Plan de développement

Projet L3D2 - Systèmes proies-prédateurs

Année 2022-2023

***Les informations d’identification du document***

| Référence du document: | PD\_L3D2\_v2 |
| --- | --- |
| Version du document: | 2 |
| Date du document: | 28/02/2023 |
| Auteurs: | Fu Suntanqing  Rossignol Paul  Xu Remy  Ye Frédéric |

***Les éléments de vérification du document***

| En attente de validation: | LATTAUD Claude |
| --- | --- |
| Soumis le: | 02/03/23 |
| Validé le: | \_\_/\_\_/23 |
| Type de diffusion: | Document électronique (.pdf) |
| Confidentialité: | Réservé au jury, professeurs encadrants et aux étudiants de l’Université Paris Cité. |

**SOMMAIRE**

[**I. Introduction 4**](#_1t6ubza3m5sg)

[A-Objectifs et méthodes 4](#_bxy4yogkclym)

[B-Documents de référence 4](#_o1kjsmguzcfm)

[**II. Guide de lecture 4**](#_vvxsrbe2g7jj)

[A-Maîtrise d’oeuvre 4](#_19vsstnbtv60)

[B-Maîtrise d’ouvrage 5](#_f98umtwlh41h)

[**III. Concepts de base 5**](#_nox9jijdmfrr)

[**IV. Organisation 6**](#_ri2s7qtz4rmm)

[A-Décomposition en tâches 6](#_ohjci8y0nic1)

[B-Structure de l’équipe 6](#_yrtsrf7hqtd)

[**V. Planification 7**](#_voky31qic9az)

[**VI. Cycle de vie 8**](#_k8k6huiwjdgh)

[**VII. Méthodes et outils 9**](#_exoylcc2sbun)

[**VIII. Conception générale 10**](#_64h0uumv45vj)

[A-Diagrammes de cas d’utilisation 10](#_p78azlp2ccct)

[1-Contrôle d’un agent 10](#_looyf3yute4p)

[2-Mimétisme. 11](#_nzkpxiy3n7fw)

[3-Mimétisme, en utilisant le cas de mimétisme, stocké dans la base de données 12](#_6dk15rou39oy)

[4-Changement du comportement de l’agent en fonction de ses besoins 13](#_qx4zgasjrqra)

[5-Système de peur 14](#_hmm6r3fiw35b)

[6-Système de saison 15](#_lvq9r0go0m8k)

[7-Interaction avec l’eau 16](#_4zq4443gtdia)

[8-Système de météo et ses impacts sur les agents. 17](#_5gg2p9afxvqy)

[9-Modification des traits de l’agent en fonction de ses besoins. 18](#_7cfyhmae3jil)

[B-Diagrammes de Séquence : 19](#_hng46scvsfzo)

[1-Mimétisme. 19](#_2xv5y1xbxywt)

[2-Mimétisme, en utilisant le cas de mimétisme, stocké dans la base de données 20](#_4y7c0l3o4hp5)

[3-Interaction avec l’eau 21](#_n996osesby80)

[4-Changement du comportement de l’agent en fonction de ses besoins 21](#_1k7ytwt19hc6)

[5-Modification des traits 22](#_pt7h2dnm93ly)

[6-Contrôle d’agent 23](#_i3j1dmqfzjuw)

[C-Diagramme de classe 24](#_5q4burxpdlky)

[D-Diagrammes d'état-transition 25](#_227mre6r72r1)

[1-Stockage des paramètres et des résultats dans la BDD 25](#_5kqwb3hx8s7l)

[2-Gestion du contrôle d’un agent par l’utilisateur 26](#_2bixza2uhj84)

[3-Gestion des saisons et des différentes météos 27](#_s03dy5uvxati)

[E-Diagramme de packages 28](#_fd42we6qs9vw)

[**IX. Conception détaillée 29**](#_k4nga28m8flc)

[A-Introduction 29](#_4n4mdyph7b53)

[B-Liste des classes implémentées ou modifiées 29](#_cud2kd9kpqq6)

[1- Agent 29](#_qod8p9pei2fx)

[2- Prise de contrôle 32](#_qn0htljyt89i)

[3- Mimétisme 32](#_p5nhniifkjnf)

[4- Interaction avec l’eau 33](#_q09utb1yf93u)

[5 - Monde 33](#_3549kzbm7j44)

[6 - Météo possible 34](#_mz7343431qw3)

[7 - Météo 34](#_3n40amvnerle)

[**X. Références 35**](#_3e1yypxgeciv)

[A-Tableau des figures 35](#_l87pvmrihnul)

[B-Bibliographie 36](#_9i77npm5cll3)

## I. Introduction

### A-Objectifs et méthodes

Dans la conception générale nous retrouverons les modélisations des différentes parties de notre application. Il y aura la description de grands domaines formels, l’architecture de l’application et les interactions avec les éléments internes et externes au logiciel. Ce document listera les packages, les modules, les interfaces, les classes que l’on implémente. La planification et nos méthodes de travail seront expliqués. De plus, nous décrivons dans ce document l’ensemble des outils nous aidant à développer notre logiciel.

### B-Documents de référence

Nous recommandons la lecture de notre cahier des charges et de recette.

## II. Guide de lecture

### A-Maîtrise d’oeuvre

Il est recommandé de consulter l’ensemble des sections de ce plan de développement, plus spécifiquement l’[organisation](#_ri2s7qtz4rmm), la [planification](#_voky31qic9az), le [cycle de vie](#_mehzf4o7ig8a) et les [méthodes et outils](#_exoylcc2sbun).

### B-Maîtrise d’ouvrage

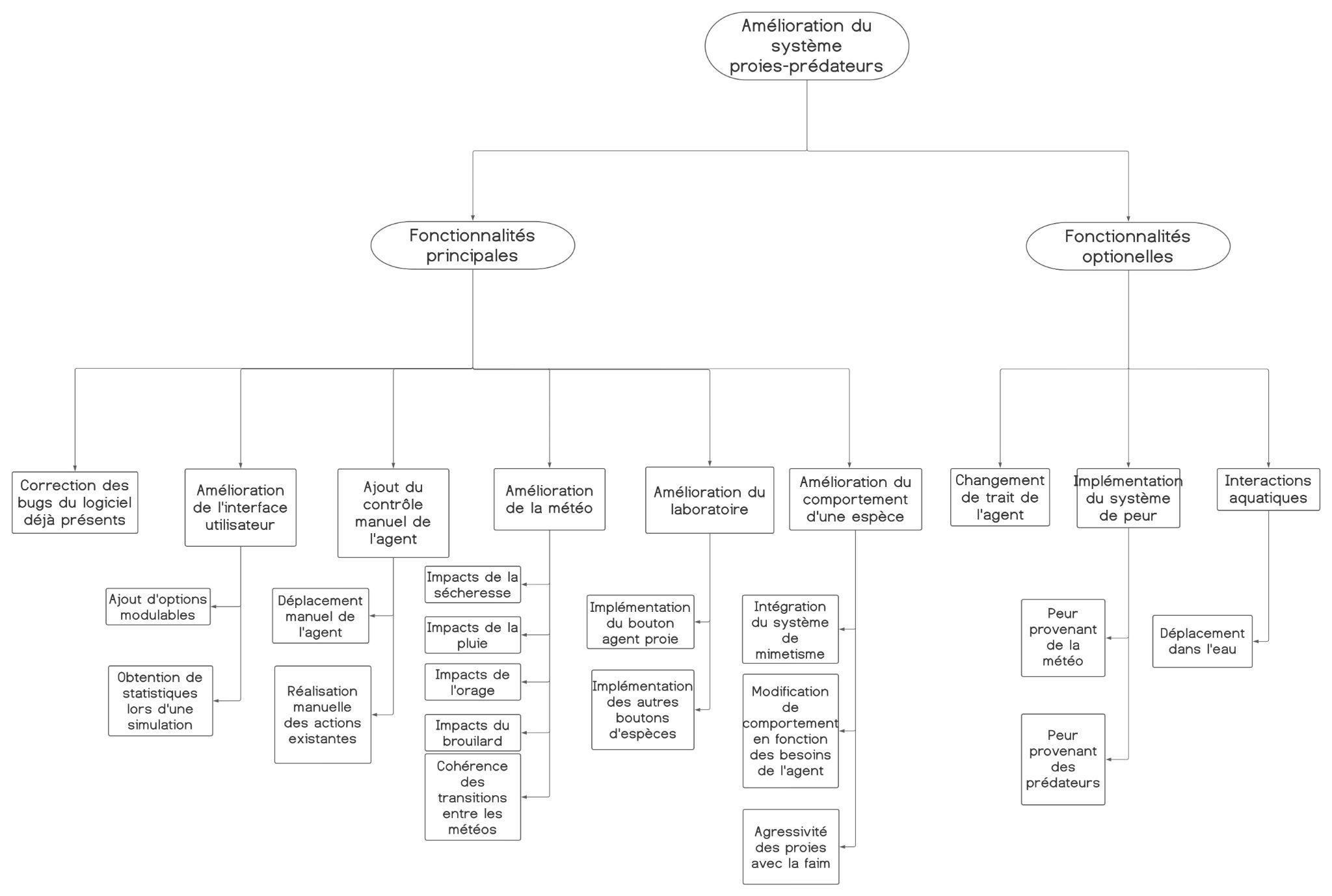
Il est recommandé de consulter l’ensemble des sections de ce plan de développement, plus spécifiquement la [conception générale](#_64h0uumv45vj) et [détaillée](#_k4nga28m8flc).

## III. Concepts de base

Pour la compréhension des documents, il est recommandé d’avoir des notions de génie logiciel pour comprendre complètement les diagrammes. De plus, tout comme le cahier des charges et le cahier de recette, il est recommandé de lire des articles sur l'équilibre proies-prédateurs et sur la prédation (voir la bibliographie dans [X.Références](#_3e1yypxgeciv)).

