Rapport de projet

January 2023

RG2A

Rapport de projet

RG2A

Redjeb Gabin

Gallardo Alexandre

Saadi Akram

Contents

1 Présentation du Groupe

1.1 Redjeb Gabin

Je suis Gabin Redjeb chef de projet auto-proclamé et adoubé par tous. J'aime faire adhérer une équipe autour d'un projet et manager la motivation de chacun au cours du temps afin que chacun puisse donner le meilleur de lui même au service d'un projet commun RG2A.

1.2 Gallardo Alexandre

Je m'appelle Alexandre Gallardo et pour ma part j'entame mon premier s4, c'est donc pour moi une nouvelle étape a franchir avec de nouvelles difficultées avec toujours comme aboutissement l'amélioration de mes compétences notament pour des projets en groupe comme celui ci.

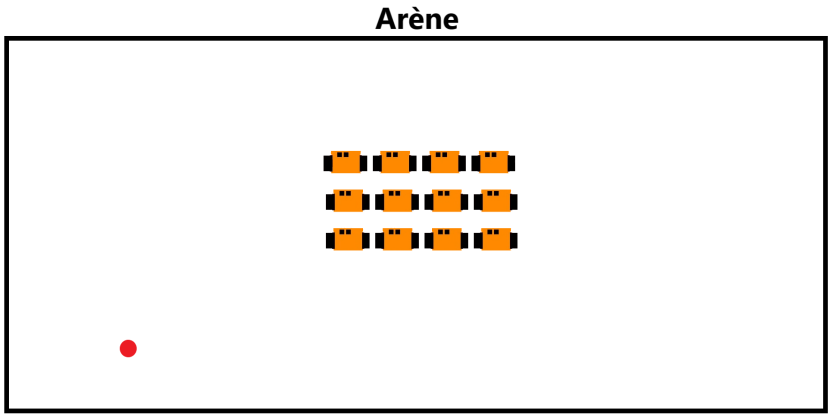
1.3 Saadi Akram

Quand j'ai rejoint epita, l'infographie était l'un de mes principaux intérêts en informatique, j'ai toujours voulu construire quelque chose à partir de rien, réinventer la roue, transformer le traitement informatique en images virtuelles pixel par pixel. J'étais aussi très intéressé par la modélisation et les simulations, c'est pourquoi j'ai choisi de faire ce projet car c'est un bon début pour mon parcours en infographie et modélisation. Je suis très investi dans ce projet et je suis sûr que je vais apprendre beaucoup.

2 Présentation du Projet

2.1 Description

Notre projet consiste à créer une simulation d'exploration où :
Des robots doivent être capables de trouver le plus rapidement possible, même dans les pires conditions, une cible positionnée dans une arène. Les robots sont tous autonomes, il n'y a pas de hiérarchie et ne peuvent communiquer entre eux qu'à partir d'une certaine distance (pour pouvoir prévenir les autres robots de la position de la cible). La cible est placée dans l'arène de manière aléatoire. L'utilisateur pourra ajouter des obstacles dans l'arène permettant de complexifier la recherche (`arène modulable`). Pour mener à bien la recherche de la cible nous simulons de la vie artificielle, en utilisant le principe de robotique d'essaim (les comportements des essaims). Pour reproduire ces comportements, nous allons utiliser l'algorithme de Boids.



2.2 Inspirations

L'algorithme de Boids est utilisé dans divers secteurs:

- l'infographie, fournissant des représentations réalistes de volées d'oiseaux et d'autres créatures, telles que des bancs de poissons ou des troupeaux d'animaux.:

- la cinématographie: permettant d'automatiser l'animation de créatures dans un décor comme des oiseaux, un banc de poissons ou une foule (Seigneurs des Anneaux)

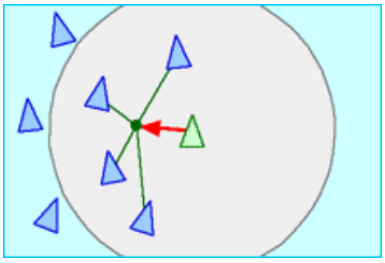
- le contrôle et la stabilisation de véhicules sans pilote: permettant d'explorer des lieux inconnus qu'ils soient terrestres, aériens ou sous-marins.

2.3 Intérêt algorithmique

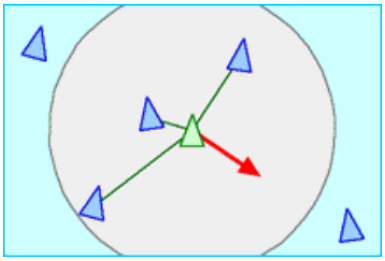
Boids est un programme de vie artificielle, qui simule le comportement de vol des oiseaux et qui permet de modéliser un comportement émergent, c'est-à-dire une complexité comportementale qui résulte de l'interaction d'agents individuels respectant un nombre limité de règles simples.

Règles:

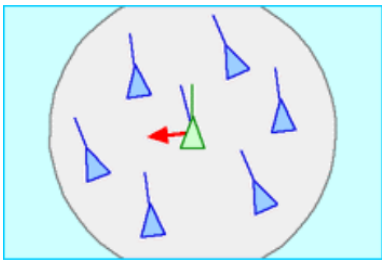
Cohésion: pour rester groupés, les robots essaient de suivre un même chemin.



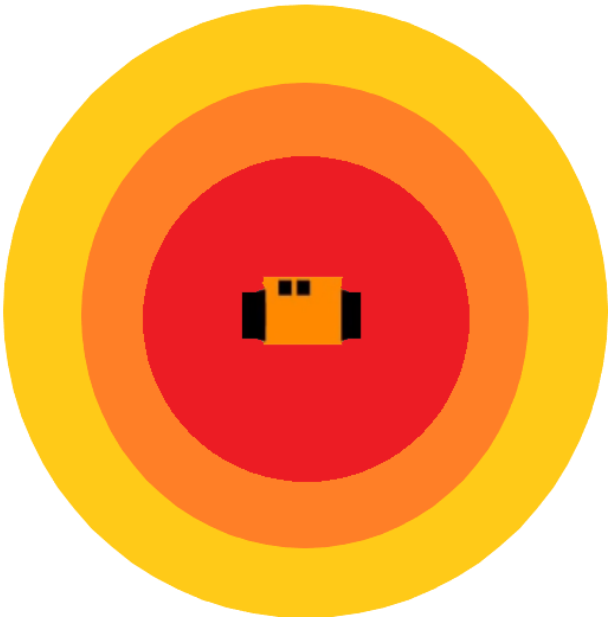
Séparation: deux robots ne peuvent pas se trouver au même endroit au même moment donc pour éviter de s'entasser chaque robot doit s'éviter.



Alignement: pour rester groupés les robots essaient de suivre le même chemin. 4



Dans les faits, cela se traduit par la réaction du robot à son voisinage. Ce voisinage se divise en trois zones, de la plus proche à la plus éloignée:



une **zone de répulsion**, une **zone d'orientation** et une **zone d'attraction**. Lorsqu'un voisin entre dans sa zone de répulsion, le robot s'éloigne (séparation). Si le voisin est dans sa zone d'orientation, le boids* le suit (alignement). Et enfin, s'il est dans la zone d'attraction, le robot s'en rapproche (orientation).

boids : vient du mot bird-oid, humanoid pour un oiseau

3 Répartition des charges

3.1 Prévu

Tâches	**Gabin**	**Alexandre**	**Akram**
Implémentation algo Boids :	-	-	X
Implémentation de la recherche de la cible :	X	X	X
Algorithme de colonies de fourmis	-	-	X
Algorithme path finding	X	X	-
Représentation graphique :	X	X	X
Représentation des boids	-	-	X
Représentation des vecteurs	-	-	X
Représentation des obstacles	X	-	-
Représentation de plusieurs équipes	X	X	-
Interface Graphique :	X	X	-
Paramétrage algorithme de boids	X	-	-
Paramétrage obstacles	X	-	-
Paramétrage path finding	-	X	-
Paramétrage équipes	X	-	-
Site Web :	X	-	-

