# API - MOSIMA M2

Octobre 2023

## 1 Introduction

Le projet *Ecotopia* repose sur une approche systémique : une société humaines est un système complexe, où chaque dimension (économie, écologie ...) est impactée par les autres.

Parmis toutes les activités humaines, nous proposons de nous focaliser sur quatre dimensions :

- L'urbanisme ($H$)
- L'énergie ($E$)
- L'agriculture ($A$)
- Le transport ($T$)

Ces dimensions (qu'on nomera des "blocs") ont le plus gros impact sur l'exploitation du territoire et les émissions de GES[1]. Leur impact économique est également très élevé, mais nous n'intégrons pas l'économie dans le projet.

La population (e.g. démographie) devra également être représentée, on considère donc un composant supplémentaire "résidents" ($R$). L'ensemble de ces éléments forme le système *EARTH*.

## 2 Architecture logicielle

L'organisation du projet nécessitera que plusieurs de ces blocs soient développés parallèlement. Cependant, les blocs peuvent être inter-dépendants (e.g. la production agricole a besoin d'énergie).

Il est donc nécessaire d'anticiper ces inter-dépendances et de concevoir une architecture logicielle adaptée. L'architecture retenue permet la séparation des différents blocs, qui ne seront accessibles qu'au travers d'une API[2]. Un coordinateur se chargera de connecter les blocs entre eux et de les synchroniser. L'API est présentée dans la section 3.

[1] Gas à effet de serre

[2] Application Programming Interface

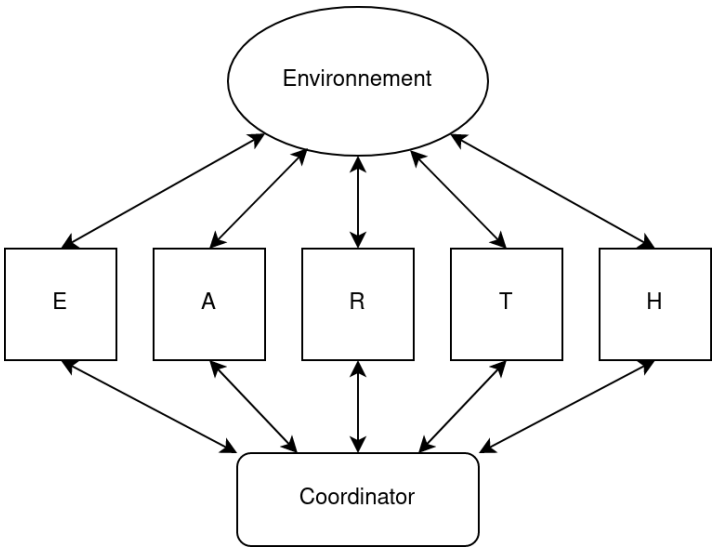


Figure 1: Illustration des composants du système *EARTH*

Chaque bloc aura deux composantes :

- la **production** : inclus tout le système de production du bloc (infrastructures, topologie, impacts ...)
- la **consommation** : inclus le comportement de consommation de la population (besoins, mécanisme de décision ...)

La méthodologie que nous proposons pour l'implémentation des blocs est itérative :

1. Implémenter une version minimale de chaque bloc, permettant de fournir aux autres blocs les éléments qui leurs sont nécessaires
2. Approfondir chaque bloc parallèlement
3. Intégrer régulièrement une version stable de chaque bloc dans le système complet

Cette méthodologie nécessite un respect strict de l'API du système.

# 3 API fournie

Nous détaillons dans cette section l'API fournie pour le projet. Cette API est implémentée en *Gama*, ainsi que des blocs minimalistes servant d'exemple.

## 3.1 Concept général

Chaque bloc (*HEAT*) ainsi que le composant de démographie ($R$) seront respectivement implémentés sous la forme d'agents, héritant de la *species Bloc* de l'API.

Les composants de production et de consommation de chaque bloc seront également représentés sous la forme d'agents. Pour chaque bloc, on instanciera un producteur virtuel sous la forme d'un unique agent, agrégeant toute la production du bloc[3]. De la même manière, il y aura un unique agent par bloc pour la consommation, implémentant le comportement de consommation de la population relatif à ce bloc[4].

