



# Simudémie — Livrable 2

## Modèle de conception & Architecture logique

**Génie logiciel orienté objet**

GLO-2004

### Travail réalisé par l'équipe 9

Henri Bernard; 111285905; [henri.bernard-st-laurent.1@ulaval.ca](mailto:henri.bernard-st-laurent.1@ulaval.ca)  
Frederick Hughes; 111008245; [frederick.hughes.1@ulaval.ca](mailto:frederick.hughes.1@ulaval.ca)  
Jonathan Roy-Noel; 536 776 368; [jonathan.roy-noel.1@ulaval.ca](mailto:jonathan.roy-noel.1@ulaval.ca)  
Michael Vermette; 536 835 891; [michael.vermette.1@ulaval.ca](mailto:michael.vermette.1@ulaval.ca)

### Date de remise

01-03-2021

## Table des matières

Mise en contexte .....	4
Diagramme de classe de conception .....	5
Architecture logique .....	8
Diagramme de séquence de conception .....	10
Création pays de forme irrégulière .....	10
L'affichage de la vue du monde .....	11
Fonctionnement d'un pas dans le temps dans la simulation mettant à jour l'information .....	12
Diffusion de l'information concernant un pays/région sous la souris .....	13
Diffusion de l'information concernant un pays/région sous la souris (suite) .....	14
Pseudocode d'un algorithme .....	15
Plan de travail .....	18
Diagramme de Gantt .....	18
Suivi du développement .....	20
Équipe 9 .....	21
Contribution .....	21
Répartition des tâches .....	21
Comptes rendus .....	23
Compte rendu no.1 .....	23
Compte rendu no.2 .....	23
Compte rendu no.3 .....	24
Compte rendu no.4 .....	24
Compte rendu no.5 .....	25
Annexes .....	26
1.1 — Esquisses v2.1 .....	26
1.2 — Thème et design v2.1 .....	28
2.1 - Vision .....	29
2.2 – Modèle de domaine .....	30
2.2.1 — Diagramme de classes conceptuelles .....	30
2.2.2 — Description de classes conceptuelles .....	31
2.3 — Modèle des cas d'utilisation .....	32
2.3.1 — Diagrammes des cas d'utilisation .....	32
	35

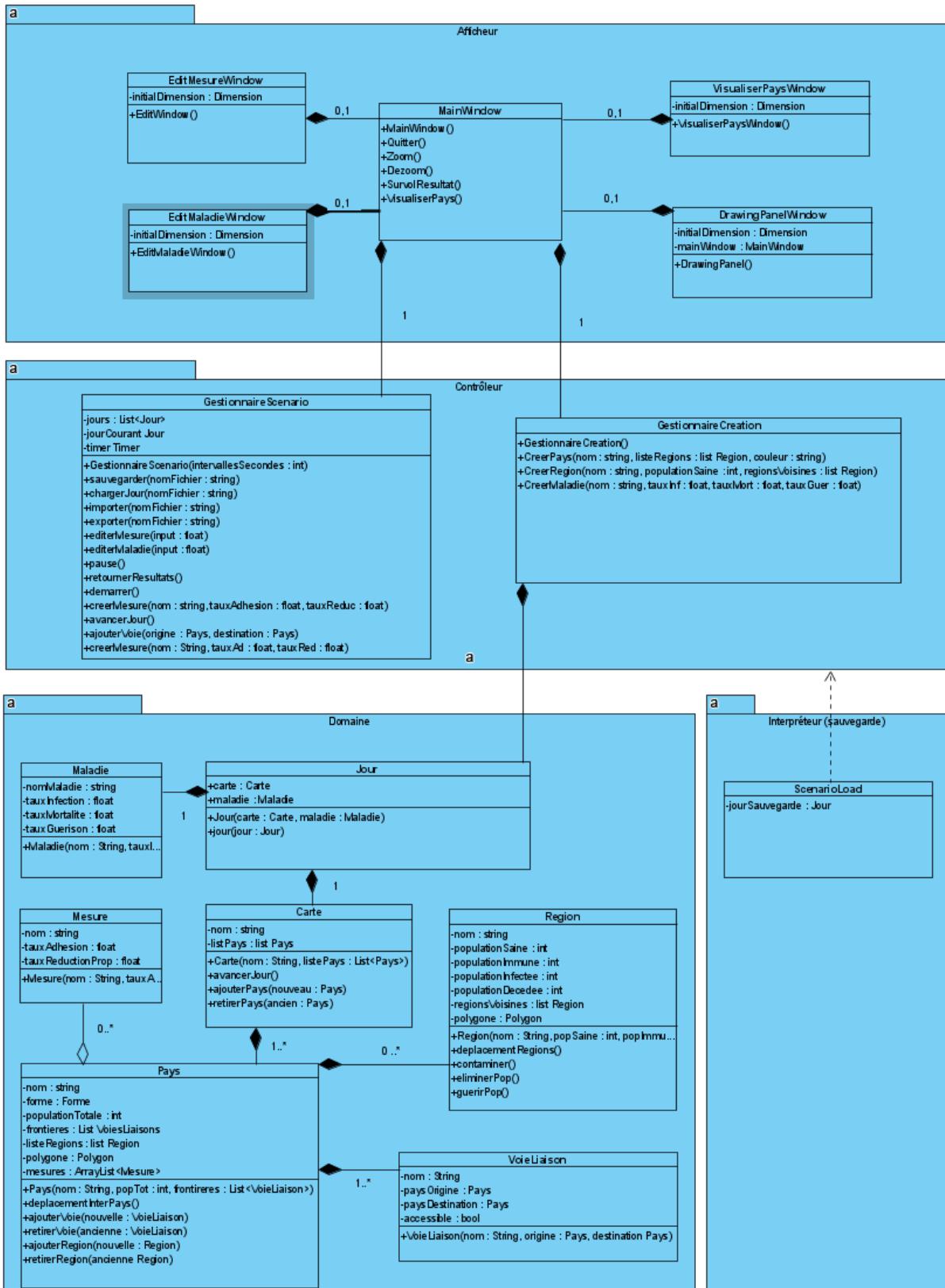
2.3.2 — Description des cas d'utilisation .....	36
2.3.3 — Diagramme de séquence système (DSS).....	58
3.1 — Esquisses des interfaces utilisateur v1.0.....	62
3.2 — Thèmes & design v1.0.....	76

## Mise en contexte

Pour donner suite à la remise du livrable 1, nous avons obtenu la rétroaction du client et nous avons effectué certains changements. Nous avons choisi de poursuivre le projet selon les diagrammes « solutions » qui nous ont été suggérés. Les différences entre nos diagrammes et ces derniers n'étaient pas importantes, mais nous avons préféré suivre les désirs du client. Dans le précédent livrable, nous avons offert plus de fonctionnalités que nécessaire et nous avons probablement ajouté une charge de travail supplémentaire au client. Nous nous concentrerons donc sur les éléments principaux du livrable 2 et discuterons avec le client des fonctionnalités supplémentaires lorsqu'une première ébauche fonctionnelle sera remise.

Dans le présent rapport se retrouve le diagramme de classe de conception ainsi qu'un texte explicatif de chaque classe, le diagramme de packages représentant l'architecture logique et finalement, les diagrammes de séquence de conception. À la suite d'une interrogation du client, nous avons inclus un pseudocode d'algorithme comme solution au problème de « cliques » dans les formes irrégulières. Nous avons aussi effectué une mise à jour sur le diagramme de Gantt du premier livrable.

## Diagramme de classe de conception



Avant de détailler chacune des classes du diagramme de classe de conception, nous souhaitons mentionner que, par souci de lisibilité, nous avons omis tous les **gets** et les **sets** ainsi que les constructeurs avec paramètre(s). Nous avons simplement laissé le constructeur par défaut.

Dans le package afficheur, on retrouve toutes les classes de type fenêtre. **EditMesureWindow** et **EditMaladieWindow** sont les fenêtres qui servent, comme leur nom l'indique, à modifier les mesures et la maladie. La classe **VisualiserPaysWindow** permet de voir une version plus détaillée du pays accompagné des informations pertinentes. Les deux dernières classes sont les plus importantes, car il s'agit de la fenêtre principale (**MainWindow**) dans laquelle on peut suivre l'évolution de la maladie « en temps réel » et l'interface de création des pays (**DrawingPanelWindow**). Toutes les fenêtres communiquent avec les classes du domaine par l'entremise du contrôleur (séparé en deux contrôleurs pour faciliter la lisibilité et éviter une classe interminable en termes de fonctions). Nous avons choisi de ne pas inclure les méthodes présentes dans les fenêtres d'interface, car celles-ci sont très nombreuses et se retrouvent majoritairement sous la forme d'**event** (événement sur un clic ou un survol, par exemple).

La première classe du contrôleur est le **GestionnaireCreation** qui fait le pont entre les interfaces accessibles par l'utilisateur et le domaine. Dans cette classe, nous avons regroupé toutes les fonctions de « départ » ou de création (à l'exception des mesures qui peuvent être créées tout au long de la simulation). Ce gestionnaire ne devrait plus être utilisé lorsque la simulation a été démarrée par l'utilisateur.

Le second contrôleur est le **GestionnaireScenario** et celui-ci contient toutes les méthodes pouvant être complétées par le biais des interfaces. Cela inclut le démarrage du scénario, la mise sur pause, toutes les éditions de valeurs possibles, ainsi que les sauvegardes et chargements. Autrement dit, cette classe gère tout ce qui peut se faire suivant le démarrage. Comme il sera mentionné dans la section suivante du rapport, c'est ici que se déroulera l'avancement journalier. Nous avons inclus un attribut *timer* qui va exécuter l'avancement de journée après chaque intervalle de secondes «x». Au même moment, la méthode *chargerJour()* va être appelée pour ajouter l'objet de type Jour à notre liste de jours. C'est ainsi que nous pourrons gérer les jours précédents (revenir plusieurs jours en arrière).

Dans le package interpréteur, nous avons mis la classe **ScenarioLoad**. Il nous manque encore certaines informations quant au bon fonctionnement de celle-ci, mais c'est la classe qui sera responsable des sauvegardes/exportations/importations/chargements. Nous n'avons pas encore effectué des tests, mais nous devrions « sérialiser » un scénario pour être en mesure de le récupérer ultérieurement.

L'objet pays est au centre de tout ce qui tourne dans l'application. Chaque pays est constitué de voies de liaisons qui les relient à d'autres pays, d'une liste de région qui le constitue et finalement

d'une forme de type Polygon qui pourra être sauvegardé et chargé plus facilement sur l'interface. Le pays dispose de quelques méthodes. La première (*DeplacementInterPays()*) permet de déplacer une partie de sa population de pays en pays par le biais des voies de liaison. Les suivantes serviront principalement à la gestion des différentes listes du pays. Cela inclut ajouter et retirer des éléments à ces listes. Les **VoieLiaison** sont les objets qui connectent les pays et ces derniers ont un nom, un pays d'origine et un pays de destination et une valeur booléenne pour déterminer si cette voie est accessible ou si elle a été fermée.

La classe **Region** est la classe qui va rassembler toutes les informations les plus importantes, car c'est dans celle-ci que va se faire le traitement statistique. Comme dans la classe **Pays**, on retrouve une méthode qui déplace une portion de la population entre les régions (d'où la présence d'un attribut liste *regionsVoisines*). Toutes les autres méthodes de la classe région, servent à appliquer les différents taux dictés par les mesures ainsi que la maladie. Ainsi, la fonction contaminée va faire appel au taux d'infection de la maladie et s'appliquer sur la population saine et transférer les personnes infectées dans la catégorie population infectée. De la même manière, la fonction *éliminerPop*, s'applique sur la population infectée et déplace les gens décédés dans la catégorie population décédée, etc. La classe Région possède aussi un attribut de type Polygon (comme la classe Pays).

Les deux classes, **Mesure** et **Maladie**, possèdent des attributs multiplicateurs qui pourront être employés sur la population des différentes régions. Outre leurs attributs (et les *gets/sets*), ces deux classes ne font rien d'autre pendant la simulation, à moins que l'utilisateur effectue des changements.

