 – Vecteur "avant" en rouge, vecteur "cible" en jaune, vecteur "préférentiel" du membre en bleu. Le membre a donc ici un angle préférentiel de  $-90^\circ$ .

Les créatures devraient donc pouvoir stimuler plus ou moins certaines articulations en fonction du contexte, afin d'adopter un mouvement adapté : avancer tout droit, tourner, sauter, etc.

### B .3.4.3 Synchronisation

Lorsque deux articulations sont stimulées différemment, leurs fonctions sinus se désynchronisent naturellement, car elles n'oscillent plus à la même vitesse ; lorsque les deux articulations retrouvent un même degré de stimulation, leurs oscillations se retrouvent décalées, plus ou moins chaotiquement. Cela pourrait poser problème aux créatures pour maintenir un mouvement cohérent et fonctionnel au court du temps.

Pour empêcher cet effet de se produire, nous avons ajouté un système d'horloge interne à notre créature. Le script `CreatureBrain` possède un tableau de 100 valeurs  $mod2\pi$  (correspondant aux 100 valeurs possibles de vitesse) qui augmentent plus ou moins rapidement au cours du temps. Ces valeurs servent de référence pour les paramètres des fonctions sinus de toutes les articulations. Comme précédemment, ces paramètres augmentent plus ou moins vite en fonction du niveau de stimulation courant de l'articulation ; mais leur augmentation peut accélérer ou décélérer légèrement, de sorte à essayer de se caler sur la valeur de référence de l'horloge interne correspondant à la vitesse courante.

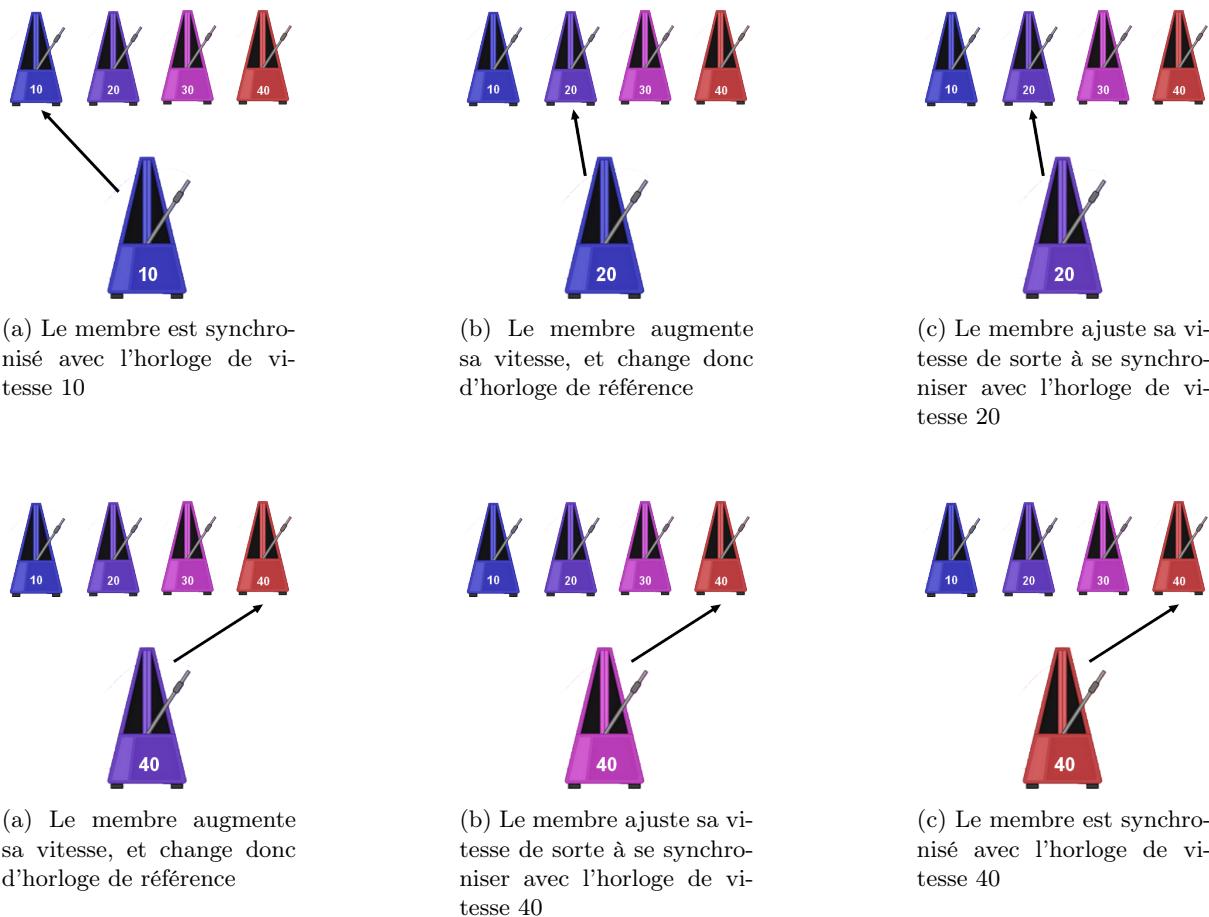


FIGURE B .26

Nos créatures peuvent donc à tout moment passer d'un mode de mouvement à un autre et changer de contexte à volonté, sans perdre la synchronisation entre leurs membres.

## B .3.5 Physique des créatures

### B .3.5.1 Problèmes avec la physique de base

Les créatures sont donc censées pouvoir se déplacer en faisant interagir leurs articulations avec le sol : elles peuvent se pousser, se tirer, se faire sauter, etc.

Mais la physique de base de Unity ne nous a pas permis d'obtenir des résultats fonctionnels : les forces engendrées par collisions entre le sol et les articulations sont trop faibles, et ne parviennent pas à contrer la friction.

Ils nous a donc été impossible d'obtenir des créatures capables d'avancer de cette manière.

### B .3.5.2 Solution

Pour régler ce problème, nous avons ajouté une interaction supplémentaire, par dessus la physique de collisions de base de Unity.

Lorsqu'une articulation entre en collision avec le sol, une force est appliquée à la créature, dans la direction opposée au mouvement propre de l'articulation. Cette force est pondérée par le produit scalaire entre la normale du sol et ce même vecteur de mouvement propre.

Un paramètre "courbe" permet d'appliquer une puissance à l'intensité du vecteur mouvement, et donc de rendre des mouvements rapides plus importants que des mouvements lents.

$$\begin{aligned} \vec{vect} &= -\text{normaliser}(mv\vec{m}_a) \times \text{norme}(mv\vec{m}_a)^{\text{courbe}} \\ \vec{angle} &= \vec{\text{normale}} \cdot -\text{normaliser}(mv\vec{m}_a) \\ \vec{force} &= \vec{vect} \times \vec{angle} \times \text{intensite} \end{aligned}$$

"vect" = vecteur force

"mvmp\_a" = mouvement "propre" de l'articulation (dans le repère de la créature)

Le mouvement "propre" est d'abord calculé par chaque articulation comme étant la distance parcourue par celle-ci entre chaque frame. Elle prend donc naturellement en compte l'accumulation des oscillations des articulations parents, mais aussi la vitesse globale de la créature, ainsi que sa vitesse angulaire.

Ensuite, lors de la collision, on soustrait à ce mouvement propre le mouvement global de la créature, ainsi que le mouvement engendré par le potentiel moment angulaire de la créature, afin d'obtenir le mouvement propre de l'articulation dans le repère créature.

$$\begin{aligned} \vec{vect\_a} &= p\_a - p\_c \\ \vec{v\_angle} &= -(rotation \times \vec{vect\_a}) \\ mv\vec{m}\_a &= mv\vec{m}p\_a - mv\vec{m}\_c - \vec{v\_angle} \end{aligned}$$

"p\_a" = position de l'articulation

"p\_c" = position de la créature

rotation" = vitesse angulaire de la créature

"v\_angle" = vitesse de l'articulation résultante de la rotation de la créature

"mvmp\_a" = mouvement absolu de l'articulation dans le repère monde

"mvmp\_c" = mouvement de la créature dans le repère monde

On obtient ainsi un vecteur représentant la direction et la vitesse à laquelle se déplace l'articulation par rapport au centre de gravité de la créature.

La méthode `Rigidbody.AddForceAtPosition(force, point, mode)` permet d'appliquer une force en un point donné, comme si on poussait l'objet en ce point, ce qui peut entraîner une rotation. Cette interaction supplémentaire nous a permis d'obtenir la physique que nous souhaitions et de permettre à nos créatures de se déplacer comme prévu.

## B .4 Simulation



### B .4.1 Apparitions des créatures

Avant que la simulation ne commence, l'utilisateur sélectionne le biome dans lequel il veut lancer la simulation :



FIGURE B .27 – Menu sélection du biome

Ensuite, il peut placer des zones où les créatures vont apparaître. Pour chaque zone, le type et la quantité (entre 1 et 10) de créatures peut être spécifié :



FIGURE B .28 – Menu de définition des zones d'apparitions des créatures.

Quand c'est fait, la simulation peut commencer.

Les créatures apparaissent sur le terrain autour de leurs zones d'apparitions avec un niveau d'énergie de 50%.



FIGURE B .29 – Machine à état du cycle de vie d'une créature

### B .4.2 Cycle de vie d'une créature

Les créatures ont un cycle de vie modélisé par une machine à état :

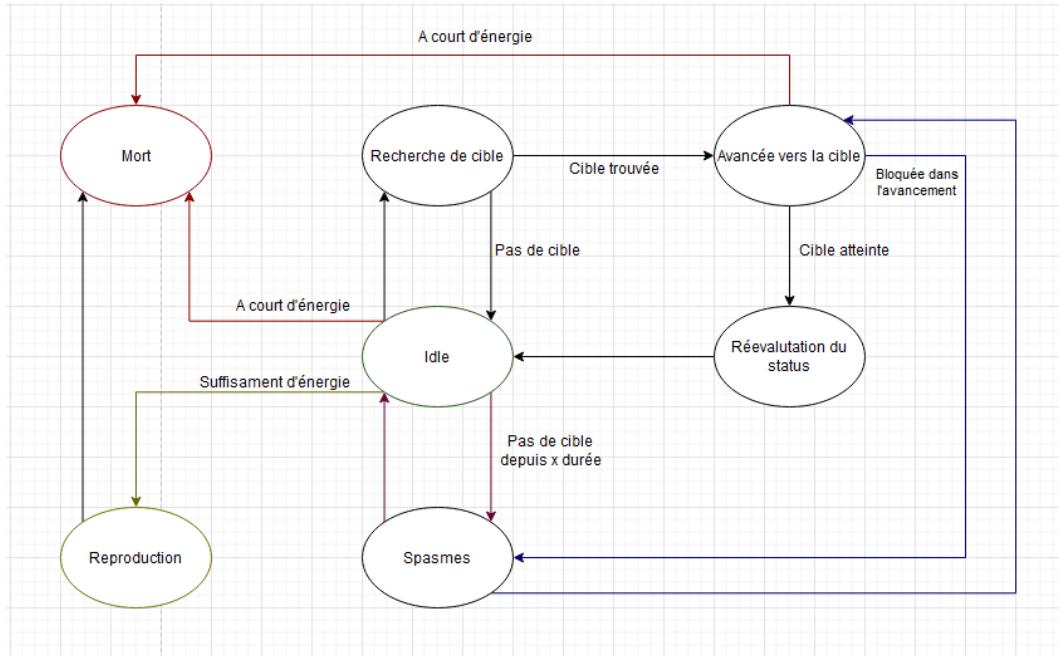


FIGURE B .30 – Schéma machine à états

L'état de base est "Idle" (nom générique pour un état de base).

