Projet IN104 : simulation d'un banc de poissons

Guillaume de Romèmont

April 2, 2024

1 Contexte

Pour de nombreuses espèces, notamment pour les oiseaux et les poissons, se mouvoir efficacement en large groupe est une question de survie. La dynamique du groupe doit pouvoir s'adapter à toutes les circonstances: : déplacement rapide et coordonné du troupeau, formation de vortex ou simple agrégation sans mouvement global. Ces mouvements collectifs sont dit émergent car le mouvement ne nécessite aucune forme de hiérarchie au sein du groupe mais proviens seulement de l'interaction locale entre chaque individus. L'enjeu est alors d'identifier la nature des interactions et les mécanismes liés aux changements de régimes dynamiques. [Cha]

2 Objectifs

Le projet se déroule en deux parties, la première partie consistera à réaliser la simulation d'un banc de poissons comme décrit dans [CKJ+02], ou équivalent au niveau 2 dans la vidéo de fouloscopie "5 niveaux de difficulté : Comment fonctionne un banc de poissons ?" (https://www.youtube.com/watch?v=Ch7VxxTBe1c&t=5s). La deuxième partie du projet sera totalement libre, à partir de la base de simulation précédemment construite, toute nouvelle idée sera la bienvenue.

La première partie devrait ressembler à cette simulation : https://www.complexity-explorables.org/explorables/flockn-roll/.

2.1 Détails sur la simulation d'un banc de poisson

L'article original sur la simulation d'un banc de poissons provient de Aoki en 1982 [Aok82]. Mais le modèle est mieux expliqué dans [CKJ+02]. Le but est de déterminer un nouvelle direction privilégiée $\boldsymbol{d}_i(t+\tau)$ en fonction de chacun de ses voisins.

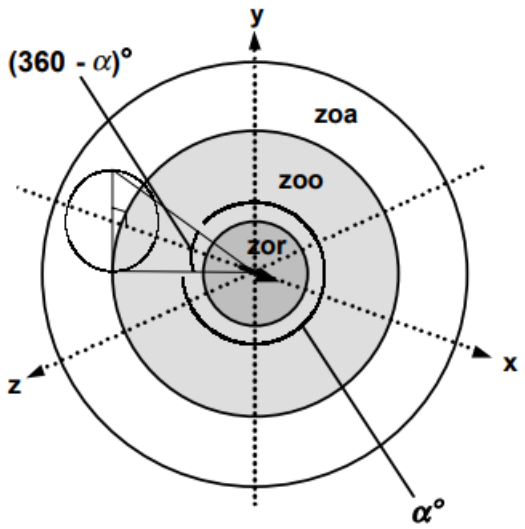

Figure 1: Représentation d'un individu dans le modèle centré sur l'origine : zor : zone de répulsion, zoo : zone d'orientation, zoa : zone d'attraction. L'éventuel 'angle mort' derrière un individu est également représenté. α : champ de perception.

Dans la zone de répulsion, on définit pour tout les poissons :

$$\boldsymbol{d}_r(t+\tau) = -\sum_{j\neq i}^{n_r} \frac{\boldsymbol{r}_{ij}(t)}{|\boldsymbol{r}_{ij}(t)|}$$

Avec $\boldsymbol{r}_{ij}(t) = (\boldsymbol{c}_j - \boldsymbol{c}_i)/|(\boldsymbol{c}_j - \boldsymbol{c}_i)|$

Dans la zone d'orientation, on définit pour tout les voisins tels quel $r_r \leq |(\boldsymbol{c}_j - \boldsymbol{c}_i)| \leq r_o$:

$$\boldsymbol{d}_o(t+\tau) = -\sum_{j=1}^{n_o} \frac{\boldsymbol{v}_j(t)}{|\boldsymbol{v}_j(t)|}$$

Dans la zone d'attraction, on définit pour tout les voisins tels quel $r_o \leq |(\boldsymbol{c}_j - \boldsymbol{c}_i)| \leq r_a$:

$$\boldsymbol{d}_a(t+\tau) = \sum_{j\neq i}^{n_r} \frac{\boldsymbol{r}_{ij}(t)}{|\boldsymbol{r}_{ij}(t)|}$$

Ici, $\boldsymbol{d}_r$,$\boldsymbol{d}_o$ ou $\boldsymbol{d}_a$ définissent des directions privilégiées que va adopter le poisson central en fonction de ses voisins. La zone de répulsion peux s'interpréter comme une zone ou le poisson maintient un espace personnel, la zone d'orientation va inciter le poisson à s'aligner avec ses voisins tandis que la zone d'attraction tend les organismes à se joindre en groupes.

