Apis mellifera vs Vespa crabo

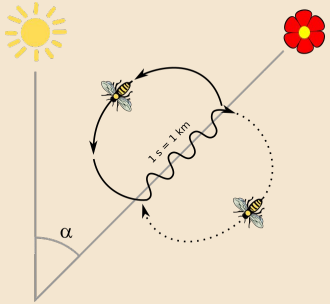
Modélisation par un système multi-agents - AI30

Alexandre Bidaux, Julie Chartier, Clément Martins, Quentin Valakou

Sommaire

Les abeilles : des insectes fascinants

- La danse comme moyen de communication



- Organisation de la ruche
  - Reine
  - Ouvrières
  - Faux bourdons

- Un objectif commun : la survie de la colonie
  - Récolte du nectar
  - Production de miel
  - **Défense contre les prédateurs**

Choix du sujet et problématique

Apis mellifera

Abeilles européennes à miel

vs

Vespa crabo

Frelons européens

Quelles proportions d'abeilles à miel faut-il à une colonie pour prospérer face au danger des frelons ?

Règles

- Une **abeille** vit dans une ruche.
- Une **abeille** a trois métiers au cours de sa vie : ouvrière, gardienne, butineuse.
- Une **butineuse** est la seule pouvant sortir de la ruche.
- Une **butineuse** trouve des fleurs à butiner.
- Une **butineuse** dépose le nectar à la ruche en fonction de la quantité qu'elle peut transporter.
- Une **butineuse** qui rencontre une fleur retient sa position.
- Une **butineuse** connaissant la position d'au moins une fleur partage ses connaissances aux autres butineuses.
- Une **butineuse** peut butiner une fleur à la fois et une fleur ne peut être butinée que par une butineuse à la fois.
- Une **gardienne** reste à la ruche et ne produit pas de miel.
- Une **gardienne** alerte toutes les abeilles de la ruche lors d'une attaque de frelon.
- Une **ouvrière** transforme le nectar en miel.
- Les **abeilles** fuient les frelons (mais défendent la colonie au prix de leur vie).
- La **ruche** contient une reine qui produit des abeilles en consommant du miel.
- La **ruche** meurt si elle n'a plus de reine.
- Une **ruche** peut tuer un frelon si il y a suffisamment d'abeilles à l'intérieur.
- Un **frelon** chasse les abeilles.