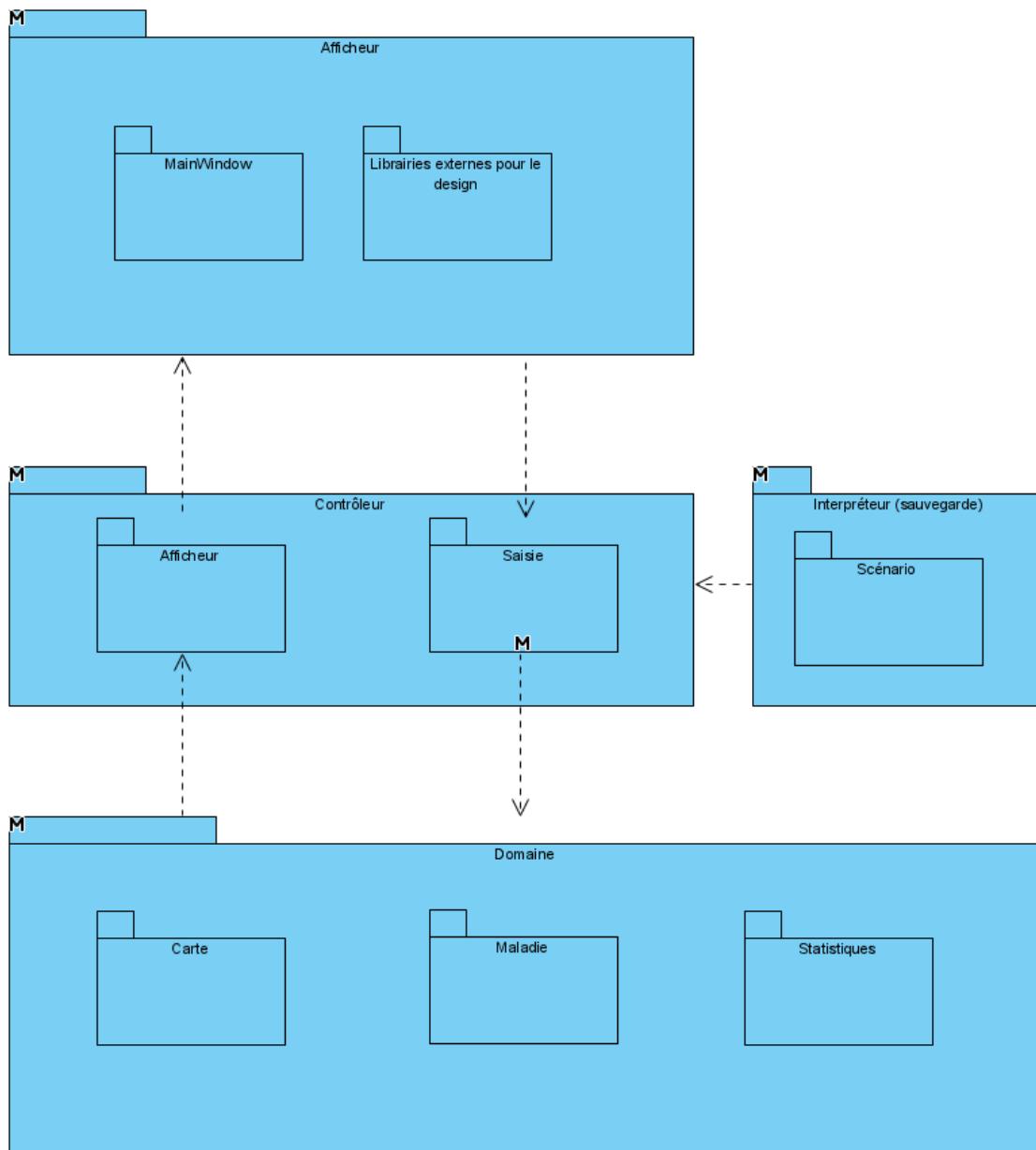
La classe **Carte** comporte tous les pays et les classes dont Pays hérite. Cette classe sert principalement de structure pour regrouper les différents éléments qui doivent être présents sur la carte. Ainsi, les seules méthodes présentes servent à faire des ajouts ou retraits de pays à la liste.

Notre dernière classe est **Jour**. C'est probablement la classe la plus importante du package Domaine, car elle regroupe tous ses éléments. Elle englobe donc la carte et ses pays, mais aussi la liste des mesures mises en place par l'utilisateur et la maladie choisie. Cela nous permet d'avoir une capture d'écran de la simulation au jour désiré. Tout comme la classe Carte, la classe Jour agit comme une structure qui sert de conteneur.

Nous croyons avoir fait le tour des classes et méthodes que nous allons employer, mais il est fort probable que cela soit appelé à changer. Si par exemple, le client souhaitait que l'on produise un rapport Excel, il faudrait créer une nouvelle classe Excel pour permettre la création d'objets Excel et ainsi faire le traitement souhaité. Il est évidemment possible qu'un aspect de l'application nous ait échappé et qu'il faille le rectifier ou alors que l'ajout d'un attribut ou d'une méthode permette d'alléger le code ou le simplifier.

## Architecture logique

Comme on pouvait le constater sur le diagramme de classe de conception, les classes se retrouvent à l'intérieur de package qui représente des systèmes globaux. Le plus important dans l'architecture que nous proposons est l'utilisation du contrôleur. Le contrôleur sert de pont entre les classes du domaine et l'interface de l'utilisateur. En effet, il serait problématique que l'utilisateur puisse avoir un accès direct aux classes à travers la vue.



Le premier package est l'afficheur (qu'on pourrait appeler interface utilisateur aussi) et contient toutes les fenêtres auxquels l'utilisateur a accès. Le second package est le contrôleur qui gère les requêtes de saisies et d'affichage du bas vers le haut et vice-versa. Comme mentionné plus haut, il sert d'intermédiaire ou de pont avec le package domaine.

Les derniers packages sont le domaine qui contient le « moteur » de notre application et toutes les classes qui interagissent avec la carte, la maladie et les mesures. Finalement, nous avons ajouté le package interpréteur pour représenter les fichiers *serialized* qui nous serviront de fichier de sauvegarde pour les importations et exportations. À noter que l'interpréteur devra passer par le package contrôleur comme n'importe quelle autre commande.

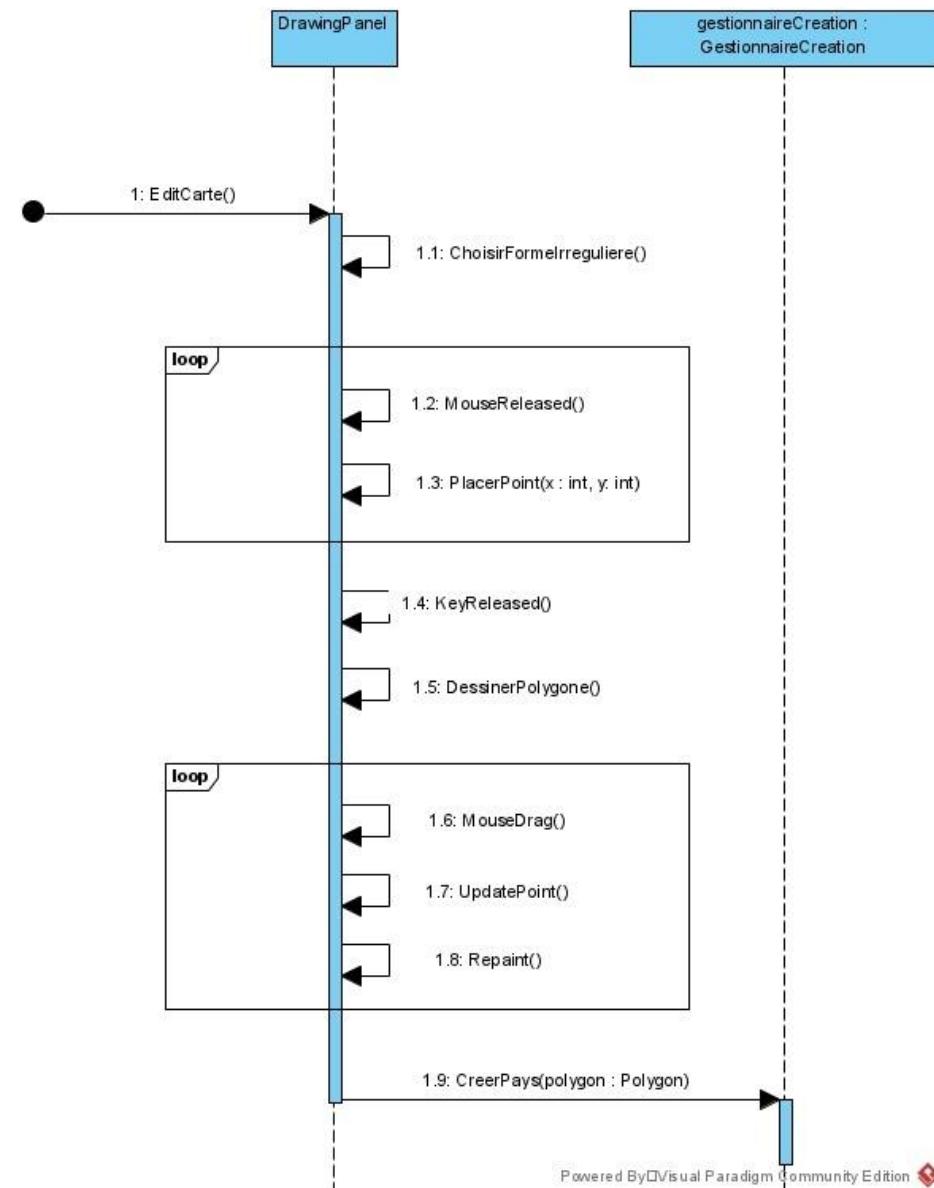
## Diagramme de séquence de conception

### Création pays de forme irrégulière

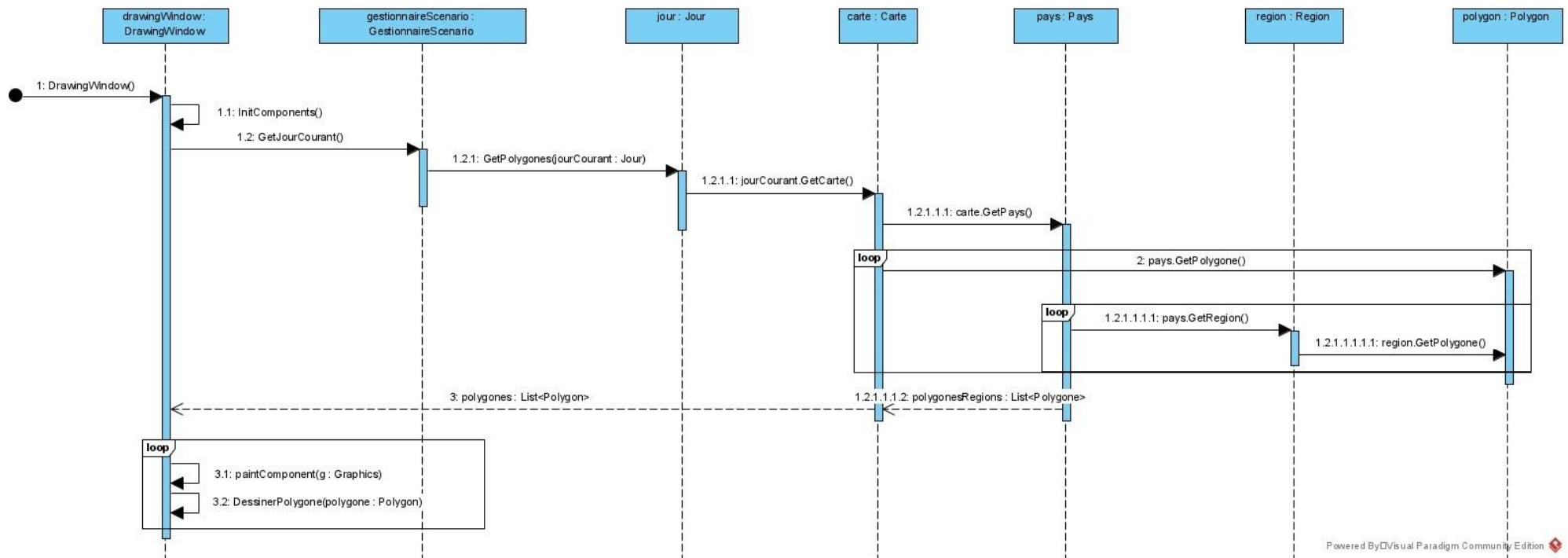
La création d'un pays de forme irrégulière se représente par la communication de deux classes, le *DrawingPanel* et le *GestionnaireCreation*. La création commence par l'appel de la fonction *EditCarte()* qui change le mode de vue pour le mode de création de cartes.