Table 1: Répartition des tâches

3.2 Réalisé

Tâches	**Gabin**	**Alexandre**	**Akram**
Implémentation algo Boids :	-	-	X
Implémentation de la recherche de la cible :	X	X	-
Algorithme de colonies de fourmis	X	-	-
Algorithme path finding	-	X	-
Représentation graphique :	X	X	X
Représentation des boids	-	-	X
Représentation des vecteurs	-	-	X
Représentation des obstacles	X	-	-
Représentation de plusieurs équipes	X	X	-
Interface Graphique :	X	X	-
Paramétrage algorithme de boids	X	-	-
Paramétrage obstacles	X	-	-
Paramétrage path finding	-	X	-
Paramétrage équipes	X	-	-
Site Web :	X	-	-

Table 2: Répartition des tâches

X : ajout d'une tâche

- : désistement d'une tâche

4 Planning

Tâches	**Soutenance 1**	**Soutenance 2**	**Soutenance 3**
Avancement	réalisé	réalisé	prévu - réalisé
Implémentation algo Boids	90%	100%	100%
Implémentation de la recherche	10%	40%	100% - 100%
Algorithme de colonies de fourmis	-	10%	100% - 100%
Algorithme path finding	20%	70%	100% - 100%
Représentation graphique	70%	75%	100% - 75%
Représentation des boids	100%	100%	100%
Représentation des vecteurs	100%	100%	100%
Représentation des obstacles	80%	100%	100%
Représentation de plusieurs équipes	-	-	100% - 0%
Interface Graphique	100%	100%	100%
Coefficients de Boids	100%	100%	100%
Obstacles	100%	100%	100%
Path finding	100%	100%	100%
Paramétrages équipes	100%	100%	100%
Site Web	25%	60%	100% - 100%

Table 3: Répartition des tâches

5 Précision des tâches

5.1 Implémentation algo Boids

On a déjà parlé de cet algorithme plus haut dans ce document, nous voulons l'utiliser afin d'établir les règles de déplacement des "robots".

5.2 Implémentation de la recherche de la cible

Pour implementer la recherche de la cible on va utiliser des algorithmes de recherche de chemin, cela consiste a trouver un chemin entre un point A a un point B en prenant en compte differentes contraintes. Il en existe plusieurs mais nous allons utiliser ces deux principaux : l'algorithne de Dijkstra qui sert a chercher le plus court chemin lorsque l'on connait tout les chemins possible. l'algorithme A* qui est très utile lorsque on ne connait pas les chemins possibles. l'implementation de ces deux algorithmes vont permettre la simulation de condition environnementaux connu ou non.

5.3 Représentation graphique

Représentation des boids :
Afin de rendre l'experience utilisateur plus ergonomique, nous implémenterons une façon de visualiser de les "boids"

Représentation des vecteurs :
Même principe que pour les "boids", il nous faut un moyen de représenter les directions de chacun des éléments

Représentation des obstacles :
Pour ajouter un peu de difficultée à la tache de nos petits robots, nous allons ajouter des obstacles empechant leurs progression. Il faudra donc bien évidemment trouver un moyen d'implementer ceux-ci et de les representer sur l'interface pour rendre la simulation plus esthetique.

Représentation de plusieurs équipes :
Afin de tester plusieurs l'algorithmes, nous mettrons plusieurs équipes en concurrence. Cela nous permettra de tirer des conclusions sur l'efficacité des différents algorithmes en fonction des configurations de l'arène (obstacles).

5.4 Interface Graphique

Paramétrage algorithme de Boids :
L'algorithme de Boids se base sur 3 règles : la cohésion, la séparation et l'alignement. Notre interface graphique permettra à l'utilisateur de varier ses paramètres pour pouvoir voir l'impact sur la simulation et donc sur le comportement des boids.

Paramétrage obstacles :
En accessionant cette fonctionnalité, l'utilisateur pourra dessiner des obstacles dans l'arène à l'aide de son curseur.

Paramétrage pathfinding :
Cette fonctionnalité permettra à l'utilisateur d'ordonner aux boids de trouver le chemin jusqu'à la position pointée par le curseur de sa souris.

Paramétrage équipes :
Cette fonctionnalité permettra de choisir le nombre d'équipes et les paramétres à associer à chaque équipe.

5.5 Site Web

Nous allons implémenter un site web afin de pouvoir y consigner nos avancées, partager le projet avec d'autres gens et donner de la visibilité a notre projet de manière plus générale.

6 Outils de travail

Nous utilserons la bibliotèque SDL2 pour représenter la simulation et la bibliotèques GTK pour l'interface graphique.

7 Soutenance 1

7.1 Implémentation de l'algorithme de Boids

Comment les boids sont-ils représentés dans notre implémentation?

Il n'existe pas de meilleure approche pour représenter un boid, mais dans notre cas, nous avons utilisé des triangles isocèles. Cela a été fait non seulement pour des raisons esthétiques mais aussi pour rendre l'algorithme plus simple à mettre en œuvre et à comprendre.

Pourquoi un triangle ?

Nous pouvons mettre en œuvre le comportement souhaité en déterminant si un autre point se trouve sur le côté gauche ou droit de notre boid en utilisant les deux points qui constituent la base du triangle. L'autre point de notre triangle définira la largeur du champ de vision du boid, et plus le champ de vision est large, plus la hauteur de notre boid doit être grande.

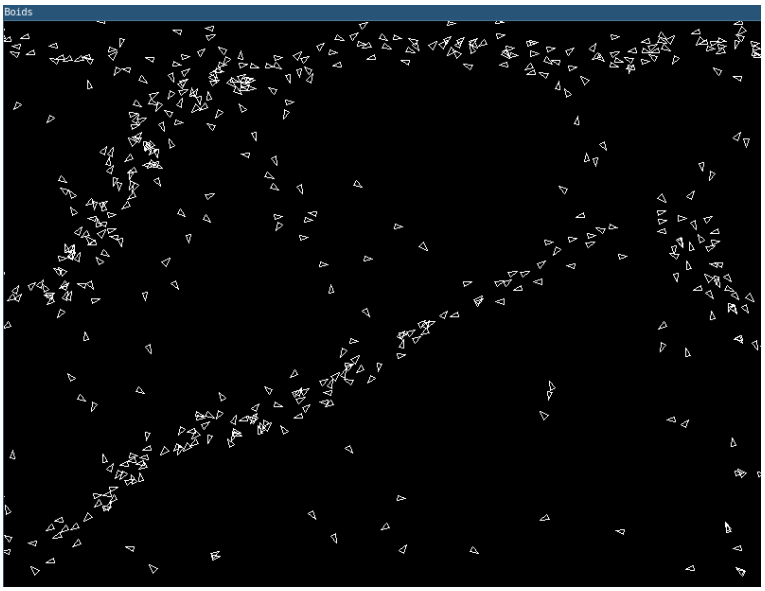
Quels sont les principaux comportements des boids?

Il y a trois règles :

L'alignement: pour rester regroupés les boids essaient de se suivre sur le même chemin.

La séparation: les boids évitent autant que possible de se coller l'un à l'autre.

La cohésion: les boids ont tendance à se regrouper entre eux.