Agents

Abeilles 

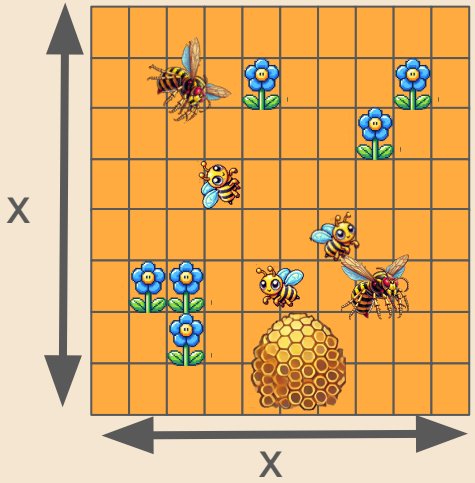
Frelons 

Objets

Ruche 

Fleurs 

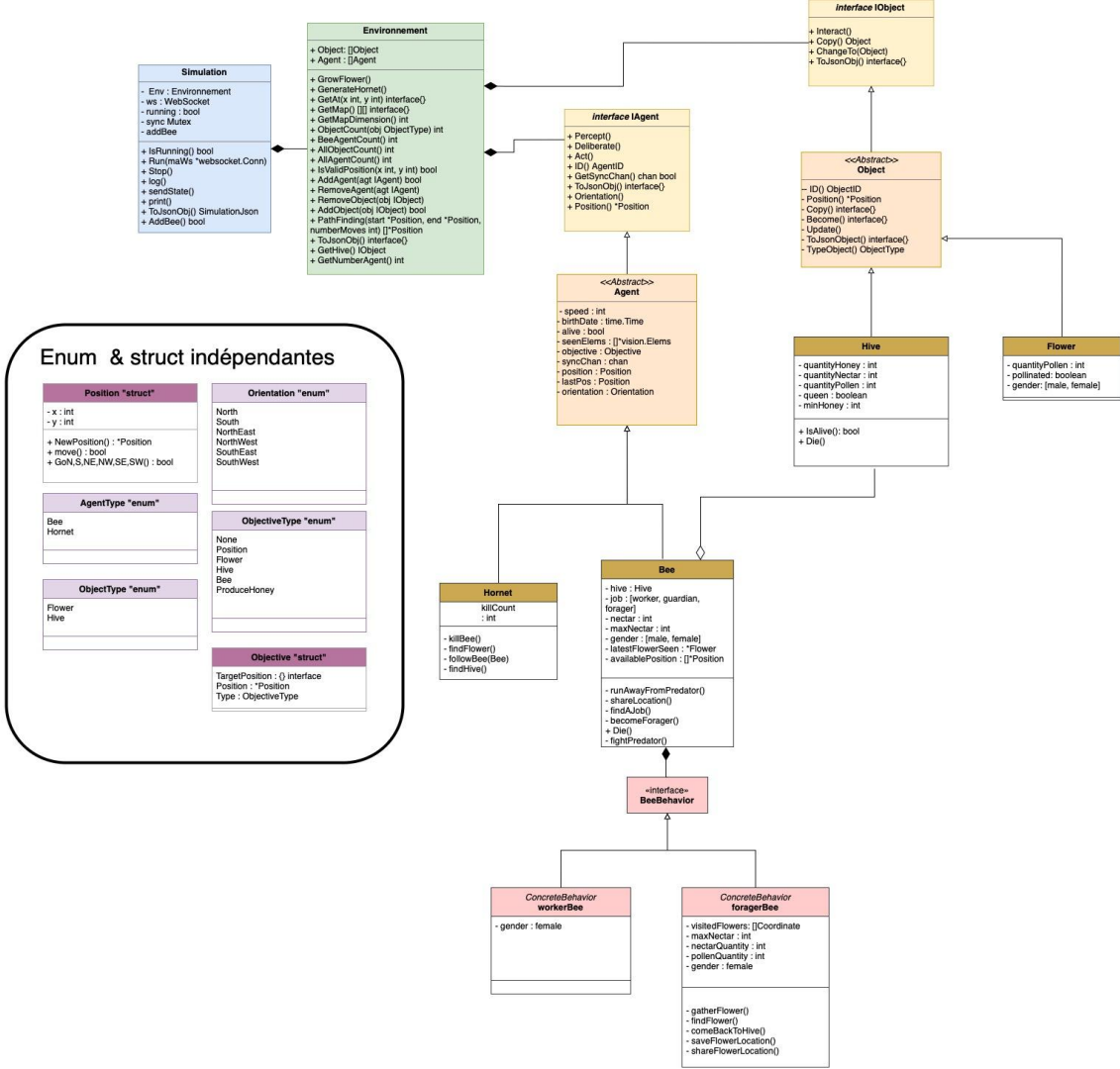
Environnement



Modélisation UML

Points clés :

- Interface pour les agents et les objets.
- Deux types d'agent : abeilles et frelons.
- Deux types d'objet : fleurs et la ruche.
- Une abeille peut avoir plusieurs comportements (design pattern Strategy).



Structures & interfaces

Environment *struct* :

- grid [][]interface{}
- agts []IAgent
- objs []IObject
- sync Mutex

\+ De nombreuses méthodes dont : ajouter, supprimer, envoyer Json, gestion de la grille (get, isValid, count), pathFinding.

