RG2A

Rapport de soutenance 2

Redjeb Gabin

Gallardo Alexandre

Saadi Akram

Table des matières

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.

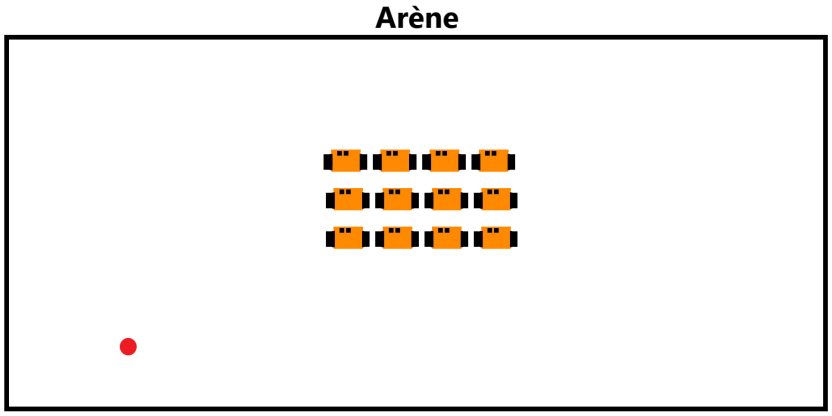
1.3 Saadi Akram

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 :
12 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)

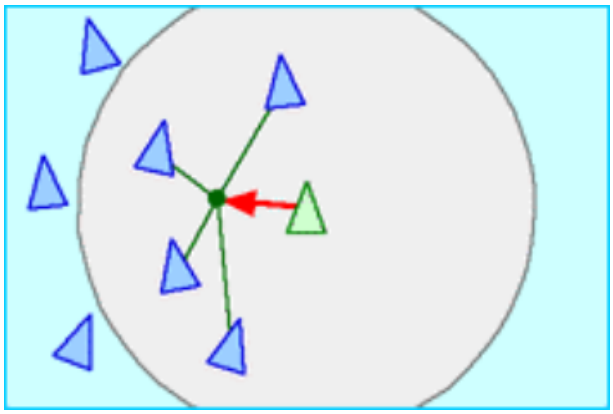
— le contrôle et la stabilisation de véhicules sans pilote : permettant d'explorer des lieux inconnus qu'ils soient terrestres, aériens ou sous-marins.

2.3 Intérêt algorithmique

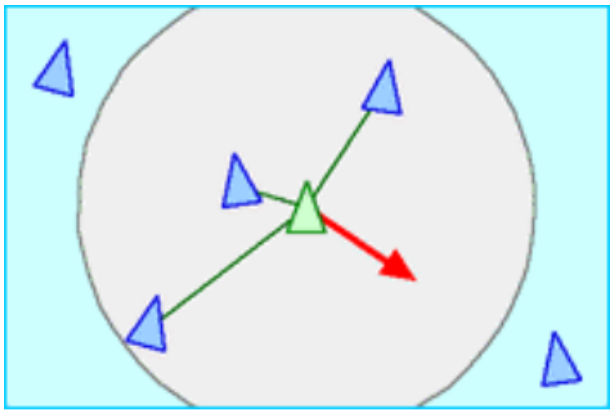
Boids est un programme de vie artificielle, qui simule le comportement de vol des oiseaux et qui permet de modéliser un comportement émergent, c'est-à-dire une complexité comportementale qui résulte de l'interaction d'agents individuels respectant un nombre limité de règles simples.

Règles :

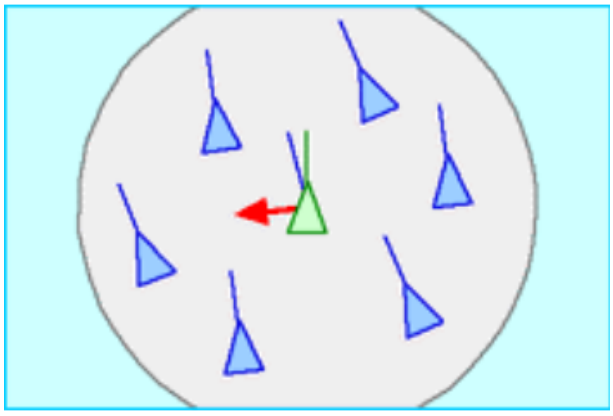
Cohésion : pour rester groupés, les robots essaient de suivre un même chemin.