La créature va ensuite passer dans l'état "Recherche de cible", où elle va lancer régulièrement une méthode de recherche, jusqu'à trouver une cible.

Une fois la cible trouvée, la créature passe dans l'état "Avancée vers la cible", où elle va appliquer les algorithmes de mouvement vus plus haut afin de s'approcher au mieux de la cible.

Si jamais la créature se retrouve "bloquée" (modélisé par le fait que sa position ne change plus suffisamment), elle va passer dans l'état "Spasmes" : c'est à dire qu'elle va activer toutes ses articulations au maximum et en même temps, afin de parvenir à se débloquer.

Lorsque la cible est atteinte, c'est à dire que le fruit ciblé est attrapé, la créature va réévaluer son statut actuel et repasser en "Idle". Si la cible est une autre créature, la créature va rester dans l'état "Avancée vers la cible" afin de continuer à l'attaquer.

Pendant l'état "Avancée vers la cible", la créature va de temps à autre lancer une nouvelle recherche de cible afin de voir si une autre cible plus intéressante se trouve à proximité.

Quand la créature repasse à l'état "Idle", elle va vérifier si elle a assez d'énergie pour se reproduire.

Dans ce cas, elle passe à l'état "Reproduction", ce qui va lancer une reproduction, et faire disparaître la créature.

Sinon, elle recommence son cycle.

A tout moment, la créature peut mourir si elle n'a plus assez d'énergie.



## B .4.3 Énergie

### B .4.3.1 Consommation

Les créatures dépensent leur énergie pour stimuler leurs membres.

A chaque instant, la consommation est calculée comme suit :

$$C = \sum_{i=0}^n (sx[i] \times c) + (sy[i] \times c)$$

Où "C" est la consommation instantanée, "n" le nombre d'articulations, "sx" et "sy" les stimulations de chaque articulation, et "c" la valeur de consommation par articulation

La consommation par articulation est définie par une constante multipliée par la consommation globale. La consommation globale dépend du biome (voir section B.2.1), mais aussi du nombre de créatures vivantes.

Pour éviter une explosion du nombre de créatures, qui empêcherait la simulation de tourner correctement, la consommation globale augmente si le nombre de créature atteint un certain intervalle : au dessus de 85 créatures et jusqu'à 110 créatures, la consommation est interpolée entre  $\times 1$  et  $\times 100$ .

Cela permet de "nettoyer" la simulation si les créatures se mettent à se reproduire trop vite.

Avant d'être appliquée, la consommation instantanée est ensuite multipliée par un coefficient de vieillesse : plus une créature est vivante longtemps, plus elle consomme d'énergie. Ainsi, la consommation est multipliée par :

$$A = 1 + (0.1 \times age\_en\_secondes)$$

Ce qui incrémente de 1 le coefficient toutes les 10 secondes.

Cela permet également d'éliminer les créatures trop âgées afin de faire de la place pour les nouvelles créatures.

#### B .4.3.2 Nutrition et attaque

Une créature peut donc attraper des fruits pour se nourrir, mais également s'attaquer à une autre créature.

Pour manger un fruit, il suffit qu'une de ses articulation entre en contact avec : le fruit est alors consommé (il disparaît), et la créature récupère plus ou moins d'énergie, en fonction de la nutritivité du fruit.



FIGURE B .31 – Vidéo d'une créature attrapant un fruit

Pour attaquer une autre créature, il faut qu'elle la touche avec une de ses articulation : une attaque est alors détectée par les deux créatures. Pour déterminer qui "gagne", c'est à dire quelle créature va voler de l'énergie à l'autre, on va comparer la vitesse des deux articulations qui se sont entrechoquées : la créature qui a tapé le plus fort "gagne".

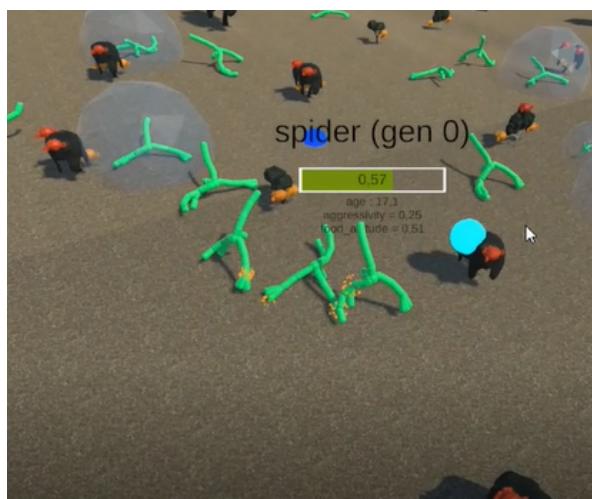


FIGURE B .32 – Vidéo de créatures en train de se frapper



## B .4.4 Paramètres comportementaux

Lors de la simulation, plusieurs paramètres comportementaux sont observés comme l'agressivité des créatures et l'altitude préférentielle de la nourriture.

### B .4.4.1 Agressivité

L'agressivité des créatures est définie comme un paramètre dans leur code génétique qui varie entre 0 et 1. Elle détermine si la créature a plus ou moins tendance à prendre pour cible une autre créature si elle est suffisamment proche. Elle est utilisée comme un seuil : si

$$1 - \frac{dist(fruit)}{dist(creature)} \geq 1 - aggressivite$$

alors la cible sélectionnée sera la créature, sinon, ce sera le fruit.

A noter que le fruit sera toujours sélectionné en priorité s'il est plus proche que la créature ciblée.

L'agressivité intervient aussi dans les mécanismes d'attaque et de nutrition : une créature agressive récupérera plus d'énergie sur la cible qu'elle attaque, mais en obtiendra moins en mangeant des fruits :

$$coefficient\_fruit = 1 + (1 - aggressivite)$$

$$coefficient\_attaque = 1 + aggressivite$$

Ce taux d'agressivité fait donc partie du code génétique de la créature, et varie donc au fur et à mesure des générations.

### B .4.4.2 Altitude préférentielle de la nourriture

L'altitude préférentielle de la nourriture est l'altitude des fruits préférée par la créature. Celle-ci va donc prendre pour cible en priorité les fruits qui sont à une altitude proche de ce paramètre. La valeur de ce paramètre est définie dans le code génétique et varie entre 0 (hauteur du sol) et 1 (valeur maximale prédéfinie).

Le fruit sélectionné sera donc celui qui minimise ce score :

$$score = distance \times (1 + |altitude\_fruit - altitude\_pref|)$$

Comme les fruits en hauteur sont plus nutritifs que les fruits au sol, ce paramètre peut avantager les créatures capables d'attraper des fruits hauts.

## B .4.5 Reproduction et mutations

### B .4.5.1 Conditions de reproduction

Pour pouvoir se reproduire, les créatures doivent parvenir à atteindre un niveau d'énergie suffisant (90%), qui peut varier en fonction du biome. Elles doivent donc réussir à attraper des fruits, et/ou à voler de l'énergie aux autres créatures en les attaquant.

Quand une créature réussi à atteindre ce seuil, elle disparaît, et fait apparaître deux nouvelles créatures à côté d'elle : une créature identique, et une créature mutée.

Les créatures enfants apparaissent dans un "oeuf", et avec une taille réduite. Elles "grossissent" avec l'oeuf jusqu'à atteindre une taille normale, et peuvent ensuite commencer leur cycle de vie.

Cela permet de garantir à chaque créature enfant d'avoir suffisamment d'espace, et de ne pas apparaître les unes sur les autres, ce qui peut poser des problèmes de collisions, et/ou de créatures qui s'entre-tuent à peine sorties de l'oeuf.



FIGURE B .33 – Vidéo d'une créature dans son oeuf



### B .4.5.2 Mutations

Nous avons mis au point plusieurs types de mutations, afin d'avoir un panel assez large de mutations possibles, et d'obtenir plus de diversités dans nos créatures au cours de leurs évolutions.

**Ajout d'une articulation :** Duplique l'articulation courante et la place juste après (ordre+1). Pour conserver la hiérarchie des articulations suivantes, on décale également leur ordre de 1 :

---

**Algorithm 4** Ajouter

---

```
Require: gene(Gene)
liste_genes.add(gene)
gene2 ← copier(gene)
gene2.ordre ← gene2.ordre + 1
liste_genes.add(gene2)
for g in genes_suivants do
    if g.ordre > gene.ordre then
        g.ordre ← g.ordre + 1
        liste_genes.add(g)
    else
        arret
    end if
end for
```

---

**Retrait d'une articulation :** Supprime l'articulation courante. On décale toutes les articulations enfants pour conserver la hiérarchie (ordre - 1) :

---

**Algorithm 5** Retirer

---

```
Require: gene(Gene)
ne pas ajouter gene à la liste
for g in genes_suivants do
    if g.ordre > gene.ordre then
        ne pas ajouter g à la liste
    else
        arret
    end if
end for
```

---



**Duplication :** Duplique l'articulation courante ainsi que tous ses enfants :

---

**Algorithm 6** Dupliquer

---

```
Require: gene(Gene)
        liste_genes.add(gene)
        liste_duplicats ← [gene]
        for g in genes_suivants do
            if g.ordre > gene.ordre then
                liste_genes.add(g)
                liste_duplicats.add(g)
            else
                for g2 in liste_duplicats do
                    liste_genes.add(g2)
                end for
                arret
            end if
        end for
```

---

**Suppression :** Supprime l'articulation courante ainsi que tous ses enfants :

---

**Algorithm 7** Suppression

---

```
Require: gene(Gene)
        ne pas ajouter gene à la liste
        for g in genes_suivants do
            if g.ordre > gene.ordre then
                ne pas ajouter g à la liste
            else
                arret
            end if
        end for
```

---

**Mutation des valeurs :** Cette mutation va simplement incrémenter ou décrémenter les valeurs d'une articulation, afin de modifier son comportement :

