



# Squadron Bêta

Marty Hugo & Rouiller Cyril

oct.25 2024

# Introduction

- `oort.rs` est un jeu de programmation qui utilise le langage Rust afin de contrôler une flotte de vaisseaux dans différents défis et duels.

# Introduction

- Le tournois se déroulant sur le duel de fighter.
- Nous avons donc utilisé son programme par défaut comme base, puis nous l'avons amélioré jusqu'à l'amener dans le leaderboard.

# Sommaire

- Introduction
- Équipements
- R&D
- Comportements
  - Modes / États
  - Mouvements
- Conclusion

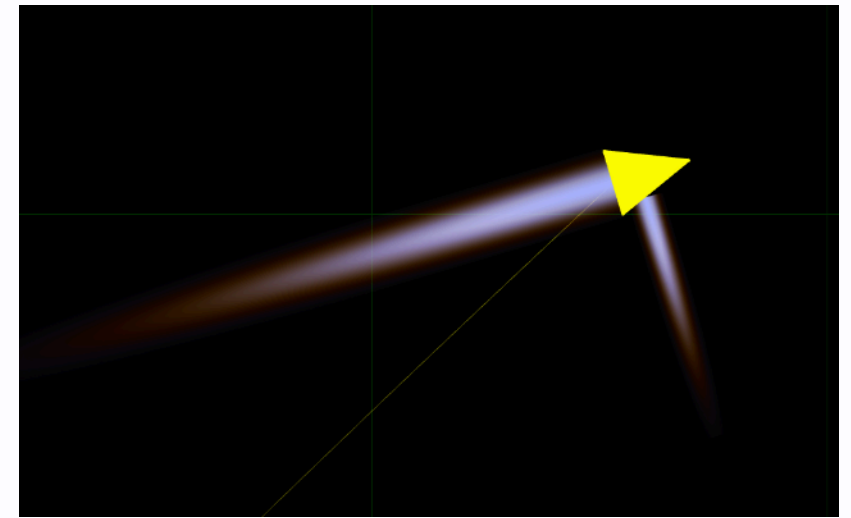
# Équipements

- Notre vaisseau dispose de cinq éléments:
  - des réacteurs
  - un radar
  - une radio
  - une mitrailleuse
  - des missiles

# Équipements :

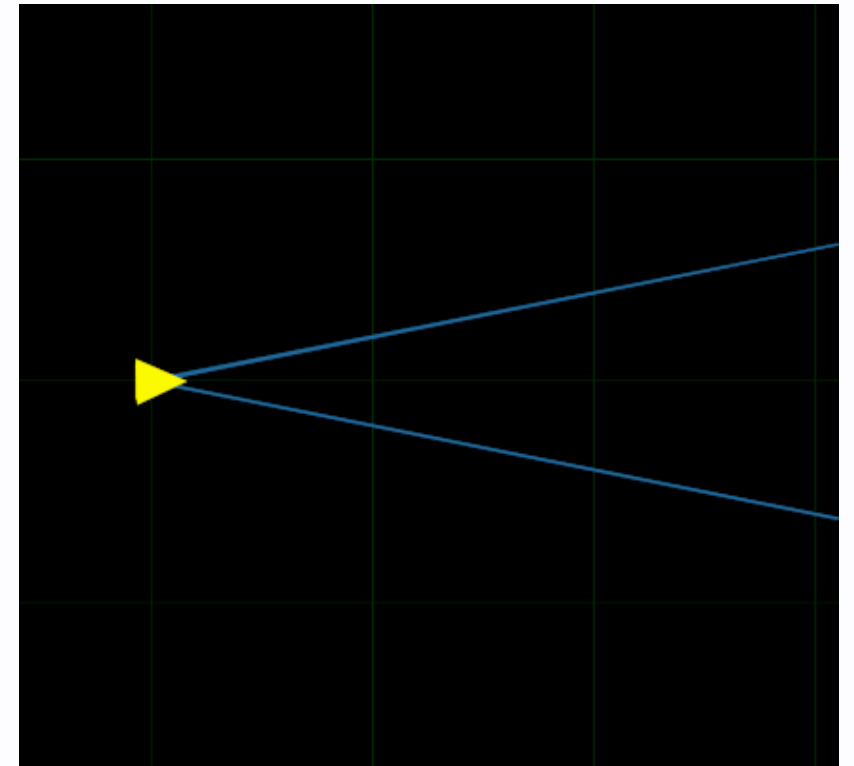
## *Réacteurs*

Les réacteurs permettent au vaisseau de se déplacer dans l'espace ainsi que d'effectuer des rotations sur lui même.