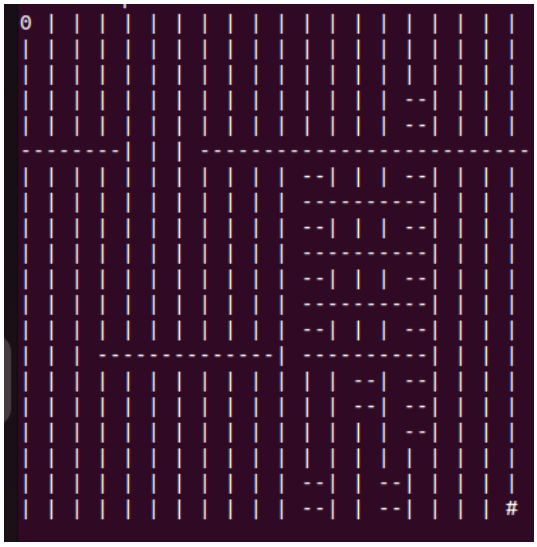


7.2 Implémentation Path finding

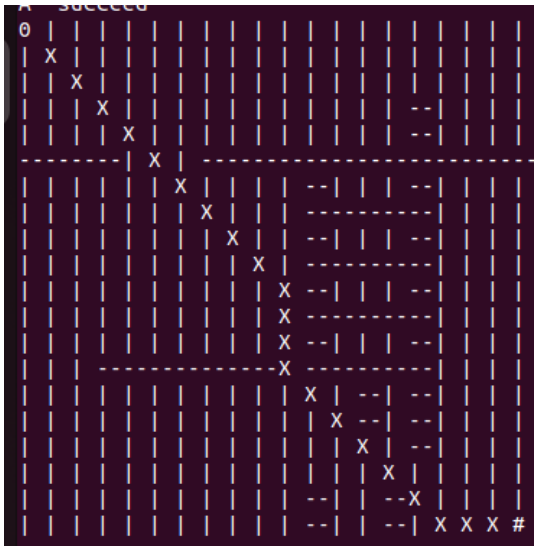
Pour l'implementation de l'algorithme de pathfinding nous avons d'abord choisis d'implementer celui de A* qui correspond au mieux a nos demande pour un algorithme de pathfinding le plus general possible. En effet celui-ci recherche le chemins le plus cours entre un point A et un point B en considerant des obstacles. Pour ce faire a cette premiere soutenance l'implementation de cette algortihme et faite sur vecteur de int avec comme valeur possible 0 pour un emplacement libre et 1 pour un emplacement ocuppee. L'algorithme ce base sur une structure fifo avec ordre de prioritee basée sur une valeur heuristic. La valeur heuristice se calcule pour chaque noeud u par heuristic = cout + distance(u, destination), le cout correspond au deplacement total depuis le point de depart.

```
struct pqueue{
    struct pqueue* next;
    int x;
    int y;
    int heur;
    int cout;
};
```

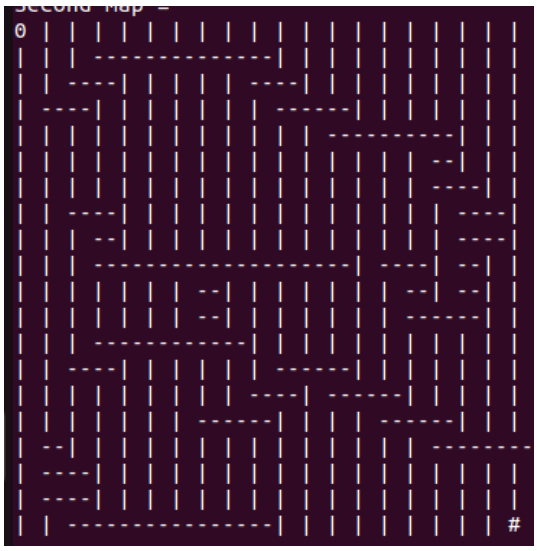
Ici la structure de donnée qui contiendra chaque pixel qui sera atteint par l'algorithme. Ci-dessous une map avec un point de depart (0) et un point d'arrivé (#) ainsi que les obstacles (–) et les postitions possible (—) :



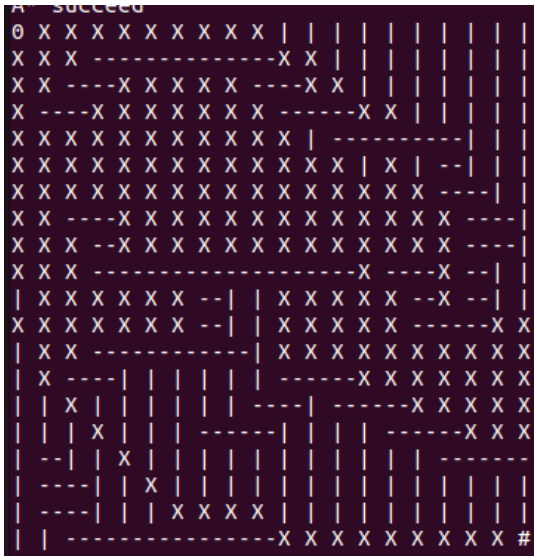
La meme map qui a été resolue par l'algorithme :



Maintenant le probleme de l'algorithme etant qu'il ne retrouve pas tout seul le chemin et sur de map plus complexe comme celle-ci :

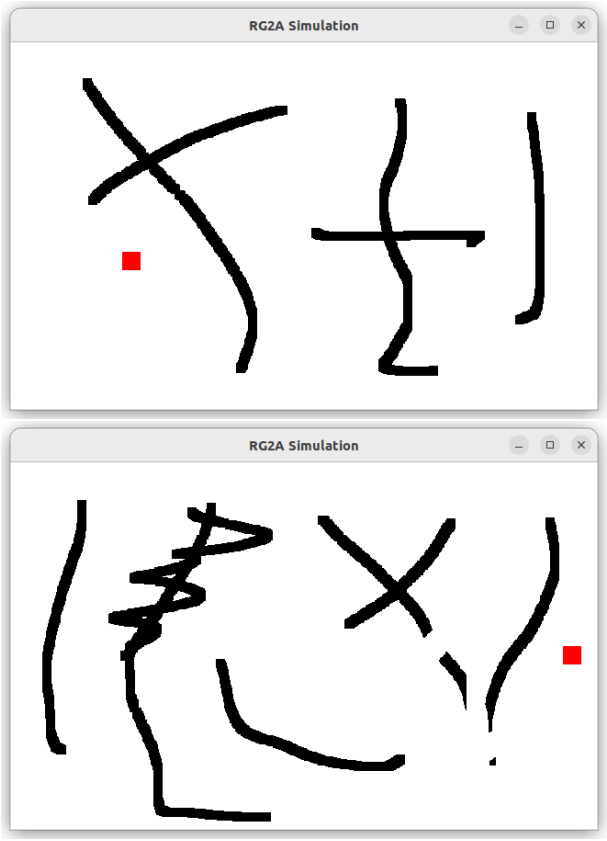


Il va renvoyée le chemin mais aussi les cases qui ne sont pas supposée etre pris en compte :



7.3 Représentation graphique

Représentation des obstacles - Paint :
Paint est la première version de l'implémentation des obstacles. Notre Paint fonctionne sur le même principe que le vrai Paint, dessiner sur un plan. A la différence que notre version dessine des rectangles noir en guise d'obstacles sauf sur la cible rouge positionnée de façon aléatoire lorsque l'on presse une touche du clavier.
Le Paint s'activera lorsque le checkbutton obstacle(de l'interface graphique) est activé. Lorsque le click gauche est pressé, et que le curseur de la souris n'est pas en collision avec la cible rouge, un rectangle noir de dimension 10x10 est dessiner sur l'arène (en guise d'obstacle). Le click droit à la même fonctionnalité que le click gauche mais celui-ci efface seulement les obstacles.



7.4 Interface graphique

L'interface graphique est composé de 3 parties :

- Les paramètres généraux
- Les paramètres de l'équipe 1
- Les paramètres de l'équipe 2

Le paramétrage général permet de paramétrer la simulation de manière générale, ces modifications s'appliquent sur toutes les équipes.
Il est composé de 2 boutons et de 2 checkbuttons:

- Le bouton Start/Pause : permet de lancer la simulation ou de la mettre en pause.
- Le bouton Stop : permet d'arrêter la simulation.
- Le checkbutton Obstacle : permet à l'utilisateur de dessiner des

obstacles dans l'arène. Lorsque ce checkbutton est actionné, ceux du pathfinding de l'équipe 1 et 2 sont désactivés, pour que le curseur de la souris ne soit pas utilisé pour plusieurs actions.

- Le checkbutton Vector : permet d'afficher les vecteurs des robots.

Le paramétrage des équipes permet d'appliquer les différentes modifications (coefficient boids, différents algorithmes de recherche, pathfinding) sur chacune des équipes. Il est composé de 3 silder et de 3 checkbutton et d'un bouton.

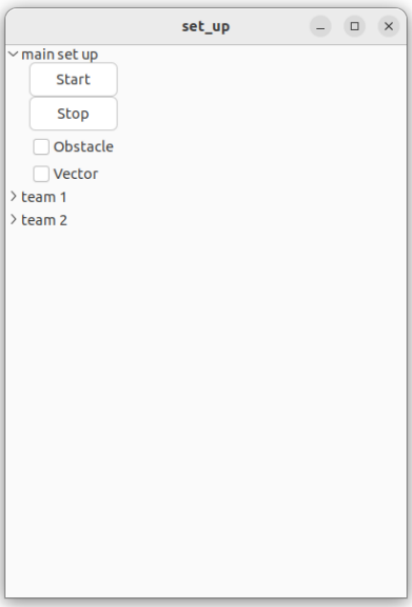
- Le cohésion/séparation/alignement slider : permettent de modifier les différents coefficients de l'algorithme de boids pour chaque équipe.