Par la suite, la fonction *ChoisirFormeIrreguliere()* dit au programme que les prochaines interactions avec la fenêtre seront pour la création d'une forme irrégulière. Ensuite, l'utilisateur peut, à répétition, faire un clic dans la fenêtre qui ajoute un point dans le polygone en cours de création. Lorsque l'utilisateur appuie sur une touche de clavier précise comme la touche «espace», la fonction *DessinerPolygone()* est ensuite appelée pour faire afficher le polygone à l'écran.

Ensuite, la deuxième boucle contenant les fonctions *MouseDrag()*, *UpdatePoint()* et *Repaint()* se chargent d'ajuster le point que l'utilisateur est en train de déplacer. Pour finir, la fonction *CreerPays(Pays pays)* crée le pays selon le polygone en paramètre.

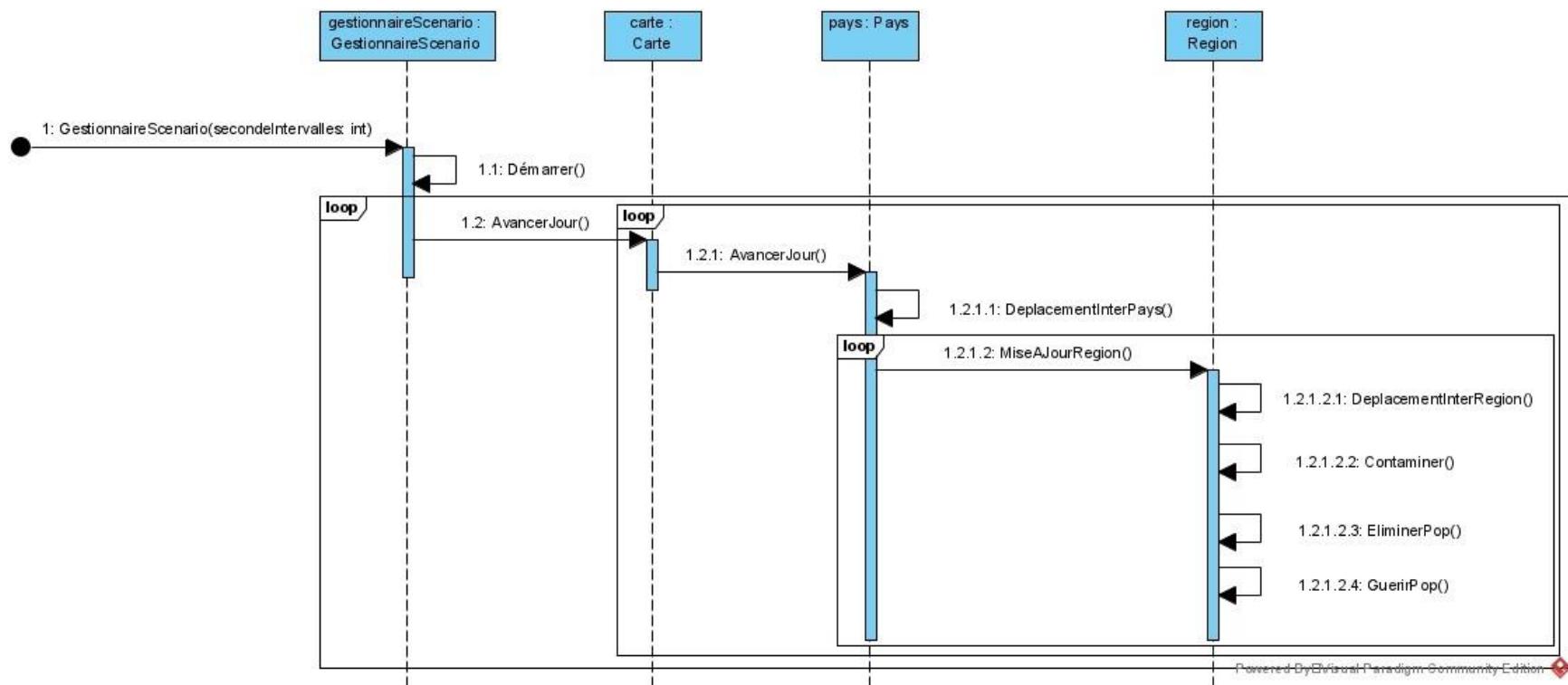


## L'affichage de la vue du monde



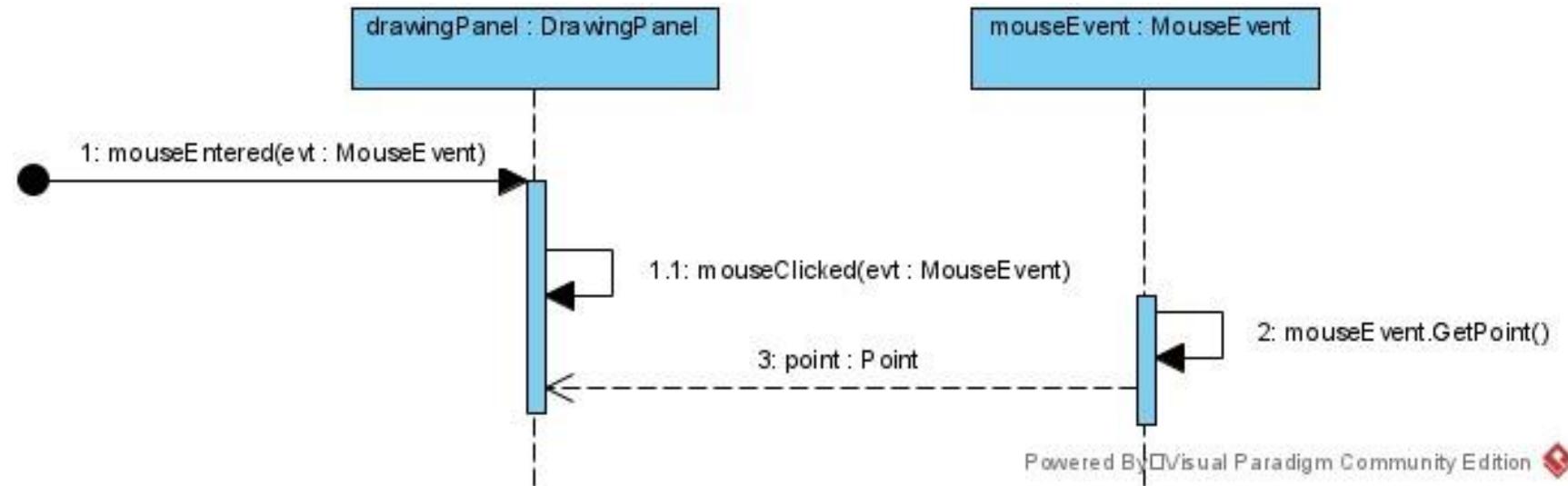
Le diagramme commence lorsque le système fait appel au **DrawingWindow()**. Il initialise avec **InitComponents()** qui appelle par la suite **GetJourCourant()**. Par la suite on appelle **GetPolygones()** sur ce jour courant, ce qui ira chercher toutes les informations relatives à la représentation de la carte mondiale du jour courant. Lorsque que l'algorithme est allé chercher la carte du jour courant avec **GetCarte()**, il pourra boucler sur **GetPolygone()** des pays puis le retourner au **DrawingWindow()** sous forme de liste *List<Polygon>*. Par la suite, le **DrawingWindow()** dessine chacun des polygones de la liste à l'aide de ses méthodes **paintComponent()** & **DessinerPolygone()**.

## Fonctionnement d'un pas dans le temps dans la simulation mettant à jour l'information



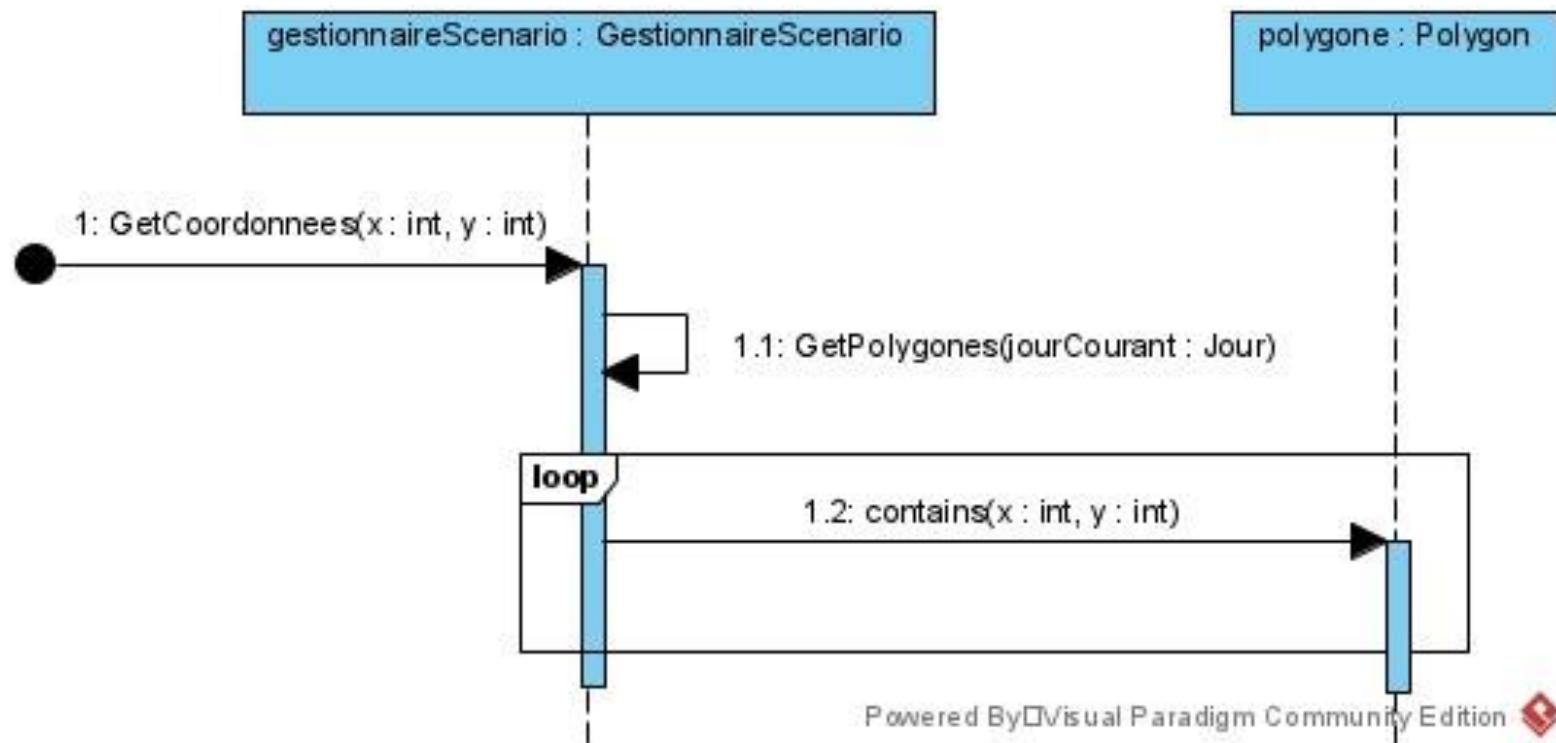
Le diagramme commence avec l'appel au constructeur du **GestionnaireScenario**. Celui-ci appelle ensuite *Démarrer()* qui commence l'appel en boucle de la méthode *AvancerJour()* sur la carte courante. Cette carte va ensuite également appeler *AvancerJour()* sur tous les pays de la carte. Ces pays vont par la suite calculer leurs déplacements avec l'appel *DeplacementInterPays()*. Chaque région de tous les pays exécute la fonction *MiseAJourRegion()*. À leur tour, les régions se chargent de déplacer, contaminer, éliminer et guérir la population avec l'appel des fonctions respectives.

## Diffusion de l'information concernant un pays/région sous la souris



Le diagramme commence lorsque la souris entre sur le panneau de dessin **DrawingPanel**. À tout instant, ce panneau est à l'écoute d'évènement dont celui qui nous intéresse le plus, *mouseClicked(MouseEvent evt)*. La méthode *GetPoint()* est liée à cet évènement et retourne le point avec les coordonnées x et y de la souris au moment du clic.