# Équipements : *Radar*

Le radar est un outil qui donne la possibilité de scanner dans une direction donnée. Il renvoie ensuite les coordonnées et la vélocité du contact ennemi.



# Équipements : *Radio*

La radio est un élément invisible qui nous permet d'envoyer et de recevoir des messages sur un canal choisi.

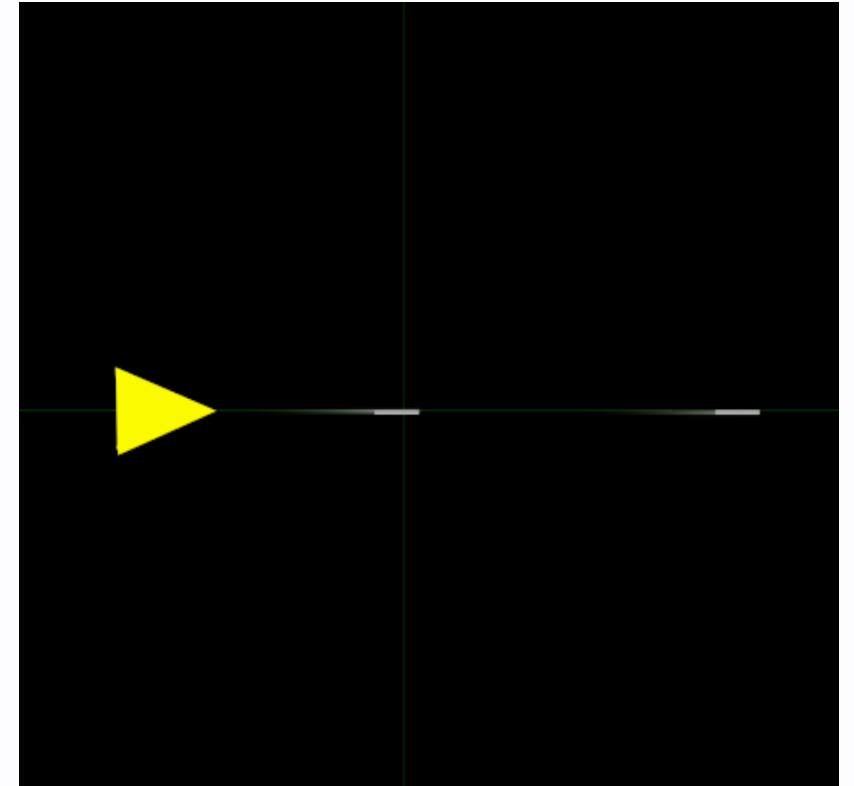
Les canaux sont communs aux deux équipes et sont au nombre de 10 (0..9)