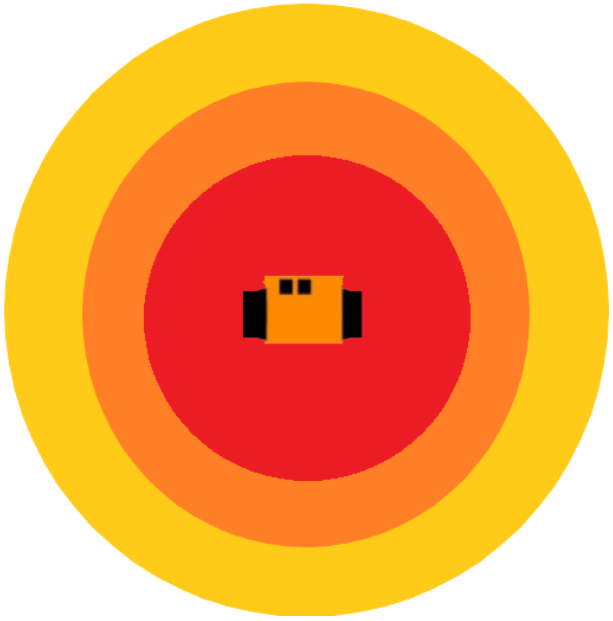
Séparation : deux robots ne peuvent pas se trouver au même endroit au même moment donc pour éviter de s'entasser chaque robot doit s'éviter.



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

Tâches	**Gabin**	**Alexandre**	**Akram**
Implémentation algo Boids :	-	-	X
Implémentation de la recherche de la cible :	X	X	X
Algorithme de colonnies 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	-
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

4 Planning

Tâches	**Soutenance 1**	**Soutenance 2**	**Soutenance 3**
Avancement	réalisé	prévu - réalisé	prévu
Implémentation algo Boids :	90%	100% - 100%	100%
Implémentation de la recherche de la cible :	10%	50% - 40%	100%
Algorithme de colonnies de fourmis	-	20% - 10%	100%
Algorithme path finding	20%	60% - 70%	100%
Représentation graphique :	70%	75% - 75%	100%
Représentation des boids	100%	100%	100%
Représentation des vecteurs	100%	100%	100%
Représentation des obstacles	80%	90% - 90%	100%
Représentation de plusieurs équipes	-	-	100%
Interface Graphique :	100%	100%	100%
Paramétrage algorithme de boids	100%	100%	100%
Paramétrage obstacles	100%	100%	100%
Paramétrage path finding	100%	100%	100%
Paramétrage équipes	100%	100%	100%
Site Web :	25%	60% - 60%	100%

TABLE 2 – 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 2

7.1 Implémentation de l'algorithme de colonnies 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.

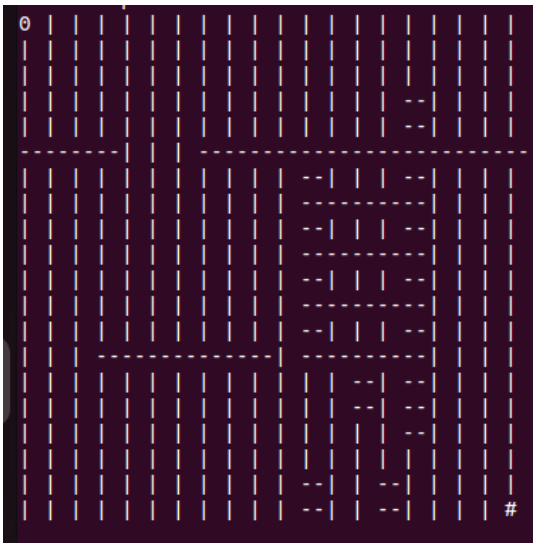
Après plusieurs tentatives pour combiner l'algorithme de Boid et l'algorithme de la colonie de fourmis, je n'ai pas été en mesure de trouver une solution optimale. Le comportement des composants individuels et leurs interactions étaient difficiles à contrôler et compliqués à comprendre. J'ai également 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.

