Simulation et optimisation de créatures basiques soumises à des contraintes simples.

Théorie et pratique des algorithmes génétiques.

Orélian Kohler

Travail de maturité 2018-2019

Introduction

Les algorithmes génétiques sont à la base de nombreuses inventions modernes. Cette technologie est, par exemple, devenue indispensable à l'optimisation et au perfectionnement des bases de données et des mouvements des robots contemporains. Bien que ces applications paraissent particulièrement complexes en raison de leur utilisation dans des milieux scientifiques très spécialisés, il est possible d'en comprendre les fonctionnements avec des applications relativement basiques et d'utiliser les principes des algorithmes génétiques pour résoudre des problèmes de la vie courante. C'est pourquoi j'ai personnellement choisi, pour mon travail de maturité en informatique, ce sujet qui me tenait à cœur, liant à la fois sciences complexes et vie réelle.

Dans cette optique, j'ai orienté mon travail de maturité dans cette direction et j'ai decidé de travailler sur un projet un projet à mon niveau qui me permettrait d'utiliser les outils informatiques je j'ai pu développer durant mes années de maturité tout en me faisant face à de nouveaux défis à relever. Je souhaitais également utiliser de nouveaux outils avec lesquels je n'avais encore jamais travaillé. Mon choix final s'est porté sur une simulation d'un environnement simple. Cela me permuterait aussi de pouvoir appliqué ce que j'ai pu voir pendant mes cours de physique, de mathématique et aussi d'informatique. Suite à la vue d'une vidéo où quelqu'un avait crée une simulation semblable avec des créatures qui évoluait de génération en génération. J'ai voulu tenter moi-même l'expérience et comprendre à mon tour les mécaniques sous-jacentes et pouvoir en expliquer les fondements. Mon choix final s'est donc porté sur une simulation de créature basique qui serait capable de se mouvoir dans un environnement. L'enjeu principal sera donc pour mon projet de déterminer la créature la plus optimisé pour faire la tâche sur laquelle on l'aura jugé.

Mon projet peut se diviser en plusieurs parties. Dans un premier temps, il faudra crée un monde avec des règles physiques où les créatures seront testées. Ensuite, il faudra créer des créatures basiques qui auront seront presque identiques dans leurs structures, mais qui différeront dans leurs aspects comportementaux. Il faudra crée un moyen de déterminer qu'elles sont les créatures qui ont le mieux réussi la tâche sur laquelle elles ont été jugées. Après les sélections les meilleures, celle-ci prendront plus de place dans la population des créatures totale, elles supprimeront celles qui auront moins bien réussi et pour que une optimisation se produise dans la population, les créatures seront modifiées et croisé avec des autres pour créer un forme de génétique propre aux individus de la simulation.

Hypothèse → Je suis parti de l'hypothèse suivante…

Je suis parti de l'hypothèse suivante : selon moi, dans un monde assez simple avec des variables parfaitement contrôlables et avec des intervenants simples également, il sera facile de déceler qu'elles sont les paramètres les plus importants à optimiser et que après un certain nombre de génération les éléments de la simulation réussiront tous les tests mieux que celle de la première génération créée aléatoirement.

Ce dossier éclairera plusieurs point de réflexion que j'ai eu pendant mes temps de recherches que ce soit au niveau théorique ou dans la programmation de mon travail de maturité.

Ce dossier va éclairer la manière dont je suis parvenu à démontrer mon hypothèse. En premier j'y expliquerai… dans un second temps...

→ plan travail (3 phrases)

Théorie des algorithmes génétiques

Le but d'un algorithme génétique est un outil théorique qui tente d'apporter une méthode de résolution à de problèmes variés où les solutions ne sont ni clairement définie par une méthode exacte ou nous sont simplement inconnues comme par exemples : la résolution d'une fonction quelconque. Bien que les résultats ne soient jamais véritablement garantis, cet algorithme nous propose une démarche qui est susceptible de résoudre des problèmes complexes.