# Équipements :

## *Mitrailleuse*

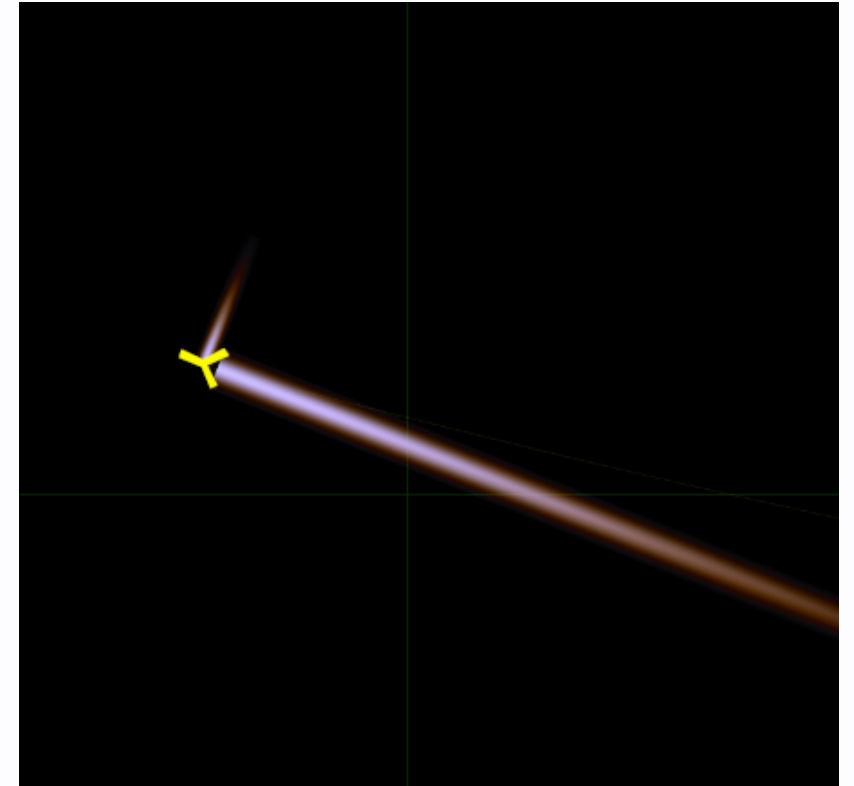
La mitrailleuse permet au vaisseau de tirer des balles qui sont des projectiles, elles ne sont plus contrôlables une fois tirées.



# Équipements :

## *Missiles*

Les missiles agissent comme des mini vaisseaux, ils sont dirigeables et comportent la radio ainsi que le radar.

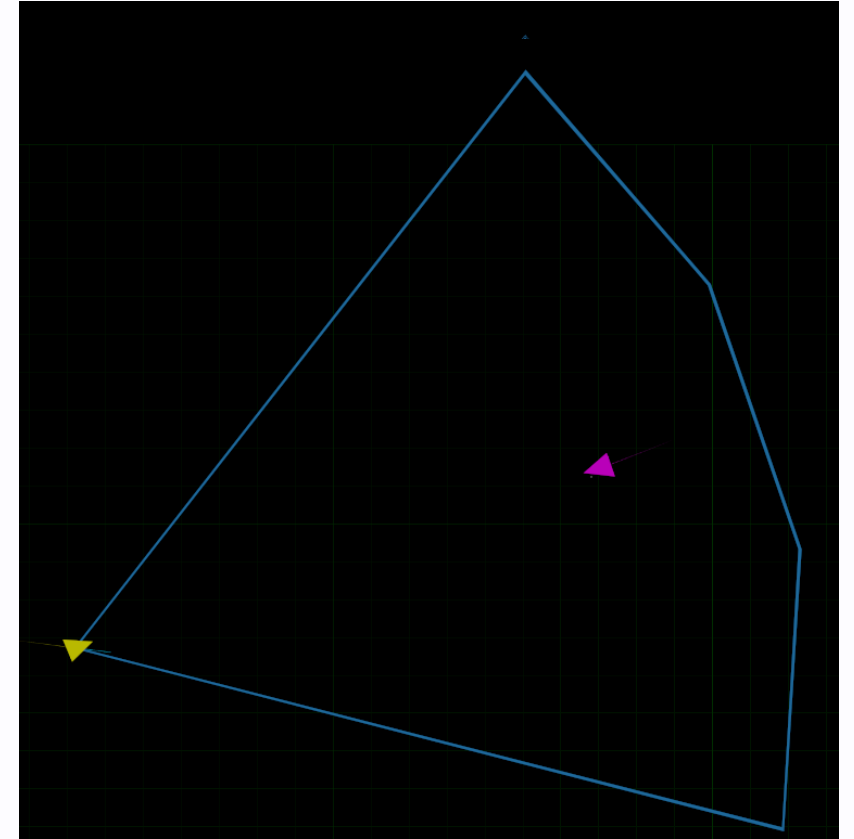


# R&D

- À partir du code de base, nous avons amélioré les éléments suivants:
  - Radar
  - Radio
  - Missiles