IAgent *interface* :

- ID() AgentID
- Position() *Position
- Orientation() Orientation
- GetSyncChan() chan AgentID
- ToJsonObj() interface{}
- Start()
- Percept()
- Deliberate()
- Act()
- Type() AgentType

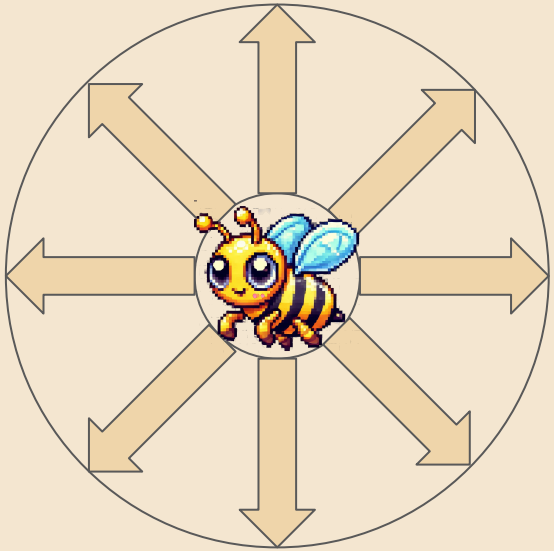
IObject *interface* :

- ID() ObjectID
- Position() *Position
- Copy() interface{}
- Become(interface{})
- ToJsonObj() interface{}
- Update()
- TypeObject() ObjectType

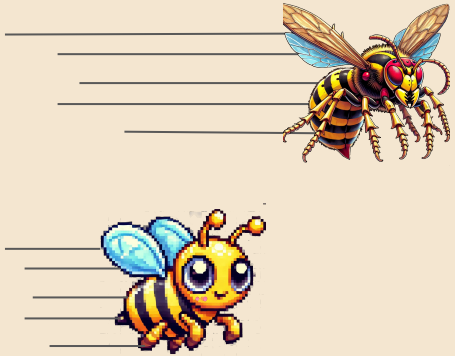
Modélisation des agents

Déplacement des agents

- 8 orientations

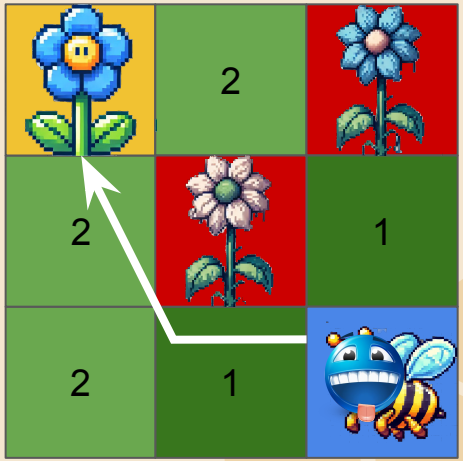


- Vitesse



- Objectif

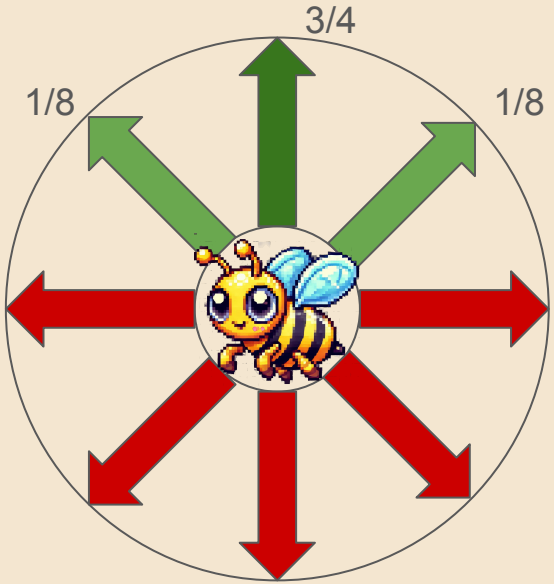
Ex : Vitesse = 2



Modélisation des agents

Déplacement des agents

- Aucun objectif …



Ex : orientation Nord

- … mais trop au bord

Ex : Vitesse = 2

LIMITE			
	NO	1	2
		1	2
	NO	1	2

- … ou chemin obstrué

Ex : Vitesse = 2

Modélisation des agents

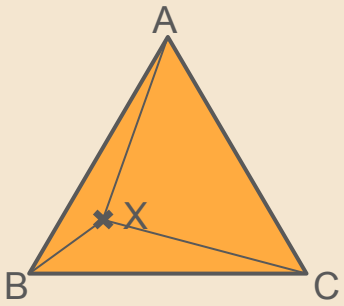
Vision des agents

- Un triangle à partir de 4 données
  - Orientation de l'agent
  - Hauteur du triangle
  - Longueur de la base de mesure
  - Coordonnée du point opposé à la base