Historique

Réflexions

La réflexion sur laquelle se passe l'algorithme génétique et la même qu'en biologie. Les individus les plus à mème de survivre à leur environnement seront les plus à même de se reproduire et de survivre de génération en génération. On dira que les algortihme génétique sont bio-inspirés. Partant de ce principe, la but d'un algorithme génétique est de mettre en compétition un certain nombre d'individu crée artificiellement et de juger en fonction de leur capacité à résoudre un problème donné. Dans cette optique, cette méthode essayera de mimer de perfectionner la sélection naturelle opérante dans la nature en la transposant à des problème lié à l'informatique.

Base

Le fonctionnement de base d'un algorithme génétique est le suivant. À l'initialisation de l'algorithme, une première génération de *n* individus est créée une épreuve leurs est soumise et la première itération de l'algorithme est génétique est ainsi effectué. Ensuite pour chaque nouvelle interaction de l'algorithme, l'on créera une nouvelle génération par rapport aux résultats de l'épreuve des individus de la génération précédente.

Étant baser sur les travaux effectués en biologie, les génétiques utilisent un vocabulaire similaire. On dira que chaque individu d'une simulation à un génome qui lui est propre et à chaque fin d'itération, ce génome sera légué au individu de la génération suivante selon trois paramètre.

- La sélection : plus l'individu aura obtenu un bon résultat à l'épreuve demandée plus le génome sera susceptible d'être légué aux individus de la génération suivante.
- L'enjambement : les génomes de deux individus peuvent se mélanger de façon aléatoire pour créer un troisième individu dans la génération suivante.
- La mutation : une partie infime du génome de l'individu peut être modifié aléatoirement de génération en génération. Cette mutation a pour but de simuler les mutations opérant dans le vivant et d'obtenir des résultats plus concluant en apportant des modifications locales au génome.

Applications

Il existe plusieurs applications intéressantes au algorithme génétique et concrète. Il est ainsi facilement possible de trouver un zéro d'une fonction en utilisant un algorithme génétique. Il suffit de définir un zone de recherche où un zéro serait susceptible d'être présent et de créer une première

génération donc chaque individu aura x comme génome et f(x) comme résultats et plus le critère de réussite de l'épreuve étant de s'approcher de 0. Et en utilisant, la sélection pour garder que les meilleurs résultats, l'enjambement pour obtenir la moyenne entre deux x différents set la mutation pour affiner les recherches. Il est également possible de rechercher les extremas d'une fonction de la même manière en prenant cette fois-ci, f'(x) comme résultat.

Bien que cette méthode ne soit pas aussi efficace que d'autre algorithme de recherche de zéro, elle est une application facile à mettre en place pour comprendre les mécanismes d'un algorithme génétique.

Il est également possible de trouver d'autres exemples plus complexes tels que des robots qui ont appris à marcher par eux-même grâce à un algorithme génétique tel que le robot d'exploration martien de la Nasa Pathfinder ou encore le robot Aibo de Sony. Ou encore dans différents domaines, comme l'urbanisation pour trouver une configuration de rues optimales dans une ville ou encore dans la gestion de données d'un serveur. Les applications ne manque pas et cette méthode est un outil efficace dans bien des domaines tant qu'il est possible de créer une simulation de la situation étudiée dans un environnement informatique.

Schéma de mon projet

Dans l'idée de créer mon propre exemple d'algorithme génétique. J'ai réfléchi à la simulation que je voulais modéliser. J'ai dans un premier temps du défini mes individus que j'ai sombrement nommés créature. Celle-ci devront apprendre à se déplacer par elle-même et seront juger sur la distance totale qu'elles auront effectue en un laps de temps donné.

Créatures

Chaque créature serait composé de trois nœuds et de trois liens qui relierait ses nœuds. Chaque nœud et chaque lien ont plusieurs paramètres. Les nœuds auront leur poids et leur coefficient de frottement qui seront variables. Et chaque lien aura un muscle qui pourra s'allonger pendant une période définie avec une force qui lui sera propre et chaque lien aura également agira comme un ressort avec une longueur initiale et une constante de raideur. Ainsi les liens entre les nœuds pourront maintenir la forme de la créature et pourront également lui fournir un source de mouvement le but étant d'optimiser de tout pour parcourir la plus grande distance.