## R&D : *Radar*

- Le radar a été amélioré afin de se redimensionner de manière à être le plus large tout en gardant une distance de sécurité derrière l'ennemi



## R&D : *Radar pt2*

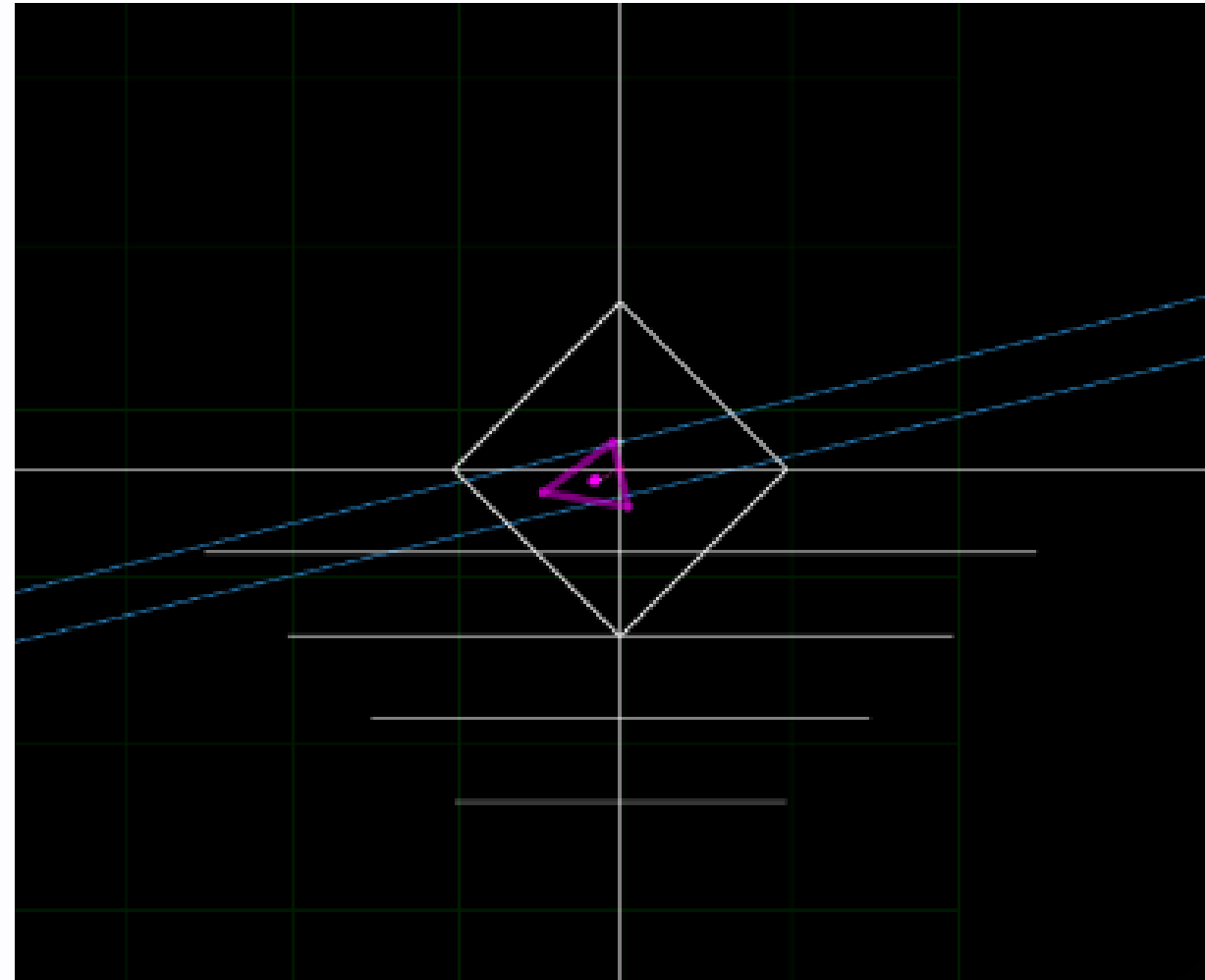
- Le radar ne disposant de méthode que pour donner la largeur.
- Nous avons donc fait notre propre méthode qui donne la largeur du radar en prenant en paramètres sa longueur.
- Formule :  $1 - (((distance\_voulue * 1.25) - 20'000) / 10) * 0.00125)$