[3]Nous imposons l'usage d'un unique agent producteur dans un premier temps. Lorsque le modèle sera fonctionnel, vous pourrez si souhaité étendre ce composant à plusieurs agents de production.

[4]Ceci pour des raisons de structuration : ainsi, tout ce qui est relatif à un bloc est implémenté dans ce dernier. Il est tout de même possible d'implémenter des comportements individuels détaillés, en utilisant par exemple les attributs de chaque individu.

Afin de pouvoir adapter ces composants de production et de consommation aux besoins des différents blocs, chaque bloc définira deux *micro-species*[5] *[Bloc]Producer* et *[Bloc]Consumer* héritant des *species Production* et *Consumption* de l'API : ainsi, les fonctionnalités spécifiques au bloc ne seront pas accessible depuis un autre bloc.

L'environnement sera composé de différents agents représentant les éléments du GIS : frontières, montagnes, forêts, points d'eau (rivières et lacs), villes. Les fichiers GIS correspondants seront fournis prêt à l'emploi. Vous pourrez également les enrichir ou définir des variables globales, selon votre convenance.

Un unique agent coordinateur gèrera la simulation : il connectera les blocs entre eux et les synchronisera. La fonction *simulationStep()* du coordinateur déclenchera le tick des différents blocs, cette fonction sera un *Reflex* conditionné par l'état du coordinateur (démarré/arrêté), il s'exécutera ainsi à chaque pas de temps de la simulation automatiquement. L'agent coordinateur est fourni prêt à l'emploi (*species Coordinator*).

## 3.2 Signature de l'API

**Bloc**

- getName() : String
  Retourne l'identifiant du bloc
- getProducer()[6] : Production
  Retourne le composant de production du bloc. Utile pour les requêtes de production aux fournisseurs, en provenance d'un autre bloc
- tick(List<Human> pop) : void
  Exécute un tick : applique la consommation de la population et la production correspondante
- getInputResourcesLabels() : List<String>
  Retourne les labels des resources utilisées par le bloc pour produire
- getOutputResourcesLabels() : List<String>
  Retourne les labels des ressources produites par ce bloc

**Production**

- produce(Map<String, float> demand) : boolean
  Produit les quantités demandées
- getTickInputsUsed() : Map<String, float>
  Retourne les quantités de chaque ressource utilisée ce tick pour la production
- getTickOutputProduced() : Map<String, float>
  Retourne les quantités produites ce tick
- setSupplier(String product, Bloc bloc)
  Définit un fournisseur externe pour une ressources utilisée pour produire

**Consumption**

- consume(Human h) : void
  Détermine et exécute le comportement de consommation de cet humain
- getTickConsumption() : Map<String, float>
  Retourne les quantités consommées ce tick par la population

[5] Une *micro-species* en Gama est semblable à une classe *nested* en Java : elle n'est accessible qu'au sein de l'espèce parent et peut également accéder à cette dernière.

[6] Remarque : on ne retrouve pas de getConsumer() dans l'API : il n'est pas nécessaire de pouvoir accéder aux composants de consommation depuis l'extérieur du bloc.

**Human**

- getAge() : int
  Retourne l'age (en année) de l'humain
- getGender() : String
  Retourne l'identifiant du genre de l'humain
- getAdditionalAttributes() : Map<String, String>
  Retourne les autres attributs de l'humain (e.g. CSP, préférences)

**Coordinator**

- registerBloc(String name, Bloc b) : void
  Enregistre le bloc, il sera inclus dans la simulation et géré par le coordinateur
- affectSuppliers() : void
  Affecte à chaque composant de production l'ensemble de ses fournisseurs, pour les ressources utilisées qui sont issues d'un autre bloc
- registerAllBlocs() : void
  Enregistre automatiquement tous les blocs existants
- start() : void
  Démarre la simulation
- stop() : void
  Arrête la simulation (pause)
- simulationStep() : void
  Démarre le tick suivant. Cette méthode sera rappellée automatiquement à chaque fin de tick si la simulation n'est pas stoppée. Ordonne aux blocs d'exécuter un nouveau tick.