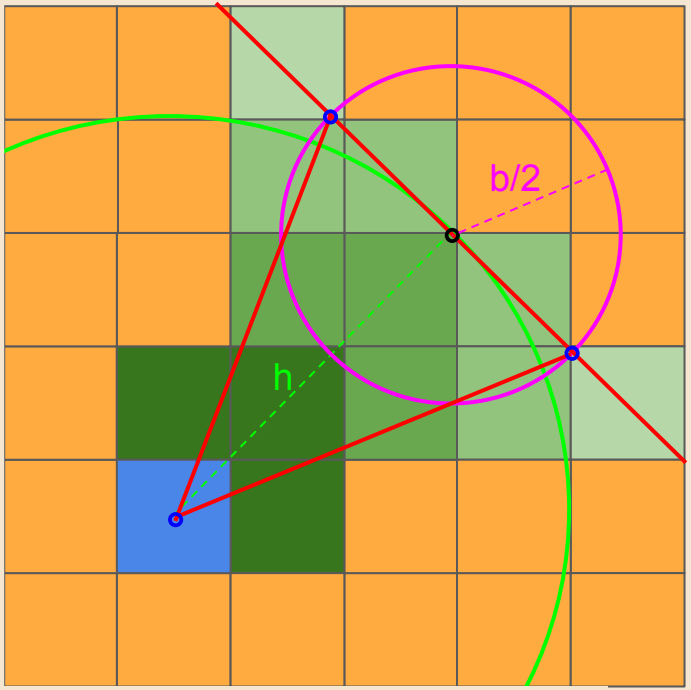
- Calculer si un élément est dans le triangle

Si Aire(ABX)+Aire(ACX)+Aire(BCX) = Aire(ABC)