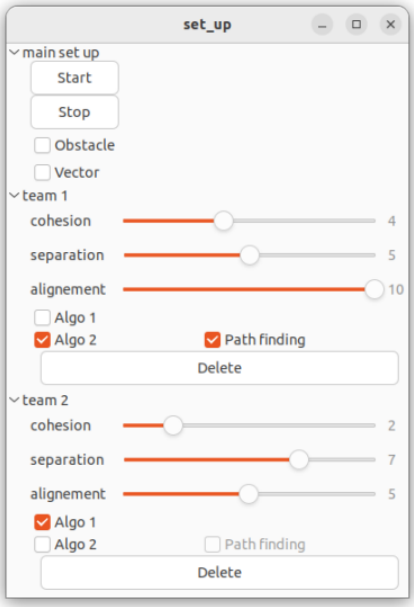
- Le checkbutton algo1 : active l'algorithme numéro 1 celui qui se base sur l'algorithme de colonies de fourmis et désactive l'algo2 s'il a été activé.

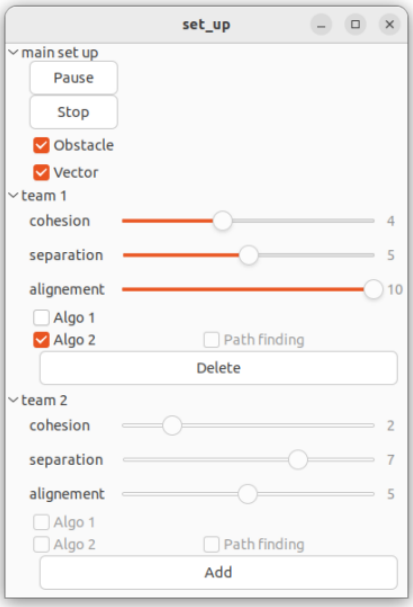
- Le checkbutton algo2 : active l'algorithme numéro 2 qui s'appuie sur la recherche de chemin (pathfinding) et désactive l'algo1 s'il a été activé.

- Le checkbutton pathfinding : peut être activé que si le checkbutton obstacle et algo1 sont désactivés. Lorsqu'il est activé les robots associés au pathfinding de l'équipe de dirige vers le curseur de la souris.

- Le bouton Delete/Add : supprime/ajoute les robots associés à l'équipe.


set_up
main set up
Start
Stop
Obstacle
Vector
team 1
team 2

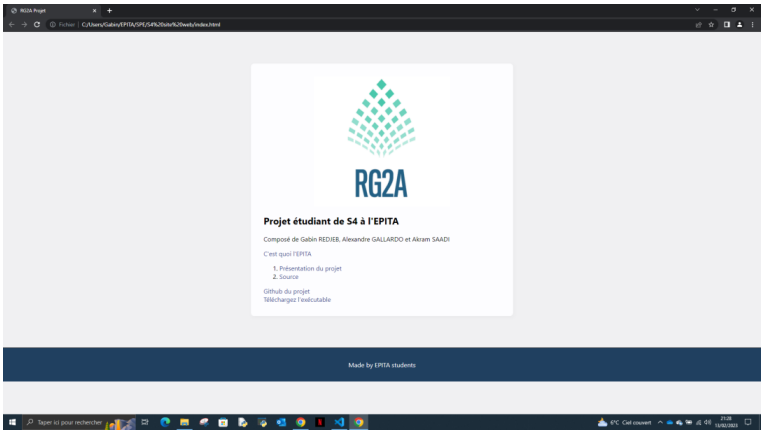

set_up
main set up
Start
Stop
Obstacle
Vector
team 1
cohesion
4
separation
5
alignement
10
Algo 1
Algo 2
Path finding
Delete
team 2
cohesion
2
separation
7
alignement
5
Algo 1
Algo 2
Path finding
Delete


set_up
main set up
Pause
Stop
Obstacle
Vector
team 1
cohesion
4
separation
5
alignement
10
Algo 1
Algo 2
Path finding
Delete
team 2
cohesion
2
separation
7
alignement
5
Algo 1
Algo 2
Path finding
Add

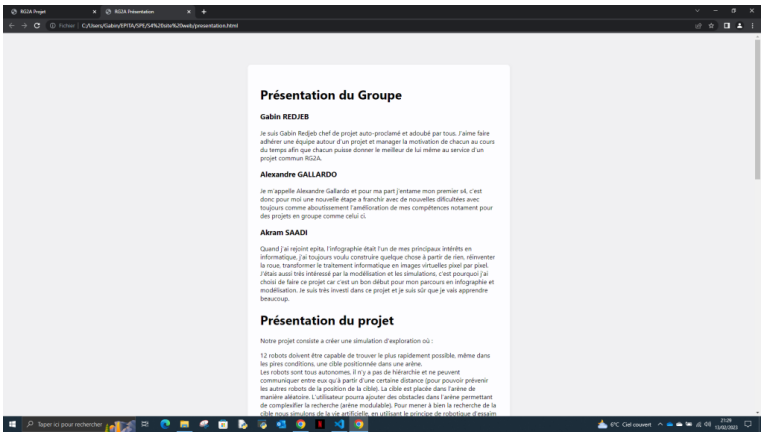
7.5 Site Web

Le site web est composé d'une page d'accueil, d'une page de présentation du groupe et du projet et d'une page amenant aux sources utilisées pour la réalisation du projet.
La page d'accueil comporte 5 liens qui amène à :

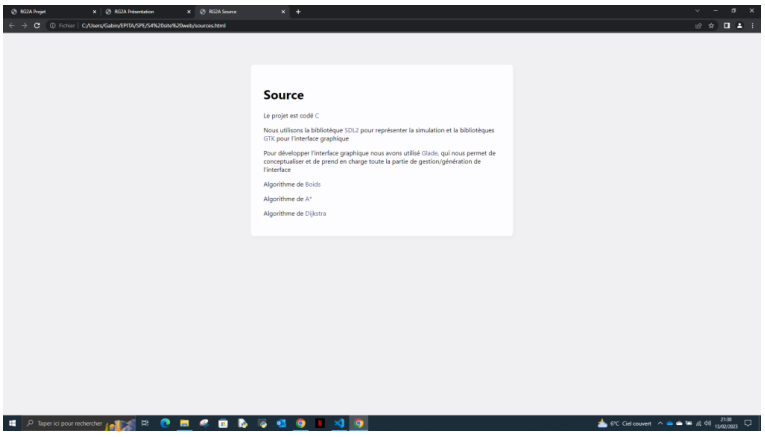
- La page de l'EPITA
- La page de présentation du groupe et du projet
- La page source
- La page de projet GitHub
- L'installation de l'exécutable du projet



La page de présentation résume l'idée du projet et les objectifs personnels de chaque membre du projet.



La page source explicite les outils utilisés pour le projet, qui sont le langage C, SDL2 pour modéliser l'arène et les robots, GTK/Glade pour conceptualiser l'interface graphique qui va permettre de rendre la simulation modifiables et les algorithmes que nous utilisons tels que Boids, A*, Dijkstra et colonie de fourmis.



8 Soutenance 2

8.1 Implémentation de l'algorithme de colonies de fourmis

L'algorithme Boid imite le comportement collectif d'animaux tels que les oiseaux en vol, les poissons en bancs et les essaims d'insectes. Il utilise des règles simples basées sur le mouvement des objets environnants, ainsi que sur le comportement local, pour générer le mouvement de chaque objet. L'algorithme de la colonie de fourmis simule le comportement de fourmis qui explorent un territoire et utilisent des phéromones pour communiquer. Il est basé sur l'idée que les fourmis recherchent continuellement de la nourriture et marquent leur chemin avec des phéromones. Au fur et à mesure que les fourmis explorent, elles accumulent les pistes de phéromones, ce qui leur permet de trouver un chemin plus efficace vers la source de nourriture.