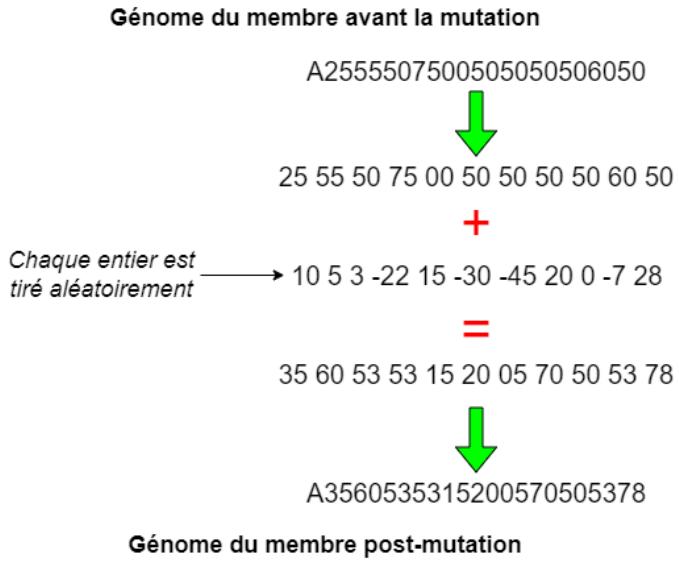


FIGURE B .34 – Schéma du fonctionnement de la mutation des valeurs

Pour contrôler la probabilité des différentes valeur d'incrément, nous l'avons calculé ainsi :

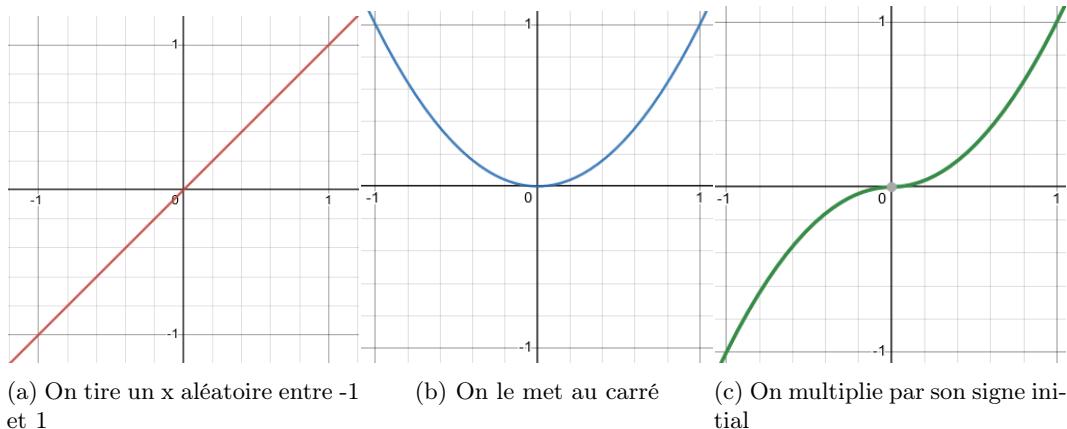


FIGURE B .35

Puis on multiplie la valeur obtenue par la valeur maximale d'incrément.  
Ainsi, on obtiendra plus souvent des valeurs faibles que des valeurs élevées.

### B .4.5.3 Crossovers

Le principe du crossover est de fusionner les génomes de deux créatures. Pour cela, on choisit un membre "pivot" dans le premier génome. On conserve les membres se situant avant le membre pivot dans l'arborescence, ce qui correspond à une première sous-chaîne de caractères qui va former la première moitié du nouveau génome. Ensuite, on recherche dans le second génome, le premier membre de niveau inférieur au pivot du premier génome - si le membre pivot est A, on va sélectionner le premier membre B dans le second génome - et tous les membres situés après celui-ci vont constituer la seconde moitié de notre nouveau génome. Si aucun membre n'est trouvé dans le génome, on remonte d'un niveau dans l'arborescence - si on ne trouve aucun B, on va remonter et chercher le premier A -. Enfin, on concatène ces deux sous-chaînes de caractères, et on obtient un génome finale, qui représente une créature "fusion" des deux créatures sélectionnées à l'origine.

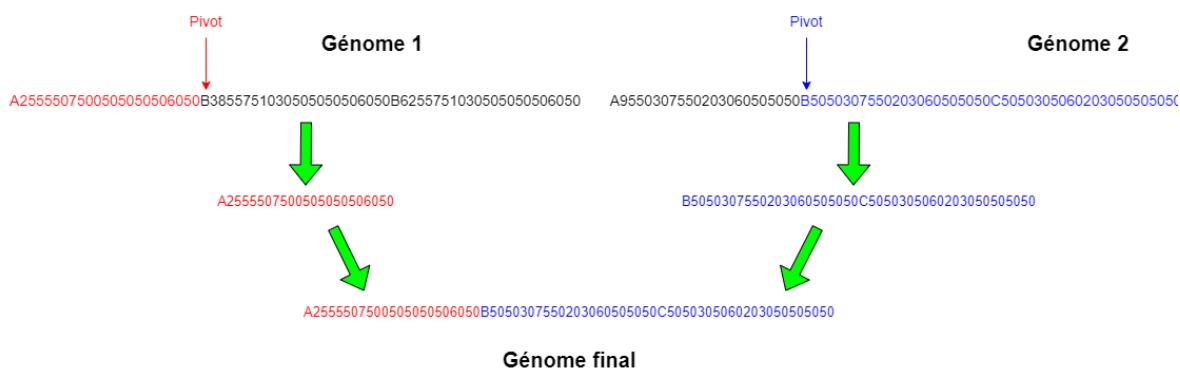


FIGURE B .36 – Schéma du fonctionnement du crossover

Karl Sims, lui, présente une méthode de crossover. Cette méthode implique la conservation des noeuds du premier parent jusqu'à un point de croisement déterminé. Ensuite, le même "pivot" est sélectionné dans le second parent, et les noeuds qui suivent ce pivot sont concaténés avec les noeuds sélectionnés du premier parent. Il est possible de sélectionner plusieurs points de croisement et d'effectuer des allers-retours entre les deux parents. Les connexions entre les deux groupes de noeuds sont conservées si cela est possible, sinon elles sont réaffectées de manière aléatoire. Au final, nous obtenons un graphe enfant qui possède des noeuds et des connexions hérités de chacun de ses parents. Nous nous sommes partiellement inspirés de la méthode de crossover de Karl Sims et l'avons adaptée à la structure que nous avons choisie pour représenter les informations de nos créatures, à savoir leur génome.

## B .4.6 Statistiques

### B .4.6.1 Graphiques

Afin de mieux analyser le déroulement d'une simulation, nous avons implémenté un système de statistiques dynamiques.

Lors d'une simulation, on peut afficher trois différents graphiques : la répartition du niveau d'agressivité, d'altitude préférentielle de la nourriture, et de la consommation d'énergie.

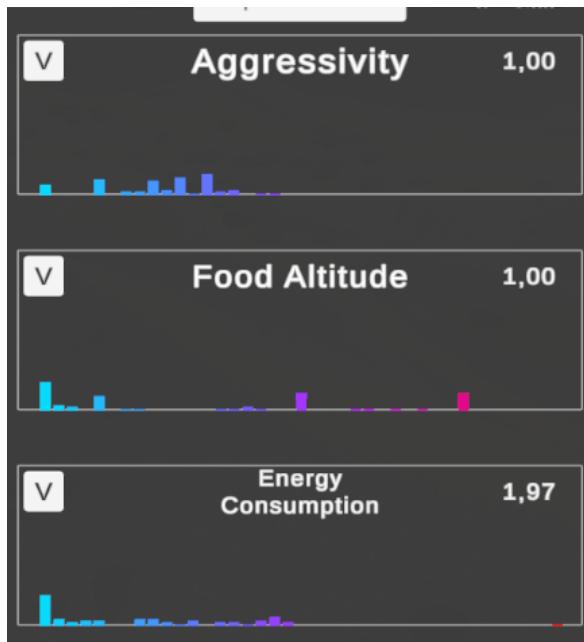


FIGURE B .37 – Capture d'écran de l'affichage des graphiques

Dans le cas de l'agressivité et de l'altitude (Figure B .37), les graphiques présentent en abscisses la valeur du paramètre (entre 0 et 1), et en ordonnées le nombre de créatures à ce niveau. C'est un diagramme à bâtons, chaque bâton représente une plage de valeurs (ici on a 20 bâtons, avec des plages de 0.05 de large) dans lesquelles les créatures sont regroupées.

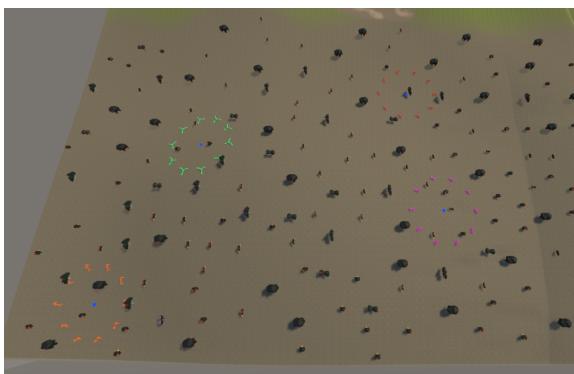
Le diagramme de consommation d'énergie fonctionne de la même manière, mais la plage de valeurs en abscisses est dynamique : la valeur maximale est fréquemment actualisée pour représenter au mieux la répartition des consommations, qui peuvent être très variables.

#### B .4.6.2 Coloration des créatures

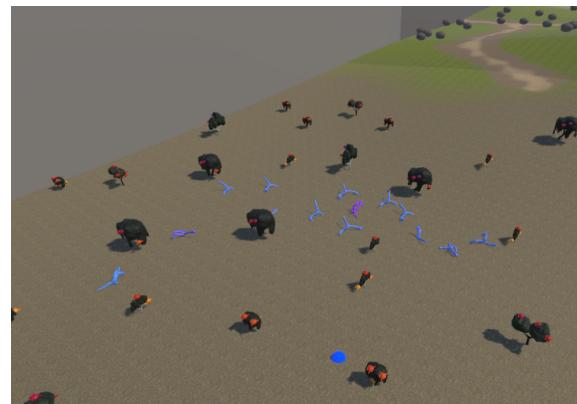
Afin d'avoir une vision d'ensemble de nos créatures et de leurs caractéristiques lors de la simulation, nous avons ajouté la possibilité de colorer les créatures en fonction d'un paramètre.

Par défaut, chaque espèce de créature possède une couleur, ce qui permet de voir rapidement quelles créatures survivent ou non.

En cliquant sur le bouton "V" d'un des graphiques, les créatures vont se colorer en fonction de la valeur du paramètre correspondant. Cela permet de voir rapidement quelles créatures ont quels niveaux d'agressivité, quelles consommations d'énergie, etc.