## 3.3 Implémentation en Gama fournie

### 3.3.1 Installation

1. Récupérer le code du projet sur le dépot gitlab[7]
   - Option 1 : télécharger l'archive zip de la branche *main* du dépot depuis un navigateur web, puis l'extraire
   - Option 2 : initialiser un répo git sur votre machine, ajouter le dépot comme *remote*, puis *pull* la branche *main* sur votre machine
2. Importer le projet sur l'IDE *Gama*, cf Figure 2

[7]Il existe également un plugin Git pour l'IDE *Gama* : `https://gama-platform.org/wiki/InstallingPlugins#git`

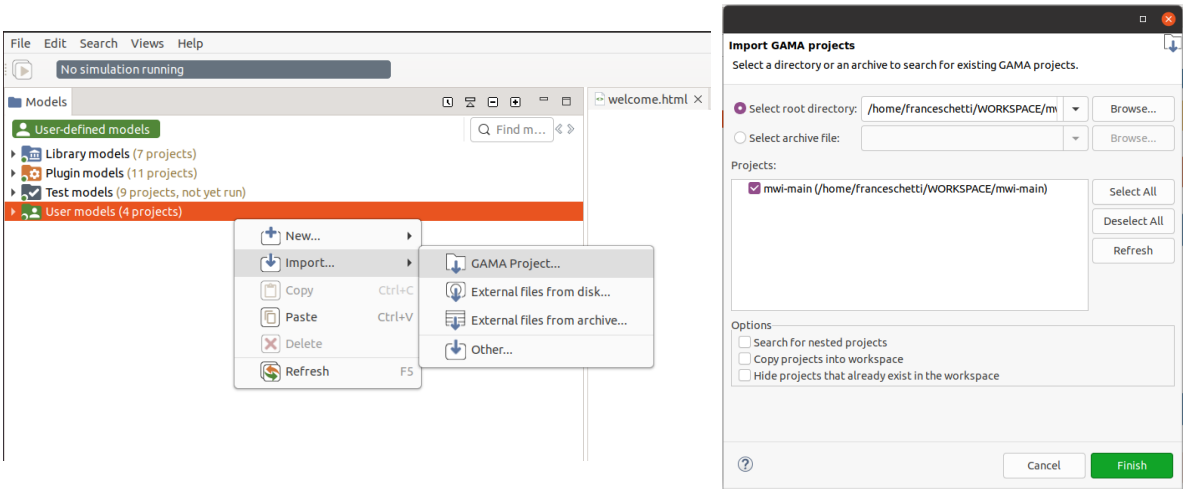


Figure 2: Import du projet *Gama* dans l'IDE

### 3.3.2 Structure

La structure du projet *Gama* fourni est détaillée en Figure 3. Le répertoire *includes* contient les données utilisés (ici, les fichiers GIS), le répertoire *models* contient l'API ainsi que des blocs minimaux implémentés : Agriculture, Energie et Démographie. La simulation sera lancée depuis le modèle *Main*, qui instancie les différents blocs et fait tourner la simulation.


Figure 3: Structure du projet *Gama*

### 3.3.3 Exécution

Pour pouvoir lancer une simulation en *Gama*, il faut définir dans le modèle une *experiment*[8] à exécuter. Dans chaque bloc, une *experiment* définira les tracés qui lui sont relatifs (e.g. les tracés relatifs à la démographie seront codés dans le fichier *Demography.gaml*). Ces *experiment* seront alors visibles depuis le modèle *Main* une fois les différents blocs importés, et pourront être exécutées (cf Figure 4). Si vous souhaitez combiner les tracés de plusieurs blocs à la fois, il faudra définir dans le modèle *Main* une *experiment* les combinant.

Remarque : l'instanciation du coordinateur et des différents blocs s'effectue dans le modèle *Main*. Les *experiment* ne peuvent donc êtres exécutées que dans ce contexte. Si une *experiment* est lancée

[8]Structure définissant entre autre les éléments à afficher (champs de saisie, moniteurs, tracés) pendant la simulation


Figure 4: Expérimentations disponibles une fois les blocs importés

directement depuis le modèle d'un bloc, aucun agent ne sera instancié. Pour éviter toute confusion, une erreur sera levée si une *experiment* est lancée depuis un autre modèle que le *Main* (cf Figure 5).