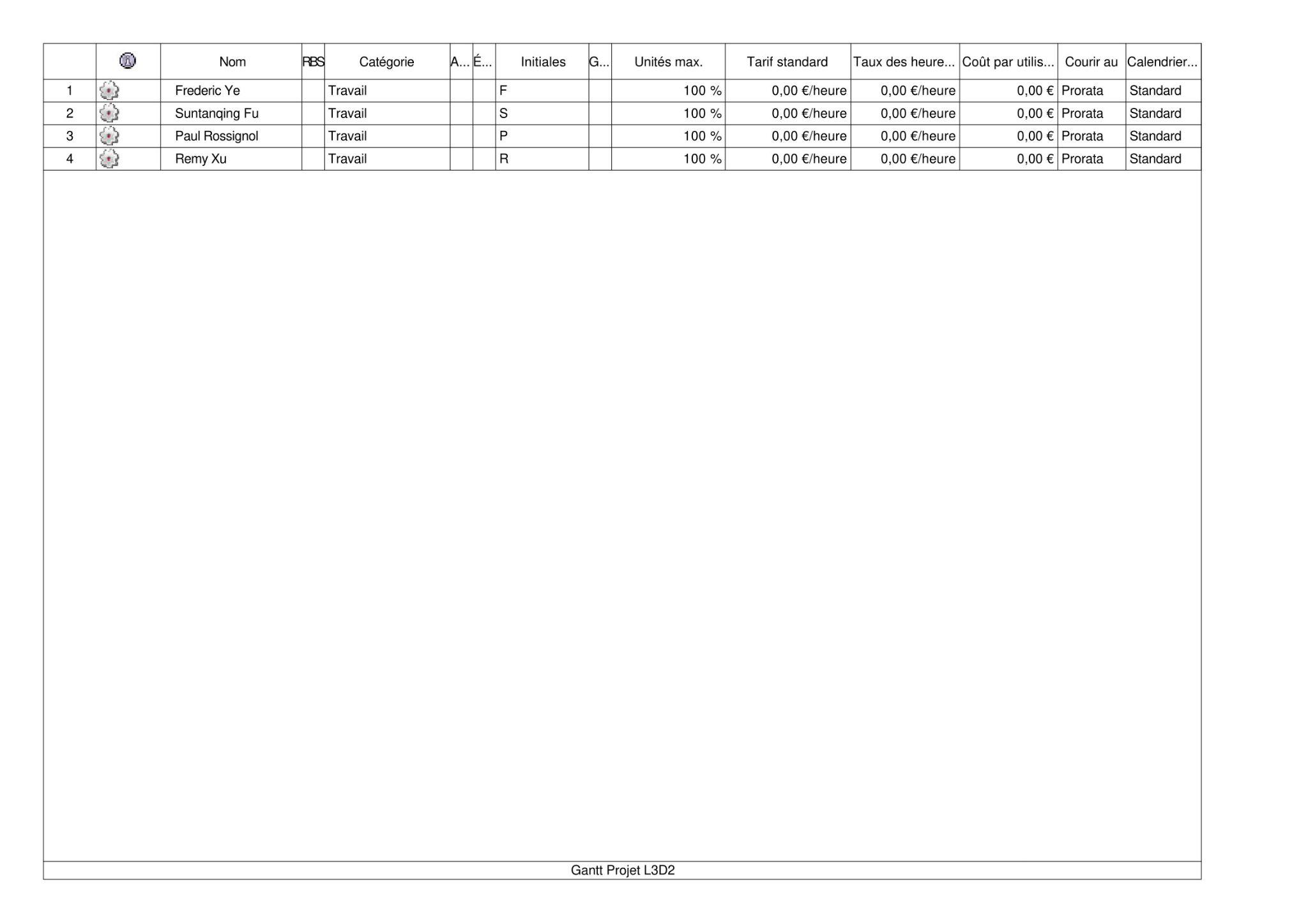
## IV. Organisation

### A-Décomposition en tâches

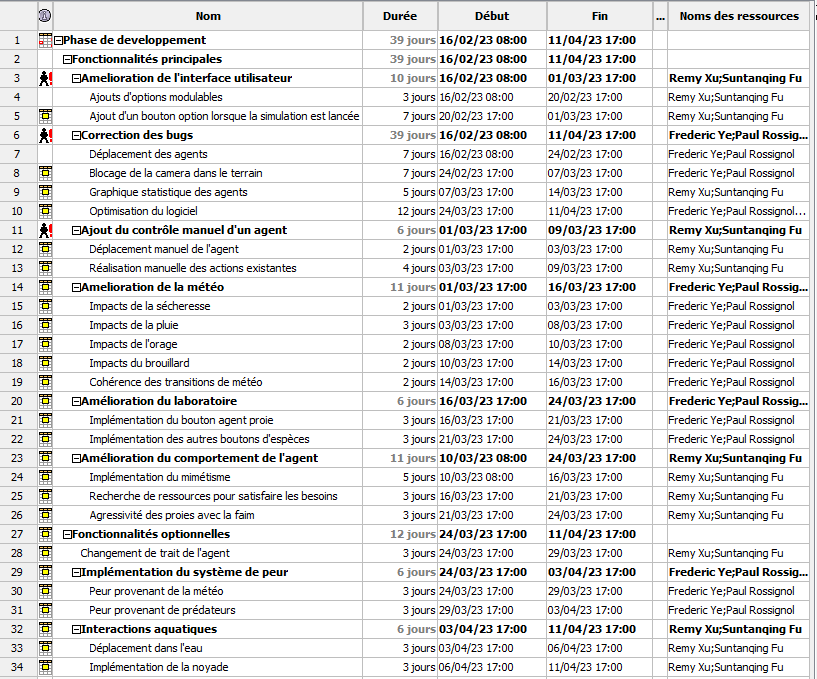
*Figure 1: Arbre WBS du projet*

### B-Structure de l’équipe

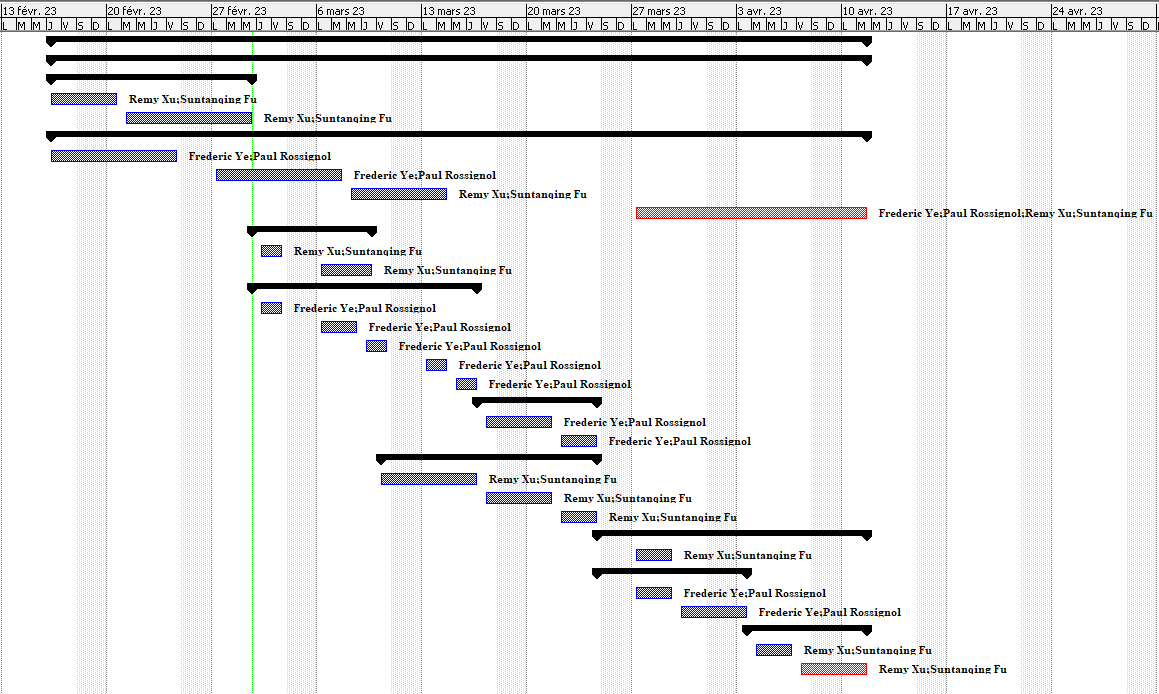
Notre équipe est composée de 4 développeurs, travaillant sur les différentes tâches à réaliser.

*Figure 2: Ressources de l’équipe du diagramme de Gantt*

## V. Planification

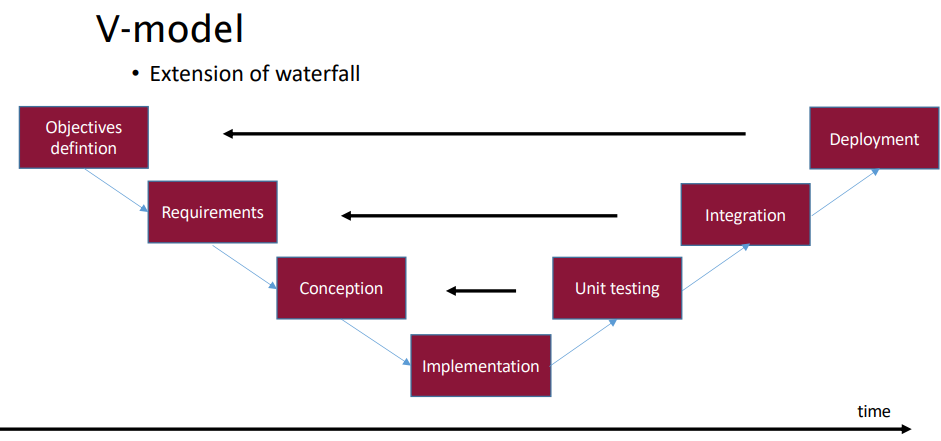
Diagramme de Gantt représentant les différentes tâches réalisées par qui, pour quelle durée en fonction du temps:

*Figure 3: Tableau du diagramme de Gantt*

*Figure 4: Graphique du diagramme de Gantt*

## VI. Cycle de vie

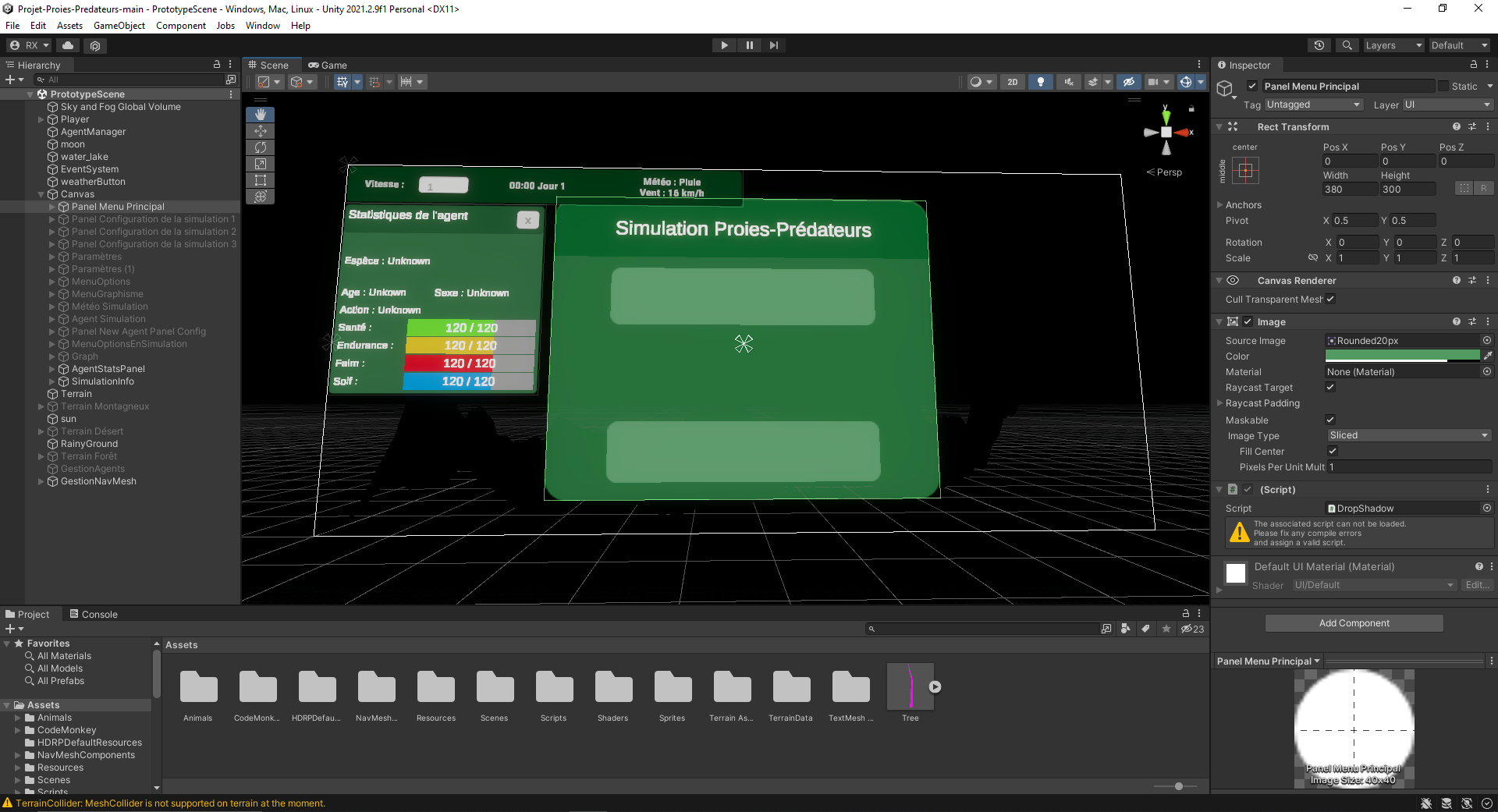
Pour notre projet, nous allons utiliser un cycle de vie en modèle V, dû à notre manque d’expérience en tant qu’étudiants. On sera amenés à régulièrement faire des modifications ultérieures aux rendus conseillés des documents. Cela nous permettra d’affiner les différentes étapes du projet.