Environnements

Ensuite comme les créatures ont été définies, il faut envisager l'environnement dans lequel elles vont devoir se mouvoir. J'ai pensé à un environnement très simple qui serait composé d'un sol et également chaque nœud de la créature serait attiré pas le sol à cause de la gravité.

Algorithmes

Après avoir défini l'épreuve sur laquelle les individus de ma simulation seront jugés et sur leurs caractéristiques, il est possible de mettre en place un algorithme génétique sur cette simulation. Pour commencer, il faudra créer une première génération composée de *n* créature qui devront essayer de parcourir la plus grande distance pendant un certain nombre de temps grâce aux muscles. Et ensuite, un certain nombre des meilleures créatures de la première génération seront sélectionnées pour ensuite être mélangé entre elles-avec l'enjambement de leur génome qui sera ensuite légèrement muté. La génération suivante sera donc formée des créatures ainsi obtenues qui referont l'épreuve et qui seront à leur tour muté et croisé et le processus sera ainsi répété jusqu'à l'obtention de résultats satisfaisants.

Le plan de mon projet en tête, je peux maintenant m'atteler à la programmation.

Programmation

Dans ce chapitre, je vais parler de tout l'aspect programmation de mon travail de maturité. Je vais expliquer avec quels outils j'ai réalisé mon projet. Comment j'ai programmé la logique de mon projet comment j'ai codé l'aspect graphique pour pouvoir voir les résultats de ma simulation en direct. Ensuite, j'expliquerai comment j'ai récolté les données obtenues de mes simulations et enfin comment j'ai créé une interface pour pourvoir voir les résultats et les analyser.

Outils utilisés

Mon travail sera fait presque exclusivement à l'ordinateur puisque les ressources, dont j'ai besoin pour réussir mon projet, sont informatiques et non matérielles. Pour pouvoir commencer correctement à travailler, il me fallait cependant définir plusieurs outils que j'allais utiliser pendant mon développement. Dans un premier temps, l'outil principal était le langage de programmation. C'est un choix assez compliqué propre à chacun et à ces capacités. Je ne voulais pas spécialement apprendre un langage de programmation juste pour ce projet puisque l'aspect logique de celui-ci était pour moi la difficulté la plus intéressante. J'ai choisi de programmer en Javascript et de manipuler des `<canvas>` HTML pour les animations et l'aspect graphique. Ce choix, c'est principalement fait par

choix d'efficacité, je souhaitais perdre le moins de temps entre chaque test de code. C'est pour cela, que coder en Javascript sur Firefox ou Chrome qui ont les deux de très bons debuggers, me semblait une très bonne solution. Et connaissant déjà la communauté très active sur internet des utilisateurs des Javascript, je savais que si j'avais des problèmes peu spécifiques, je trouverais facilement une solution en ligne ce qui fut le cas à de nombreuses reprises.

Pour avoir un plan de travail en ordre et organiser, j'ai utilisé le plus vite possible un outil informatique très connu et très utile : git. Pour faire simple, git est un outil de gestion de version de programme. Il permet en autre de garder un historique de modification des anciennes versions du projet, de pouvoir retourner à des versions antérieures du projet, et de faire des modifications à part du proget dans les phases de test pour ne pas altérer ce qui fonctionne. Et dans mon cas, en utilisant un service en ligne de git, je peux sauvegarder, mon projet de manière sécurisée en ligne. C'est un outil qui offre plein de possibilités, mais demander un temps d'apprentissage assez grand.