## R&D : *Radar pt3*

- Nous avons aussi fait une approximation de la position de l'ennemi.
- À chaque visualisation de l'ennemi on décale tout le tableau de un cran vers la gauche et on push la nouvelle valeur à la fin du tableau.
- La position utilisée par tout le code est une moyenne non pondérée des 10 dernières positions enregistrées de l'ennemi.

## R&D : *Radar pt4*

- Pour plus de clareté visuelle un viseur est également dessiné sur l'ennemi / à sa dernière position connue.



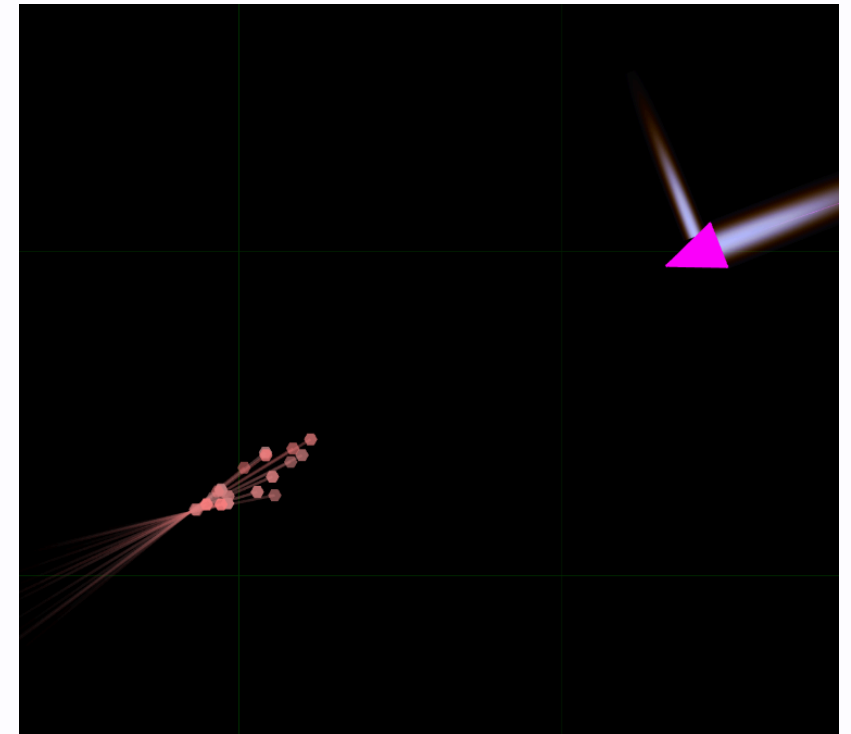
## R&D : *Radio*

- La radio change aléatoirement de canal à chaque tick afin de ne pas interférer avec l'ennemi.
- Utilisation de `oorandom::Rand32`
- Donne un nombre aléatoire suivant une "*seed*" et un incrément.