*Figure 5: Capture d’écran du cours de génie logiciel, cycle de vie modèle en V*

## VII. Méthodes et outils

Nous allons améliorer le système proies-prédateurs, en développant sur Unity 3D, comme ce projet dans les années précédentes. Il s’agira de la version 2021.2.9f1. Il est très important de développer sous cette version, car il s’agit de la version de l’année précédente. De plus, certaines fonctionnalités rencontreront des problèmes tels que l’immobilité des agents et le non-affichage des statistiques des agents.



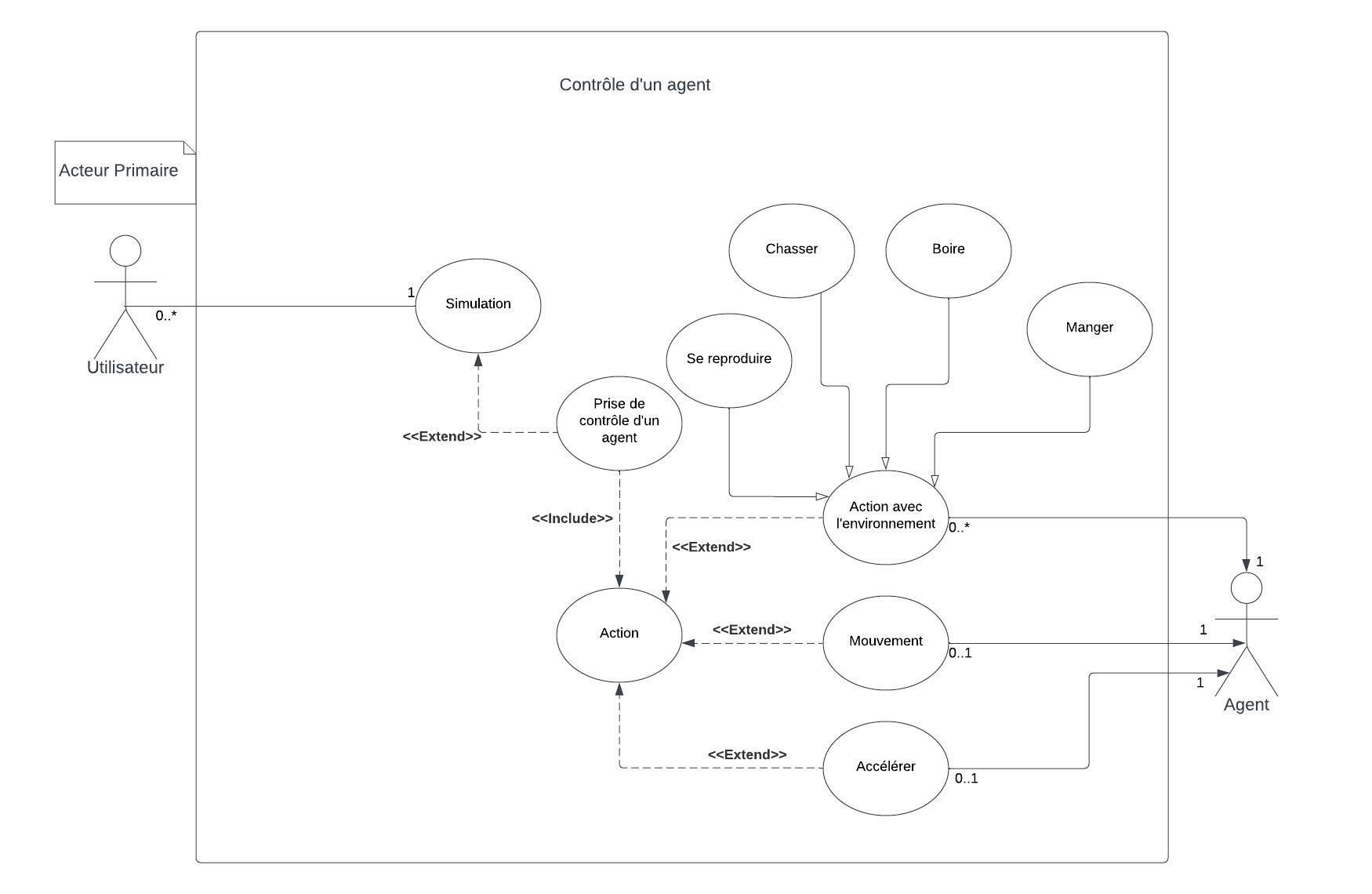
*Figure 6: Capture d’écran de Unity en version 2021.2.9f1*

## VIII. Conception générale

### A-Diagrammes de cas d’utilisation

#### 1-Contrôle d’un agent

Un agent peut être pris en contrôle manuellement par l’utilisateur s’il le souhaite. Il peut soit le contrôler avec les commandes qu’il a configurées lui-même ou avec les commandes de base. Il pourra lui faire effectuer des actions comme boire, manger, attaquer une proie. Il pourra aussi accélérer en consommant plus rapidement son endurance d’un certain pourcentage.

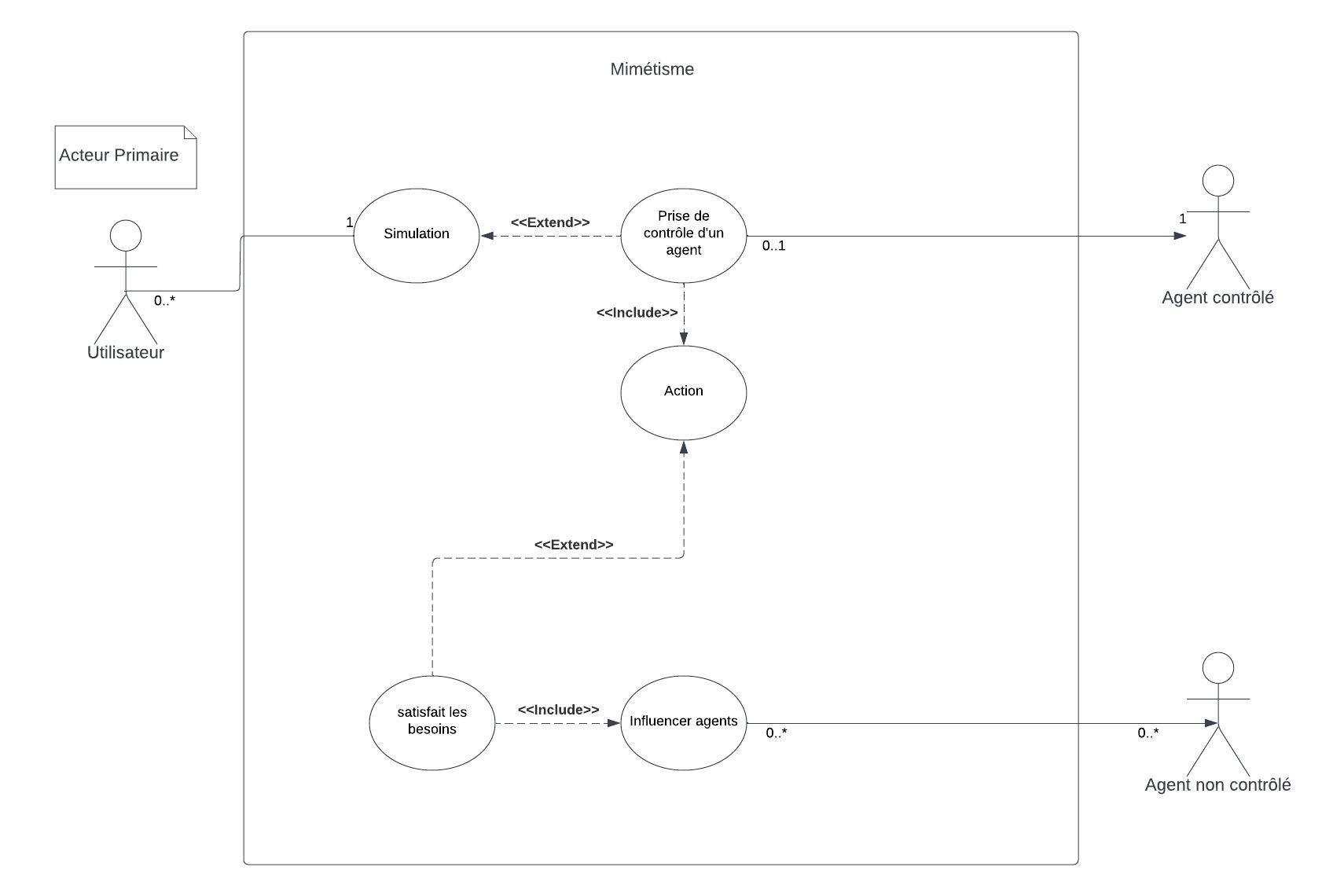


*Figure 7: Diagramme de cas, contrôle d’un agent*

Un agent peut être pris en contrôle, mais il doit aussi agir de manière autonome. On lui ajoute des nouvelles fonctionnalités pour qu’il soit le plus réaliste possible. Les quatre diagrammes suivants seront à ce sujet-là.

#### 2-Mimétisme.

Le mimétisme permet à un agent autonome de pouvoir copier les actions de l’agent contrôlé. Il copiera seulement s’il estime que l’action aura un impact positif sur sa survie.