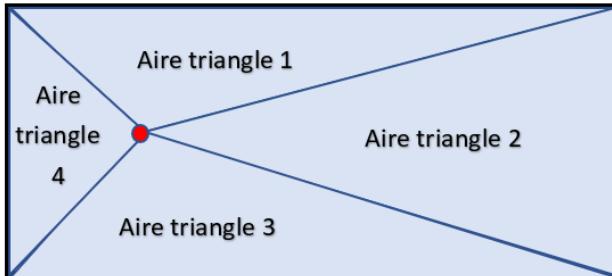
## Diffusion de l'information concernant un pays/région sous la souris (suite)



L'obtention d'information concernant un pays ou une région qui se trouve sous la souris commence par l'envoi des coordonnées x et y de la souris au contrôleur **GestionnaireScenario**. Ce contrôleur récupère ensuite tous les pays et régions. Une fois le tout récupéré, on boucle sur toutes les formes récupérées afin de savoir si le point se trouve dans la forme à l'aide de `Polygon.contains(int x, int y)`.

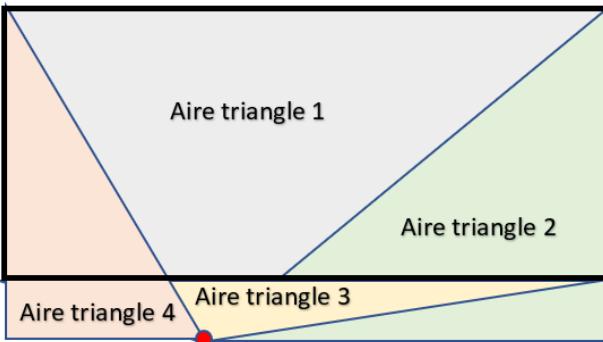
## Pseudocode d'un algorithme

Le client nous a posé le problème suivant : Comment saurons-nous si l'utilisateur a cliqué sur une forme irrégulière (pouvant être convexe par exemple) ? Nous avions déjà solutionné l'énigme des formes rectangulaires en calculant les aires des quatre triangles formés par le point et les sommets. En effet, la somme de l'aire des quatre triangles produits à partir du point et des sommets doit égaler l'aire totale du rectangle.



$$\text{Aire rectangle} = A1 + A2 + A3 + A4$$

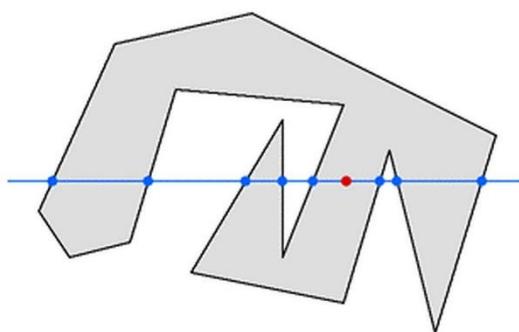
Donc le point est à l'intérieur de la forme.



$$\text{Aire rectangle} < A1 + A2 + A3 + A4$$

Donc le point est à l'extérieur de la forme.

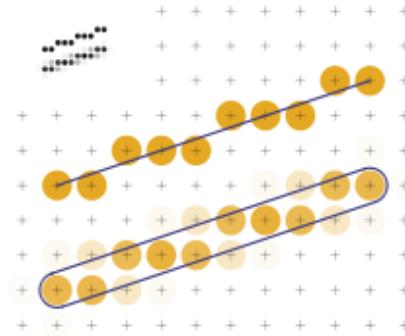
Pour la situation d'une forme irrégulière, les polygones convexes deviennent problématiques. La solution est donc de tracer une ligne sur l'axe des X à l'endroit Y où passe le point. Il suffit ensuite de regarder le nombre d'intersections entre les arrêtes et la droite. Si le nombre d'intersections est impair de chaque côté du point, le point est bel et bien à l'intérieur de la forme irrégulière. Voici une image qui illustre bien le principe (source ; [alienryderflex.com/polygon/](http://alienryderflex.com/polygon/)) :



Ce qui nous a demandé une certaine réflexion est la méthode permettant de trouver les points contenus dans les arrêtes. La solution que nous proposons ici n'est pas optimale, mais était suffisamment intrigante pour l'essayer. Puisqu'une droite est constituée de deux sommets, voici comment les points seront dessinés (source wiki) :

```
dx = x2 - x1
dy = y2 - y1
listePoints [ ]

for x from x1 to x2 do
    y = y1 + dy × (x - x1) / dx
    listePoints.add (x, y)
```



Nous obtenons ainsi une liste contenant tous les points qui font le contour de la forme. Nous pourrons donc implémenter notre algorithme décrit précédemment et effectuer la gestion d'exception (par exemple si un point se trouve sur un sommet).

```
lePoint = (x, y)
listeIntersectionsDroite []
for (i = x; i < extremiteTableau; i++)
    if (listePoints. contains (lePoint (x + i, y)))
        listeIntersectionsDroite.add (lePoint)
```

Nous avons donc ajouté toutes les intersections à la droite du point d'origine. Nous allons ensuite effectuer la même opération pour les intersections de gauche.

```
lePoint = (x, y)
listeIntersectionsGauche []
for (i = 0; i < x; i++)
    if (listePoints. contains (lePoint (x - i, y)))
        listeIntersectionsGauche.add (lePoint)
```

Il ne nous reste plus qu'à nous assurer que le nombre d'intersections contenues dans chaque liste est impair ou que le point est dans la liste.

```
If (listeIntersectionsGauche.contains(lePoint)
    or listeIntersectionsDroite.contains(lePoint)
        Return True //À l'intérieur de la forme
Else if ((listeIntersectionsGauche.size Mod 2
== 1) and (listeIntersectionsDroite.size Mod 2 == 1))
    Return True
Else
    Return False //La pièce est à l'extérieur
```

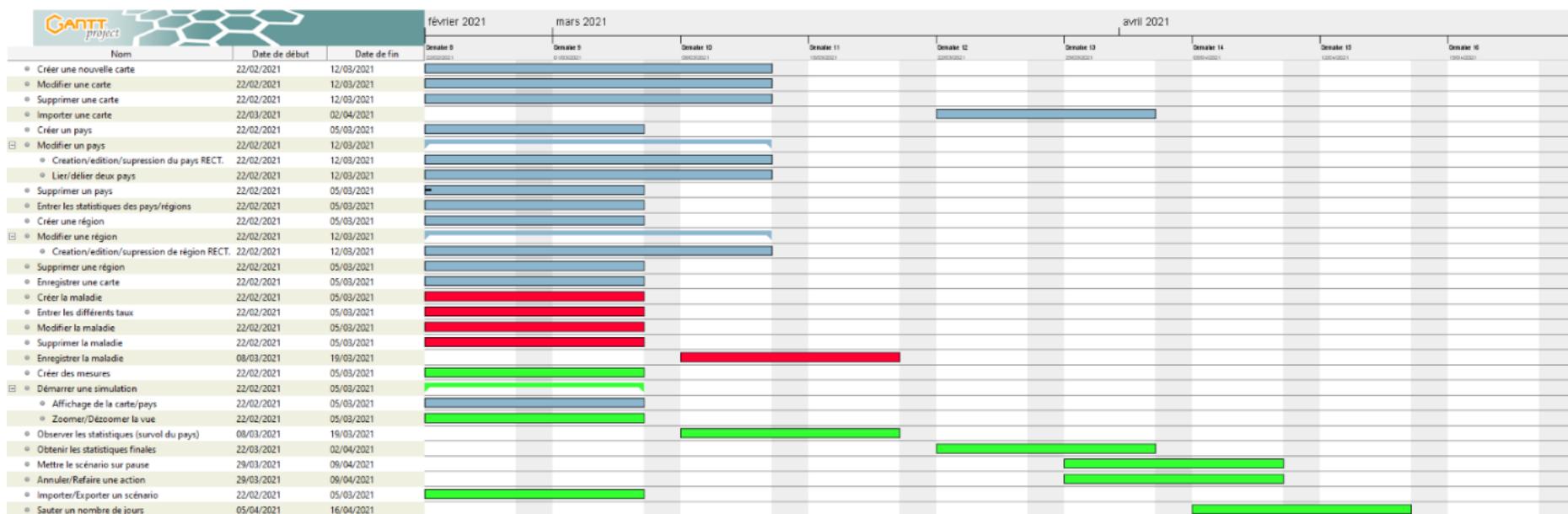
Nous obtenons finalement un booléen représentant la présence de notre point à l'intérieur de la forme irrégulière. Évidemment que ce processus n'est pas optimal, car il doit être effectué pour chaque pièce et à chaque clic. La solution simple est de « remplir » chaque point de l'objet Polygon créé et faire une simple validation sur le polygone pour savoir si le point en fait partie à l'aide de Polygon.contains (int x, int y).

## Plan de travail

Notre plan de travail a très peu changé en termes de dates. À l'exception de zoomer/dézoomer, que nous pensions implémenter en fin de projet, les autres fonctions désirées par le client pour le livrable 3 vont dans la même direction que nos attentes.

Nous avons toutefois pris du retard et nous espérons avoir comblé ce retard d'ici la production du présent rapport. Le diagramme de Gantt ci-dessous a été produit le 17 février 2021. Bien qu'une bonne partie du design des interfaces ait été produit, très peu d'éléments backend ont été complété. Si tout se passe comme prévu, toutes les classes seront créées pendant la semaine de lecture (soit une semaine après la remise du livrable).

### Diagramme de Gantt



## Tâches

Nom	Date de début	Date de fin
Créer une nouvelle carte	22/02/2021	12/03/2021
Modifier une carte	22/02/2021	12/03/2021
Supprimer une carte	22/02/2021	12/03/2021
Importer une carte	22/03/2021	02/04/2021
Créer un pays	22/02/2021	05/03/2021
Modifier un pays	22/02/2021	12/03/2021
Creation/editon/supression du pays RECT.	22/02/2021	12/03/2021
Lier/délier deux pays	22/02/2021	12/03/2021
Supprimer un pays	22/02/2021	05/03/2021
Entrer les statistiques des pays/régions	22/02/2021	05/03/2021
Créer une région	22/02/2021	05/03/2021
Modifier une région	22/02/2021	12/03/2021
Creation/editon/supression de région RECT.	22/02/2021	12/03/2021
Supprimer une région	22/02/2021	05/03/2021
Enregistrer une carte	08/03/2021	19/03/2021
Annuler une action	05/04/2021	16/04/2021
Créer la maladie	22/02/2021	05/03/2021
Entrer les différents taux	22/02/2021	05/03/2021
Modifier la maladie	22/02/2021	05/03/2021
Supprimer la maladie	22/02/2021	05/03/2021
Enregistrer la maladie	08/03/2021	19/03/2021
Créer des mesures	22/02/2021	05/03/2021
Démarrer une simulation	22/02/2021	05/03/2021
Affichage de la carte/pays	22/02/2021	05/03/2021
Zoomer/Dézoomer la vue	22/02/2021	05/03/2021
Observer les statistiques (survol du pays)	08/03/2021	19/03/2021
Obtenir les statistiques finales	08/03/2021	19/03/2021
Mettre le scénario sur pause	08/03/2021	19/03/2021
Annuler/Refaire une action	22/03/2021	02/04/2021
Importer/Exporter un scénario	05/04/2021	16/04/2021
Sauter un nombre de jours	05/04/2021	16/04/2021

---

## Suivi du développement

Malgré le léger retard précédemment mentionné, nous sommes très satisfaits du produit que nous pouvons présenter au client. En effet, notre squelette d'interface est sublime et nous avons commencé à dessiner les formes rectangulaires. Pour chaque élément du projet qui n'a pas encore été fait, nous avons des esquisses, des modèles ou des algorithmes auxquels nous avons songé. Si un quelconque problème venait à survenir, nous avons déjà pensé à de multiples solutions que nous avons classées en ordre de priorité et complexité. Ainsi, si le temps venait à manquer, nous sommes certains de pouvoir apporter des modifications dans le code qui vont simplifier nos méthodes et faciliter la production d'une itération du projet.

Il nous reste quelques questions à poser au client concernant l'utilisation de librairies externes pour l'embellissement de l'interface et certaines questions concernant le cœur de l'application. Par exemple, nous souhaitons valider l'impact qu'aura chaque voie de liaison ainsi que la méthodologie de séparation du pays. Si tout se déroule comme prévu, ces questions seront répondues le jeudi 25 février 2021 ou le jeudi suivant.