(a) Coloration par espèces



(b) Coloration par agressivité

FIGURE B .38

#### B .4.6.3 Informations individuelles

Il est également possible de sélectionner une créature, afin de passer en mode caméra orbitale autour d'elle, et d'afficher différentes informations :



(a)



(b)

On peut ainsi voir l'espèce de la créature sélectionnée, sa génération, son niveau d'énergie actuel accompagné de sa consommation, son âge, et ses paramètres comportementaux : agressivité et altitude.

## B .5 Pré-entraînements

Les créatures de base que nous avons créées (voir section C.1.0.2) ne sont pas directement aptes à se déplacer correctement et à se nourrir, ce qui pose problème pour la simulation. C'est pourquoi nous avons mis en place une série de pré-entraînements permettant aux créatures d'apprendre à avancer, à tourner, à aller plus vite et à optimiser leur énergie. En somme, cela leur permet d'avancer de manière optimale vers une cible (i.e. fruit ou créature), et donc de rendre la simulation fonctionnelle et intéressante.

Pour pré-entraîner les créatures, on place 100 créatures dans des enclos séparés. Dans chaque enclos, on trouvera une créature et une cible. En fonction du type d'entraînement, la cible sera positionnée différemment dans l'enclos. Le but de la créature est de s'en rapprocher le plus possible.



FIGURE B .40 – Enclos d'entraînements.

Après un certain temps (100 secondes dans un premier temps, puis on réduit pour augmenter la difficulté), on sélectionne les 10 meilleurs individus et on les fait muter pour obtenir 90 nouvelles créatures (en plus de 10 meilleures). On peut donc relancer un entraînement avec ces individus là. On continue à relancer des entraînements jusqu'à ce que les créatures aient acquis la compétence recherchée.

En comparaison, Karl Sims choisit de laisser deux créatures se battre pour le contrôle d'une cible (en l'occurrence, un cube) afin de leur attribuer un score proportionnel à leur avancement vers celle-ci. La créature ayant obtenu le meilleur score va subir une mutation, puis de nouveau être mise en compétition avec une autre créature. Ce type d'entraînement permet finalement de faire évoluer les créatures.

### B .5.1 Apprendre à avancer

Le but est d'apprendre aux créatures de se rapprocher d'une cible située tout droit en face d'elles. Pour cela, elles doivent minimiser la fitness-fonction suivante :

$$F = \text{distance\_cible}$$

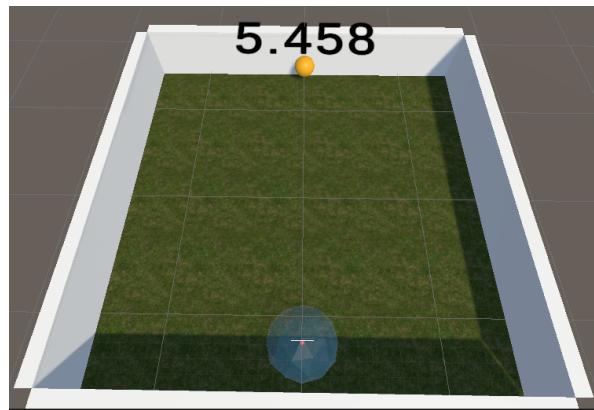


FIGURE B .41 – Enclos d'entraînement pour permettre à la créature d'apprendre à avancer tout droit. En jaune : la cible. Dans la bulle : la créature.

### B .5.2 Apprendre à tourner

Le but est d'apprendre aux créatures à atteindre des cibles qui ne sont pas forcément tout droit, et donc à développer des modes de déplacement leur permettant de tourner.

Pour cela, elles doivent maximiser la fitness-fonction suivante :

$$F = (\text{nombre\_cibles\_atteintes} \times \text{distance\_entre\_cibles}) - \sqrt{\text{distance\_prochaine\_cible}}$$

Dans ce type d'entraînement, la cible va se déplacer dès que la créature la touche. Ce qui va obliger la créature à tourner afin de la toucher à nouveau.

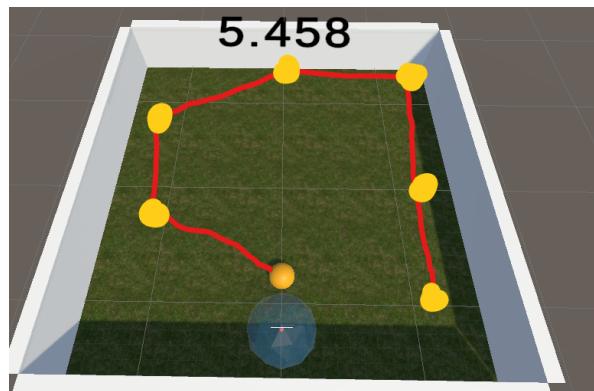


FIGURE B .42 – Enclos d'entraînement pour permettre à la créature d'apprendre à tourner. En jaune : la cible et son avancement. En rouge : le circuit parcouru par la cible. Dans la bulle : la créature.



### B .5.3 Aller plus vite

Le but est de faire en sorte à ce que la créature entraînée soit de plus en plus rapide à atteindre l'objectif. Le terrain d'entraînement est semblable à l'entraînement pour apprendre à tourner. Les créatures doivent maximiser la fitness-fonction suivante :

$$F = \left( \sum_{i=0}^n \frac{distance\_entre\_cibles}{1 + duree\_cible[i]} \right) - \sqrt{distance\_prochaine\_cible}$$

Où 'n' est le nombre de cibles atteintes, et  $duree\_cible[i]$  est le temps qu'a mis la créature à atteindre la cible n°i à partir de la cible n°(i-1) (ou du point de départ)

### B .5.4 Optimiser l'énergie consommée

Le dernier entraînement consiste à apprendre à la créature à optimiser son énergie. C'est à dire que nous voulons qu'elle garde les mêmes performances qu'aux précédents entraînements mais en réduisant la taille de son code génétique (et donc son nombre de membres) et en minimisant l'utilisation de certains membres peu utiles.

Les créatures doivent maximiser la fitness-fonction suivante :

$$F = \left( \sum_{i=0}^n \frac{2 \times distance\_entre\_cibles}{(1 + duree\_cible[i]) + (1 + energie\_cible[i])} \right) - \sqrt{distance\_prochaine\_cible}$$

Où  $energie\_cible[i]$  est la quantité d'énergie dépensée pour atteindre la cible n°i à partir de la cible n°(i - 1) (ou du point de départ)

## Résultats et analyses



## C .1 Création des créatures et pré-entraînements

### C .1.0.1 Comment créer des créatures ?

Afin d'observer l'évolution des créatures au fur et à mesure des générations, il nous faut tout d'abord générer des créatures initiales qui pourront ensuite évoluer. Chacun de nous a donc imaginé une créature fictive avec une certaine structure et un mouvement de base. Nous avons donc inventé quatre créatures au total. Tout d'abord, il y a le Manodon : créature à quatre membres qui tente de reproduire le mouvement du crawl. Ensuite, il y a la Spider et le Mille-pattes, qui se veulent ressemblants aux animaux portant ce nom. Enfin, il y a le Phoque : créature à quatre membres d'abord inspiré de l'animal, puis finalement tourné en paire de ciseaux. Pour les créer, nous avons écrit à la main un code génétique donnant naissance à ces créatures. Ce code a dû être modifié petit à petit jusqu'à obtention d'une créature potentiellement capable d'évoluer de manière optimale, à partir d'une prémissse de déplacement fonctionnant un minimum.

### C .1.0.2 Présentation de nos créatures

Voici finalement le résultat obtenu :

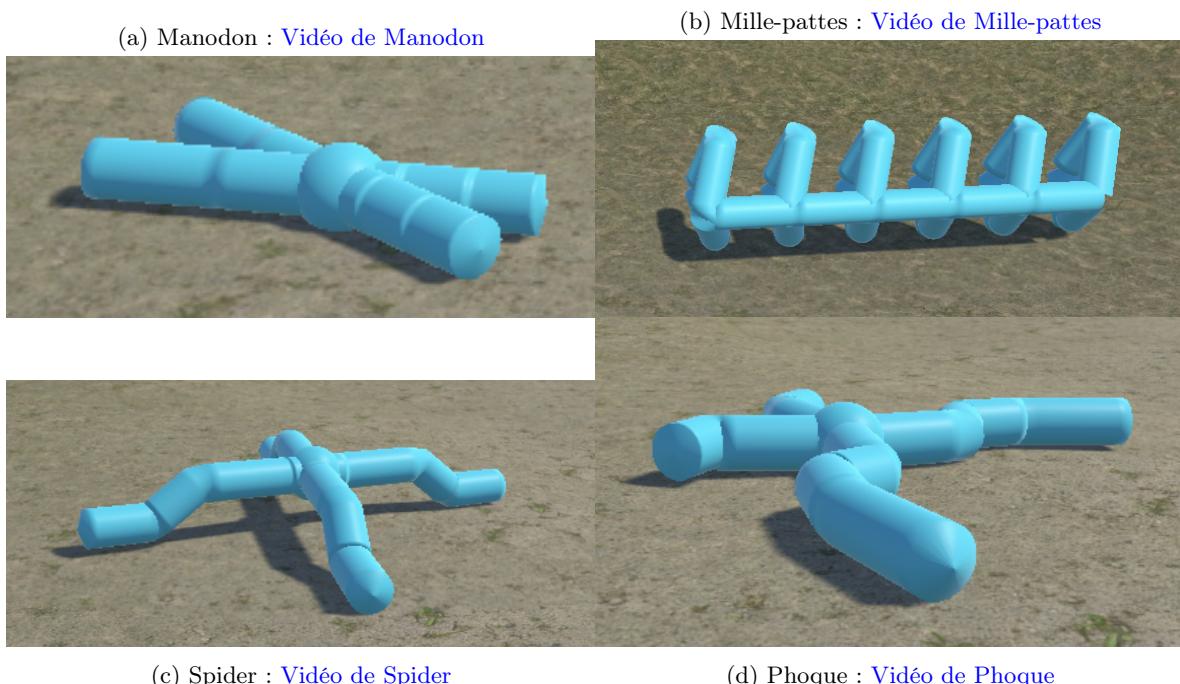


FIGURE C .1 – Créatures de base

### C .1.1 Créatures après entraînements

#### C .1.1.1 Apprendre à avancer

**Manodon :**



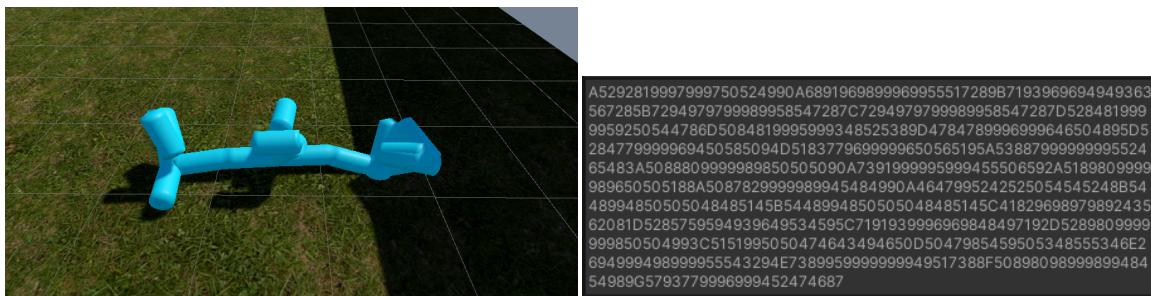
(a) Vidéo de Manodon après 1 entraînement pour apprendre à avancer

(b) Code génétique

FIGURE C .2

Le Manodon a désormais plus de membres, il utilise ses pattes avant pour avancer et il laisse traîner ses pattes arrières qui lui servent plutôt de support. Son code génétique s'est compléxiifié.