J'ai essayé de combiner l'algorithme de boids et l'algorithme de colonies de fourmis en un seul algorithme unique et bien défini, en utilisant les mêmes paramètres et les mêmes lignes directrices. L'objectif était de développer des règles et des variables simples qui, lorsqu'elles sont modifiées d'une manière spécifique, transformeraient un boid en fourmi et vice versa.

Après plusieurs tentatives pour combiner l'algorithme de Boid et l'algorithme de la colonie de fourmis, je n'ai pas été en mesure de trouver une solution optimale. Le comportement des composants individuels et leurs interactions étaient difficiles à contrôler et compliqués à comprendre. J'ai également rencontré des difficultés à mettre en place un environnement permettant aux deux algorithmes de coexister, d'interagir et de bénéficier l'un de l'autre. En raison de ces limitations, de la complexité de la tâche et de l'utilisation difficile de la bibliothèque SDL, il s'est avéré trop difficile pour moi de réaliser des progrès significatifs, c'est pourquoi je m'excuse de reporter cette

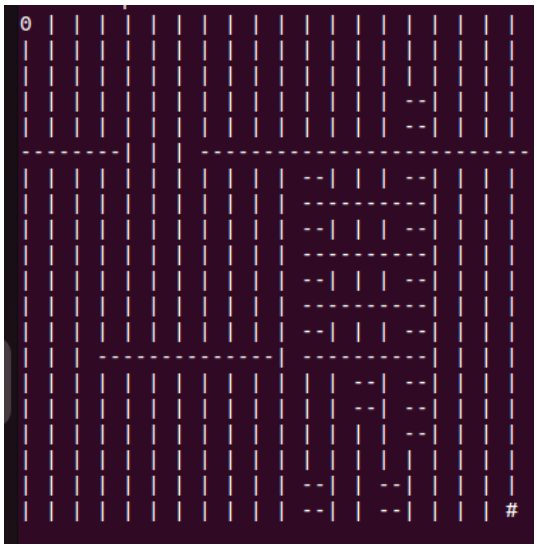
partie à la soutenance prochaine.

8.2 Implémentation du Path Finding

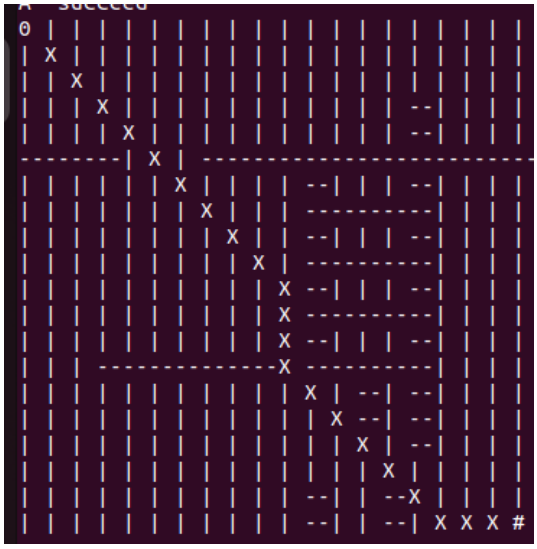
Pour l'implementation de l'algorithme de pathfinding nous avons d'abord choisis d'implementer celui de A* qui correspond au mieux a nos demande pour un algorithme de pathfinding le plus general possible. En effet celui-ci recherche le chemins le plus cours entre un point A et un point B en considerant des obstacles. Pour ce faire a cette premiere soutenance l'implementation de cette algortihme et faite sur vecteur de int avec comme valeur possible 0 pour un emplacement libre et 1 pour un emplacement ocuppee. L'algorithme ce base sur une structure fifo avec ordre de prioritee basée sur une valeur heuristic. La valeur heuristice se calcule pour chaque noeud u par heuristic = cout + distance(u, destination), le cout correspond au deplacement total depuis le point de depart.

```
struct pqueue{
    struct pqueue* next;
    int x;
    int y;
    int heur;
    int cout;
};
```

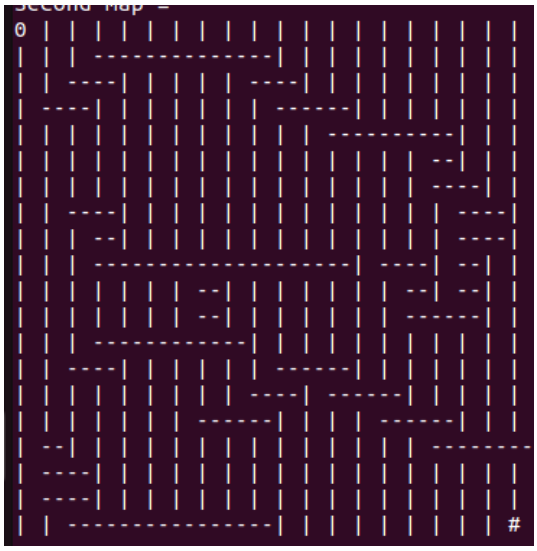
Ici la structure de donnée qui contiendra chaque pixel qui sera atteint par l'algorithme. Ci-dessous une map avec un point de depart (0) et un point d'arrivé (#) ainsi que les obstacles (–) et les postitions possible (—) :



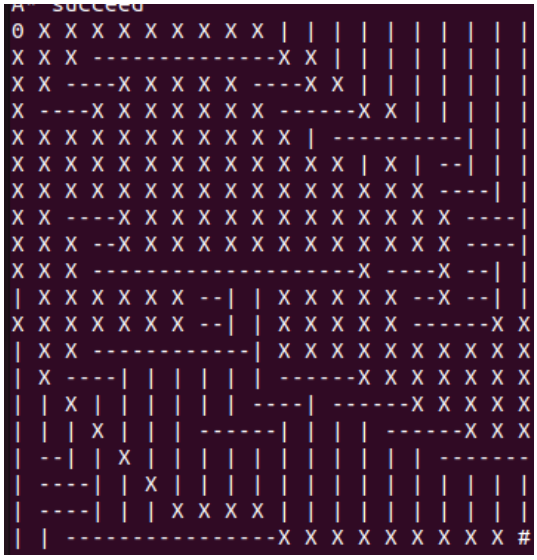
La meme map qui a été resolue par l'algorithme :



lors de la soutenance une le probleme de l'algorithme etais qu'il ne retrouve pas tout seul le chemin et sur de map plus complexe comme celle-ci :

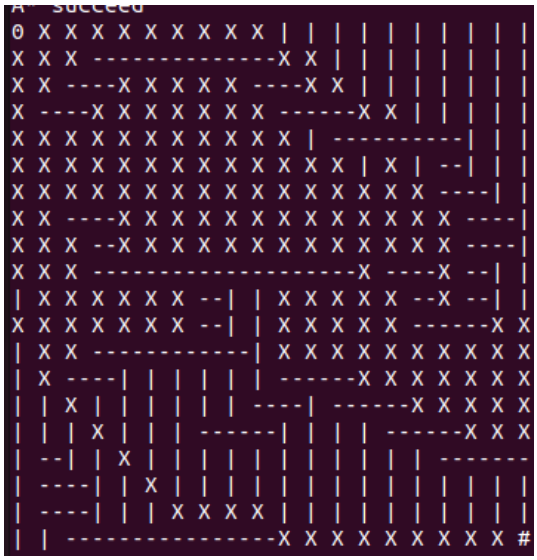


Il va renvoyée le chemin mais aussi les cases qui ne sont pas supposée etre pris en compte :