Ensuite, j'ai cherché une API pour facilité le développement de la physique de mon programme. Une API ou interface de programmation applicative en français est en quelque sort des bouts de code que l'on peut greffer à son programme et que l'on peut réutiliser. On peut également nommer une API une librairie. Dans mon cas, je vais utiliser plusieurs API. Mais la première, dont je vais parler, s'appelle *P5.js*. Sans renter dans les détails, cette librairie permet facilement de dessiner des traits, des cercles et des formes en 2D sur un canvas html. Elle offre également la possibilité de manipuler facilement des vecteurs en deux dimensions, de les additionner et de les multiplier, de calculer sa norme, etc.

Enfin, en vrac, quelques autre outil que j'ai utilisé :

- Chart.js qui permet de créer des graphiques simplement avec des fichiers de données. Cette libraire m'a été utile pour synthétiser de manière visuelle les résultats que j'ai obtenus.
- Socket.io est une librairie qui offre plein de possibilité pour tout ce qui touche aux communications locales ou aux communications via internet. Je ne veux pas trop entrer dans les détails, socket.io m'a servi à transférer les données

obtenues de l'interface client pour les données au serveur local et qu'il puisse les sauvegarder.

- Node.js est un interpréteur de Javascript. Il m'a permis de créer un serveur local qui donne accès à l'interface graphique et peut recevoir des données et de les sauvegarder.
- Bootstap qui sert à avoir facilement un meilleur visuel sur les pages HTML que celui proposer par défaut par le navigateur. J'ai utilisé cette librairie pour améliorer ma compréhension du CSS et également d'avoir un meilleur rendu des pages de l'interface utilisateur.

Aspects logiques

Comme j'ai décidé de coder mon projet en javascript, j'ai profité des nouvelles mise à jour du langage et des nouvelles fonctionnalités disponibles pour pourvoir faire de l'orienté objet. Pour expliquer rapidement l'orienté objet en programmation, cela permet de structurer le code de façon que chaque concept correspond à une classe qui aura ces méthodes attitrées à l'intérieur d'elle-même et les classe peuvent être étendu de façon qu'une classe mère donne ces attributs à une ou plusieurs classes filles. Pour donner un exemple, on peut parfaitement définir la classe mère *Animal* qui aura plusieurs attributs comme la taille de l'animal, le poids, etc. Et ensuite, un classe fille d'animal C*hat* qui en plus d'avoir les même variables que celles d'*Animal* aurait également des méthodes spécifiques aux chats tel que le miaulement ou autre chose.

J'ai donc élaboré mon projet en donner une classe spécifique à chaque composante de ma simulation.

En commençant par l'algorithme, il forme une classe qui comporte plusieurs variables comme la durer d'une génération, le nombre d'individus dans une génération. Cette classe comporte plusieurs fonctions comme une pour l'initialisation de l'algorithme, une pour le calcul de résultats de la génération précédent, une pour créer la génération suivante en utilisant la sélection, l'enjambement et la mutation. Et enfin, une fonction update() qui mets à jour toutes les créatures présentes de la génération.

Ensuite viens les créatures en elle-même, celles-ci sont définies par les trois nœuds et les trois liens qui les composent. La classe Créature n'a que deux fonctions principale : la première étant la fonction qui permet de générer les mutations et de les appliquer sur les nœuds et les liens et la deuxième

étant la fonction update() qui comme pour l'algorithme met à jour les états de touts les nœuds et liens de la créature.

Les liens sont donc formés de deux classe différentes chacune pour une fonctionnalité du lien. Une jouera le rôle du muscle et l'autre celui du ressort. Les deux classes auront dans tous les cas en mémoire les deux nœuds auxquels elles sont attachées. Pour la classe ressort, elle sera définie par une constante de constante de raideur *k* et une longueur du ressort initiale *l*. Quand la fonction update() est appelée une force est applique sur les deux nœuds attachés est au ressort d'intensité $F = k \cdot \Delta l$.

Quant aux muscles, ils sont définis par la force F de celui-ci, par deux temps d'activation : t_0 t_1 et par une horloge interne au muscle *c*. La fonction update() pour la classe muscle applique une force F sur les deux nœuds étant reliés à celui-ci et les repousse l'un de l'autre et cela quand $c \in [t_0 ; t_1]$.