**Mille-pattes :**

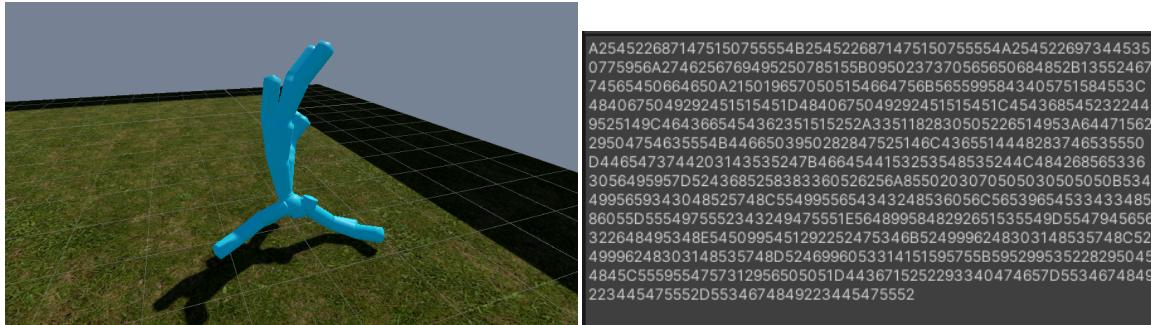


(a) Vidéo de Mille-pattes après 1 entraînement pour apprendre à avancer

(b) Code génétique

FIGURE C .3

Le Mille-pattes a perdu certaines de ses pattes, il a gardé celles se trouvant derrière lui, et se sert d'elles pour générer une impulsion lui permettant de faire de grands sauts vers l'avant. Il a également conservé des pattes à l'avant, et au milieu de son corps, ce qui lui permet d'être plus stable.

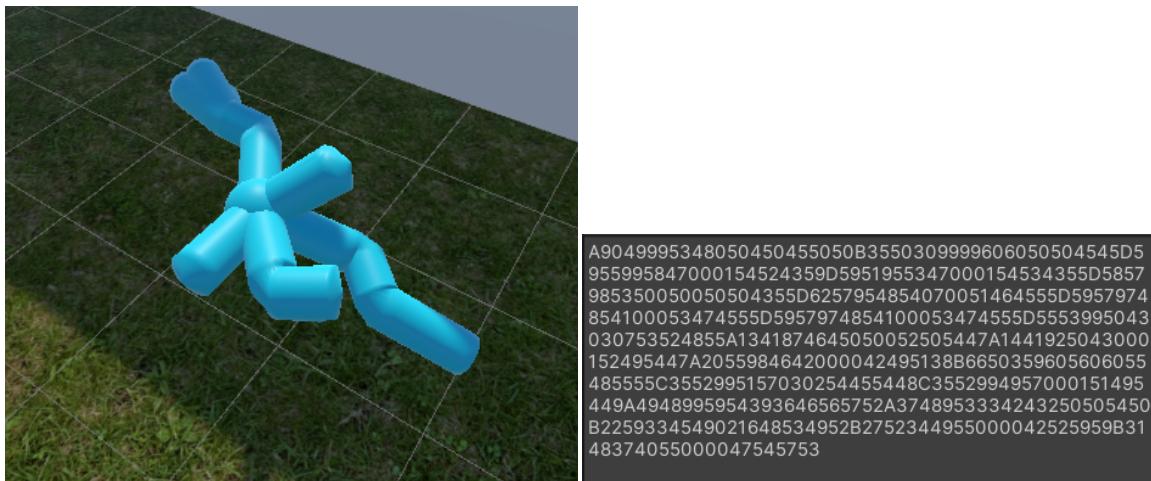
**Spider :**


(a) Vidéo de Spider après 2 entraînements pour apprendre à avancer

(b) Code génétique

FIGURE C .4

La Spider a désormais plus de pattes, et certaines sont beaucoup plus longues. Elle se sert des plus petites pour se déplacer et des plus grandes pour se stabiliser. Son génome est beaucoup plus complexe à présent.

**Phoque :**


(a) Vidéo de Phoque après 3 entraînements pour apprendre à avancer

(b) Code génétique

FIGURE C .5

On peut voir que le phoque arrive désormais à atteindre la cible assez rapidement. Ses pattes avant se sont raccourcies, et ses pattes arrière étendues sur les côtés. Elle utilise ses deux pattes arrière pour se propulser, et ses pattes avant pour se stabiliser. Elle parvient même à redresser sa trajectoire pour rester alignée à la cible. Son code génétique s'est légèrement complexifié.

### C .1.1.2 Apprendre à tourner

**Manodon :**



(a) Vidéo de Manodon après 1 entraînement pour apprendre à tourner

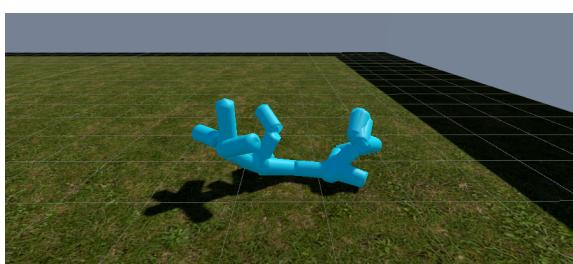
```
A9751297945232968585645B4250377053204053535151B43523773652344  
51575852B455742736213643576249C4350354761153747345246C485031  
4856163653365245C464132554123148334849C51473455613325138504  
7C5147345556133251385047C504431553093255405450C404832505520  
2860355052C4151374354232463464848C4948384959222962435147C494  
5345458202655444950C514736495221325444951C44453250572839594  
04346C4141354554143156434646C4141354554143156434646A124124817  
0173767474360A1454238162193564383758B445537855227347045574586  
267318351233263444862B5155307553172763435357C4056295965271950  
523949C4056295965271950523949C5048324147203158525658C49464352  
57183646405744C3952345649232144535847C4056315755202443525155  
A5149796241978308454748A474479784937603524851A46488478499581  
05495244A4347847644907707495351A4853808547798808584757B475691  
6255978500345846B4756916255978500345846B606485766086740347484  
5B5764917854907800475451A544990805099354484653B485087704086  
8547504846B4850877040868547504846851548874459288534248508485  
6907843938546483945
```

(b) Code génétique

FIGURE C .6

Le Manodon arrive à se déplacer légèrement sur les côtés grâce à ses pattes avant.

**Mille-pattes :**



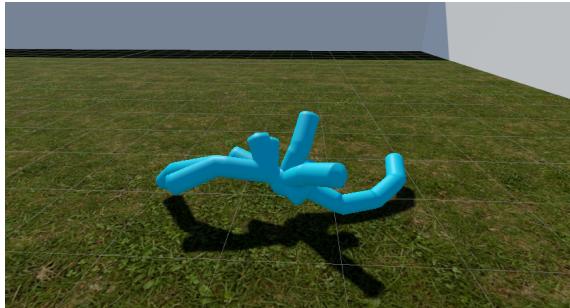
(a) Vidéo de Mille-pattes après 1 entraînement pour apprendre à tourner

```
A5199819999989848524991B539981999999852565090B5299819996999852  
555090A779195999949956467089B759198999939863607292B769299989  
999962587890B76929989999962587890C749797949999958547683D51  
80869798959349584385D50880989298856554690D4584859497942475  
55691D498277991999059524899D5181759099949950595295C7294979699  
979958527787D508182979999352584687D5085829995948857554789D508  
4819995999348525389D4782779991999155545199D5181759499989505952  
9645283829797949844494888A524699534548553545448B635199434853  
384652347B5950994350544247525343B49519649554550534444498605  
1964450444150525146C348396499999649652988D53797590939495359  
4786C7699959698919944506295C7096959991989945536493D5194759989  
49953494590D5393909591919949535095C5459993852523743494349D56  
57994352514145514352C555599455554042494750D2796979699998545  
43496D6992969293989949597789D6992969293989949597789E4896839699  
929846455077F6399679498949751553993
```

(b) Code génétique

FIGURE C .7

Le Mille-pattes n'a plus de membres au milieu de son corps, il a maintenant un gros amas de membres à l'avant et à l'arrière, et il peut maintenant faire des sauts légèrement orientés sur les côtés ce qui lui permet de se déplacer dans différentes directions.

**Spider :**


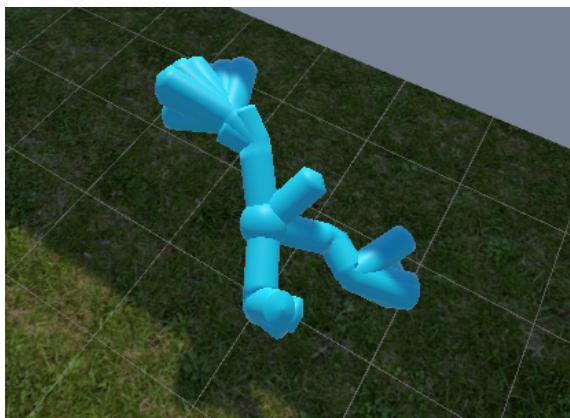
(a) Vidéo de Spider après 2 entraînements pour apprendre à tourner

```
A3652176666564456786146A365217666564456786146B284724666753575
4765650B284246667555855785653A1843286073455152774850A21471663
71495253704959B5458985433365748603751C5040684856242359526455D
4843674856252354546354E444173455272555505363A324719272448502
0534956B3247192724485020534956A39624634571828474742448506348
3546262937524849A5038655161292754395661B4740685161302655445861
A4852705551433467546156B4852705551433467546156A44486857564134
63566257A754721417154533251384486246886554353050566849858494
5258284043545957C5457995649332659585346C584997496129254743584
7C5849974961292547435847D5947995246332654445247D5747995346292
656455449B4937755852252937464653B4938755851303035464656B50316
65946243050465759
```

(b) Code génétique

FIGURE C .8

Certaines des pattes les plus longues de la Spider ont rétrécies, il ne lui en reste plus qu'une très longue. Elle arrive désormais à s'orienter dans plusieurs directions, et semble avoir adopté un mode de déplacement similaire à celui de Phoque.

