Systèmes Multi-Agents

Janvier 2022

Simulation d'un jeu de pêche

Rapport Final

Crédit : La Voix du Nord

Louis SOTO
Alexandre CONSTANTIN
Jean DESCAMPS
Vivien LECLERCQ

Première partie

Introduction

Dans ce rapport nous nous intéressons à l'implémentation d'un jeu de pêche. Nous présentons le travail que nous avons effectué pour le projet final l'UV IA04 à l'Université de Technologie de Compiègne.

1. Genèse du projet

Avant de commencer, il convient de vous présenter la manière dont nous sommes parvenu au choix de ce sujet. Au début, nous n'étions pas tous d'accord sur le projet à choisir.

1.1. Rendu graphique musical

Une des premières motivations de notre projet a été tout d'abord d'étudier la manière dont un système multiagent pouvait donner à voir de la musique. L'idée étant de visualiser plusieurs centaines d'agents dans un environnement qui répondent de manières différents aux stimuli que leur envoie la musique.

1.2. Fourmi

Une autre première motivation de notre projet a été de vouloir simuler le fonctionnement d'une fourmilière. Il s'agit d'agents dont le but est de devoir chercher de la nourriture dans un environnement donné afin de garantir le bon fonctionnement de la colonie.

2. Solutions explorées

Il a alors été fait le choix d'un système multi-agent dans lequel la musique a une place importante. Le but étant de créer des agents se déplaçant dans un environnement dans lequel la musique joue le rôle d'agent perturbateur. Il s'agit donc de choisir un moyen que vont utiliser les agents pour se déplacer dans cet environnement. Notons que ces deux solutions ont bel et bien été implémentés.

2.1. Slime

Une première solution envisagé a été le fait d'utiliser un slime capable de réagir à la musique. Le slime est une structure créée par des milliers d'agents. Ces agents, à la manière des fourmis, se déplacent dans l'environnement en suivant des traces qui sont laissées dans

l'environnement par d'autres agetns. Ils disposent également d'un facteur de divergence qui leur permet d'explorer de nouvelles régions de l'espace. En se basant sur ce principe, et en jouant avec les propriétés des agents, il est possible de voir émerger une structure que l'on appelle Slime.
L'idée première a donc été de vouloir créer un slime capable de réagir à la musique en modifiant, par la musique, les propriétés des agents qui le composent.

2.2. Boid

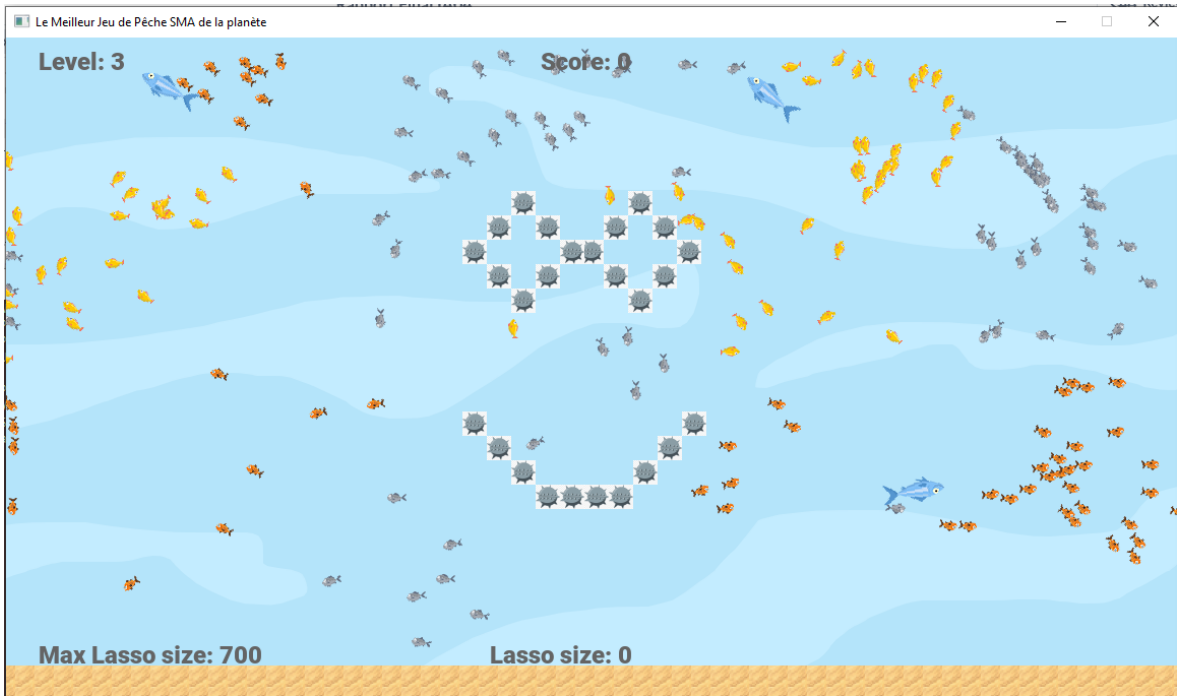
Une autre solution envisagée a été d'implémenter l'algorithme Boid. Il s'agit d'un algorithme qui a pour but de simuler le mouvement d'oiseaux ou de poissons dans un espace donné. Il se base sur le champ de vision des différents agents et sur un facteur de divergence.