## Équipe 9

### Contribution

#### Répartition des tâches

Tâches	Fait par	Révisé par
<b>Le diagramme de classe de conception</b>		
Diagramme + Texte	Frederick	Michael & Henri & Jonathan
<b>L'architecture logique</b>		
Diagramme + Texte	Frederick	Michael & Henri & Jonathan
<b>Diagrammes de séquence de conception (DSC)</b>		
3.1 + Texte	Jonathan & Henri	Michael & Frederick
3.2 + Texte	Jonathan	Henri
3.3 + Texte	Jonathan & Henri	Michael & Frederick
3.4 .1 + Texte	Jonathan & Henri	Michael & Frederick
3.4.2 + Texte	Jonathan & Henri	Michael & Frederick
<b>Le pseudocode d'un algorithme</b>		
Pseudocode + Texte	Frederick	Michael & Jonathan & Henri
<b>Plan de travail</b>		
Diagramme de Gantt + Texte	Frederick	Michael

Finalisation		
Contribution & Comptes rendus	Henri	Frederick & Michael & Jonathan
Mise en page & Correction + Remise	Henri	Frederick & Michael & Jonathan
Version fonctionnelle		
Menu Principal	Jonathan & Henri	Michael
Création Carte	Henri	Jonathan
Simulation	Jonathan	Henri

## Comptes rendus

### Compte rendu no.1

Réunion no. 1	
Titre	Mise en point du livrable 2
Lieu	Discord — Serveur privé pour l'équipe
Date & Heure	Lundi 15 février 2021 – 16 h 30
Participants	Henri Bernard, Jonathan Roy-Noel, Frederick Hughes & Michael Vermette
Contenu	<ul style="list-style-type: none"><li>• Lecture et compréhension en groupe du livrable 2</li><li>• Séparation des tâches</li><li>• Avancement de la version fonctionnelle du projet</li></ul>

### Compte rendu no.2

Réunion no. 2	
Titre	Remue-méninges d'un pseudocode et d'une architecture logique
Lieu	Discord — Serveur privé pour l'équipe
Date & Heure	Jeudi 18 février 2021 – 16 h 30
Participants	Henri Bernard, Jonathan Roy-Noel, Frederick Hughes & Michael Vermette
Contenu	<ul style="list-style-type: none"><li>• Élaboration d'un pseudocode d'un algorithme</li><li>• Élaboration de l'architecture logique du projet</li><li>• Avancement de la version fonctionnelle du projet</li></ul>

Compte rendu no.3

Réunion no. 3	
Titre	Avancement du squelette
Lieu	Discord — Serveur privé pour l'équipe
Date & Heure	Lundi 22 février 2021 – 16 h 30
Participants	Henri Bernard, Jonathan Roy-Noel, Frederick Hughes & Michael Vermette
Contenu	<ul style="list-style-type: none"><li>• Première implémentation des classes du domaine</li><li>• Avancement de la version fonctionnelle du projet</li><li>• Objectif d'avoir terminé tous les points du livrable 2 d'ici la prochaine réunion d'équipe, soit le 25 février 2021 à 16 h 30</li></ul>

Compte rendu no.4

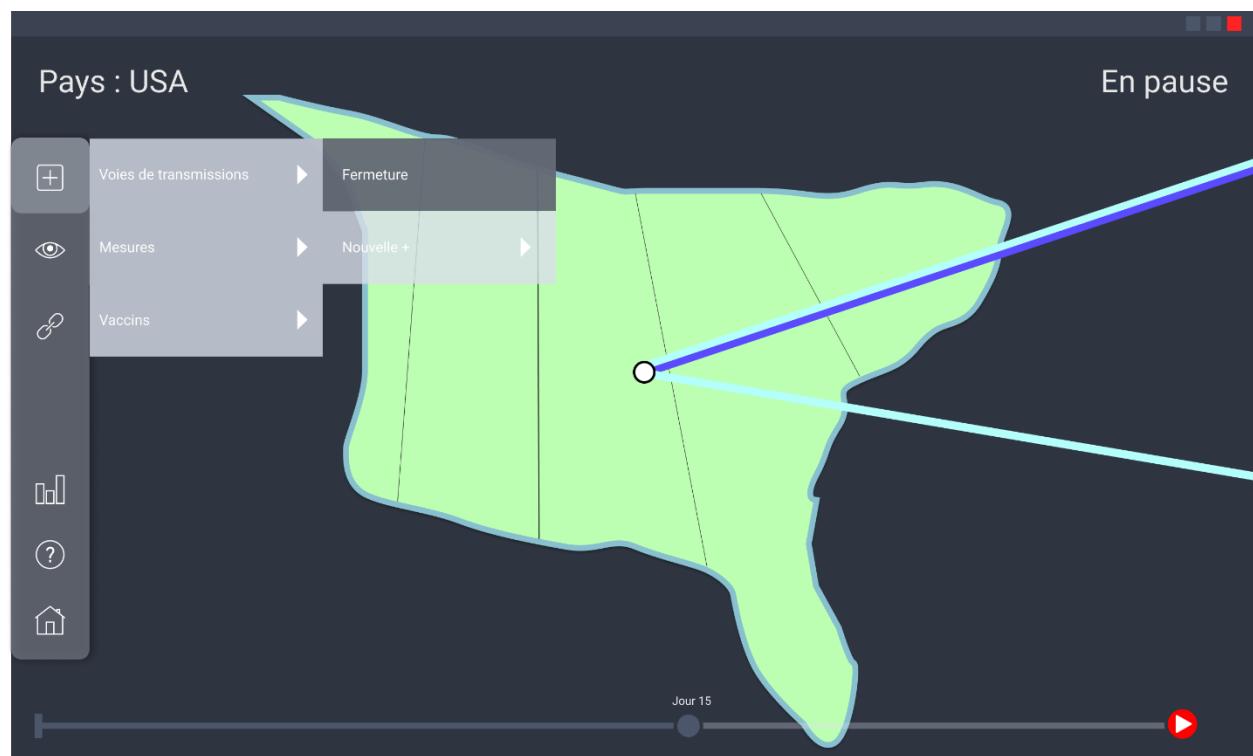
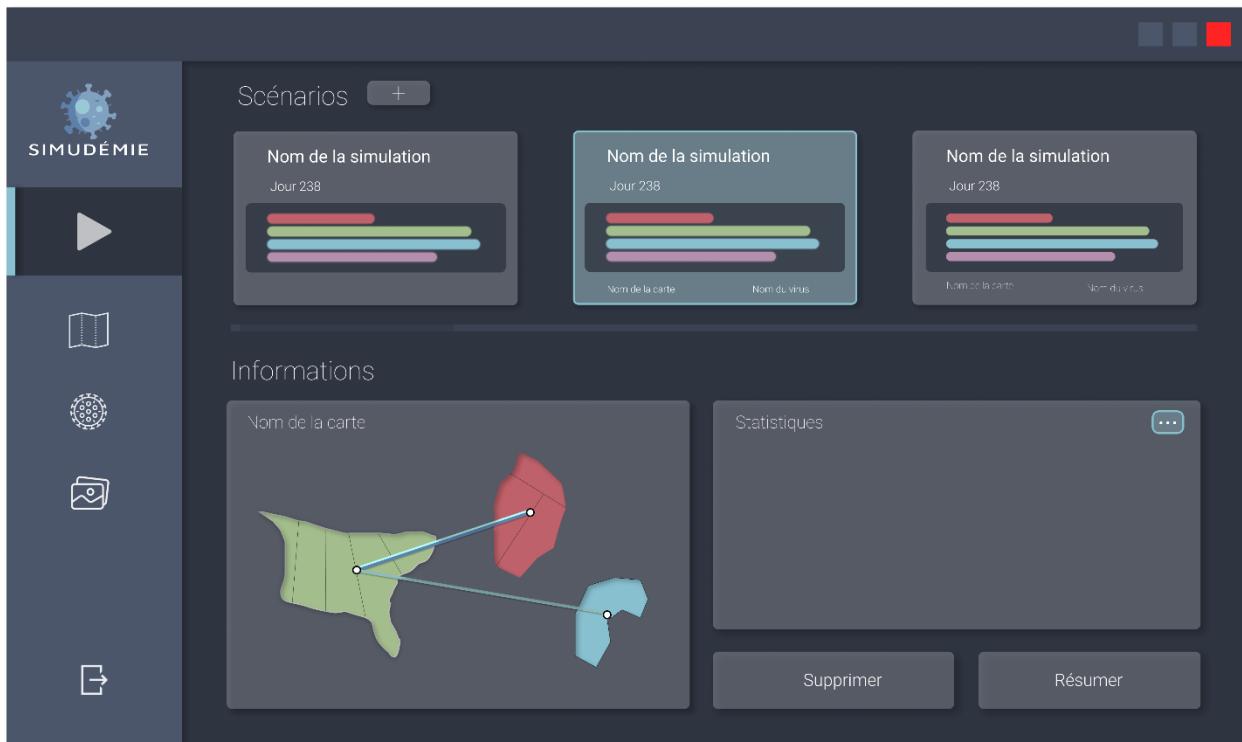
Réunion no. 4	
Titre	Uniformisation & finalisation
Lieu	Discord — Serveur privé pour l'équipe
Date & Heure	Jeudi 25 février 2021 – 16 h 30
Participants	Henri Bernard, Jonathan Roy-Noel, Frederick Hughes & Michael Vermette
Contenu	<ul style="list-style-type: none"><li>• Uniformisation des points du livrable 2</li><li>• Finalisation des points</li><li>• Avancement de la version fonctionnelle du projet</li></ul>