- Alors X ∈ ABC
- Sinon X ∉ ABC



Ex : orientation Nord-Est (45°)
hauteur(h) = 3 et base (b) = 2



Modélisation des agents

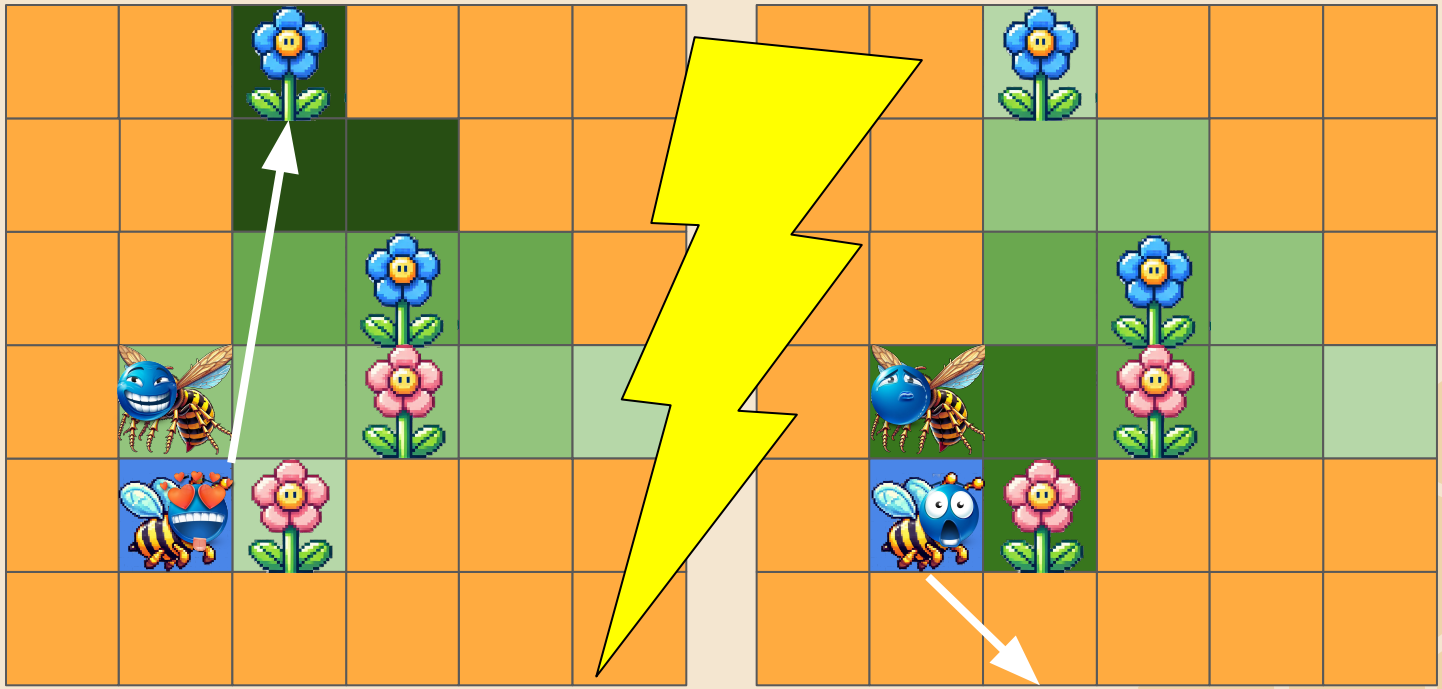
Vision des agents

- Une liste triée par proximité
  - Permet de prioriser les objectifs

Ex : On parcours les cases ligne par ligne

VS

On parcourt les cases de la plus proche à la plus éloignée



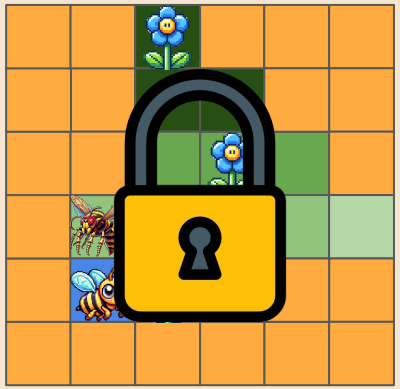
Modélisation des agents

Communication entre abeilles

- Communication des abeilles au sein de la ruche
  - Un pile (LIFO ou FILO au choix)
  - Permet de préserver la “fraîcheur” de l’information
  - Missionne une autre abeille pour communiquer son information

Modélisation backend

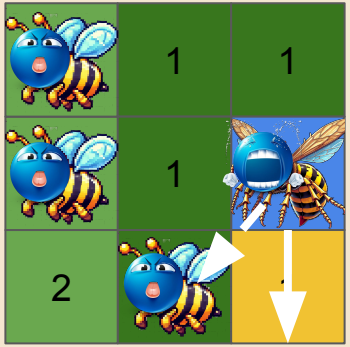
- Gestion de l’asynchrone et de la concurrence
  - Un agent = une goroutine & un channel de synchronisation géré par la simulation
  - **La perception, délibération & action peut être interrompue par un autre agent**
    - Environnement = Mutex
    - Perception ? Verrouillage de l’environnement.
    - Délibération ? Verrouillage de l’environnement.
    - Action ???? Verrouillage 👏 de 👏 l’environnement. 👏



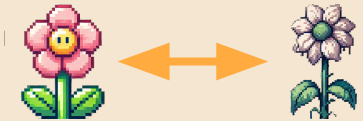
- Utilisation d’un fichier de configuration yaml
  - Dimensions de l’environnement
  - Distance de perception des agents
  - Nombre d’éléments a créer
  - …