**Phoque :**


(a) Vidéo de Phoque après 3 entraînements pour apprendre à tourner

```
A8943984958070941485749B4353319893585852463754B43
53319893585852463754B3550339995595251504049B35503
39995595251504049B4549349992635145473651B45493499
92635145473651B3949319992595950444343B39493199925
95950444343D6758997048030758584654D6260956932040
559584155D586095624304135947253D5860956243041359
474253D6264966151031061484456D6263966348020862544
265D6259966446060864594267D5258996247061059614457
D5761994754140755434655D5456994857140655444753D50
55964964170059415352D4854974760170059394752D46509
95554130057415053D4652995456210054455057D45529252
39070352503060D505399494215065743375D50539949421
50657433757D4855995041130657483759D44599247450801
52423561D4159954342090052423859D41599543420900524
23859D475490474507005343356A0742884145091348425
538A1756914542000140634742B6650359605606055485555
C3450935666030253444735C3256945264010448474131C3
456945663050251473833C2856945166000047474436C2856
945166000047474436C35539455607035744441C365094
6056020052464946C2855985754060253434744C234699525
0030248575737A4253946151373350585458B357529395801
0345514260B3135335165110534455955B354431445805064
3515365B2949383655000551505346
```

(b) Code génétique

FIGURE C .9

On peut voir que le phoque parvient à se tourner et à se déplacer latéralement pour atteindre les différentes cibles. Ses pattes arrières ont grossies, et elles sont stimulées différemment en fonction de la direction de la cible pour pouvoir tourner. Le phoque n'a plus qu'une patte avant stabilisatrice. Son code génétique s'est grandement complexifié.

### C .1.1.3 Aller plus vite

**Manodon :**



(a) Vidéo de Manodon après 1 entraînement pour apprendre à aller plus vite

```
A9951297944192870585447B4149406748233453515654B42483573682541
5253545284557417557193943606046B4557417557193943606046C43503
35064154047365644D4352344666153749305146C4139315551153545334
447C4850344957173555364948C5646465856203045424944C4841315557
062756425550C3953324959232954335450C3959334355212862505452C4
743345459253055445250C5342364752132450514752C424728535830435
4363747C4037354958143057435041C454334425013345644642D4643
364755112956435040C4350354761153747345246D435035476115374734
5246C4239315548143545354848C5049365157173154364948C554343595
6153144374744C4744305558053156425950C4053294756232857325250C
4155364356232559495352C4945345458202655444950C54463649521829
54474850C424528537284355354148C414135455413156434646C41413
54554143156434646D4141364954122857434643A12412481701737674743
60A0954268361183560363854B3955358647273471475444B636735855127
3262475163B5155307553172763435357C4056295965271950523949C3860
256265301253483754C5348344147163258525454D504932414620355953
5658C5047445257193846365246C3952345649232144535847C435633565
1222443505152A5149796241978308454748B5149796241978308454748A52
44787850957404584149A4248837546958106514944A434980740907906
465156A4349807040907906465156B4349836841907904465654B4349817
244907507465555A5450759052788708554852B4353996460948800325845
B5564917954957603525654
```

(b) Code génétique

FIGURE C .10

Le Manodon a beaucoup plus de membres qu'avant, et sur ceux qu'il avait déjà, des ramifications se sont formées, ce qui lui permet d'être un peu plus rapide. On constate également que certains de ses membres sont un peu plus longs ce qui lui offre plus de stabilité sur le sol.

**Mille-pattes :**



(a) Vidéo de Mille-pattes après 1 entraînement pour apprendre à aller plus vite

```
A539982969999852515192B5899829897979949544293A779195999949956
467089B709296949898983576891C709296949898983576891B729395949
9969960627587C7294999794989962477783D508182979999352584687D51
81829997989252554789D5084819995999348525389D4780769991999156575
095D51807996999495050565396A4942995648475251565649B62509943505
33950535840B5950994350544247525348534799485448504944745B6
250914549484849535150B6250914549484849535150C3685999995999952
652586D51827493889956024189C74999889939941506896D509078999
7969756485090E5286739699919954515396C5059993852534241464449D5
555984255474143454452C5653994550594039494747D259099969999956
553999D719896918592751518292E509984999919457435782
```

(b) Code génétique

FIGURE C .11

Le Mille-pattes est à présent plus recourbé sur lui-même, ses membres ont légèrement évolué mais l'ensemble reste similaire. On constate qu'il est quand même un peu plus rapide qu'avant.

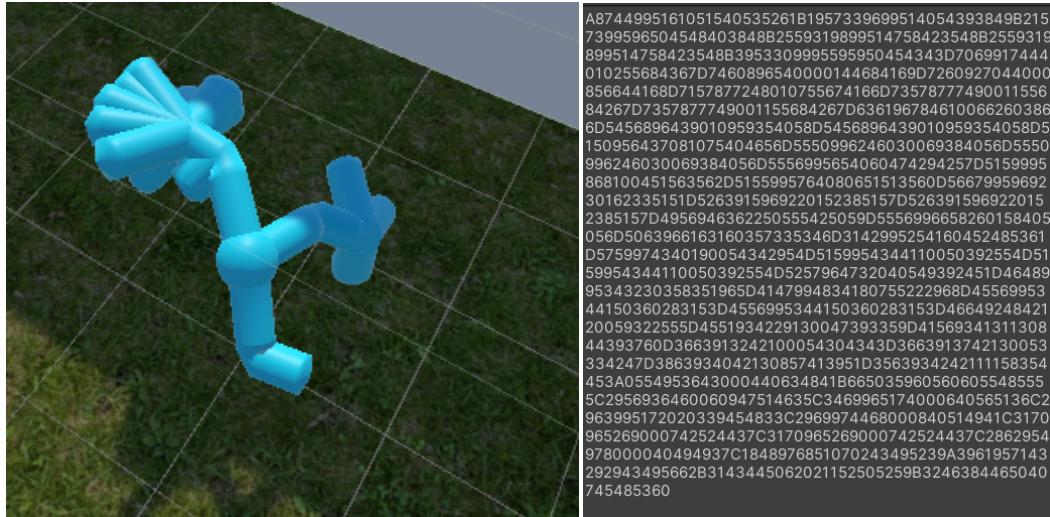
**Spider :**


(a) Vidéo de la Spider après 2 entraînements pour apprendre à aller plus vite

(b) Code génétique

FIGURE C .12

Les membres les plus courts de la Spider se sont un peu allongés, et se trouvent désormais un peu plus vers l'arrière, grâce à eux elle se déplace plus vite. On constate également que le membre le plus long qu'elle possédait a encore grandi, et qu'elle s'est déplacé vers l'avant de la Spider, agissant comme une sorte de trompe pour atteindre sa cible.

**Phoque :**


(a) Vidéo de Phoque après 3 entraînements pour apprendre à aller plus vite

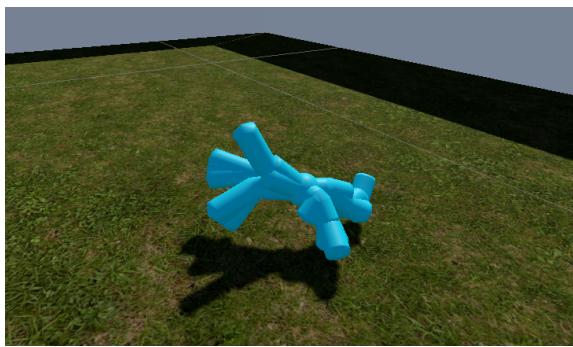
(b) Code génétique

FIGURE C .13

On peut voir que le phoque parvient désormais à atteindre les différentes cibles plus rapidement. Ses pattes arrières se sont encore élargies. Son code génétique s'est encore un peu complexifié.

#### C .1.1.4 Optimiser l'énergie consommée

**Manodon :**



(a) Vidéo de la version finale de Manodon

```
A9746338350232968555846B4353387055204448535052B4551357365234  
44854584984557427362213643576249C3951354657153747314949C4950  
304959123653385249C4950304959123653385249C463930585212264933  
5047D4639305852122649335047C544832555143447375049C505132565  
2083155465945C3652314754253061355251C4151374354232463464848C  
4846424959233269415244C486424959233269415244C5649354951632  
56394356C394533557293858404642C3945335557293858404642C3742  
354553142655453446C3742354553142655454346A1354268659193561353  
758B4356378456293670445645B616627835225263454862B47542975531  
72661415358C4156306463291953523649C5048324147203158525658C584  
547505813946326144C3952345649232144535847C425830625720214249  
5152A5149756741998711454747A444578574972600484552B47447978479  
37603524851A4248817547987905465047A434484764889790751571A95  
3808548838806604757B4756916255978500345846B5258916256988901335  
946B5862987651947504475949B5862987651947504475949A54499080509  
99354484653B495089704089840474746B4953896944868547514846B51  
5488744592885342485084857897844968545453845
```

(b) Code génétique

FIGURE C .14

Le Manodon est plus polyvalent dans ses déplacements, son génome s'est très largement complexifié, sans doute dû au nombre de membres élevé qu'il possède à présent. **Mille-pattes :**



(a) Vidéo de la version finale de Mille-pattes

```
A7791959999949956467089B7297949093999754597983C739499979297976  
244738C739499989597986341738D4983859699939651564790D5382849  
890898852535289D458079999199915956294D5087769995949855545296D  
5086759859499555250950508182979999352584687D51818298979596525  
95091D5084819995999348525389D4379769991999156545092D5180796999  
949505653964453999604749525156049B5955994046533549535543862  
5395354661405051504386048894951474849534751B6250914149504455  
45150B6250914149504445545150C378499995999753682486C458373919  
095946260042945083709691979666584089C7195999887929944516893D4  
888749699919952505795C4860983854544241464750C545098425054041  
524647D239094969998950553798E239094969998950553798D719995868  
2939651518691E4599789995879761385984E4599789995879761385984
```

(b) Code génétique

FIGURE C .15

La version finale du Mille-pattes est assez impressionnante. Il utilise ses pattes arrières (deux en particulier) pour effectuer ses sauts, mais leur recourbement lui offre aussi plus de stabilité. Sa grande patte avant assez mobile lui permet également de "galoper" et lui fait gagner en vitesse.