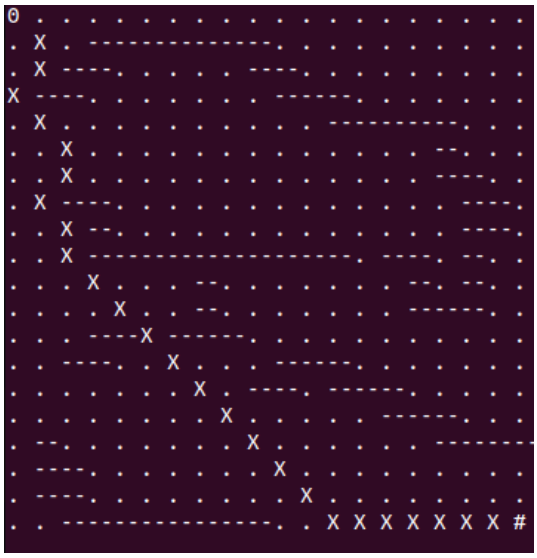


Maintenant lors de cette deuxieme soutenance j'ai resolue le probleme lorsque il y'a plusieur obstacle sur le trajet. Pour ce faire j'ai pensée a plusieurs solutions :
- Premiere idée a etait d'implementer une liste de pere et de distance comme djikstra sur le chemins qu'il renvoyé, cela marche mais et tres couteux lors du lancement de l'algorithme et donc sur une petit matrice

le temps de calcul ce faisait deja ressentir.
- Deuxieme idée, apres avoir implementé la remonté du pere sur tout les chemins parcourue, j'ai pensé a implementer directement dans la recherche du chemin une liaison lorsque j'ouvre un emplacement libre avec son l'emplacement actuelle pour pouvoir ensuite retrouver le chemin direct.
Voici donc une map sur l'aquelle il ne trouvais pas le chemin le plus rapide :



Et maintenant le meme chemin mais en appliquant le nouvelle algorithme :



8.3 Représentation des obstacles

La fois précédente, j'ai déplacé et fait pivoter des boids en utilisant des coordonnées polaires parce que c'était plus simple à mettre en œuvre, mais ajouter de la vitesse et de l'accélération au comportement des boids n'était pas extensible ou efficace en utilisant ces coordonnées.
J'ai dû refaire presque tout le projet pour l'affiner, j'ai utilisé des coordonnées polaires et cartésiennes, j'ai ajouté la vitesse et l'accélération à l'algorithme.
J'ai également segmenté l'espace pour réduire les calculs et faciliter la détection des collisions et les interactions avec les autres boids.
J'ai utilisé le centre de chaque boid pour la détection des collisions et cela fonctionne bien quand on entre en collision avec les bords de la fenêtre, le seul problème est la possibilité de tunneling en cas d'autres obstacles.

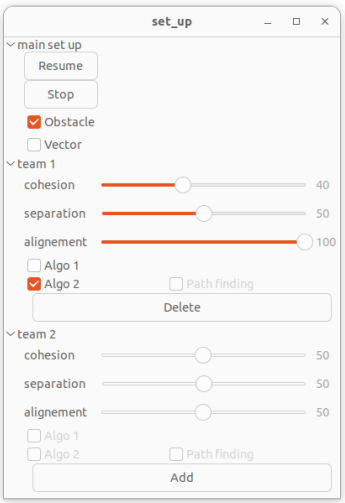
8.4 Implémentation de la console de pilotage de la simulation

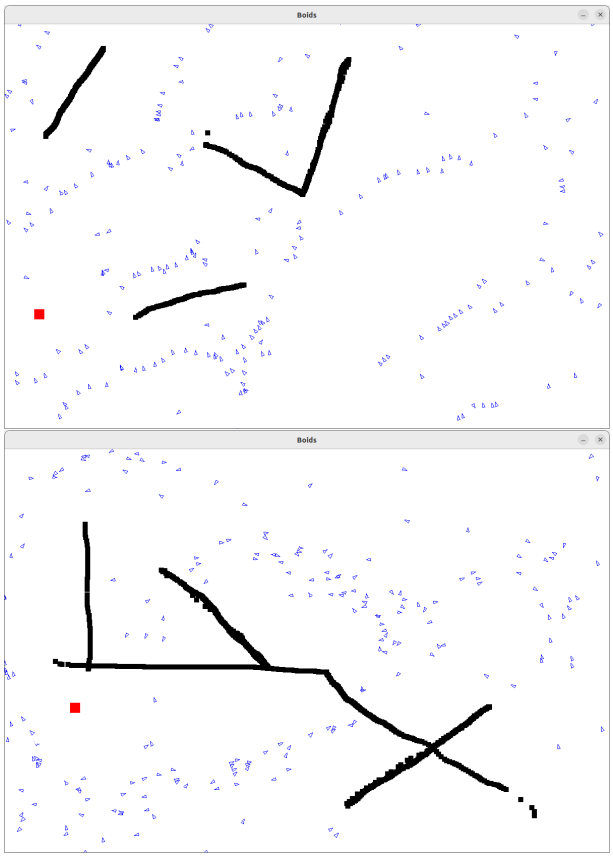
Lors de la soutenance 1, nous avons réalisé une interface graphique, implémenté l'algorithme de Boid, du pathfinding et du paint. Pour cette deuxième soutenance nous avons lié l'algorithme de Boid et le paint à l'interface graphique.

L'interface graphique / console de pilotage : permet de paramétrer la simulation.

- Le bouton Start/Pause : permet de lancer la simulation ou de la mettre en pause.
- Le bouton Stop : permet d'arrêter la simulation.
- Le checkbutton Obstacle : permet à l'utilisateur de dessiner des obstacles dans l'arène.
- Le cohésion/séparation/alignement slider : permettent de modifier les différents coefficients de Boids.
- Le checkbutton algo1/algo2 : n'active pas un algorithme en particulier, il établit des coefficients à l'algorithme de Boid.

Toutes ces fonctionnalités sont maintenant fonctionnelles et agissent sur la simulation.





Pour mener à bien cette partie, nous avons due utiliser les threads. Pour pouvoir executer l'inteface graphique et la simulation nous avons utilisé un thread qui lance la simulation lorsque le boutton Start est pressé et s'arrête lorsque le boutton Stop est pressé.
En plus de pouvoir executer deux programmes simultanément, nous avons due échanger des informations, de l'interface graphiqe à la simulation (Pause/Resume/Stop, Obstacle, Coefficients de Boid, Algo1, Algo2). Ce que nous faisons pour échanger ces informations c'est : tout d'abord créer des variables globales, qui représentent le status de chaque option disponible de l'interface (Pause/Stop, Obstacle, Coefficients de Boid, Algo1, Algo2), et qui pourront être modifier dans un autre fichier avec des fonctions dédié pour.

8.5 Site Web

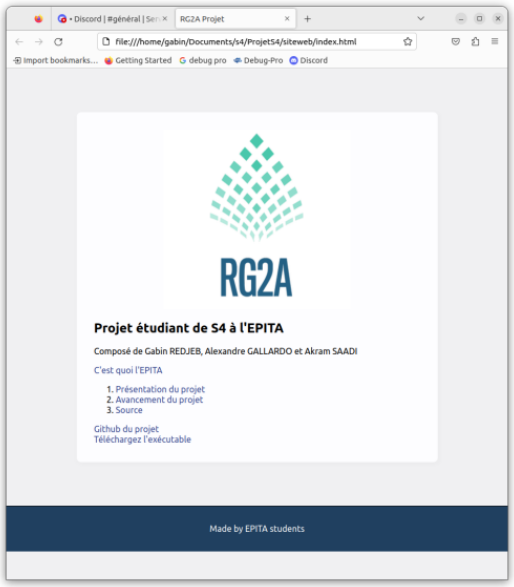
Le site web est composé d'une page d'accueil, d'une page de présentation du groupe et du projet et d'une page amenant aux sources utilisées pour la réalisation du projet.
La page d'accueil comporte les 5 liens présenté lors de la soutenance 1 qui amène à :

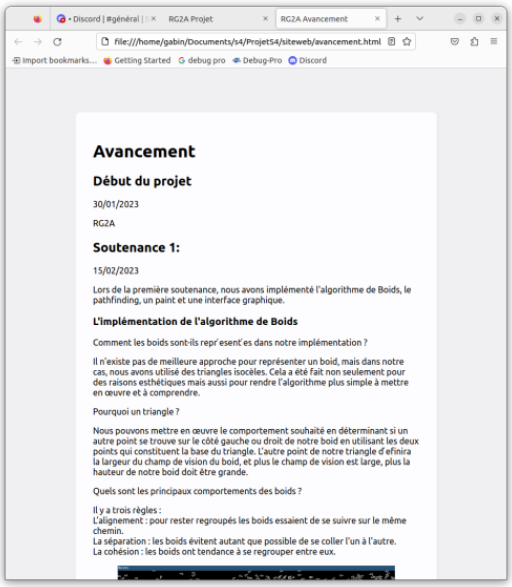
- La page de l'EPITA
- La page de présentation du groupe et du projet
- La page source
- La page de projet GitHub
- L'installation de l'exécutable du projet

Et un 6 liens:

- Avancement du projet

qui représente l'avancement du groupe dans le projet RG2A au fir et à mesure des soutenances.