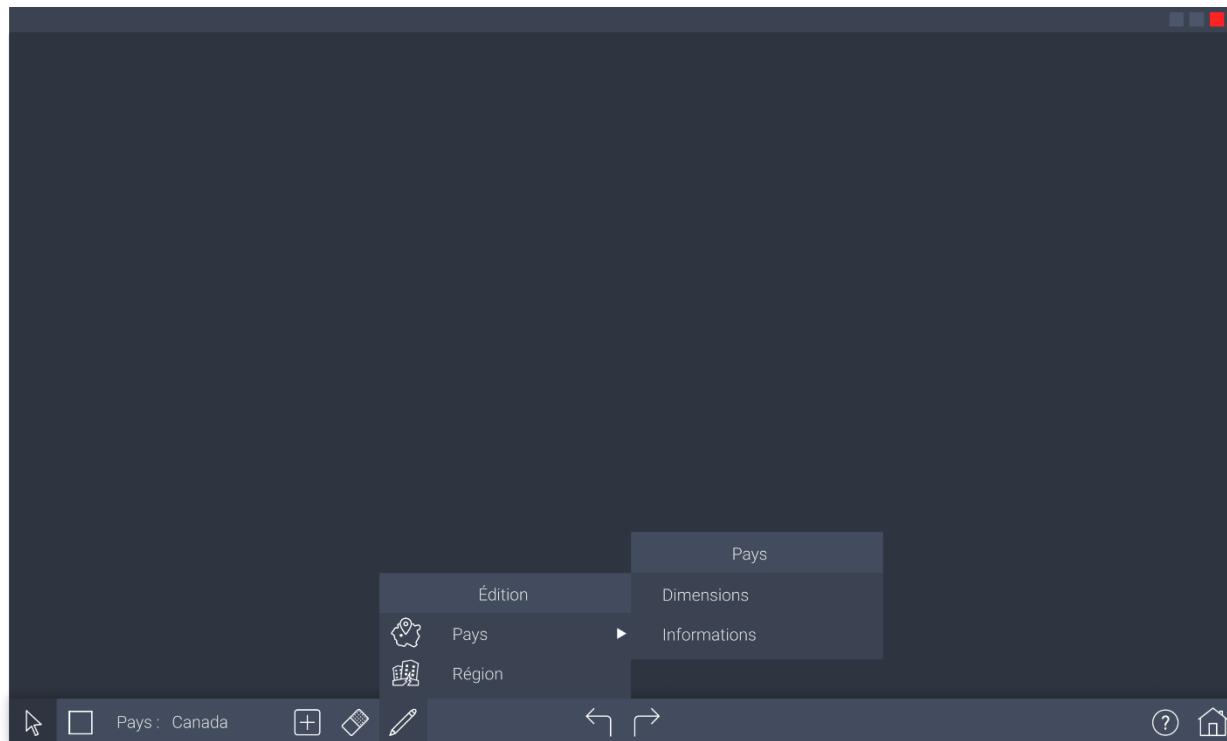
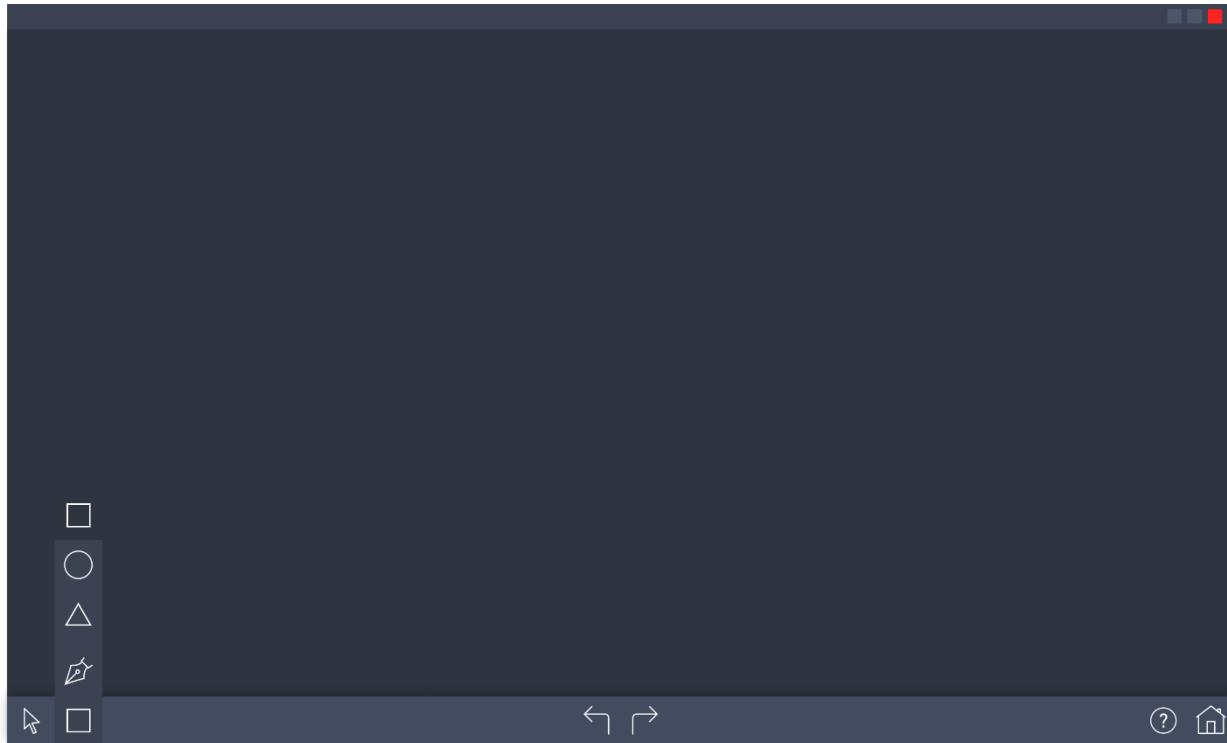
Compte rendu no.5

Réunion no. 5	
Titre	Dernières vérifications & petites retouches
Lieu	Discord — Serveur privé pour l'équipe
Date & Heure	Dimanche 28 février 2021 – 16 h 30
Participants	Henri Bernard, Jonathan Roy-Noel, Frederick Hughes & Michael Vermette
Contenu	<ul style="list-style-type: none"><li>• Correction</li><li>• Dernières vérifications</li><li>• Dernières retouches</li><li>• Préparation de la remise</li></ul>

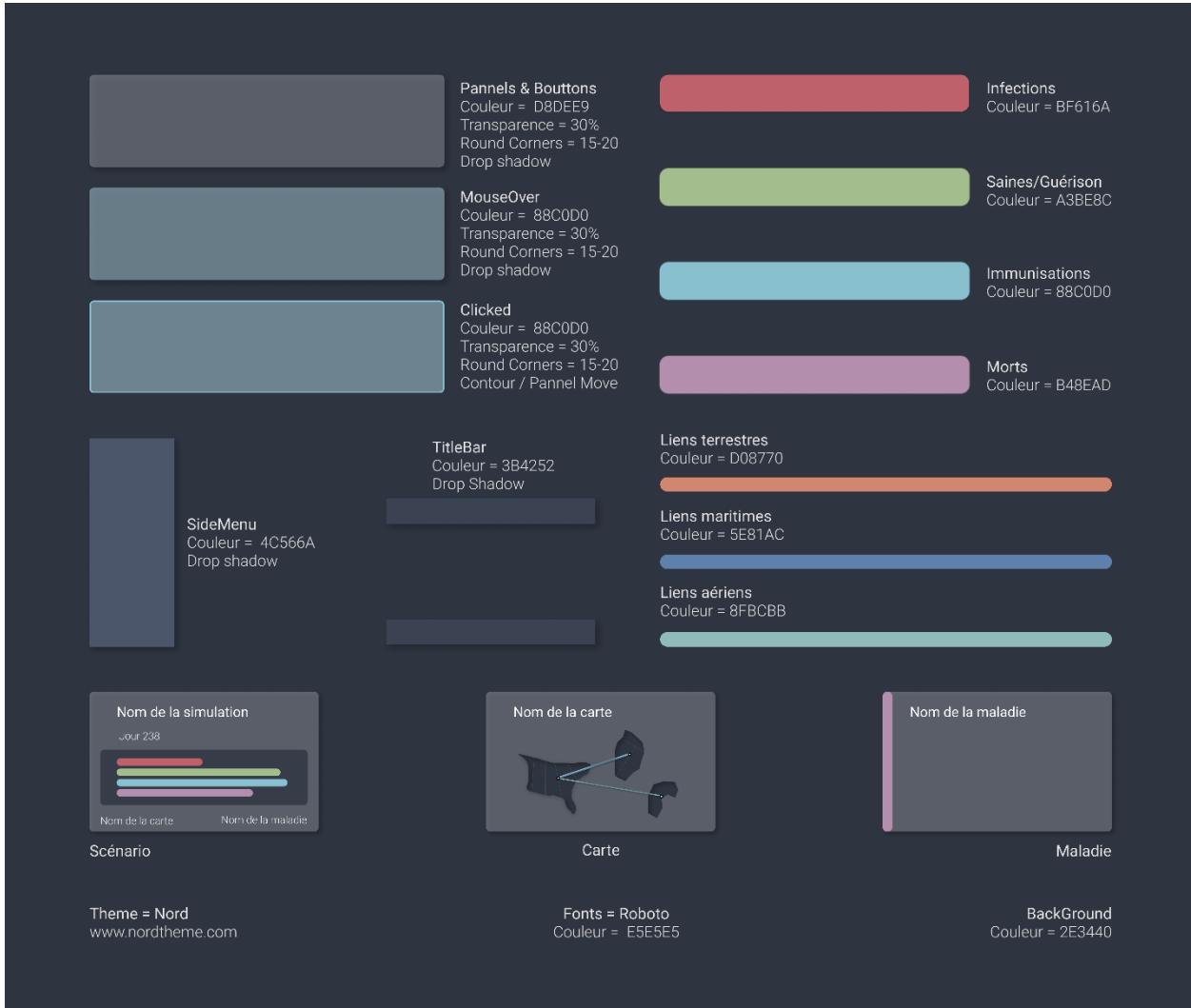
## Annexes

### 1.1 — Esquisses v2.1





## 1.2 — Thème et design v2.1



## 2.1 - Vision

L'application sera codée en Java et dotée d'une interface graphique à partir de laquelle le client pourra effectuer ses opérations. L'application a deux fonctions principales séparées.

La première est de pouvoir créer une carte mondiale avec les pays désirés (et les interconnexions entre ceux-ci). Dans ce mode de création, l'utilisateur peut utiliser des formes géométriques ou créer sa propre forme pour représenter un pays. Il peut ensuite connecter les pays par des voies terrestres, aériennes et/ou maritimes. Certaines informations devront être entrées selon les pays comme le nombre de régions, la population et la densité de population (ou la superficie).

La deuxième fonction de l'application est un mode simulation. L'utilisateur entre donc les informations concernant la maladie. Ces données sont principalement le taux de propagation, le taux de guérison et le taux de mortalité (possiblement ajouter les possibilités de mutation). À partir de ces informations, la simulation peut débuter. Le programme va s'exécuter en continu pour chaque unité de temps (qu'il nous reste à déterminer) qui représentera un « pas » dans l'application (une journée). L'utilisateur peut ainsi analyser la progression de la maladie et ensuite mettre la simulation sur pause pour introduire de nouvelles mesures et l'effet de ces mesures sur la propagation de la maladie.