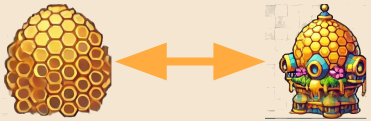
Modélisation backend

- Éviter les mauvaises manipulations avec les positions
  - Opération de prévisions sur une position : copie défensive
  - Positions protégées contre les erreurs, limite les bugs et anomalies au cours de la simulation



- Mise à jour des objets
  - Fonction Update() appelée après avoir réveillé tous les agents une fois





Modélisation front

- Décomposition des éléments d’affichage
  - Affichage des ressources : Zones pour afficher le nectar et le miel 🍯
  - Tableau des frelons : Affiche les frelons et leur "kill count" 💥
  - Zone principale : Contient un canevas géré par Pixi.js pour le rendu graphique
  - Communication avec le serveur : Boutons interactif pour communiquer

- Communication en WebSocket
  - Canal bidirectionnel sans une réouverture de connexion constante
  - Temps réel et efficacité ⚡

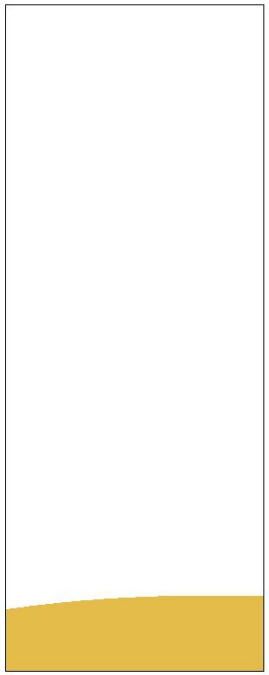
- Mise à jour des objets
  - Fonction onmessage() reçoit le Json qui nous est envoyé depuis le back via le WebSocket
  - On itère sur les agents et les objets reçu, si son **ID** est inconnu = **Nouveau agents,** sinon on **Update**

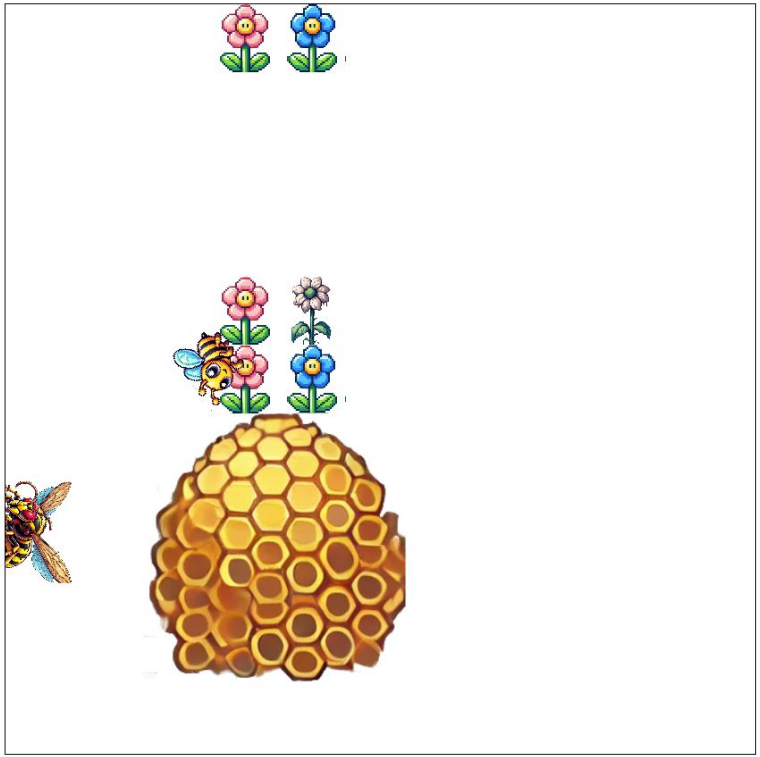
- Rendu avec Pixi.js
  - Bibliothèque légère pour les rendu 2D et rapide