*Figure 8: Diagramme de cas, mimétisme*

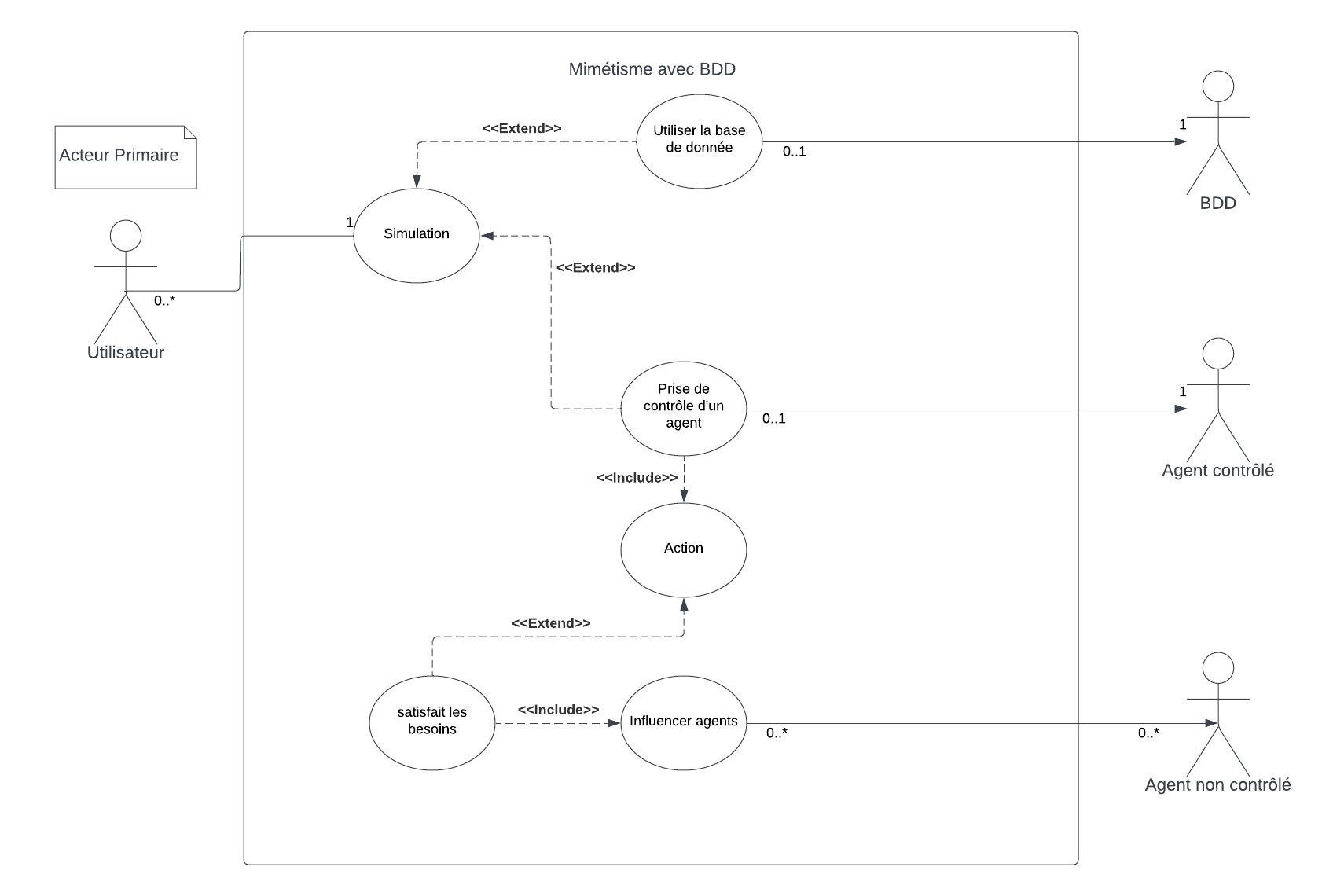
#### 

#### 

#### 

#### 3-Mimétisme, en utilisant le cas de mimétisme, stocké dans la base de données

L’utilisateur peut souhaiter enregistrer dans la base de données le mimétisme qui a été réalisé ou non.

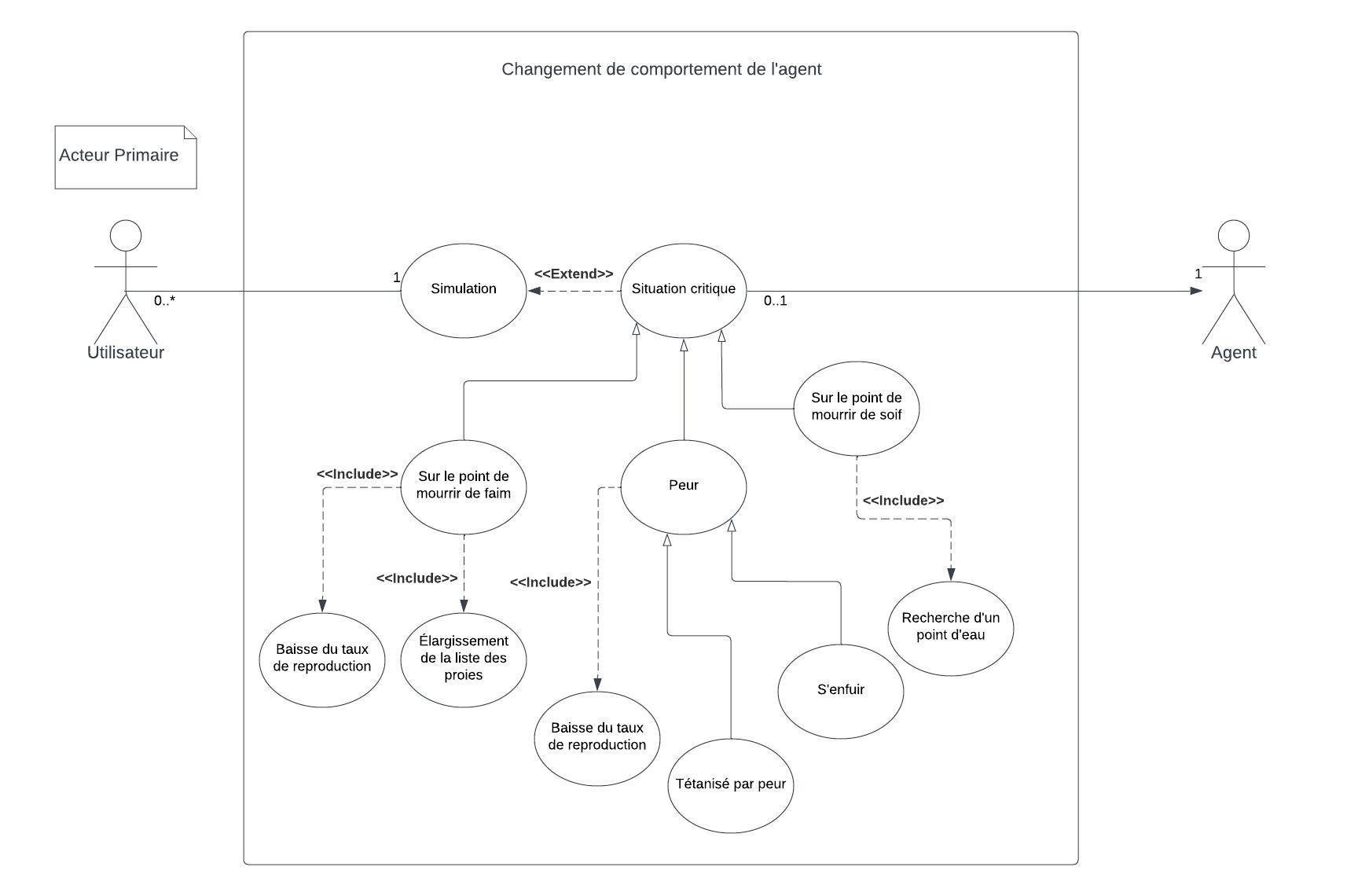


*Figure 9: Diagramme de cas, mimétisme stocké dans la base de données*

L’agent, mis à part imiter un congénère, peut aussi changer de comportement en fonction de ses besoins tout seul.

#### 4-Changement du comportement de l’agent en fonction de ses besoins

L’agent peut changer de comportement en fonction de ses besoins manquants. On lui imposera d’aller chercher sa proie (ou sa nourriture) lorsque ses besoins dépasseront un certain seuil, de fuir lorsque sa peur dépasse un certain seuil et de boire lorsque sa barre de soif dépasse un certain seuil.

*Figure 10: Diagramme de cas, mimétisme stocké dans la base de données*

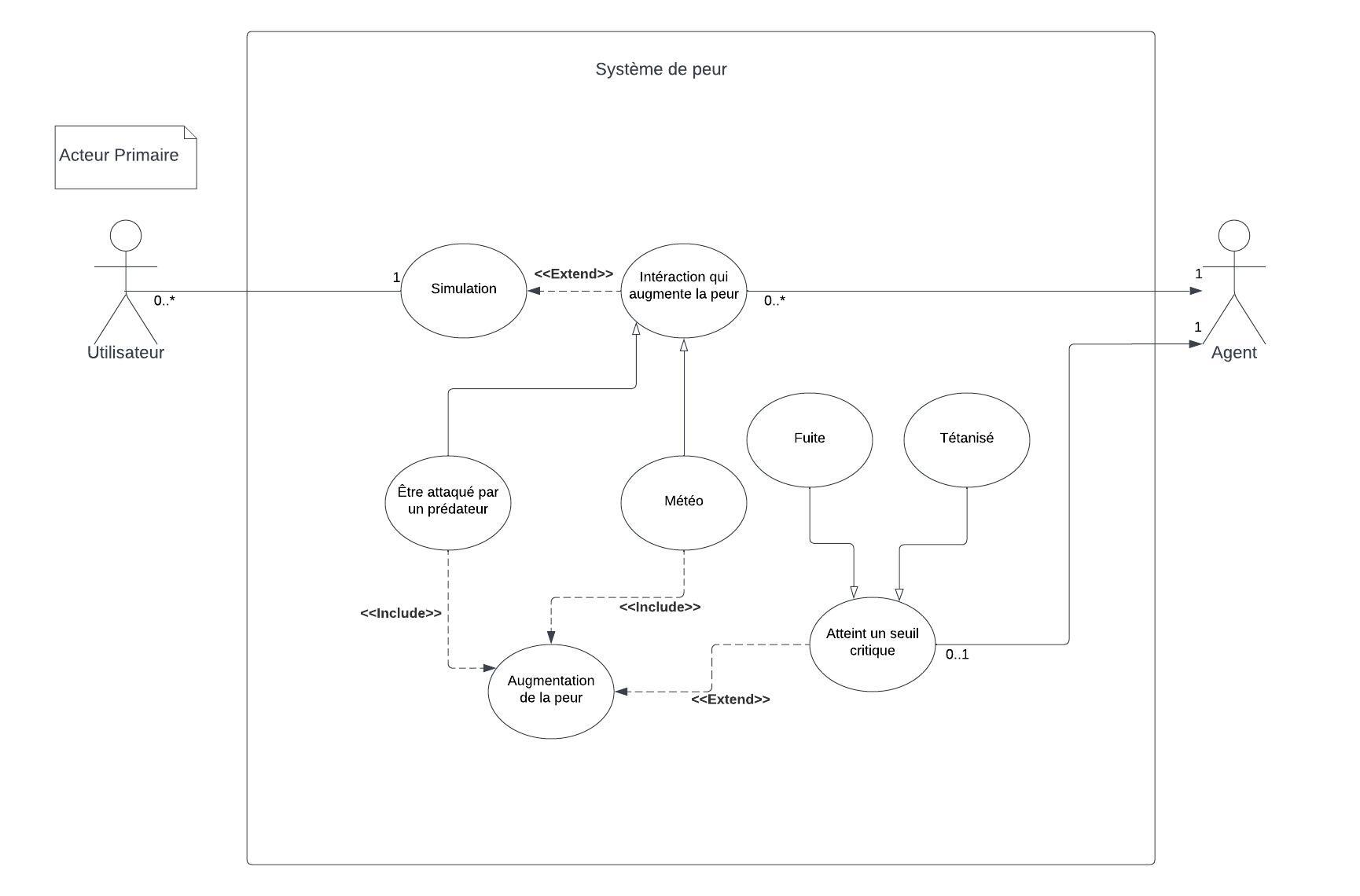
Concernant le système de peur, on précisera dans le prochain diagramme son fonctionnement.