Les nœuds quant à eux ont au tous un vecteur position : $\begin{pmatrix} x \\ y \end{pmatrix}$, un vecteur de vitesse: $\begin{pmatrix} Vx \\ Vy \end{pmatrix}$, un vecteur accélération : $\begin{pmatrix} Ax \\ Ay \end{pmatrix}$, un poids *p* et un coefficient de frottement μ. Chaque nœud possède également une fonction update() qui mets à jour le vecteur accélération en fonction de la résultante des forces exercer sur le nœud et du poids de celui-ci, elle met aussi à jour de la même façon le vecteur vitesse et le vecteur position.

Aspects graphiques

Gestion des données

Affichage des données

Résultats

Développement

Aillant décider des outils que j'allais utiliser. Je me suis mis à réfléchir comment j'allais penser mon projet. J'ai d'abord fait une ébauche des aspects qui seraient présents dans ma simulation.

Ébauche de la simulation désirée

Dans le but de tester les créatures et de pouvoir les améliorer, j'ai défini un critère arbitraire sur lequel elles seront jugées dans mon cas précis : la distance totale parcourue pendant un laps de temps donné.

Création de la simulation

Pour ce faire, il faut donc définir plusieurs critères. Tout d'abord, il faut choisir la forme et la structure des créatures et ensuite les contraintes physiques auquel elles seront soumises pour rendre la simulation intéressante.

Création des créatures

Dans mon cas, j'ai décidé que les créatures seraient composées de la façon suivante : elles auraient toutes trois boules et de trois muscles.

Boules

Chaque boule d'une créature a un poids spécifique et un coefficient de frottement qui lui est propre pour simplifier la vision dans la simulation plus le poids de la boule est grand plus elle sera grand et dans le même principe plus la boule est blanche plus les frottements avec le sol sont petits et inversement si la boule est jaune foncé.

Muscles

Chaque muscle de la créature joue deux rôles. Le premier étant de maintenir la créature en place et la deuxième étant de créer du mouvement pour que la créature puisse se mouvoir. Pour ce faire, chaque muscle a plusieurs variables qui lui sont propres : la longueur du muscle, cette longueur est la longueur que le muscle a au repos et à chaque instant le muscles exercera une force au deux boules auxquelles il est relié pour atteindre la longueur du muscle. Ensuite, il y a les moments où le muscle travaille pour éloigner les deux boules, le muscles est alors activé et devient rouge. Pendant cette phase d'activation qui est différente pour chaque muscle, une force éloignera les deux boules avec une intensité qui elle aussi lui est propre.

Contraintes physiques

Comme dit plus tôt chaque créature est composé de trois boules avec des poids et des coefficients de frottement. Dans la simulation, chaque boule est donc soumis à la gravité qui comme dans la réalité accélère les créatures vers le sol. Et également le sol qui freine en fonction du coefficient de frottement. Les créatures devront donc jouer avec ces deux contraintes pour pouvoir se mouvoir et avancer.

Evolution entre chaque génération

Maintenant que les créatures sont définies, il faut mettre en place l'algorithme génétique qui permettra d'améliorer les créatures.

Pour ce faire, il faut d'abord créer une première génération qui sera composée d'un certain nombre d'individus générer aléatoirement. Cette première génération dans la majorité des cas n'obtiendra pas de bon résultats au test. Mais, la génération suivant sera composé d'une partie des meilleurs résultats obtenu dans la génération précédente modifier légèrement, d'une partie de nouveaux individus générer aléatoirement et enfin d'un parti d'individu qui seront des mélanges des meilleurs de la génération précédente. Cette nouvelle génération ainsi obtenue sera elle aussi soumise à la simulation et une nouvelle génération pourra ainsi être générée. Le but final de cette méthode est d'obtenir une convergence vers une créature optimale pour le test auquel elle est soumise.

Conclusion

Bibliographie

Annexes