Démo

Nectar: 174

Honey: 42



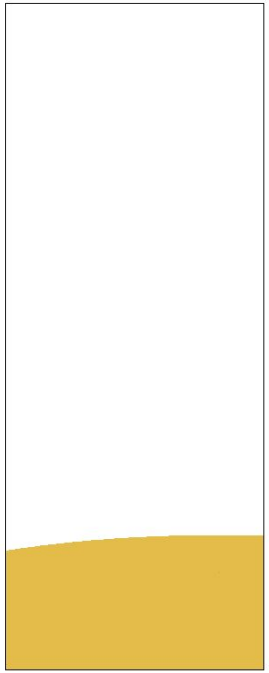


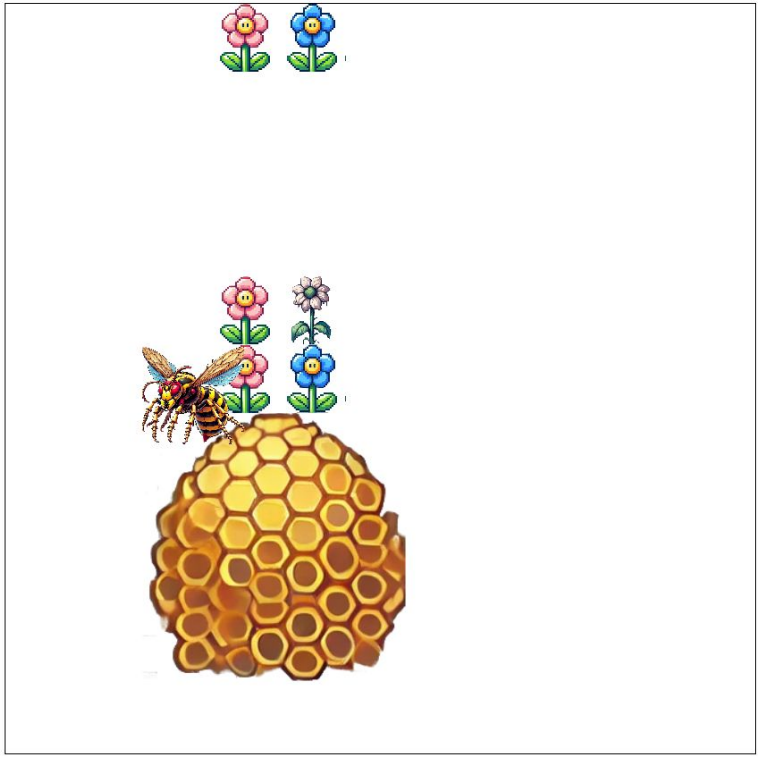
Hornet ID	Kill Count
Hornet #0	1

Démo

Nectar: 75

Honey: 101





Hornet ID	Kill Count
Hornet #0	2

Démo

Nombre d’abeilles : 10
Taille de la carte: 25
Fleurs: 10
Frelons: 1
Production des fleurs: 3

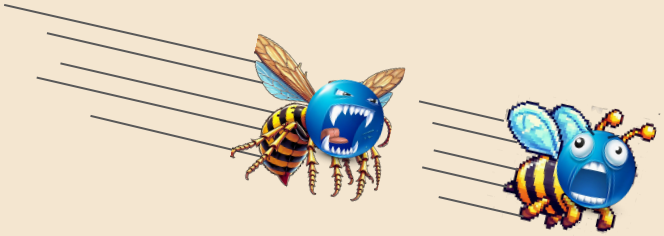
lien

Nombre d’abeilles : 10
Taille de la carte: 30
Fleurs: 10
Frelons: 3
Production des fleurs: 1

lien

Conclusion

1. Forces de la simulation.
    - Communication.
    - Temps réel.
    - Ajustable.
    - Extensible.
2. Retour critique.
    - Très sensible à la configuration initiale (nombre d'agents, emplacement de départ, quantité de nectar, etc.)
    - Importance de l'aléatoire.
    - Websockets transmettant beaucoup de données (toutes, vérification post envoi).
3. Améliorations ou adaptation.



Merci pour votre attention

Avez-vous des questions ?