#### 

#### 

#### 5-Système de peur

Un agent aura deux moyens d’augmenter sa barre de peur: par la météo, l’orage va provoquer la peur chez les proies. Sinon ce sera lorsqu’il aperçoit un prédateur ou lorsqu’il est attaqué par son prédateur.

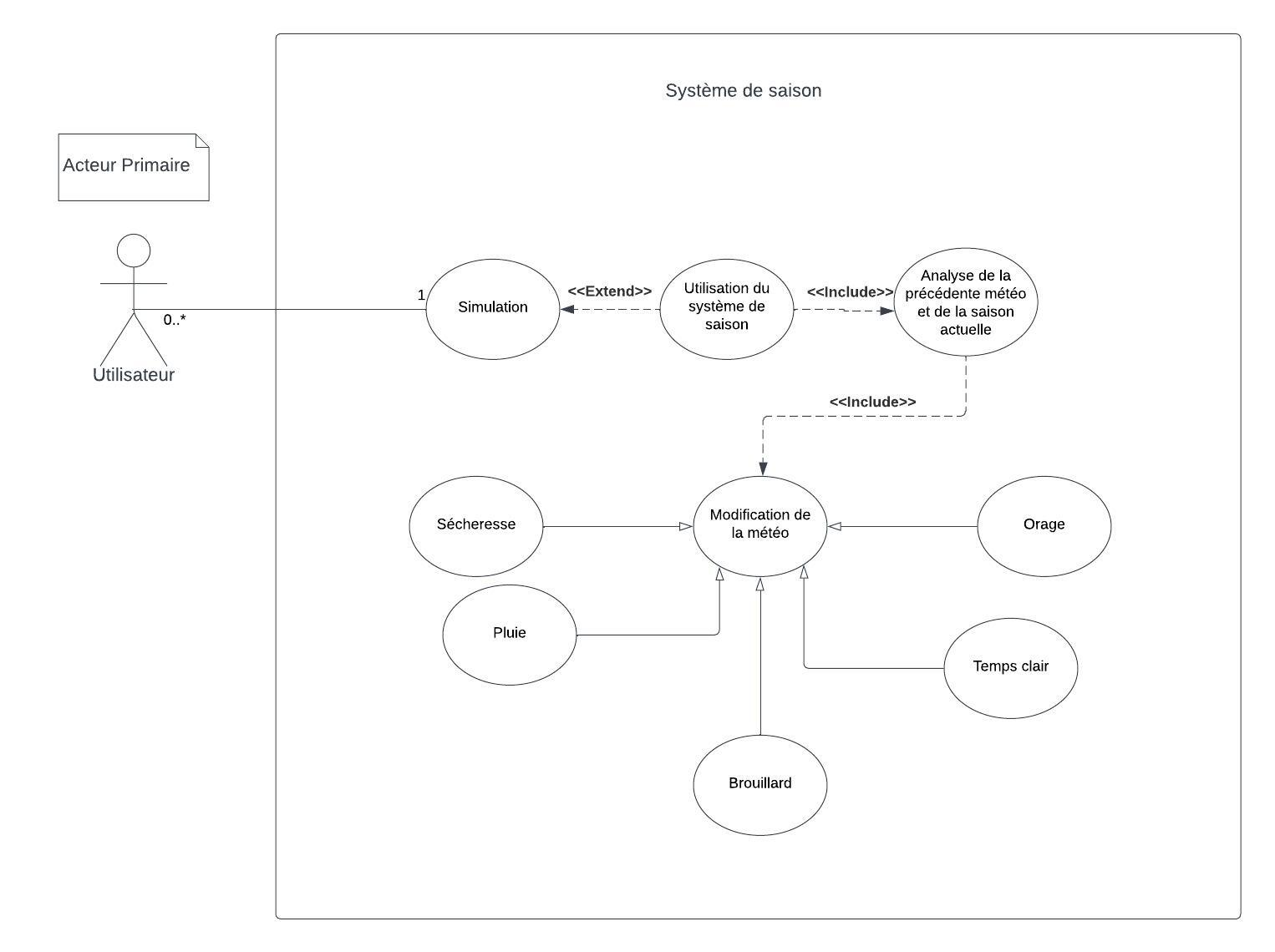


*Figure 11: Diagramme de cas, système de peur*

Après avoir vu les nouvelles fonctionnalités au niveau du comportement de l’agent. Nous verrons avec les trois diagrammes suivants la météo et les différentes interactions possibles du terrain avec les agents.

#### 6-Système de saison

Nous mettrons en place un système de cohérence entre les météos, il sera impossible de passer d’un extrême à un autre (exemple: sécheresse à orage ou tempête d’après le projet de l’an dernier). Les probabilités d’avoir certaines météos seront plus élevées ou plus basses en fonction des saisons, comme le reflète la réalité.



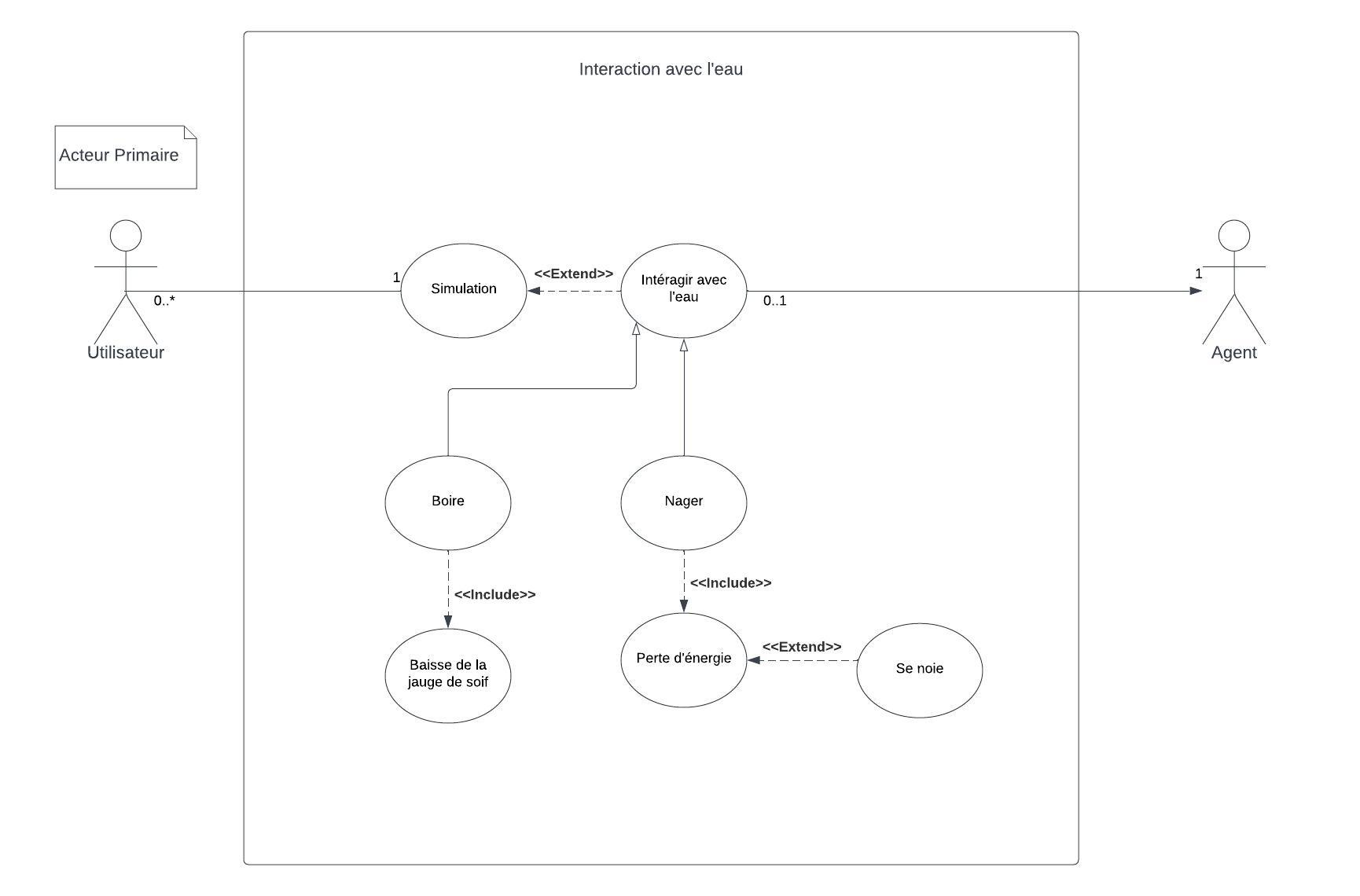
*Figure 12: Diagramme de cas, système de saison*

#### 

#### 

#### 7-Interaction avec l’eau

Les agents pourront interagir avec les sources d’eau disponibles dans le terrain. Dans notre terrain de base, ce sera principalement des lacs ou les agents pourront boire ou nager.