Si des poissons se trouvent dans la zone de répulsion, alors $\boldsymbol{d}_i(t+\tau) = \boldsymbol{d}_r(t+\tau)$.

Sinon, si les poissons voisins se trouvent uniquement dans la zone d'orientation $\boldsymbol{d}_i(t+\tau) = \boldsymbol{d}_o(t+\tau)$ et de la même façon si les voisins se trouvent uniquement dans la zone d'attraction $\boldsymbol{d}_i(t+\tau) = \boldsymbol{d}_a(t+\tau)$. Si des poissons voisins se trouvent dans les deux zones précédemment citées : $\boldsymbol{d}_i(t+\tau) = \frac{1}{2}[\boldsymbol{d}_o(t+\tau) + \boldsymbol{d}_a(t+\tau)]$.

Dans l'éventualité ou les forces résultantes sont égales à 0 ou que il n'y a pas de voisins dans toutes ces zones, $\boldsymbol{d}_i(t+\tau) = \boldsymbol{v}_i(t)$.

La prise de décision individuelle de chaque poisson est modélisée en modifiant $\boldsymbol{d}_i(t+\tau)$ en ajoutant une distribution Gaussienne.

Après obtention de la direction privilégiée $\boldsymbol{d}_i(t+\tau)$, l'individu tourne en direction de $\boldsymbol{d}_i(t+\tau)$ avec une vitesse de rotation θ (la rotation maximale est fixée avec $\theta\tau$). Pour simplifier l'analyse, on suppose que les individus se déplacent à une vitesse vitesse constante de s unités par seconde.

Les ordres de grandeurs sont donnés dans [CKJ+02].

2.2 Ajout de fonctionnalités

La deuxième partie est donc complètement libre, et ne dépend que de ce que vous voulez faire. Toute initiative sera valorisée.

Voici quelques exemples en vrac pour vous donner des idées :

- Complexifier le modèle pour le rendre plus réaliste, avec les papiers scientifiques à l'appui. Voir la vidéo de science étonnante la dessus.
- Passer à un modèle 3D pour la simulation d'une nuée d'oiseaux.
- Améliorer la rapidité d'éxécution de votre code (MPI, multithreading...) pour utiliser au mieux les ressources de votre ordinateur et créer des bancs de milliers de poissons.
- Paralléliser votre code et le porter sur GPU
- Améliorer les algorithmes de recherche de voisins de votre code (Octree...)
- Ajout d'un prédateur dans la simulation [OKMI03]
- Ajout d'une interface graphique en temps réel pour mieux contrôler la simulation
- ...

3 Critères d'évaluations

La bonne réalisation de la première partie assurera une note minimale a votre projet et la note maximale ne sera accessible que avec une bonne deuxième partie.

Le code doit être entièrement codé en C. N'oubliez pas les bonnes pratiques de codes. La lisibilité et la beauté du code seront prises en compte pour la notation :

- Indentez votre code
- Faites attention à vos noms de variables, ils doivent être explicites
- N'hésitez pas à découper le code en sous fonction voir à faire plusieurs fichiers
- N'hésitez pas à ajouter des commentaires dans votre code
- Bien utiliser la notion de structures

Les fonctions utiles seront dans un dossier include contenant les fichier *.c et *.h. Un Makefile compilera l'ensemble du projet.

Utiliser Git garantit une synchronisation du code entre les membres de l'équipe, un historique des versions et l'impossibilité de perdre une ligne de code ! Une bonne pratique est de créer au moins deux branches avec en permanence sur la branche principale appelée - origin - un code fonctionnel (qui compile et s'exécute) et sur la branche - develop - les nouvelle fonctionnalités en cours de développement.

4 Quelques pistes et aides

Les papiers cités sont disponibles sur internet (sinon avec scihub...).

Créer une structure poisson avec une fenêtre graphique pour que le rendu se fasse en temps réel. Commencez simple! D'abord une fenêtre graphique, ensuite un point, ensuite ajouter petit à patit les fonctionnalités en les testant bien une à une.

J'ai personnellement utilisé SDL2 pour créer une fenêtre graphique et afficher des images.Bien que non codé en C, la librairie Cimgui vous permet d'avoir un wrapper pour créer une interface graphique en C.

4.1 Installation de WSL

Cela permet d'installer une machine virtuelle linux sur votre PC en local. Ce système vous permet d'avoir accès à toutes les commandes et la facilité d'utilisation de linux sur un PC windows sans faire de dual-boot. Si vous installer vscode, vous pourrez également lancer vscode en local. Vous aurez une meilleure gestion des librairies à utiliser en C.