3. Choix retenu

Suite à ces différents choix d'implémentation, nous avons préférer le fait d'utiliser l'algorithme Boid afin de créer un environnement dans lequels des agents se déplacent et réagissent à la musique.
Enfin, après avoir réussi l'implémentation que nous voulions, nous nou sommes rendu compte que notre simulation n'était qu'artistique et ne répondait à aucun objectif précis. L'idée a alors été d'utiliser cette base pour créer un jeu dans lequel les agents sont des poissons qu'il faut capturer. C'est ainsi qu'est né le projet tel que nous le connaissons aujourd'hui.

Deuxième partie

Fonctionnement du jeu



1. Principe général

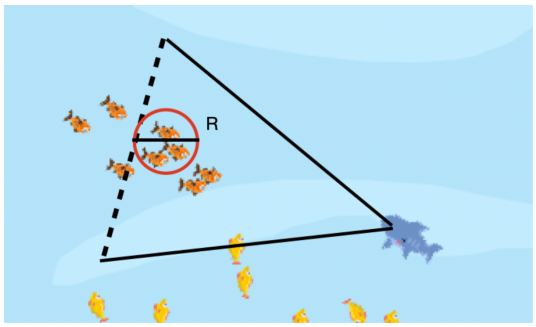
Le programme se présente comme un jeu dans lequel l'utilisateur doit capturer des poissons au moyen d'un lasso. Ces poissons sont regroupés entre différentes espèces qui possèdent des propriétés (et donc des mouvements) différentes. Le but est de capturer qu'une seule espèce. Ces poissons sont pertubés dans plusieurs aspects : par la musique, par les obstacles (murs et bombes) et par les prédateurs.

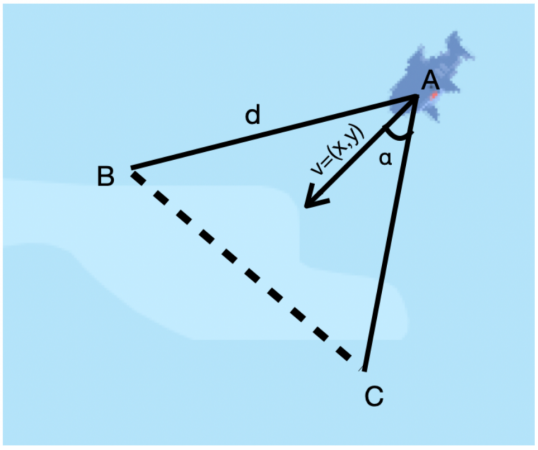
1.1. Les poissons et les prédateurs

Les poissons implémentent l'algorithme Boid que nous avons vu précédemment. Ils sont regroupés en plusieurs espèces distinctes qui possèdent des propriétés différentes. Ils peuvent être tués soit direcement par l'utilisateur par le biais du lasso, soit par les prédateurs qui existent dans l'environnement.

Les prédateurs quant à eux ont pour objectif de manger le plus de poisson possible. Ils ont pour stratégie de se déplacer lentement sur la carte et que lorsqu'une densité de

poisson dépasse un seuil dans son champ de vision, de foncer dans le banc de poisson.





1.2. La musique

Nous avons tout de même tenu au fait de garder un aspect musical dans notre projet en adoptant un agent musicale qui envoie des informations au monde en fonction de l'amplitude sonore actuelle de la musique. Cette partie du jeu a été codé en utilisant le package Beep et la librairie GoWave du package GoAudio.

1.3. Les niveaux

Le jeu est regroupé en 5 niveaux distincts. Ces niveaux différents font varier les propriétés qui composent le jeu : vitesse de déplacement, obstacles, taille du lasso...
A chaque niveau l'utilisateur se voit attribué un score qu'il se doit de maximiser. Le fait d'attrapper un poisson de la bonne espèce augmente le score, le fait d'en attrapper un de la mauvaise le fait diminuer.

Troisième partie

Choix d'implémentation

1. Choix de librairie

Pour ce projet, nous avons fait le choix d'utiliser la bibliothèque ebiten qui permet de faire de petits jeux vidéo en Go. Cette bibliothèque est tout de même limitée dans les possibilités qu'elle offre.

2. Structure du programme

Pour faire notre jeu, nous avons adopter structure en plusieurs modules distincts.

— agent : regroupe les différents agents présents dans le jeu. On compte parmis les agents : les poissons (Boid), les prédateurs et la musique.

— flock : regroupe le contexte des différents éléments qui composent le monde. Ces éléments sont les différents agents et les objets statiques.

— game : regroupe le contexte relatif au scoring et les fonctions graphiques du jeu. L'objet Game est l'élément central de notre programme qui délèque les calculent aux entités concernées. L'ensemble des fonctions d'affichages sont dans l'objet Graphics.

— utils : regroupe les fonctions usuelles ainsi que toutes les ressources du programmes (images et sons).

— worldelements : regroupe tous les éléments statiques existants dans le jeu. On y retrouve seulment l'objet Wall.