*Figure 13: Diagramme de cas, interaction avec l’eau*

#### 8-Système de météo et ses impacts sur les agents.

*Figure 14: Diagramme de cas, système de météo*

#### 

#### 

#### 

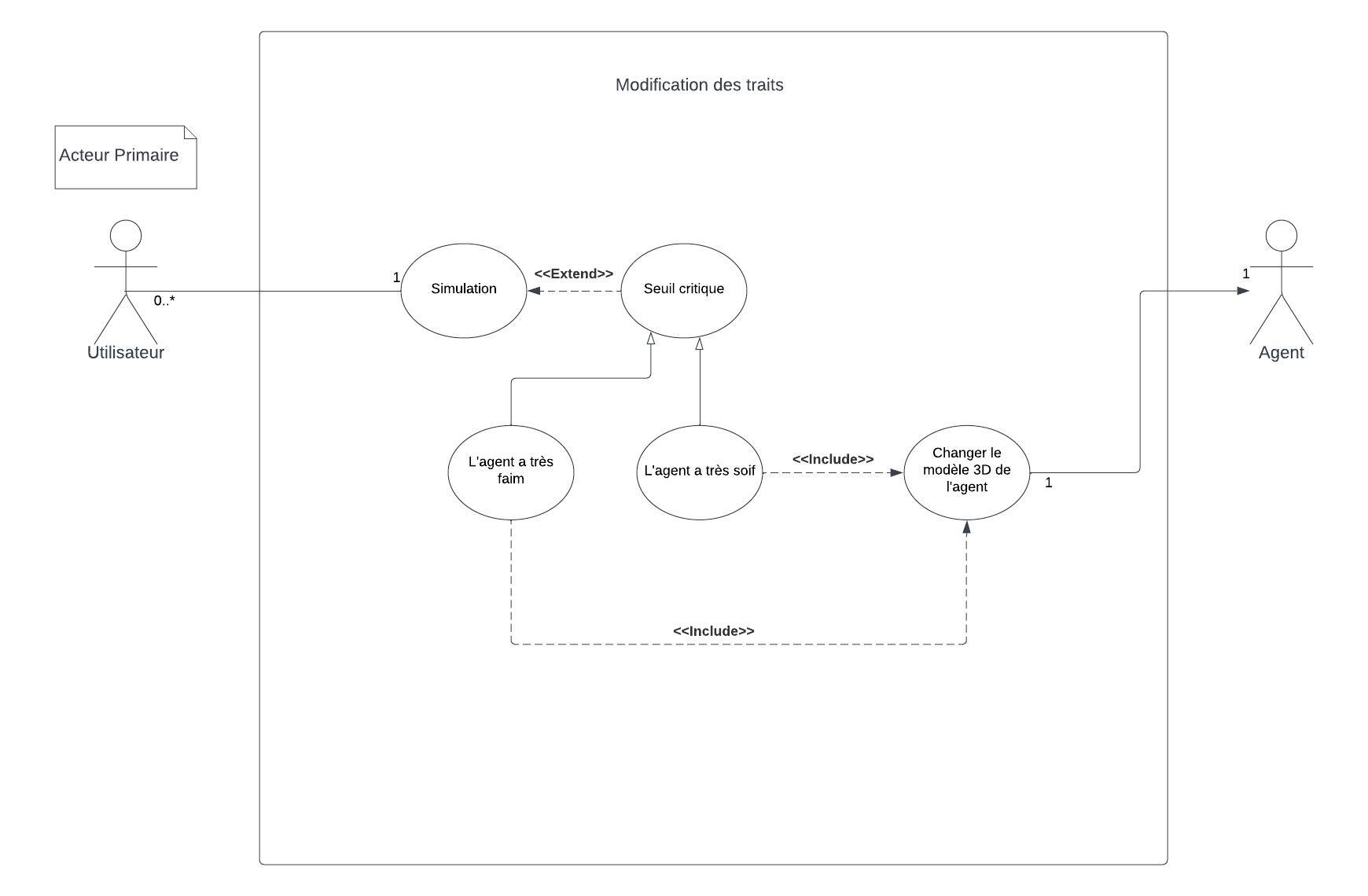
#### 

#### 

#### 

#### 9-Modification des traits de l’agent en fonction de ses besoins.

En fonction des agents, le seuil critique sera très différent. Nous mettrons en place éventuellement une variation de ce seuil critique pour une même espèce en fonction des saisons. Ensuite, nous changerons le modèle 3D de l’agent.



*Figure 15: Diagramme de cas, modification des traits*

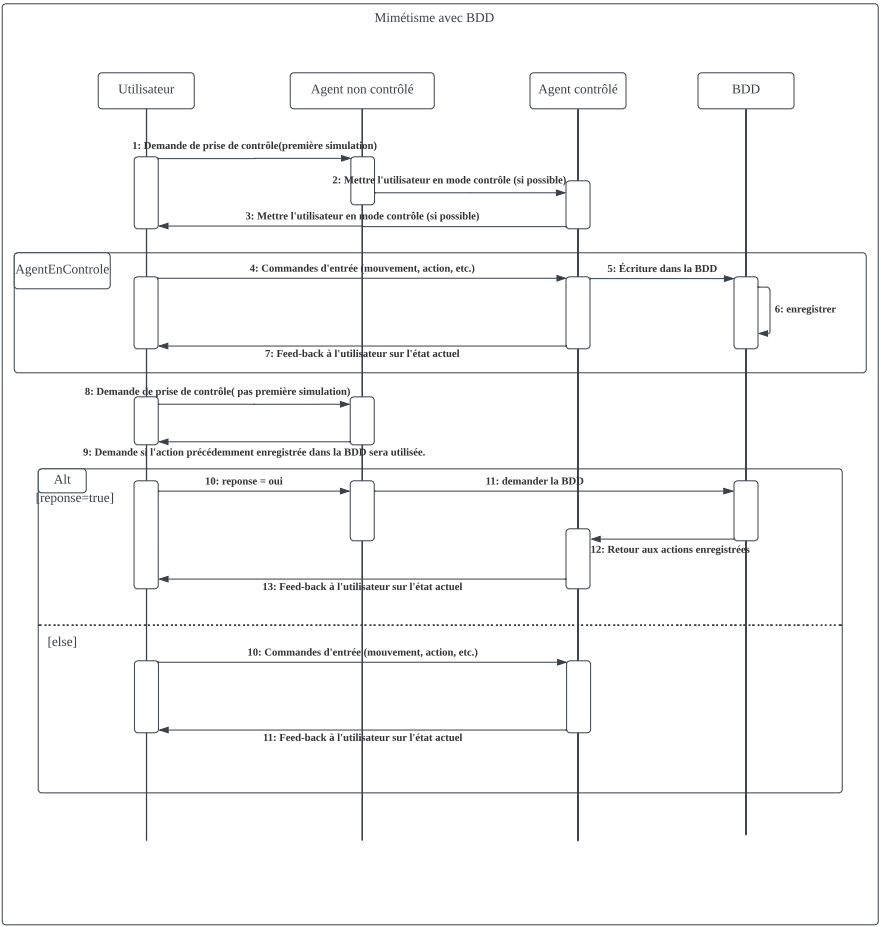
### B-Diagrammes de Séquence :

#### 1-Mimétisme.

#### 

*Figure 16: Diagramme de séquence, mimétisme*

#### 2-Mimétisme, en utilisant le cas de mimétisme, stocké dans la base de données



*Figure 17: Diagramme de séquence, mimétisme stocké dans la base de données*

#### 3-Interaction avec l’eau

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

*Figure 18: Diagramme de séquence, interaction avec l’eau*

#### 4-Changement du comportement de l’agent en fonction de ses besoins

*Figure 19: Diagramme de séquence, comportement de l’agent en fonction de ses besoins*