Pour installer WSL, suivre ce tutoriel :https://learn.microsoft.com/fr-fr/windows/wsl/install

4.2 La bibliothèque SDL2

Cette bibliothèque vous permet de créer d'afficher des éléments graphiques en C. Voici un tutoriel et quelques commandes utiles. Elle est normalement déjà téléchargée sur le serveur distant mais vous pouvez téléchargez cette librairie "sudo apt-get install libsdl2-dev" en local grâce à cette Linux.

Makefile :

```
CC = gcc
CFLAGS = -Wall -Wextra -std=c99 -g
LDFLAGS = -lSDL2 -lm -lSDL2_image
SRC = SDL_example.c
EXECUTABLE = output

all: $(EXECUTABLE)

$(EXECUTABLE): $(SRC)
  $(CC) $(CFLAGS) -o $@ $^ $(LDFLAGS)

run: $(EXECUTABLE)
  ./$(EXECUTABLE)

debug: $(EXECUTABLE)
  gdb ./$(EXECUTABLE)

clean:
  rm -f $(EXECUTABLE)
```

Code C de création d'un fenêtre :

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <stdio.h>
#include <stdlib.h>

#define IMAGE_WIDTH 1200
#define IMAGE_HEIGHT 800

// Window
void render(SDL_Renderer *renderer, SDL_Texture **texture) {
    SDL_SetRenderDrawColor(renderer, 255, 255, 255, 255);
    SDL_RenderClear(renderer);

    SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255);
  SDL_Rect rect = { 400, 400, 10, 10 };
  SDL_RenderFillRect(renderer, &rect);
  SDL_RenderCopy(renderer, *texture, NULL, &rect);
  // SDL_RenderCopyEx(renderer, *texture, NULL, &destRect, angle, NULL, SDL_FLIP_NONE)
    ;


    SDL_RenderPresent(renderer);
}

// void loadTexture(SDL_Renderer *renderer, SDL_Texture **texture) {
//    SDL_Surface *surface = IMG_Load( "/mnt/c/Users/guill/workspace/IN104/fish_bitmap
    .png");
//     if (surface == NULL) {
//         fprintf(stderr, "Failed to load image: %s\n", IMG_GetError());
//         SDL_Quit();
//         exit(1);
//     }
//     *texture = SDL_CreateTextureFromSurface(renderer, surface);
```

```
//     SDL_FreeSurface(surface);

//     if (*texture == NULL) {
//         fprintf(stderr, "Failed to create texture: %s\n", SDL_GetError());
//         SDL_Quit();
//         exit(1);
//     }
// }

int main() {
    if (SDL_Init(SDL_INIT_VIDEO) < 0) {
        fprintf(stderr, "SDL initialization failed: %s\n", SDL_GetError());
        return 1;
    }

    SDL_Window *window = SDL_CreateWindow("N-Body Simulation", SDL_WINDOWPOS_UNDEFINED
    , SDL_WINDOWPOS_UNDEFINED, IMAGE_WIDTH, IMAGE_HEIGHT, SDL_WINDOW_SHOWN);
    if (window == NULL) {
        fprintf(stderr, "Window creation failed: %s\n", SDL_GetError());
        SDL_Quit();
        return 1;
    }

    SDL_Renderer *renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
    if (renderer == NULL) {
        fprintf(stderr, "Renderer creation failed: %s\n", SDL_GetError());
        SDL_DestroyWindow(window);
        SDL_Quit();
        return 1;
    }
    SDL_Texture *texture;
    // loadTexture(renderer, &texture); // Replace "particle.png" with your image file


    SDL_Event event;
    int quit = 0;
    while (!quit) {
        while (SDL_PollEvent(&event) != 0) {
            if (event.type == SDL_QUIT) {
                quit = 1;
            }
        }



        // Render the updated positions
        render(renderer, &texture);

        // Delay to control the frame rate
        // SDL_Delay(1);
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}
```

References

[Aok82] Ichiro Aoki. A simulation study on the schooling mechanism in fish. *Bulletin of the Japanese Society of Scientific Fisheries*, 48(8):1081–1088, 1982.

[Cha] Hugo Chaté. Comment un banc de poissons passe-t-il d'un comportement collectif à l'autre ?

[CKJ+02] Iain D Couzin, Jens Krause, Richard James, Graeme D Ruxton, and Nigel R Franks. Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, 218(1):1–11, 2002.

[OKMI03] Tamon Oboshi, Shohei Kato, Atsuko Mutoh, and Hidenori Itoh. A simulation study on the form of fish schooling for escape from predator. *FORMA-TOKYO-*, 18(2):119–131, 2003.