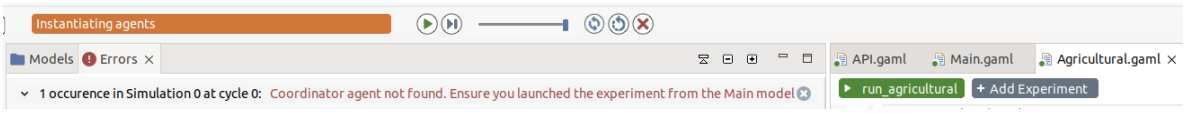


Figure 5: Erreur levée si une expérimentation est lancée depuis un modèle autre que le Main

Pour les blocs Agriculture (cf Figure 6) et Energie (vf Figure 7), on retient les tracés suivants, relatifs aux catégories de la matrice EFM (production, consommation et émission) :

- Consommation directe de la population par tick (e.g. quantités de viande et de légumes, pour le bloc Agriculture)
- Production totale du bloc par tick (e.g. quantités de viande et de légumes, pour le bloc Agriculture)
- Resources utilisées par tick pour réaliser cette production totale (e.g. quantités d'eau, d'énergie et surface au sol, pour le bloc Agriculture)
- Emissions par tick liées à cette production totale (quantités de GES émis)

Pour le bloc Démographie (cf Figure 8), on retient les tracés suivants, relatifs à la dynamique de la population et essentiels pour évaluer la stabilité de la société modélisée :

- Nombre d'hommes et de femmes dans la population, à chaque tick
- Nombre de naissances et de décès cumulés, à chaque tick
- Pyramide des ages (nombre d'individus par catégorie d'age)

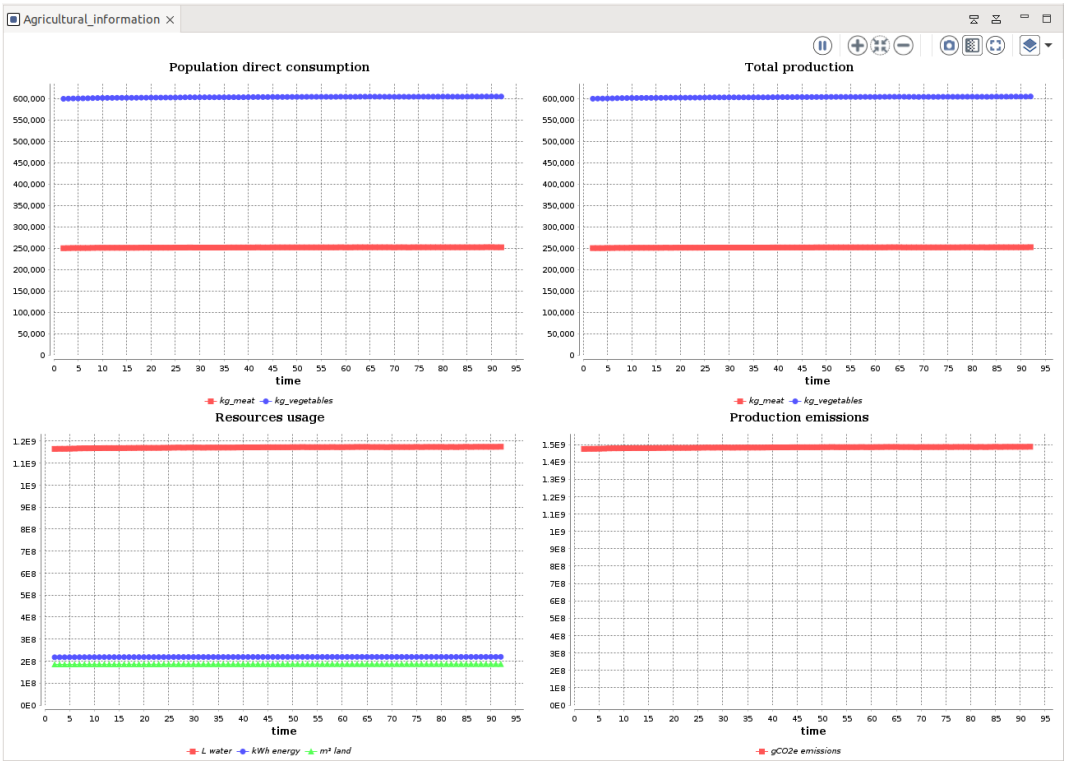


Figure 6: Tracés pour l'expérimentation du bloc Agriculture (run_agricultural)