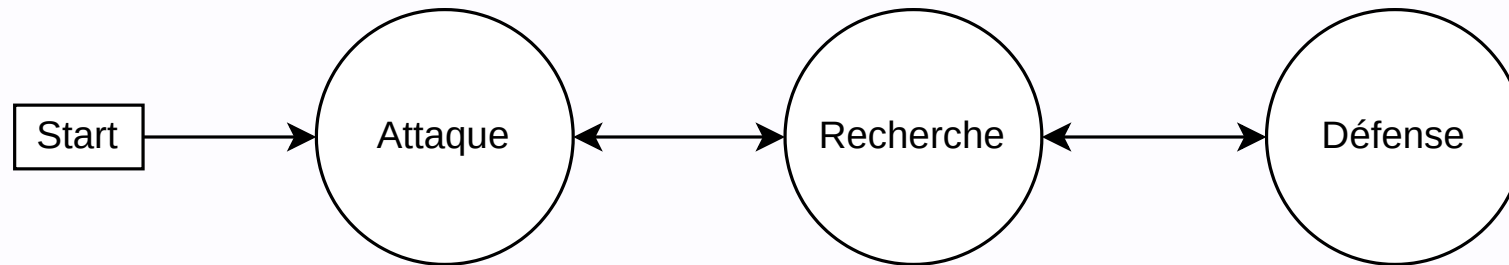
# R&D : *Missiles*

- Les missiles font des calculs pour exploser cent millisecondes avant de toucher l'ennemi.
- En faisant cela les débris augmentent la chances de toucher l'ennemi en lui infligeant tout autant de dégâts.



# Comportement

Le vaisseau comporte trois états.



# Attaque

- En mode **attaque**, le vaisseau fait rotationner son radar afin de balayer l'horizon et se dirige vers la dernière position connue de l'ennemi.
- Si il y trouve un ennemi de type "*fighter*" il lui lance un missile et communique sa position par radio à tous les missiles.
- Tous les 30 ticks le vaisseau passe en mode **recherche**

# Recherche

- Le vaisseau va mettre son radar le plus large possible (un quart) et va faire un tour de scan en 4 ticks
- Si un missile est détecté, le vaisseau passe en mode **défense**
- Sinon il retourne en mode **attaque**

# Défense

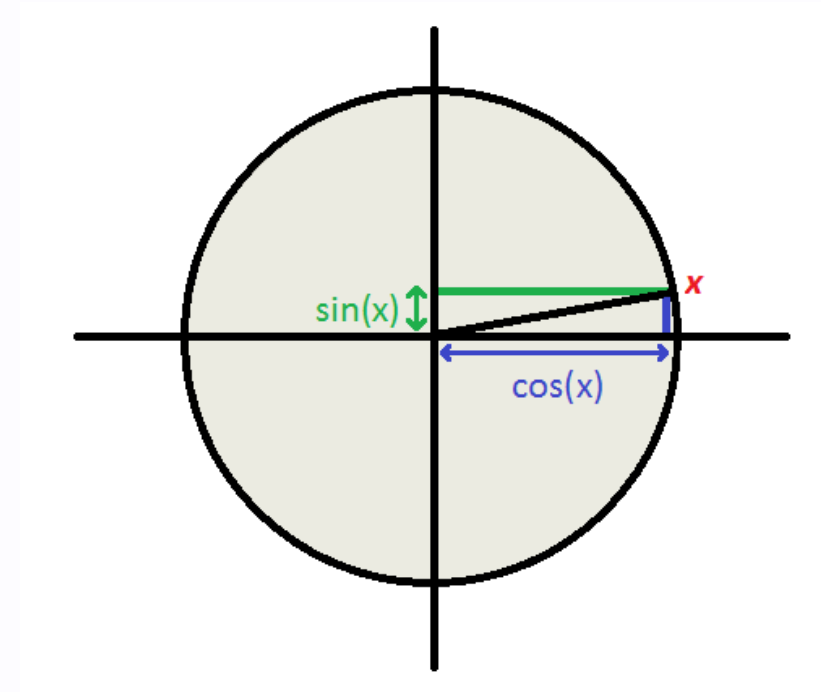
- En mode **défense**, le vaisseau va cibler le missile trouvé en recherche et essayer de le détruire
- Si le missile est détruit, perdu de vue ou nous passe à côté (*plus de 90° de différence avec l'ennemi*) on retourne en **recherche**

# Mouvements

- Le vaisseau suit deux comportements :
  - en **recherche** et **défense** on se laisse porter
  - en **attaque** on utilise la fonction ***unpredictable\_trajectory***

# *Unpredictable trajectory*

- Cette fonction est basée sur "seek" à la différence que nous utilisons comme point où se diriger notre position actuelle additionnée à :
- (en x) le cosinus de la distance entre nous et l'ennemi
- (en y) le sinus de la distance à laquelle nous souhaitons passer



# Conclusion

- Codage de vaisseau interessant
- Meilleures compréhension de la trigonométrie
- Quand-même trop de maths
- Classement pas mauvais 😊



# Questions ?