#### 5-Modification des traits

*Figure 20: Diagramme de séquence, modification des traits*

#### 

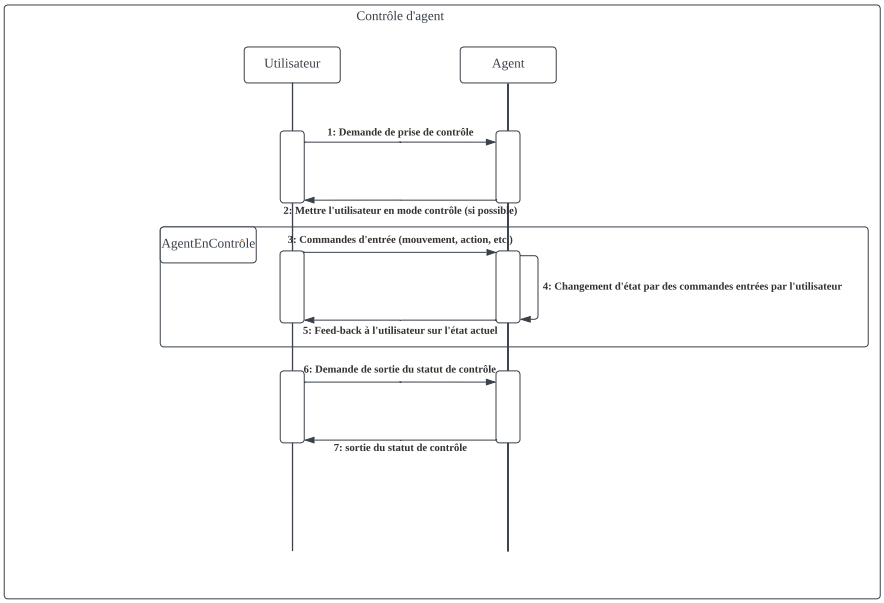
#### 

#### 

#### 

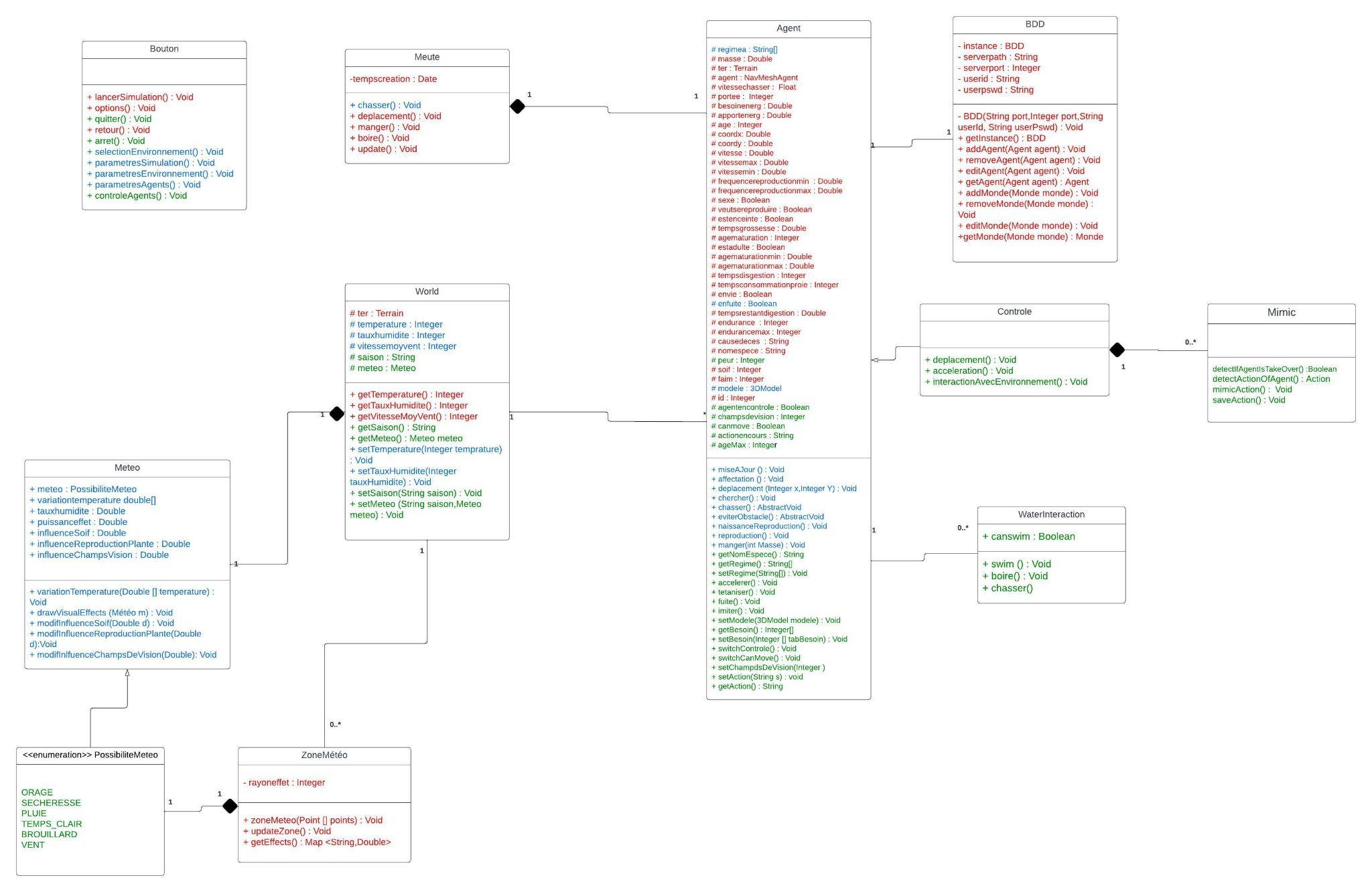
#### 

#### 6-Contrôle d’agent



*Figure 21: Diagramme de séquence, contrôle d’agent*

### C-Diagramme de classe



*Figure 22: Diagramme de classe*

*Les attributs et*

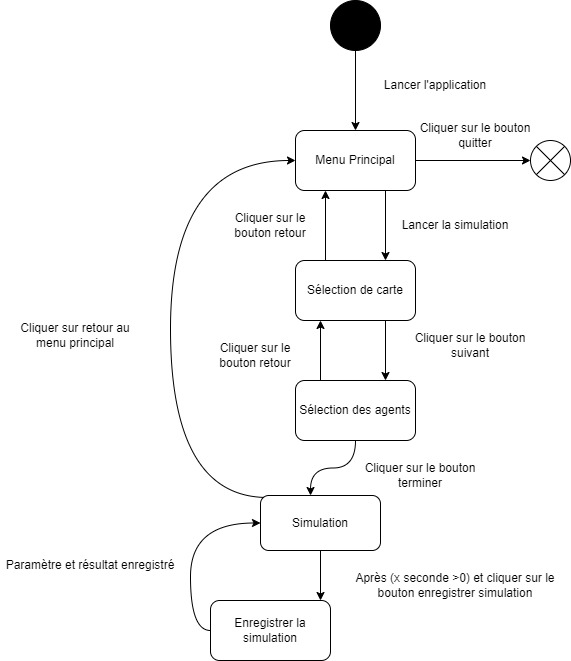
*Légende : — : Code créé par notre groupe*

*— : Modifié par rapport au groupe des années précédentes*

*— : Inchangé par rapport au groupe des années précédentes*

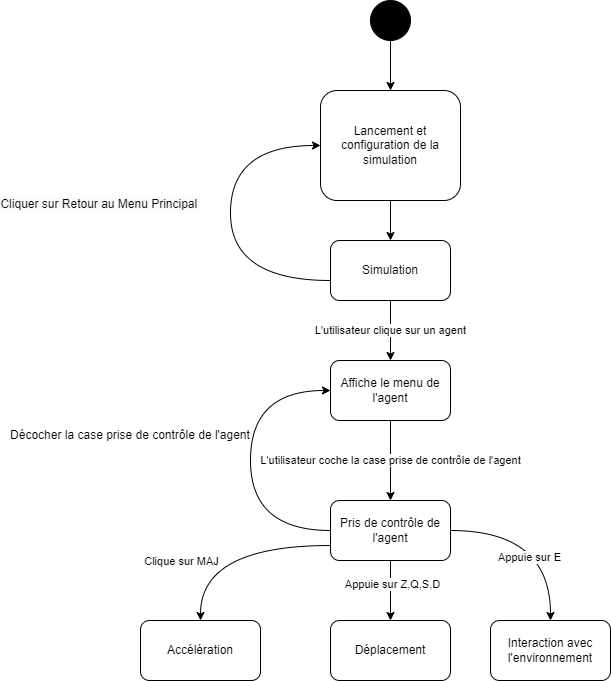
### D-Diagrammes d'état-transition

#### 1-Stockage des paramètres et des résultats dans la BDD



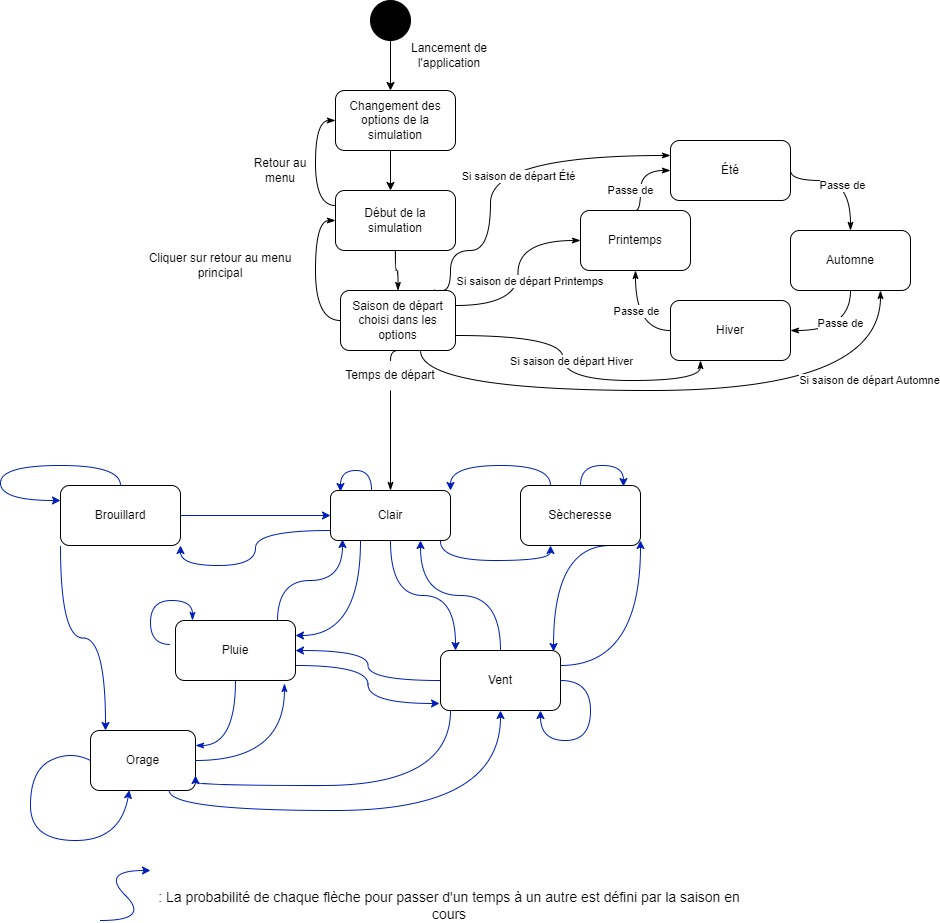
*Figure 23: Diagramme d'état- transition, stockage des paramètres et des résultats dans la BDD*

#### 2-Gestion du contrôle d’un agent par l’utilisateur



*Figure 24: Diagramme d'état- transition, gestion du contrôle d’un agent par l’utilisateur*

#### 3-Gestion des saisons et des différentes météos

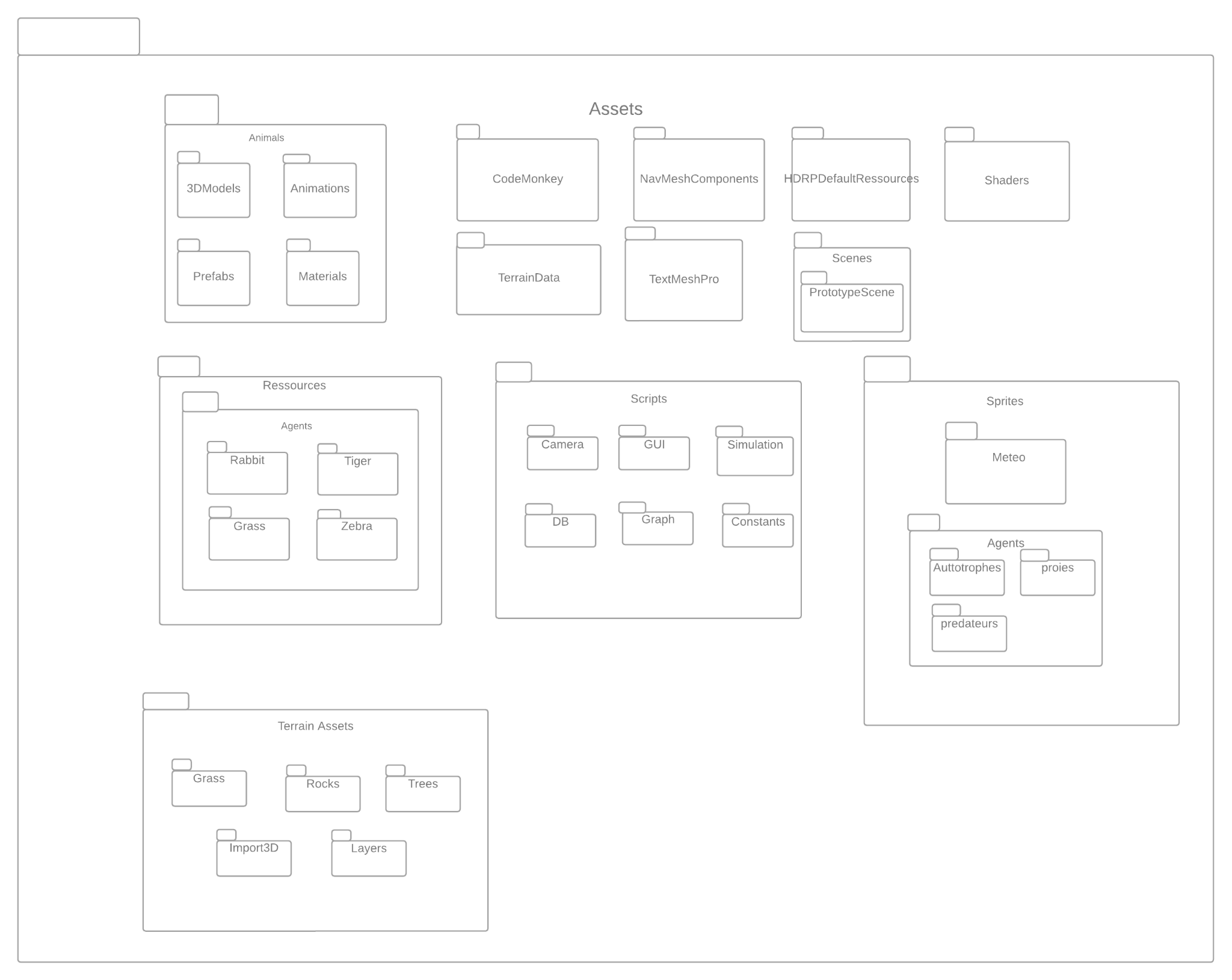


*Figure 25: Diagramme d'état- transition, gestion des saisons et des différentes météos*

#### 

### E-Diagramme de packages

Le diagramme de package représentent les packages du fichier “Assets” reçus tels quels du Github de ce projet de l’année précédente avec leurs contenus principaux.