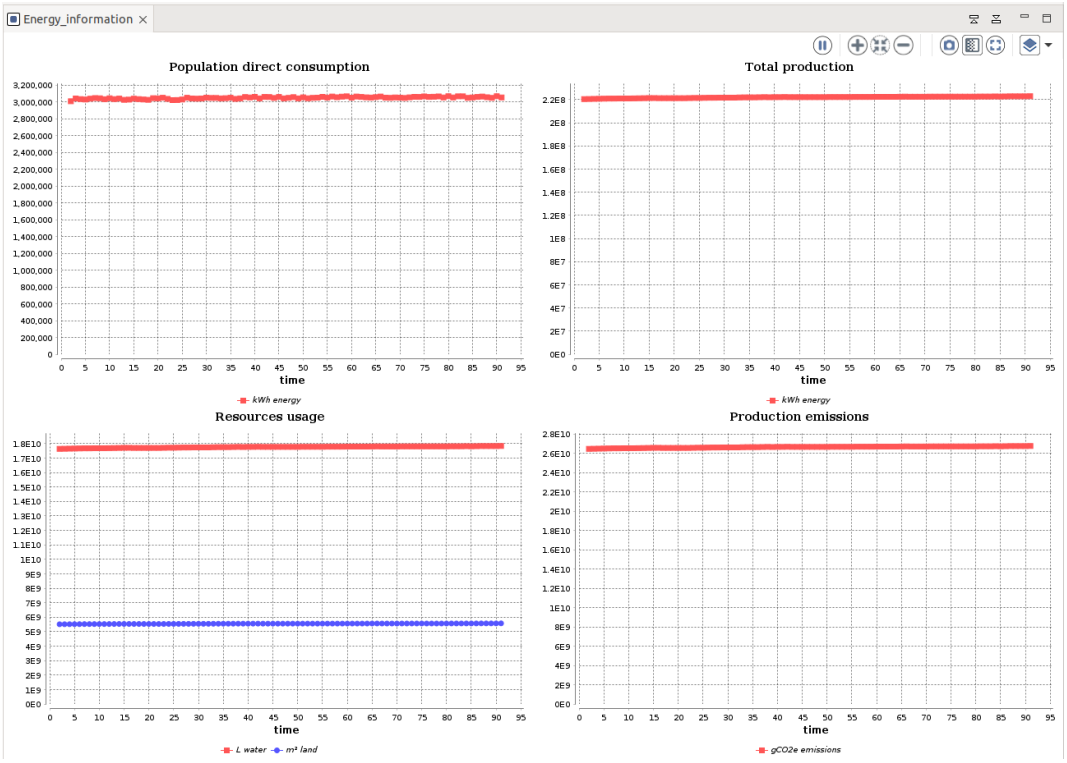


Figure 7: Tracés pour l'expérimentation du bloc Energie (run_energy)