7.2 Implémentation du Path Finding

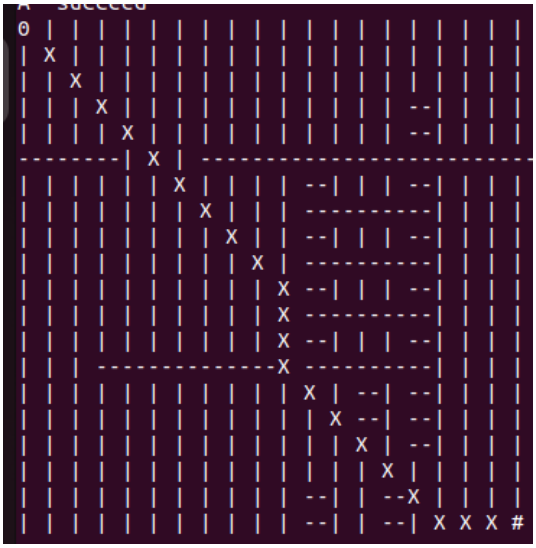
Pour l'implementation de l'algorithme de pathfinding nous avons d'abord choisis d'implementer celui de A* qui correspond au mieux a nos demande pour un algorithme de pathfinding le plus general possible. En effet celui-ci recherche le chemins le plus cours entre un point A et un point B en considerant 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;
};
```

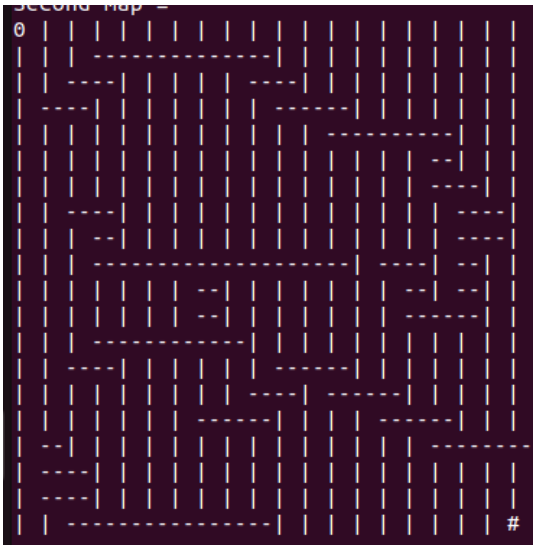
Ici la structure de donnée qui contiendra chaque pixel qui sera atteint par l'algorithme. Ci-dessous une map avec un point de depart (0) et un point d'arrivé (#) ainsi que les obstacles (–) et les postitions possible (—) :



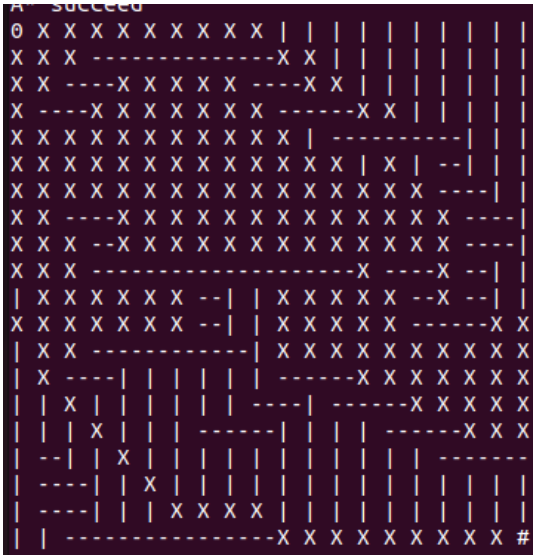
La meme map qui a été resolue par l'algorithme :



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 :

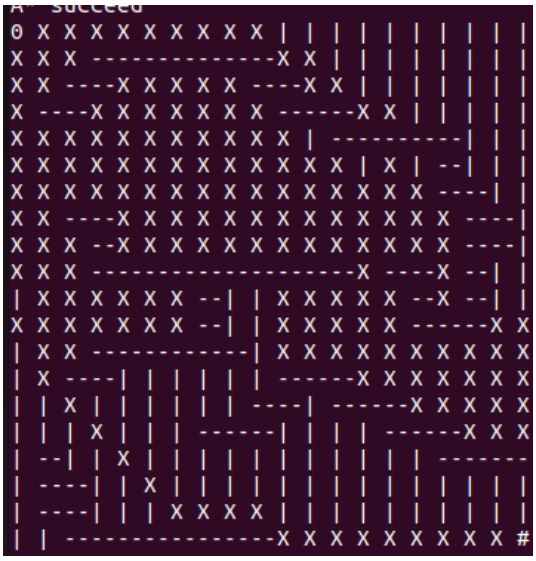


Maintenant lors de cette deuxieme soutenance j'ai resolue le probleme lorsque il y'a plusieur obstacle sur le trajet. Pour ce faire j'ai pensée a plusieurs solutions :

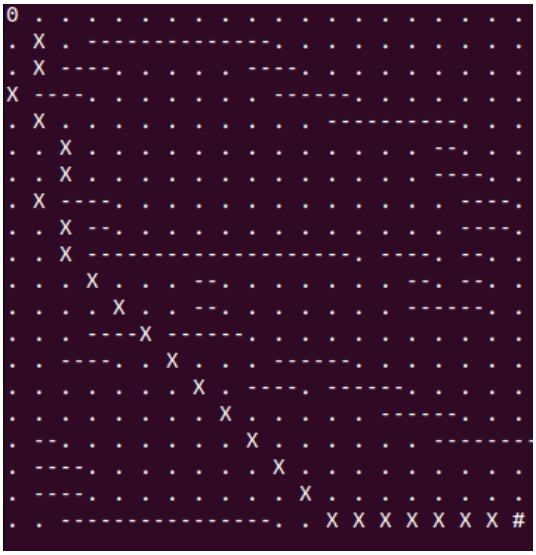