## 2.2 – Modèle de domaine

### 2.2.1 — Diagramme de classes conceptuelles

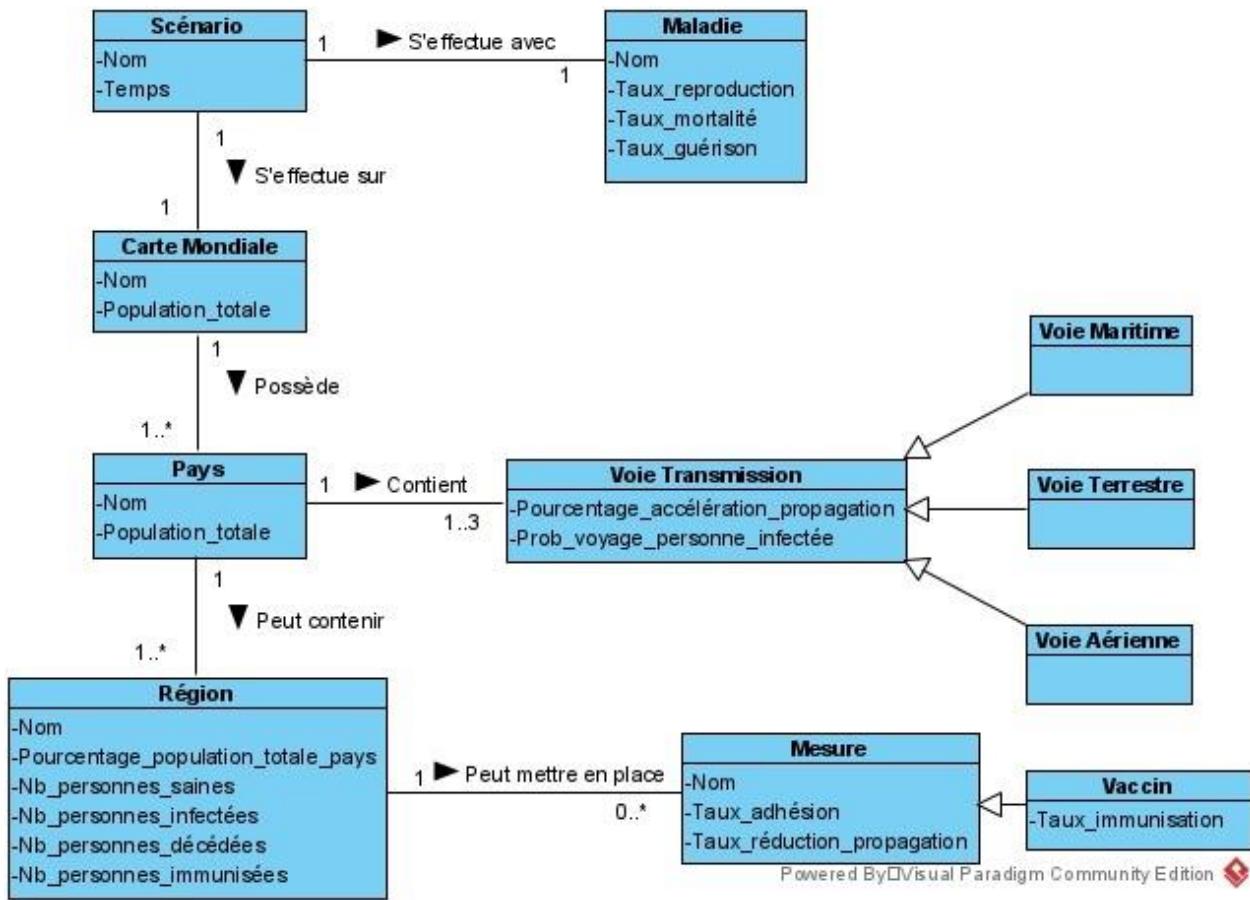


Diagramme 1.1 — Classes Conceptuelles

## 2.2.2 — Description de classes conceptuelles

CLASSE	DESCRIPTION
Scénario	Représente le lien entre la <b>Maladie</b> et la <b>Carte Mondiale</b> à utiliser pour la simulation. Possède un nom ainsi que le temps présent en jours de la simulation.
Maladie	Représente le virus et ses paramètres qui seront utilisés pendant la simulation, par le <b>Scénario</b> .
Carte Mondiale	Représente la carte préconfigurée qui sera utilisée par le <b>Scénario</b> et qui comporte plusieurs <b>Pays</b> . Compte la population totale à partir de ces derniers.
Pays	Représente une forme dans la <b>Carte Mondiale</b> qui peut être séparée en plusieurs <b>Régions</b> . Un <b>Pays</b> contient également une ou plusieurs voies de transmissions — nombre configuré par l'utilisateur.
Voie Transmission	Représente les voies de transmission de la <b>Maladie</b> pour la simulation. Une voie peut être <b>maritime, terrestre ou aérienne</b> . Chaque voie de transmission contient également son pourcentage d'accélération de propagation de la <b>Maladie</b> et la probabilité qu'une personne infectée voyage que l'utilisateur pourra configurer.
Région	Représente une subdivision d'un <b>Pays</b> . Une <b>Région</b> connaît son pourcentage de population selon la population totale de son <b>Pays</b> , le nombre de personnes saines, le nombre de personnes infectées, le nombre de personnes décédées et le nombre de personnes immunisées. Finalement, une <b>Région</b> peut aussi mettre en place diverses Mesures pour ralentir la transmission de la <b>Maladie</b> .
Mesure	Représente des opérations mises en place par l'utilisateur lors de la configuration du <b>Scénario</b> qui auront un impact sur la transmission de la <b>Maladie</b> .