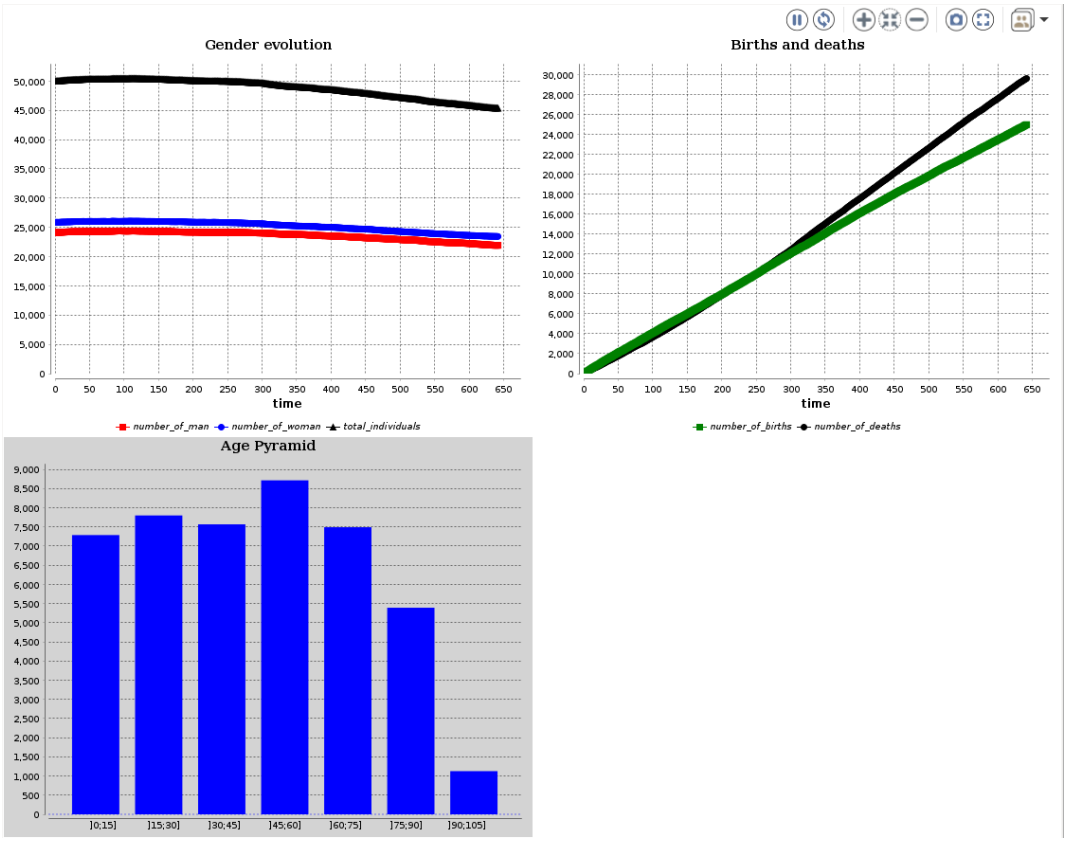


Figure 8: Tracés pour l'expérimentation du bloc Démographie (run_demography)

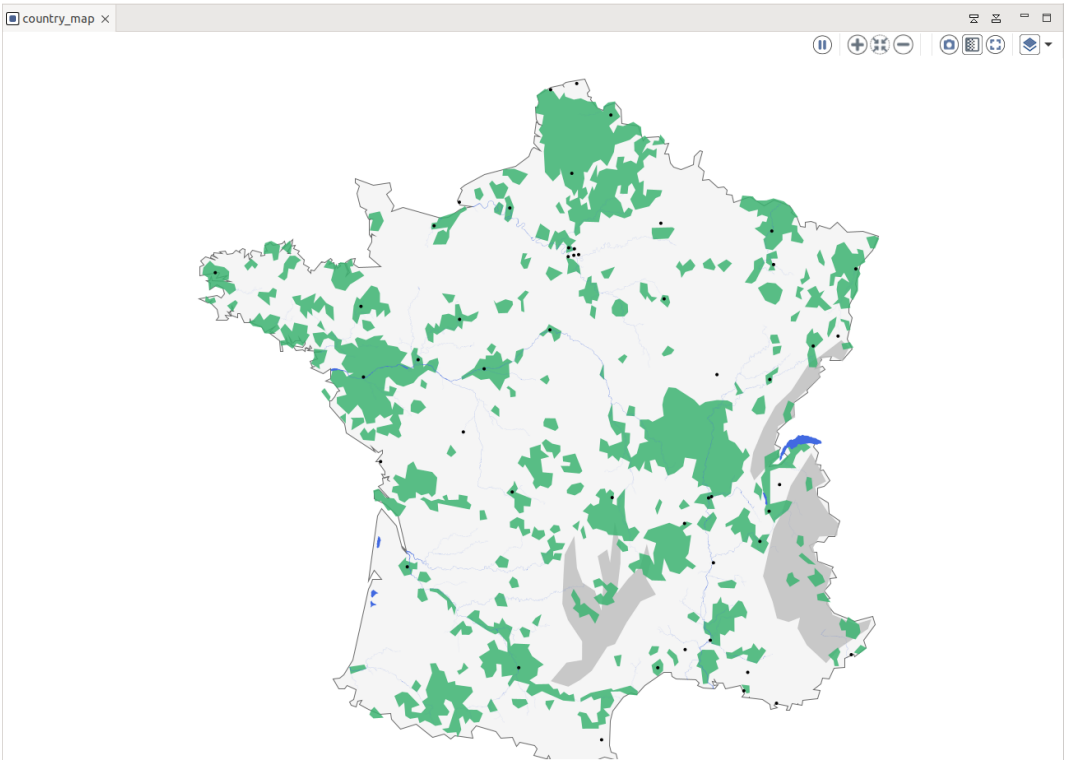


Figure 9: Affichage du GIS (display_gis)