1 - Premiere idée a etait d'implementer une liste de pere et de distance comme djikstra sur le chemins qu'il renvoyé, cela marche mais et tres couteux lors du lancement de l'algorithme et donc sur une petit matrice le temps de calcul ce faisait deja ressentir.

2 - 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 :



7.3 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.

7.4 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.

7.5 Site Web

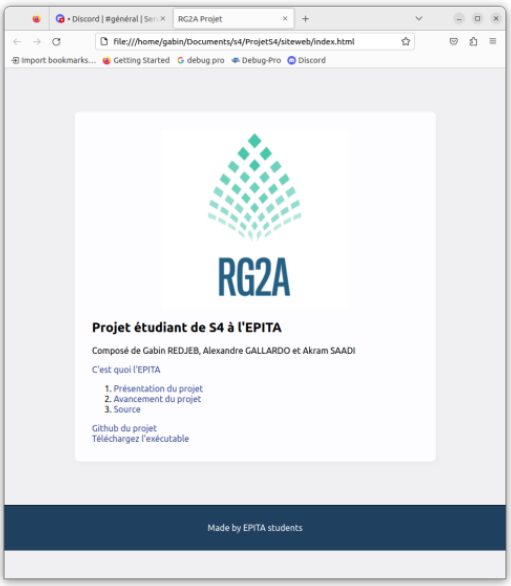
Le site web est composé d'une page d'accueil, d'une page de présentation du groupe et du projet et d'une page amenant aux sources utilisées pour la réalisation du projet.
La page d'accueil comporte les 5 liens présenté lors de la soutenance 1 qui amène à :

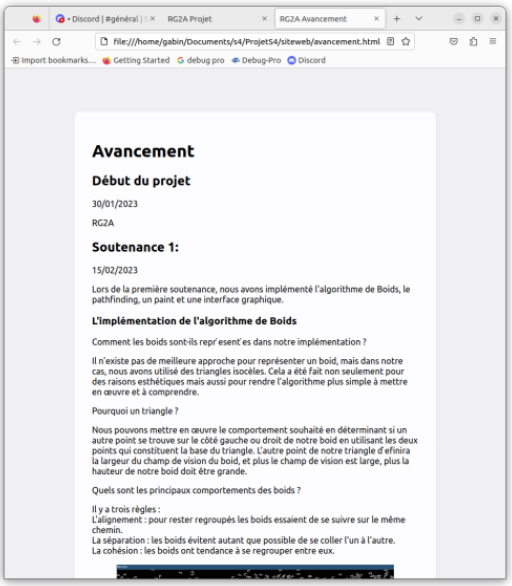
- La page de l'EPITA
- La page de présentation du groupe et du projet
- La page source
- La page de projet GitHub
- L'installation de l'exécutable du projet

Et un 6 liens :

- Avancement du projet

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





8 Conclusion

Nous sommes fier de notre travail vit a vit de cette premiere soutenance et de plus nous avons realise toutes les taches pour la premiere soutenance.