**Spider :**



(a) Vidéo de la version finale de Spider

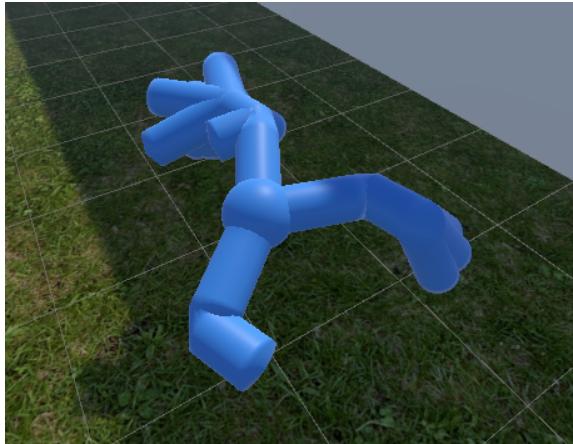
```
A2854166063514157805848B2854166063514157805848A214816666942575  
4684662B5562975634415848593048C4143594356241653636765D464560  
4160182349646565B5362995434405548623451C474173454927255553526  
2D434273444923285524662E434273444923285524662A426746315820  
2142475048B5363443249303234494849A534970555242958497157B495  
7684655433067606160C525172484873171636547D5251724848473171636  
547A7447194572953295640448595297546124373958605/C51499547592  
6244445649D574695555139244955649D646965548352349465052D52  
51995245302957445846
```

(b) Code génétique

FIGURE C .16

L'apparence de la Spider s'est beaucoup simplifiée, elle possède désormais moins de membres. Elle utilise ses deux grandes pattes arrière pour s'orienter et se déplacer, elle utilise celle du dessous comme stabilisateur, et sa trompe lui permet d'atteindre plus vite sa cible.

**Phoque :**



(a) Vidéo de la version finale de Phoque

```
A8346994661072237585657B1951359793444849434342B39  
53309995595950454343D7164937348000257694467D59589  
57442080862684068D5557957844111062694468D585199654  
6050071354059D5360996056030175255057D515599586110  
0755513458D5566955977200562375255D476588626420064  
7395452D4659954344110450432555D43509441220800434  
13964D4564852835140554304341D32549641491001504443  
48D4362943838081463364354A0550953342010035635043  
B6650359605606055485555C3268995574000640574842C32  
68995574000640574842C3368994774020635594739C3064  
975072010448544333C2562975280000540495036A405696  
7046262940495364B2950405267010440485759
```

(b) Code génétique

FIGURE C .17

On peut voir que le phoque est maintenant capable d'atteindre toutes les cibles du parcours, et même de le recommencer. Son code génétique s'est grandement simplifié, notamment parce que ses pattes arrières ont rétrécies.

Finalement, voilà à quoi ressemblent nos créatures une fois entraînées :

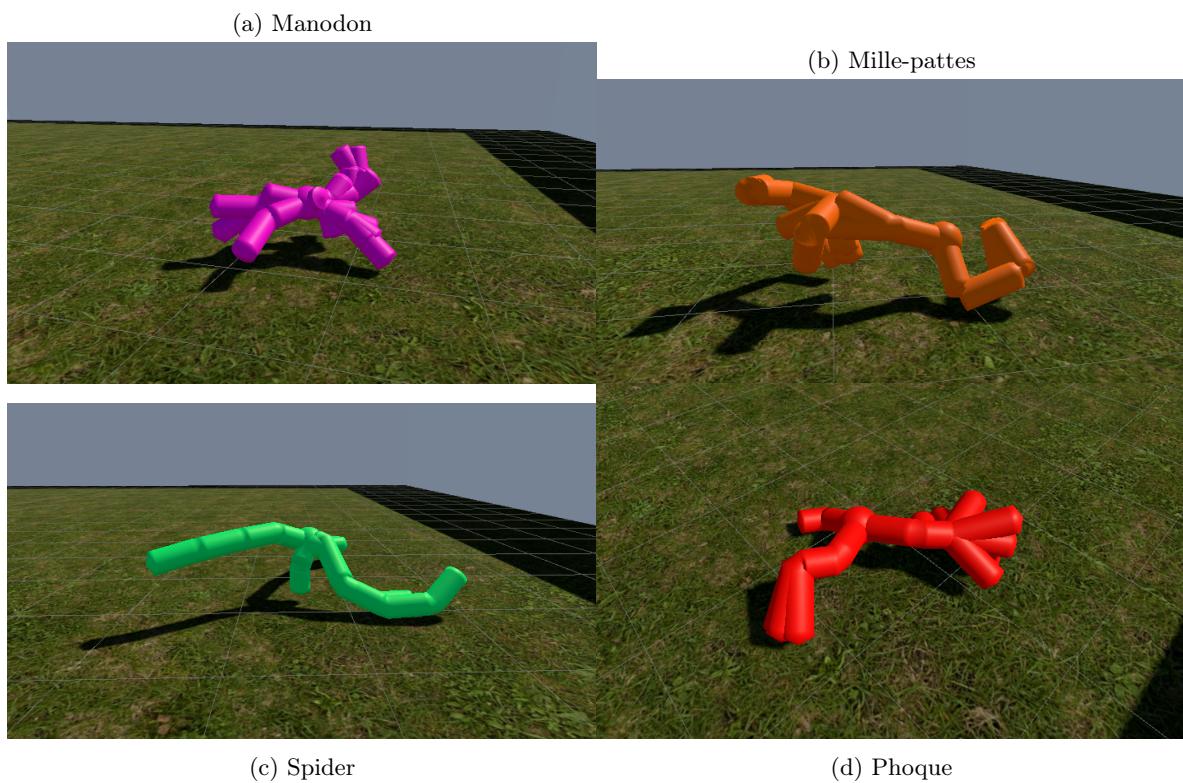


FIGURE C .18 – Créatures finales

## C .2 Simulation

Maintenant que nous avons des créatures capables de se déplacer, et d'atteindre une cible, on va pouvoir les placer dans notre environnement afin de réaliser la simulation.  
Nous avons réalisé différentes simulations dans chacun des biomes (qui sont séparés).

### C .2.1 Forêt

Rappelons les caractéristiques du biome Forêt :

- Le terrain est vallonné mais globalement plat
- Il y a beaucoup d'arbres, et donc de fruits
- On peut donc s'attendre à obtenir des créatures peu agressives qui se concentrent sur les fruits

Nous avons réalisé 5 simulations : [Simulation 1](#) [Simulation 2](#) [Simulation 3](#) [Simulation 4](#) [Simulation 5](#)

Dans la première simulation, les différentes espèces apparaissent assez éloignées les unes des autres, et on peut voir qu'elles restent séparées.

Rapidement, seuls les Phoques et les Mille-pattes survivent, puis il ne reste plus que les Phoques. A ce moment là, la démographie a fortement chutée, et l'agressivité de la population est scindée en deux modes : faible agressivité et forte agressivité. Rapidement, les créatures à forte agressivité prennent le dessus ; puis la démographie se met à augmenter progressivement en même temps que l'agressivité, qui plafonne à son maximum pour la majorité des créatures.

On arrête la simulation quand la population atteint les 80 créatures, nombre maximale pour conserver de bonnes performances. On observe à ce moment là que les Phoques en vie sont le résultat d'environ 40 générations ; ils semblent avoir évolué vers une forme assez statique, avec de longues tentacules qui bougent rapidement, ce qui facilite probablement l'attaque, ainsi que la nutrition.

Dans la seconde simulation, les différentes espèces apparaissent également assez éloignées les unes des autres.

Rapidement, seules deux populations séparées, une de Phoques, et une mixte de Spiders et de Mille-pattes survivent. Les Phoques se reproduisent rapidement, tandis que les Spiders et les Mille-pattes semblent se battre.

Les Mille-pattes finissent par gagner, avant de s'éteindre : encore, il ne reste ici plus que la population de Phoque, qui continue de s'agrandir.

Leur agressivité est à ce moment répartie uniformément sur toute la plage de valeurs. Vers la moitié de la vidéo, la population chute rapidement, et l'agressivité tend cette fois-ci vers une répartition uni-modale, et de valeur intermédiaire. Les Phoques semblent avoir développé les pattes plus rapides, ce qui les avantage probablement pour s'attaquer.

Finalement, le dernier Phoque décède, et la simulation prend fin.

La simulation 3 commence de la même manière, mais cette fois-ci, ce sont les Spider qui survivent. Leur agressivité reste faible, et leur population décroît progressivement avant de se stabiliser à quelques individus. On peut voir que les Spider semblent avoir développée de longues pattes au cours des générations, qui semblent leur servir à attraper plus facilement les fruits.

La simulation 4 se déroule de la même façon que la précédente, mais une petite population de Phoque parvient également à survivre.

L'agressivité est répartie uniformément, mais l'altitude préférentielle des fruits tend vers sa valeur maximale : on peut voir que la plupart des Spider ont une valeur élevée, mais pas les Phoques.

Les Spider semblent encore avoir développé de longue pattes pour mieux attraper les fruits, et frapper plus fort et de plus loin les autres créatures.

Les Phoques semblent être devenus immobiles, et se seraient probablement éteint rapidement.

Dans la cinquième simulation, c'est cette fois-ci une population mixte de Spider et de Mille-pattes qui survit.

Les Mille-pattes prennent rapidement le dessus, et se scindent en deux populations séparées, une avec une agressivité forte, et l'autre faible.

La population avec l'agressivité forte s'éteint, et il ne reste plus que la population pacifique, qui semble persister autour d'un arbre en mangeant ses fruits et en se bagarrant de temps à autre. Ils semblent avoir évolué vers une forme allongée, comme deux tentacules bougeant rapidement.

En mettant en communs les résultats de ces 5 simulations, on peut d'abord conclure que la créature Manodon semble avoir une faible survivabilité, en tout cas dans ce biome.

Ensuite, les Phoques et les Mille-pattes semblent le mieux s'en sortir, mais les Spider peuvent maintenir une population stable si on les laisse tranquille. Les Mille-pattes semblent être avantagés lorsqu'il s'agit de se battre, et ont tendance à éradiquer les espèces voisines.

De manière globale, dans ce biome, développer de long membres semble être un avantage, ce qui s'explique par l'avantage que cela procure en combat, mais aussi pour attraper, les fruits, qui sont abondants et quasiment toujours à portée de tentacule.

### C .2.2 Banquise

Rappelons les caractéristiques du biome Banquise :

- Le terrain est globalement plat, avec quelques buttes
- Il y a peu d'arbres, et donc de fruits
- Il fait très froid, donc les créatures consomment d'avantage d'énergie (+10%) et ont besoin de plus d'énergie pour se reproduire (+10%)
- C'est le biome le plus difficile en théorie, on peut donc s'attendre à une faible survivabilité

Nous avons réalisé 3 simulations : [Simulation 1](#) [Simulation 2](#) [Simulation 3](#)

Dans le première simulation, les différentes espèces apparaissent séparées les unes des autres.

Il ne se passe pas grand chose, et la population diminue rapidement.

On peut voir que l'agressivité semble augmenter, mais étant donné la faible démographie, il s'agit probablement de dérive génétique et non de sélection naturelle.

A la fin, il ne reste plus qu'une petite population de Spider, qui survivent tant bien que mal autour d'un arbre, et meurent si elles s'en éloignent trop.