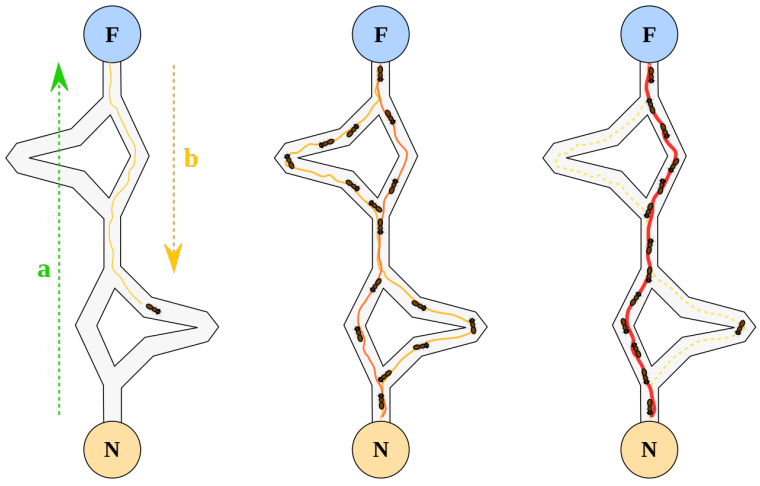


9 Soutenance 3

9.1 Implémentation de l'algorithme de colonies de fourmis

L'algorithme des fourmis est un algorithme inspiré du comportement des fourmis, ou d'autres espèces formant un superorganisme, et qui constituent une famille de métaheuristique d'optimisation.
L'idée originale provient de l'observation de l'exploitation des ressources alimentaires chez les fourmis. En effet, celles-ci, bien qu'ayant individuellement des capacités coginitives limitées, sont capables collectivement de trouver le chemin le plus court entre une source de nourriture (la cible dans notre cas) et leur nid (dans notre cas il n'y en a pas).

Des biologistes ont ainsi observé, dans une série d'expériences menées à partir de 1989, qu'une colonie de fourmis ayant le choix entre deux chemins d'inégale longueur menant à une source de nourriture avait tendance à utiliser le chemin le plus court.



Un modèle expliquant ce comportement est le suivant :
1) une fourmi (appelée "éclaireuse") parcourt plus ou moins au hasard l'environnement autour de la colonie.
2) si celle-ci découvre une source de nourriture, elle rentre plus ou moins directement au nid, en laissant sur son chemin une piste de phéronomes.
3) ces phéromones étant attractives, les fourmis passant à proximité vont avoir tendance à suivre, de façon plus ou moins directe, cette piste.
4) en revenant au nid, ces mêmes fourmis vont renforcer la piste.
5) si deux pistes sont possibles pour atteindre la même source de nourriture, celle étant la plus courte sera, dans le même temps, parcourue par plus de fourmis que la longue piste.

Dans notre cas, les fourmis nous pas de nid et donc ne permettent pas de trouver le plus court chemin entre le nid et la cible (la nourriture).
Ce que nous avons fait pour implémenter cet algorithme c'est :
Les fourmis se comportent pendant tout le long de la simulation comme des boids, en suivant l'algorithme de Boid en fonction des ces coefficients.
Lorsque une fourmis (ou boid) trouve la cible, elle va laisser derrière elle des phéromones pour indiquer aux autres fourmis le chemin à la cible. Et donc losque une fourmis passe sur l'une de ces phéromones, la fourmis se dirige vers la cible. Et lorsqu'elle trouve la cible, elle laissera derrière elle des phéromones pour indiquer aux autres fourmis etc, jusqu'à que toutes les fourmis est trouvé la cible.

Dans la simulation les fourmis qui cherche toujours la cible sont représentés en bleu foncé, lorsqu'elles ont trouvé la cible en orange et lorsqu'elle suive les phéromones en bleu fluot/clair.

9.2 Implémentation du Path Finding

Pour l'implementation de l'algorithme de pathfinding nous avons d'abord choisis d'implementer celui de A* qui correspond au mieux a nos demande pour un algorithme de pathfinding le plus general possible. En effet celui-ci recherche le chemins le plus cours entre un point A et un point B en considerant des obstacles. L'algorithme ce base sur une structure fifo avec ordre de prioritee basée sur une valeur heuristic. La valeur heuristice se calcule pour chaque noeud u par heuristic = cout + distance(u, destination), le cout correspond au deplacement total depuis le point de depart.

Lors de cette soutenance nous avons integrer cette algortihme a l'ensemble du projet, cependant l'algortihme etait extremement gourmant en ressource de calcul. Il a fallut donc optimiser l'algorithme tout en l'adaptant au mieux au projet.
Nous avions aussi pour projet d'implementer un algorithme qui prend seulement un points de départ et qui cherche une cible sur la map donnée en entrer. Cette algorithme étants glouton et ralentissez l'execution du programme en entier, nous ne l'avons donc pas retenue pour ce projet.
Le path finding sert a chaque boid de trouver leur chemin sur la carte en évitant les obstacles. Pour cela nous avions précédemment implementer deux algortihme de recherche de plus cours chemin.

En effet on avait implementer l'algorithme A* qui renvoie le chemin le plus cours entre deux points donné.
Puis une variante qui d'un point donné recherche une cible d'on les coordonnées ne sont pas connu et renvoie un chemin parmis ceux les plus cours.

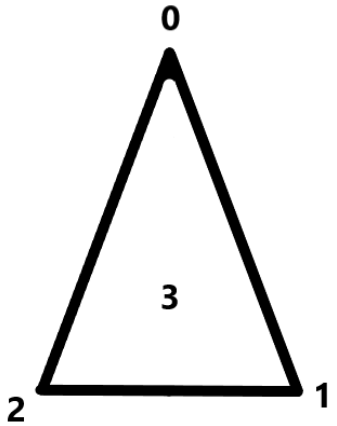
9.3 Détournement des obstacles

Le but de cette partie est de permettre aux boids de ne pas transpercer les obstacles dans l'arène. Pour se faire chaque boid va devoir analyser son environnement et va retenir l'obstacle le proche de lui dans un rayon prédéfinie.

S'il n'y a pas d'obstacle dans son rayon d'observation le boid applique l'algorithme de Boid et se comporte normalement.

Sinon il ne va pas appliquer l'algorithme de Boid sauf pour un certain cas et va se focaliser sur comment éviter cet obstacle.
Les boids sont représentés par des triangles avec 4 positions différentes:

- Position 0 : représente le sommet à l'avant du boid
- Position 1 : représente le sommet droit du triangle
- Position 2 : représente le sommet gauche du triangle
- Position 3 : représente le centre du triangle



Avec ces 4 positions et les distances entre l'obstacle et ces 4 positions, on peux savoir de quel coté le boid doit se diriger et à quelle intensité.

Si la distance obstacle/position0 est la plus grande des distances, on sait que l'obstacle n'est pas sur le chemin du boid et donc on considère qu'il n'y a pas d'obstacle.
Sinon, si la distance obstacle/position1 est plus petite que celle de l'obstacle/position2, on sait que l'obstacle est à droite et donc que le boid doit s'orienter vers la gauche.
Si la distance obstacle/position2 est plus petite que la distance obstacle/position1, on sait que l'obstacle est à gauche et donc que le boid doit s'orienter vers la droite.

De plus, plus la distance entre l'obstacle et le centre du boid (position3) est petite, plus l'angle de détournement sera important.

9.4 Interface graphique: réunification des deux algorithmes de recherche

Lors de la soutenance 1, nous avons réalisé une interface graphique, implémenté l'algorithme de Boid, le pathfinding et un paint. Lors de la soutenance 2, nous avons lié l'algorithme de Boid et le paint à l'interface graphique. Et pour cette dernière soutenance, nous avons lié les deux algorithmes de recherche (l'algorithme de colonies de fourmis et le path finding) à l'interface graphique et à la simulation.

L'interface graphique / console de pilotage : permet de paramétrer la simulation.