*Figure 26: Diagramme de packages*

## IX. Conception détaillée

### A-Introduction

La conception détaillée présente les classes, attributs et méthodes qui sont nécessaires au bon développement de l’application. En effet, elle permet de faciliter et garantir la bonne implémentation de toutes les composantes de l’application aux développeurs. Il suffira alors aux développeurs de traduire les classes en code.

### B-Liste des classes implémentées ou modifiées

#### 1- Agent

class Agent : Permet de gérer l’ensemble des agents et leurs comportements.

Attributs :

regimea : String[]

Liste de l’ensemble des proies possibles de l’agent.

masse : Double

ter : Terrain

Désigne sur quel type de terrain se trouve l’agent.

agent : NavMeshAgent

Désigne le type de l’agent (loup, tigre...).

vitessechasser : Float

portee : Integer

besoinenerg : Double

apportenerg : Double

age : Integer

coordx: Double

coordy : Double

vitesse : Double

vitessemax : Double

vitessemin : Double

frequencereproductionmin : Double

frequencereproductionmax : Double

sexe : Boolean

veutsereproduire : Boolean

estenceinte : Boolean

tempsgrossesse : Double

agematuration : Integer

estadulte : Boolean

agematurationmin : Double

agematurationmax : Double

tempsdisgestion : Integer

tempsconsommationproie : Integer

envie : Boolean

enfuite : Boolean

tempsrestantdigestion : Double

endurance : Integer

endurancemax : Integer

causedeces : String

nomespece : String

peur : Integer

soif : Integer

faim : Integer

ageMax : Integer

modele : 3DModel

La représentation de l’agent dans la simulation.

id : Integer

agentencontrole : Boolean

Si agentEnControle est true l’agent est sous le contrôle de l’utilisateur, sinon il ne l’est pas.

champsdevision : Integer

canmove : Boolean

actionencours : String

Fonction :

miseAJour () : Void

Mise à jour des coordonnées de l’agent.

affectation () : Void

deplacement (int x,int y) : Void

Permet à l’agent de se déplacer d’un endroit à un autre.

chercher() : Void

Permet à l’agent de se déplacer afin d’avoir une interaction avec le monde.

chasser() : AbstractVoid

Permet à l’agent de chasser une proie.

eviterObstacle() : AbstractVoid

Permet à l’agent d’éviter un obstacle.

naissanceReproduction() : Void

Permet à l’agent de créer un agent.

reproduction() : Void

Permet à l’agent de se reproduire.

manger(int Masse) : Void

Permet à l’agent de manger.

getNomEspece() : String

Permet de récupérer le nom de l’espèce.

getRegime() : String[]

Permet de récupérer la liste du régime alimentaire de l’agent.

setRegime(String[]) : Void

Permet de modifier la liste du régime alimentaire de l’agent.

accelerer() : Void

Permet à l’agent d'accélérer.

tetanise() : Void

Empêche l’agent de se déplacer à cause de la peur.

fuite() : Void

Permet à l’agent d’effectuer l’action de fuite.

setModele(3DModel modele) : Void

Permet de modifier le modèle 3D d’un agent, exemple si un loup meurt de faim le modèle 3D pourra être modifié.

getBesoin() : Integer[]

retourne la liste des besoins de l’agent (faim, soif, peur...) .

setBesoin(Integer [] tabBesoin) : Void

permet de modifier avec une liste les besoins de l'agent (faim, soif, peur...) .

switchControle() : Void

Changer le boolean agentencontrole.

switchCanMove() : Void

Change le boolean canmove.

setChampdsDeVision(Integer x)

Permet de définir le champ de vision d’un agent afin de détecter les obstacles et les autres agents.

setAction(String s) : void

Permet de modifier l’action en cours de l’agent.

getAction() : String

Permet de récupérer l’action en cours de l’agent.

verifAge(Integer maxAge) :

Vérifie si l’âge max est atteint, si oui l’animal meurt de vieillesse.

#### 2- Prise de contrôle

class Controle extends Agent

Permet la prise de contrôle d’un agent

Fonction :

* public void deplacement(Input commande) :

permettra à l’utilisateur de se déplacer en fonction de ses actions sur le clavier s’il ne rentre pas en conflit avec un objet de l’environnement.

* public void acceleration(Input commande) :

permettra à l’utilisateur lorsque la touche “MAJ” sera prise en entrée de passer en mode accélération, elle ne pourra cependant pas dépasser une certaines limite.

* public void interact (Input commande) :

permettra à l’utilisateur lorsque la touche “E” sera prise en entrée d’effectuer une action en fonction de l’objet devant l’agent.

#### 3- Mimétisme

class Mimic

Permet de gérer le mimétisme des agents.

Fonction :

detectIfAgentIsTakeOver() : Boolean

Permet aux agents contrôlés par une intelligence artificielle de repérer un agent contrôlé par l’utilisateur.

detectActionOfAgent() : Action

Permet aux agents contrôlés par une intelligence artificielle d’observer les actions d’un agent contrôlé par l’utilisateur.

mimicAction(): Void

Permet aux agents contrôlés par une intelligence artificielle de reproduire les actions effectuées par un agent contrôlé par l’utilisateur si elles permettent de satisfaire leurs besoins.

saveAction() : Void

Sauvegarde l’action dans la base de données dédiée.

#### 4- Interaction avec l’eau

class WaterInteraction extends Agent

Attribut :

canswim : boolean

Fonction :

swim(Integer endurance):

permet à l’utilisateur de nager, s’il le peut.

drink(Integer eau) : Void

permet à l’utilisateur de boire.

drown(Integer endurance):Void

se noie en fonction de la barre d'endurance.

#### 5 - Monde

class World

Permet la gestion de la météo et de son impact sur le monde

Attribut :

ter : Terrain

Indique le terrain utilisé pour le monde.

temperature : Integer

La température du monde.

tauxhumidite : Integer

Le taux d’humidité du monde.

vitessemoyvent : Integer

La vitesse moyenne du vent.

saison : String

La saison actuelle du monde.

meteo : Meteo

La météo actuelle du monde.

Fonction :

getTemperature() : Integer

Permet de récupérer la température.

getTauxHumidite() : Integer

Permet de récupérer le taux d’humidité.

getVitesseMoyVent() : Integer

Permet de récupérer la vitesse moyenne du vent.

getSaison() : String

Permet de récupérer la saison en cours.

getMeteo() : Meteo meteo

Permet de récupérer la météo en cours

setTemperature(Integer temprature) : Void

Permet de modifier la température.

setTauxHumidite(Integer tauxHumidite) : Void

Permet de modifier le taux d’humidité.

setSaison(String saison) : Void

Permet de modifier la saison.

setMeteo (String saison, Meteo meteo) : Void

Permet de modifier la météo.

#### 6 - Météo possible

Énumère l’ensemble des météos possibles.

enemueration PossibiliteMeteo

ORAGE

SECHERESSE

PLUIE

TEMPS\_CLAIR

BROUILLARD

VENT

#### 7 - Météo

class Meteo extend PossibiliteMeteo

Permet de modifier les différents attributs en fonction de la météo actuelle.

Attribut :

meteo : PossibiliteMeteo

variationtemperature double[]

tauxhumidite : Double

puissanceffet : Double

influenceSoif : Double

influenceReproductionPlante : Double

influenceChampsVision : Double

Fonction :

variationTemperature(Double temperature) : Void

Permet de définir les températures possibles avec borne dans un tableau.

drawVisualEffects (Météo m) : Void

Permet d’afficher dans le monde un effet visuel pour chaque meteo possible.

modifInfluenceSoif(Double d) : Void

Permet de modifier l’influence de la météo sur le taux de soif des agents.

modifInfluenceReproductionPlante(Double d):Void

Permet de modifier l'influence de la météo sur la reproduction des agents autotrophes.

modifInlfuenceChampsDeVision(Double d): Void

Permet de modifier le champ de vision des agents en fonction de la météo.

C- Packages :

L’ensemble du programme est contenu dans un dossier “Assets” contenant 12 dossiers. Les dossiers, “CodeMonkey”, “TerrainData”, “TextMeshPro”, “Shaders”, “HDRPDefaultRessources” et “NavMeshComponents” sont des packages créés automatiquement par unity afin divers outils de programmation.

Le dossier “Animals” contient l’ensemble des modèles 3D, animations et l'habillage de l’ensemble des agents de la base de données.

Le dossier “Ressources” regroupe sous le dossier “Agents” l’ensemble du code pour chaque agent présent dans la simulation dans leur propre dossier.

Le dossier “Scripts” regroupe le code permettant de gérer le graphique montrant l’évolution de la simulation, la caméra, la simulation en elle-même...

Le dossier “Sprites” s’occupe lui de la météo et de la gestion des différents types d’agents (proies, prédateurs...)

Le dossier “Terrain Assets” s’occupe de l’ensemble de l’environnement (arbres rochers...) qui permet de créer un terrain pour la simulation.

## X. Références

### A-Tableau des figures

*Figure 1: Arbre WBS du projet*

*Figure 2: Ressources de l’équipe du diagramme de Gantt*

*Figure 3: Tableau du diagramme de Gantt*

*Figure 4: Graphique du diagramme de Gantt*

*Figure 5: Capture d’écran du cours de génie logiciel, cycle de vie modèle en V*

*Figure 6: Capture d’écran de Unity en version 2021.2.9f1*

*Figure 7: Diagramme de cas, contrôle d’un agent*

*Figure 8: Diagramme de cas, mimétisme*

*Figure 9: Diagramme de cas, mimétisme stocké dans la base de données*

*Figure 10: Diagramme de cas, mimétisme stocké dans la base de données*

*Figure 11: Diagramme de cas, système de peur*

*Figure 12: Diagramme de cas, système de saison*

*Figure 13: Diagramme de cas, interaction avec l’eau*

*Figure 14: Diagramme de cas, système de météo*

*Figure 15: Diagramme de cas, modification des traits*

*Figure 16: Diagramme de séquence, mimétisme*

*Figure 17: Diagramme de séquence, mimétisme stocké dans la base de données*

*Figure 18: Diagramme de séquence, interaction avec l’eau*

*Figure 19: Diagramme de séquence, comportement de l’agent en fonction de ses besoins*

*Figure 20: Diagramme de séquence, modification des traits*

*Figure 21: Diagramme de séquence, contrôle d’agent*

*Figure 22: Diagramme de classe*

*Figure 23: Diagramme de transition, Stockage des paramètres et des résultats dans la BDD*

*Figure 24: Diagramme de transition, gestion du contrôle d’un agent par l’utilisateur*

*Figure 25: Diagramme de transition, gestion des saisons et des différentes météos*

*Figure 26: Diagramme de packages*

### B-Bibliographie

Équilibre proies-prédateurs, wikipedia:

<https://fr.wikipedia.org/wiki/%C3%89quilibres_pr%C3%A9dateurs-proies>

Prédation, wikipedia:

<https://fr.wikipedia.org/wiki/Pr%C3%A9dation>

Équations de prédation de Lotka-Volterra:

<https://fr.wikipedia.org/wiki/%C3%89quations_de_pr%C3%A9dation_de_Lotka-Volterra>

Cahier de recette du groupe L3O2 2021 :

<https://drive.google.com/file/d/19LwH4ZouJaL2TJDUdFyijDac_3spCwo9/view?usp=share_link>