La dernière Spider décède rapidement.

Dans la seconde simulation, les différentes espèces apparaissent également séparées, mais sont davantage éparpillées.

Il se passe globalement la même chose, mais ce sont cette fois ci les Mille-pattes qui survivent le plus longtemps.

Dans la troisième simulation, il se passe la même chose que dans la précédente.

Pour conclure sur ce biome, on peut voir que comme attendu, les conditions sont trop dures pour que nos créatures parviennent à survivre.



### C .2.3 Plaine

Rappelons les caractéristiques du biome Plaine :

- Le terrain est composé de collines entre lesquelles passent des chemins plats
- Il y a moins d'arbres que dans la foret mais plus que dans les autres biomes.
- Les arbres sont pour la plupart réparties sur les bords des chemins.

Nous avons réalisé 4 simulations : [Simulation 1](#) [Simulation 2](#) [Simulation 3](#) [Simulation 4](#)

La première simulation est un test sur une unique créature dans cet environnement pour ensuite anticiper les prochaines réactions et évolutions dans les prochaines simulations.

Nous avons donc choisi de prendre la créature mille-pattes pour ce premier test. Durant ce test les créatures vont apparaître sur les chemins et au milieu d'un groupement d'arbre important.

Nous pouvons vite remarquer un léger bond de l'agressivité des créatures, pour ensuite s'aplatir et être homogène au sein des mille-pattes. Assez vite dans la simulation nous remarquons que ce sont les créatures au milieu du groupement d'arbre en haut à gauche du biome qui se débrouillent le mieux. Certains mille-pattes favoriseront le combat et le vol d'énergie plutôt que de se nourrir, mais dans l'ensemble on peut conclure que le nombre conséquent d'arbres dans cette région du biome favorise grandement le développement et l'évolution des créatures. Nous pouvons aussi remarquer que les préférences d'altitudes des fruits sont extrêmement élevés.

La seconde simulation est une simulation globale avec toutes les créatures.

Nous avons décidé de mettre l'ensemble des créatures dans la simulation, et de mettre nos deux meilleures espèces dans le coin privilégié du biome. Nous aurons donc dans la partie gauche du biome les créatures phoque et spider. Manodon sera entre ces deux créatures. Mille-pattes quant à lui sera dans une intersection de deux chemins avec à sa gauche d'autres spider et au dessus de lui d'autre phoque.

Dès le début de la simulation nous avons une forte apparition de spiders et de phoques comme prévu. Mais aux bout de quelque minutes une forte chute des phoques montre une certaine difficulté de cette créatures dans cet environnement. L'agressivité totale reste plutôt faible mais à tendance à monter. Notamment à cause des Mille-pattes qui réussissent à survivre dans un coin en utilisant le combat et le vol d'énergie afin de se déculper.

À la fin de cette simulation seules deux créatures ont survécus : Spider et Mille-Pattes. Les Spider en ayant profité de leur endroit d'apparition et des arbres à proximité, les Mille-pattes en exploitant la mécanique de combat afin de combler le manque de fruit dans les alentours.

Dans la troisième simulation nous avons mis un groupe de Spider toujours au même endroit mais avec plus de Phoque à proximité. Nous avons aussi mis les Mille-pattes et les Manodons à coté sur l'intersection à droite du biome. Cette fois-ci les Phoque vont bien mieux se débrouiller en prenant beaucoup plus de place tout le long des chemins et vont rester à proximité des arbres. Ils vont même jusqu'à prendre la place des Spider et les faire disparaître soit en se battant, soit en prenant toutes les ressources disponibles. Les Manodons ne réussissent pas à appliquer la même stratégie que les Mille-pattes et disparaissent rapidement. Cependant les Mille-pattes n'arrivent plus à se reproduire de la même manière que la dernière fois. En effet leur agressivité est plus faible que précédemment et n'arrivent donc plus à appliquer la même stratégie.

Nous nous retrouvons donc avec un biome rempli alors uniquement de phoque, malgré le premier constat fait à la seconde simulation.

La quatrième simulation est un réplique de la troisième mais sans les manodons que nous avons jugés trop faibles pour cet environnement.

Cette fois-ci nous avons fait apparaître les Spider et les Phoques au milieu du biome et les Mille-pattes aux extrémités. Nous remarquons rapidement que les Spider ont du mal à se développer



sans la présence de beaucoup d'arbres et que les Phoques essayent d'appliquer la même stratégie que la dernière fois. Cependant les Mille-pattes seront pour la plupart plus agressifs que les autres espèces et finiront par les exterminer du biome.

En comparant ces simulations nous pouvons conclure que dans cet environnement, les différentes créatures appliquent différentes stratégies afin de se reproduire et de se développer. Là où la Spider a besoin de ressources à proximité et aux alentours, les Phoques s'adaptent mieux et les Mille-pattes privilégièrent le combat pour combler un certain manque de nourriture.

Ces résultats sont similaires à ceux du biome Forêt, où les Spiders privilégiaient également les fruits et avaient besoin d'être laissées tranquilles pour survivre, et où les Mille-pattes avaient aussi tendance à préférer la stratégie de combat.

## C .2.4 Désert

Rappelons les caractéristiques du biome Désert :

- Le terrain est complètement plat
- Il y a un nombre moyen d'arbres qui ne donnent que peu de fruits, mais ceci sont très nutritifs
- Le milieu du désert est globalement moins fourni en arbres et fait office de séparations entre deux zones
- Les zones de part et d'autres de cette séparations sont les plus fournies en arbres

Nous avons réalisé 3 simulations : [Simulation 1](#) [Simulation 2](#) [Simulation 3](#)

Dans la première simulation, les zones de départs de chaque espèce sont assez espacées. Il y a deux groupes de chaque espèces, sauf les mille-pattes qui sont présent à travers un seul groupe, pour limiter le nombre de créatures présentes et ne pas affecter les performances. Les Manodons sont les premiers à disparaître, suivis du groupe de Mille-pattes. A ce moment-là il ne reste plus que les Spider, qui ont diminué au sein d'un groupe, mais qui sont assez stables au sein de l'autre, et les Phoques qui sont encore présent à travers un groupe. Les derniers Manodons finissent par mourir, ne laissant plus que le dernier groupe de Spider qui vont continuer d'évoluer normalement, jusqu'à atteindre deux derniers individus qui vont mourir à leurs tours. Ces deux derniers individus auront atteint la 8ème génération, il semble qu'ils aient développés plus de membres qu'ils n'en avaient de base, et leurs membres ont tous la même longueur, abandonnant la trompe unique qu'ils possédaient auparavant. L'agressivité des créatures est resté globalement faible dans l'ensemble de la simulation, avec une ou deux créatures ayant des pics à certains moments, mais dans de rare cas exceptionnels.

Dans la seconde simulation, les zones de départs de chaque espèces sont désormais beaucoup plus proches, et elles sont réunies dans la parties supérieurs du désert, la zone ombrée. Nous avons encore une fois deux groupes de chaque espèce, excepté pour les Phoques où il n'y en a cette fois-ci qu'un. On observe que la population des Phoques, et des Manodons chute drastiquement au bout d'une minute, suivies des Spider, tandis que les Mille-Pattes conservent une très grande population et semblent dominer la zone. Ces derniers affichent une agressivité allant de faible à moyenne, et ils finissent par atteindre 16 générations pour certains des derniers individus. On remarque qu'ils ont là aussi beaucoup plus de membres qu'au départ ce qui leur a probablement permis de se déplacer plus vite, afin d'être les premiers à atteindre les fruits. Les deux derniers individus de la population finissent là aussi par mourir.

Dans la troisième et dernière simulation, les zones de départs sont à nouveau espacées, mais les groupes de chaque espèce sont réduit en population, étant de 10 individus auparavant, on passe désormais à 5 individus par groupes. On constate que les groupes de créatures se développent dans la partie inférieure du désert (la partie éclairée) sont les premiers à disparaître. La population dans la zone ombrée diminue beaucoup également, mais un groupe de Spider semble se stabiliser plus que les autres. Après quelques minutes, il ne reste finalement plus que les Spider, qui affichent un niveau d'agressivité moyen, ils semblent qu'elles priorisent la nourriture, mais sans pour autant hésiter à se battre entre elles si elles se croisent. Elles ont là aussi atteint la 8ème génération, qui semble similaire à celle de la première simulation, à savoir plus de membres mais de longueurs équivalentes. Les deux derniers individus finissent par mourir eux aussi.

Pour conclure, il semble que les espèces se développant le mieux dans le désert sont le Mille-Pattes ainsi que la Spider. Les Manodons et les Phoques ont un taux de survie assez faible. L'évolution privilégiée dans ce biome semble être l'ajout de plus de membres. Cela peut s'expliquer par le fait que les arbres et donc les fruits sont plus espacés que dans d'autres biomes, la présence de plus de membres peut donc accélérer les déplacements des créatures et les aider à atteindre les fruits les premières.

# Conclusion et perspectives

Pour conclure, ce projet de simulation d'un écosystème virtuel a été une expérience enrichissante. Nous avons appris à utiliser des algorithmes génétiques d'une manière à la fois amusante et concrète.

Nous avons pu créer une simulation simple permettant d'analyser les comportements et les structures qui favorisent la survie de nos créatures. Davantage de recherches seraient nécessaires afin d'obtenir des résultats plus concrets, mais nous sommes satisfaits de ce que nous avons accompli !

On pourrait par exemple imaginer créer davantage de créatures, et les faire passer par d'autres entraînements divers, afin d'obtenir des créatures spécialisées dans certaines tâches.

Notre environnement, très simple, pourrait être amélioré et complexifié afin d'obtenir de nouvelles interactions avec les créatures. Aussi, nos créatures ne s'accouplent pas, mais se reproduisent seules ; il pourrait être intéressant de développer un aspect "social" à nos créatures, avec différentes caractéristiques également évolutives, et in-fine utiliser le système de crossover pour fusionner deux créatures qui se reproduiraient ensembles.

# Bibliographie

- [1] Marcin L. PILAT et Christian JACOB. “Creature Academy : A system for virtual creature evolution”. In : *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. 2008, p. 3289-3297. DOI : [10.1109/CEC.2008.4631243](https://doi.org/10.1109/CEC.2008.4631243).
- [2] Karl SIMS. “Evolving 3D Morphology and Behavior by Competition”. In : *Artificial Life* 1.4 (1994), p. 353-372. DOI : [10.1162/artl.1994.1.4.353](https://doi.org/10.1162/artl.1994.1.4.353).
- [3] Tim TAYLOR et Colm MASSEY. “Recent Developments in the Evolution of Morphologies and Controllers for Physically Simulated Creatures”. In : *Artificial Life* 7 (2000), p. 77-87.

# Annexes

Vous trouverez l'ensemble de nos scripts [ici](#).  
Et un exécutable [ici](#).