- Le bouton Start/Pause : permet de lancer la simulation ou de la mettre en pause.
- Le bouton Stop : permet d'arrêter la simulation.
- Le checkbutton Obstacle : permet à l'utilisateur de dessiner des obstacles dans l'arène.
- Le cohésion/séparation/alignement slider : permettent de

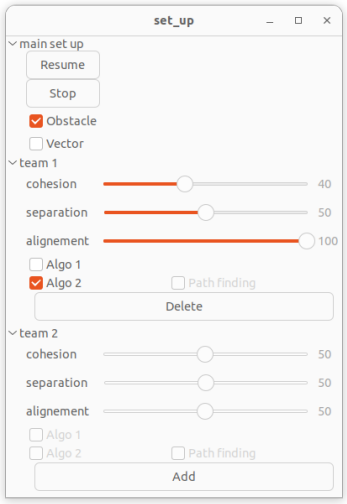
modifier les différents coefficients de Boids.

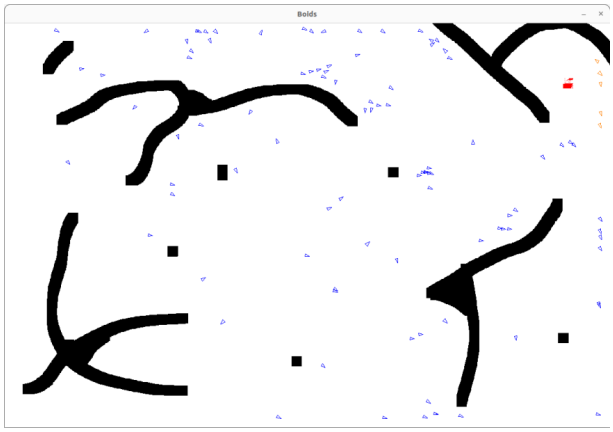
- Le checkbutton Algo Ant: active l'algorithme de recherche de la colonie de fourmis.

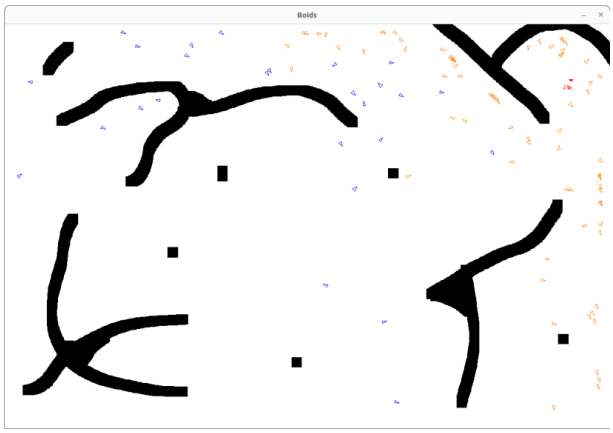
- Le checkbutton Algo Path Finding: active l'algorithme de recheche Path Finding.

- Le checkbutton Path Finding: permet de diriger tous les boids vers le curseur de la souris.

Toutes ces fonctionnalités sont maintenant fonctionnelles et agissent sur la simulation.



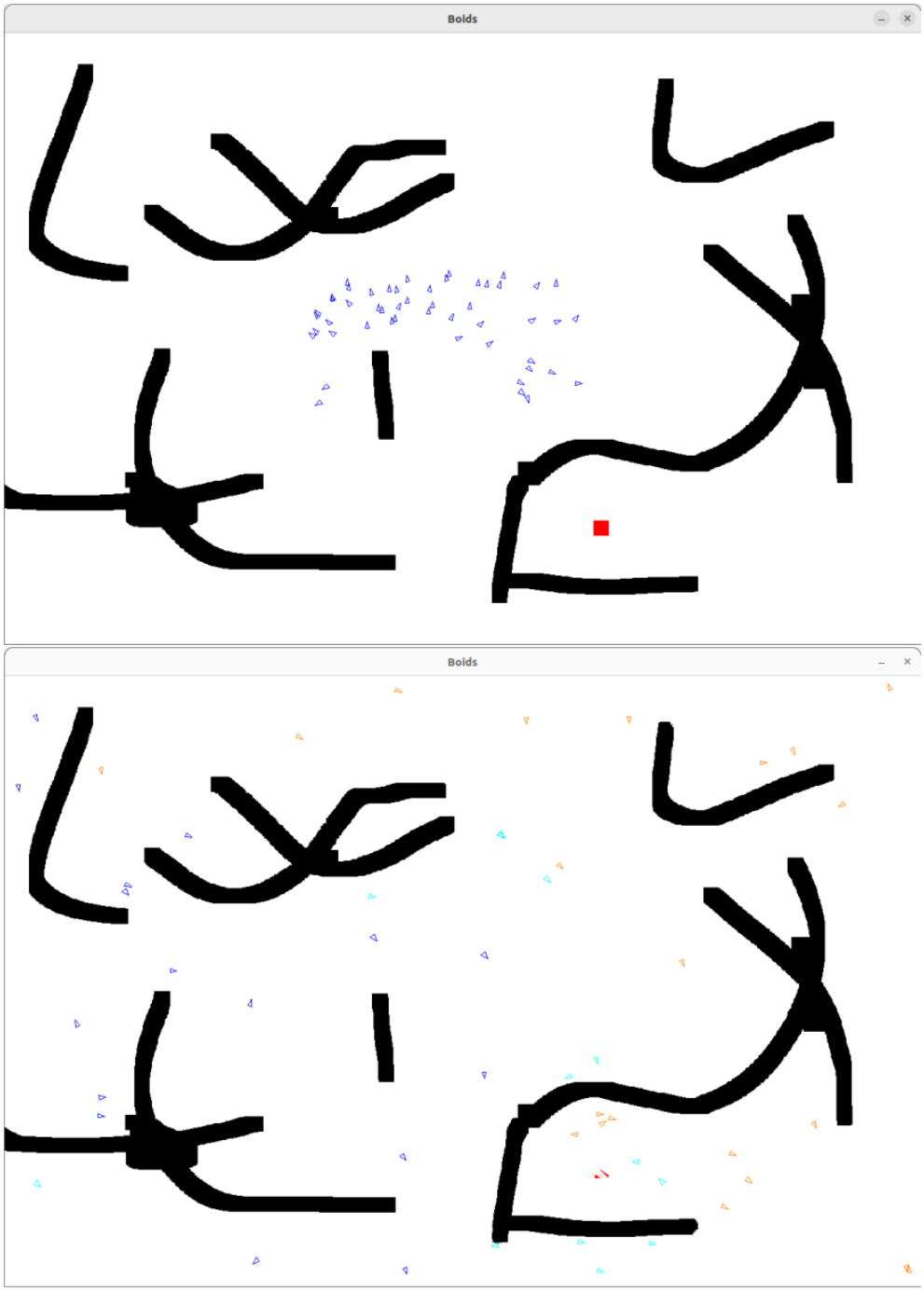




Pour mener à bien cette partie, nous utilisons une variables globales qui représente le status de l'algorithme de recherche choisit. 0 pour aucun, 1 pour l'algorithme des fourmis, 2 pour le l'algorithme de recherche path finding et 3 pour seulement le path finding d'un point A (position des boids) à B (position pointé par le curseur de la souris).

Lorsqu'un boid trouve la cible il devient orange et le communique à ses voisins les plus proches.
Lorsqu'un boid passe à coté d'une phéromone il devient bleu clair et se dirige vers la cible

9.4.1 Algo Ant



9.4.2 Algo Path Findinng

10 Conclusion

Pour conclure cette fin de projet, nous sommes globalement satisfaits de notre travail.

En effet, nous avons atteint la plupart de nos objectifs notamment l'interface graphique et différents algorithmes de recherche. Nous avons cependant dû renoncer à la représentation de plusieurs équipes.

Notre satisfaction est, cependant, nuancée car nous finalisons le projet à deux alors que l'équipe initiale était composée de quatre membres. En effet, un de nos équipiers à choisi de quitter l'EPITA en début de S4 ; un autre a abandonné le projet deux jours avant la soutenance finale.

Pour conclure, le projet a été marqué par de nombreux challenges relevés tant techniquement que humainement.

Nous avons apprécié travailler sur les sujet abordés dans le cadre de ce projet et sommes globalement satisfaits de notre travail de réalisation.