## 2.3 — Modèle des cas d'utilisation

### 2.3.1 — Diagrammes des cas d'utilisation

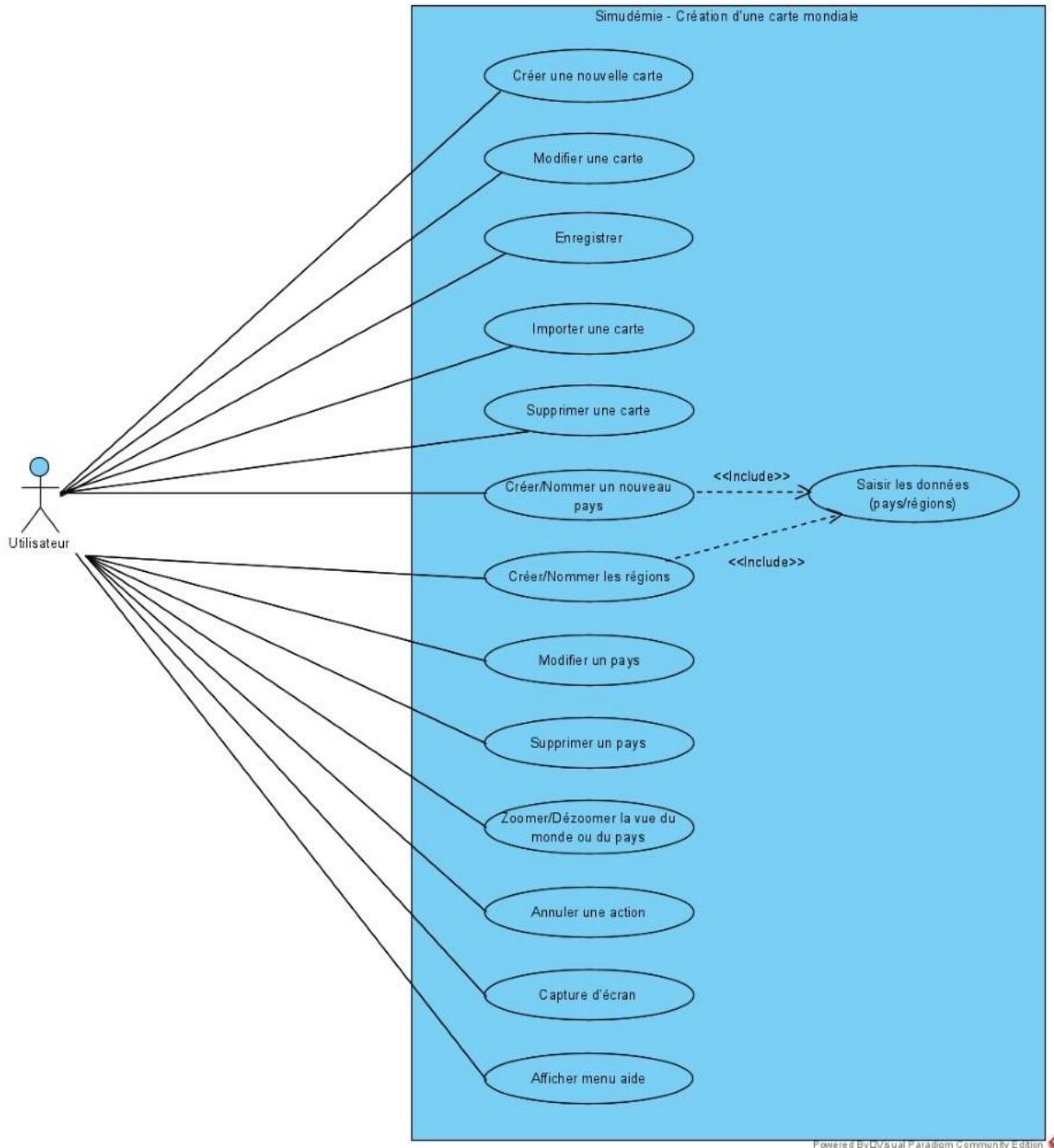


Diagramme 2.1 — Cas d'utilisation - Crédit à une carte mondiale

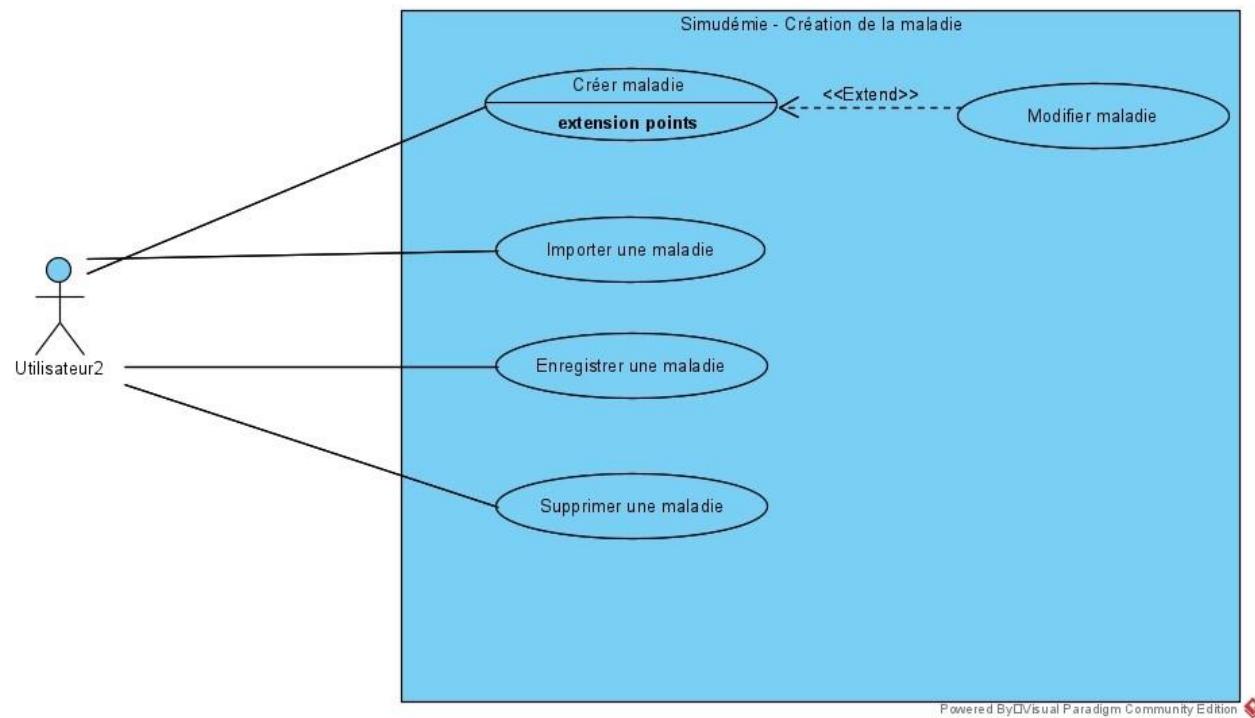


Diagramme 2.2 — Cas d'utilisation - Crédit Maladie

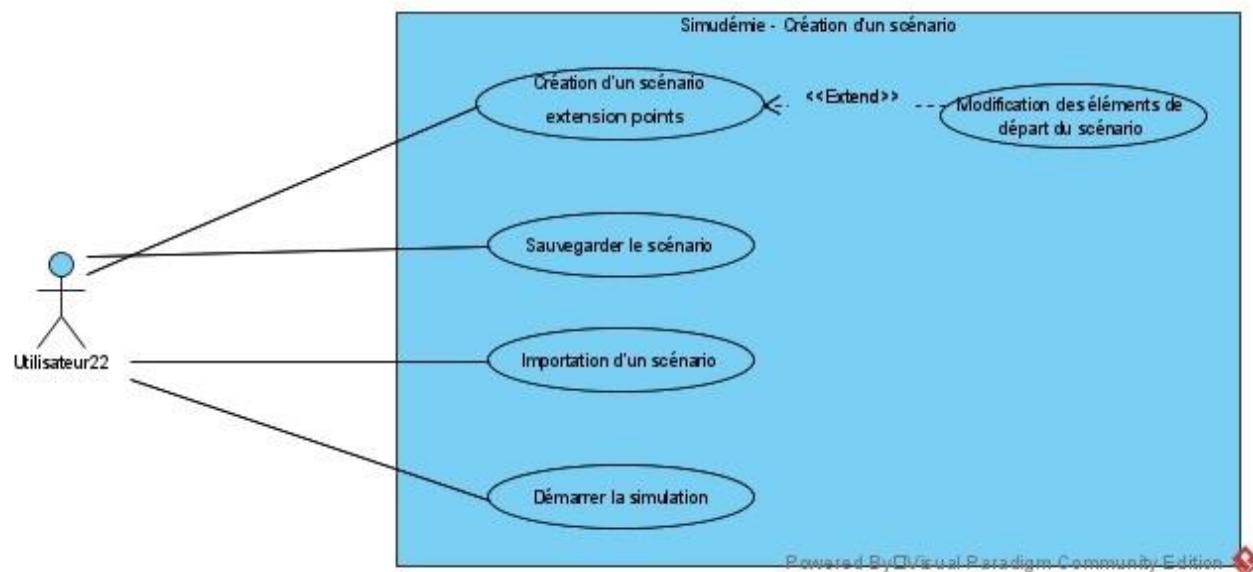


Diagramme 2.3 — Cas d'utilisation - Crédation Scénario

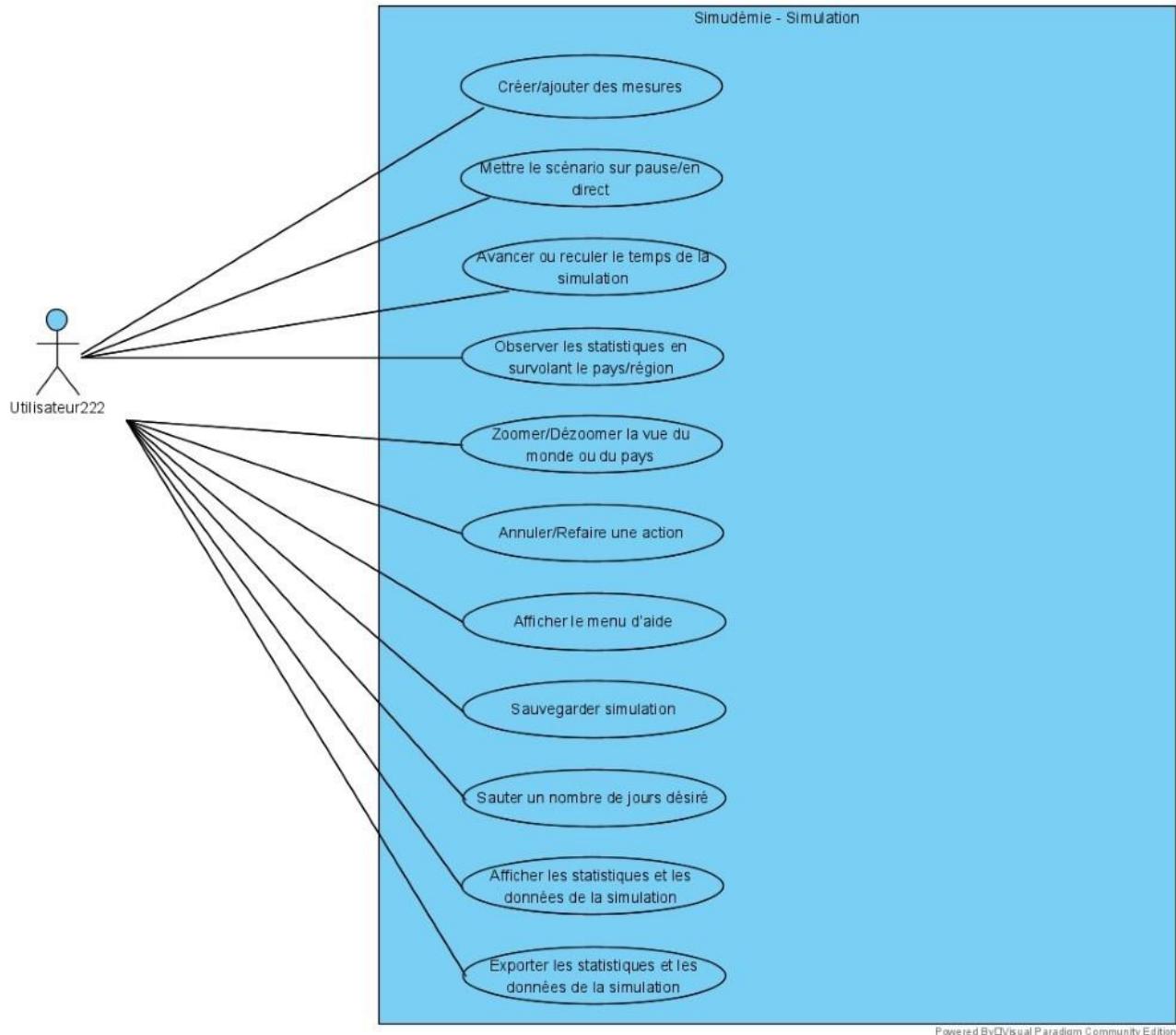


Diagramme 2.4 — Cas d'utilisation - Simulation

### 2.3.2 — Description des cas d'utilisation

#### Création carte

<b>Cas d'utilisation</b>	Créer une nouvelle carte
<b>Système</b>	Simudémie — Création de cartes
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire créer une nouvelle carte Programmeur : Offre une interface adéquate et stimulante à l'utilisateur
<b>Préconditions</b>	Aucune
<b>Garantie de succès</b>	L'utilisateur peut accéder aux interfaces suivantes (création de pays ou de régions)
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur sélectionne l'option de créer une nouvelle carte.	Ouvre le menu de création d'une carte
Saisis le nom de la nouvelle carte & enregistre	Génère un fichier de carte
<b>Scénario alternatif</b>	
Aucun	

<b>Cas d'utilisation</b>	Modifier une carte existante
<b>Système</b>	Simudémie — Création de cartes
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Veut modifier une carte Programmeur : Veut permettre la modification de la carte sans corrompre celle-ci
<b>Préconditions</b>	Il existe des cartes précédemment créées
<b>Garantie de succès</b>	La carte est modifiée sans erreur
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur sélectionne la carte existante voulant être modifiée	Ouvre le menu de création d'une carte et génère le contenu du fichier de la carte existante
L'utilisateur effectue les modifications désirées	Enregistre les modifications du fichier de la carte
<b>Scénario alternatif</b>	
Aucun	

<b>Cas d'utilisation</b>	Saisir les données (pays/régions)
<b>Système</b>	Simudémie — Création de cartes
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Souhaite entrer les statistiques aux pays et régions Programmeur : Souhaite que les statistiques soient associées aux bons objets
<b>Préconditions</b>	Pays ou région sélectionné(e)
<b>Garantie de succès</b>	Les statistiques sont enregistrées aux bons endroits
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur sélectionne pays/région	Affiche la fenêtre de saisie de données du pays/région sélectionné
L'utilisateur saisit les données	Enregistre les données
<b>Scénario alternatif</b>	
L'utilisateur saisit de mauvais paramètres	Message d'erreur demandant une nouvelle saisie

Cas d'utilisation	Importer une carte
Système	Simudémie — Création de cartes
Acteur	Utilisateur
Parties prenantes et intérêts	Utilisateur : Veut reprendre une carte précédemment sauvegarder Programmeur : Veut que le système reprenne exactement carte sauvegarder précédente en conservant les informations et les formes exactes.
Préconditions	Sauvegarde existante
Garantie de succès	La simulation est importée et lancée.
<b>Scénario principal</b>	
Utilisateur	Système Simudémie
L'utilisateur appuie sur le bouton [+]	Ouvre le menu de création d'une carte
L'utilisateur appuie sur le bouton [importer]	Ouvre le répertoire de fichiers
L'utilisateur choisit la carte désirée	Génère le contenu du fichier de la carte sélectionnée dans l'interface de création
L'utilisateur enregistre la carte	Enregistre le fichier de la carte dans le répertoire du système
<b>Scénario alternatif</b>	
L'utilisateur choisit un fichier corrompu	Message d'erreur demandant de choisir un fichier différent

---

<b>Cas d'utilisation</b>	Zoomer/Dézoomer la vue du monde ou du pays
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Secondaire
<b>Description</b>	L'utilisateur peut agrandir/réduire la vue sur la carte. En double-cliquant sur le pays désiré, la caméra effectue un zoom sur ce dernier.

<b>Cas d'utilisation</b>	Créer/Nommer un nouveau pays
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Primaire
<b>Description</b>	L'utilisateur veut créer/nommer un nouveau pays. Il choisit l'option d'ajout d'une forme au choix qui deviendra un pays et entre les informations nécessaires.

<b>Cas d'utilisation</b>	Créer/Nommer une nouvelle région
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Primaire
<b>Description</b>	Le client veut créer/nommer la nouvelle région. Il choisit l'option de découpage [bouton ciseaux] afin de découper le pays. Ce découpage deviendra une région et l'utilisateur entre les informations nécessaires.

---

<b>Cas d'utilisation</b>	Modifier un pays ou une région
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Secondaire
<b>Description</b>	L'utilisateur veut modifier un pays ou une région existante. Il sélectionne le pays ou la région à l'aide de l'option de sélection.

<b>Cas d'utilisation</b>	Supprimer un pays ou une région
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Secondaire
<b>Description</b>	L'utilisateur veut supprimer un pays ou une région existante. Il sélectionne le pays ou la région à l'aide de l'option suppression [bouton de la gomme à effacer].

<b>Cas d'utilisation</b>	Créer une voie de transmission
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Primaire
<b>Description</b>	Le client veut créer une nouvelle voie de transmission. Il sélectionne l'option des liens et ajoute la voie au choix entre les pays désirés.

---

Cas d'utilisation	Ajouter une couleur
Acteur(s)	Utilisateur
Type	Secondaire
Description	L'utilisateur veut ajouter une couleur à un pays. Il prend l'option de peinture et ajoute une couleur au pays.

Cas d'utilisation	Afficher menu aide
Acteur(s)	Utilisateur
Type	Secondaire
Description	L'utilisateur peut obtenir de l'aide en appuyant sur le bouton correspondant.

Cas d'utilisation	Capture d'écran
Acteur(s)	Utilisateur
Type	Secondaire
Description	L'utilisateur peut faire une capture d'écran des éléments présents sur la carte.

---

Cas d'utilisation	Enregistrer
Acteur(s)	Utilisateur
Type	Secondaire
Description	L'utilisateur veut enregistrer sa carte. Lorsque la carte est à son goût, il choisit l'option d'enregistrement.

## Création d'une maladie

<b>Cas d'utilisation</b>	Créer une nouvelle maladie
<b>Système</b>	Simudémie — Création de maladies
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire créer une nouvelle maladie Programmeur : Offre une interface adéquate et stimulante à l'utilisateur
<b>Préconditions</b>	Aucune
<b>Garantie de succès</b>	La maladie est créée avec succès et s'affiche maintenant dans l'interface
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur clique sur l'onglet [ajouter] dans le menu	Change d'onglet dans le menu
L'utilisateur ajoute une maladie avec le [+] dans la section respective	Ouvre la fenêtre de saisie des paramètres
L'utilisateur saisit les paramètres souhaités	Affiche la saisie des paramètres
L'utilisateur sauvegarde	Message de confirmation
<b>Scénario alternatif</b>	
L'utilisateur saisit de mauvais paramètres	Message d'erreur demandant une nouvelle saisie

<b>Cas d'utilisation</b>	Modifier une maladie existante
<b>Système</b>	Simudémie — Création de maladies
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire modifier une maladie Programmeur : Veut permettre la modification de tous les attributs de la maladie
<b>Préconditions</b>	Une maladie doit déjà exister
<b>Garantie de succès</b>	La maladie est belle et bien modifiée
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur clique sur l'onglet [ajouter] dans le menu	Change d'onglet dans le menu
L'utilisateur clique sur la maladie à modifier	Affiche la maladie
Il clique ensuite sur le bouton de modification	Ouvre la fenêtre de saisie des paramètres
Il modifie les champs souhaités et enregistre le tout après	Affiche la saisie des paramètres & Message de confirmation d'enregistrement
<b>Scénario alternatif</b>	
L'utilisateur saisit de mauvais paramètres	Message d'erreur demandant une nouvelle saisie

---

<b>Cas d'utilisation</b>	Importer une maladie
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Secondaire
<b>Description</b>	L'utilisateur veut importer une maladie déjà existante avec les informations exactes qu'elle avait.

<b>Cas d'utilisation</b>	Enregistrer une maladie
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Secondaire
<b>Description</b>	L'utilisateur veut enregistrer sa maladie. Lorsque la maladie convient, il choisit l'option d'enregistrement.

<b>Cas d'utilisation</b>	Supprimer une maladie
<b>Acteur(s)</b>	Utilisateur
<b>Type</b>	Secondaire
<b>Description</b>	L'utilisateur veut supprimer une maladie existante. Il sélectionne la maladie et clique sur la poubelle pour supprimer celle-ci.

## Création d'un scénario

<b>Cas d'utilisation</b>	Création d'un nouveau scénario
<b>Système</b>	Simudémie - Création scénario
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire créer un scénario Programmeur : Fournis une interface afin de choisir la carte, la maladie et le nom de ce scénario.
<b>Préconditions</b>	L'utilisateur devra avoir choisi une carte et une maladie.
<b>Garantie de succès</b>	L'utilisateur a maintenant un scénario prêt à rouler.
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur ajoute un scénario avec le [+]	Affiche la fenêtre de création de scénarios
L'utilisateur entre le nom du scénario, il choisit la carte à utiliser ainsi que la maladie.	Affichage visuel des éléments sélectionnés
Commencer le scénario pour générer la sauvegarde.	Génère la sauvegarde du scénario
<b>Scénario alternatif</b>	
L'utilisateur n'a pas choisi de carte ou de maladie	Message d'erreur indiquant les éléments manquants.

<b>Cas d'utilisation</b>	Modification des éléments de départ d'un scénario
<b>Système</b>	Simudémie - Création scénario
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire créer un scénario & modifier les éléments de départ du scénario Programmeur : Fournis une interface afin de choisir la carte, la maladie, le nom de ce scénario & les éléments de départ à modifier.
<b>Préconditions</b>	L'utilisateur devra avoir choisi une carte et une maladie afin de modifier ces éléments.
<b>Garantie de succès</b>	L'utilisateur a maintenant un scénario prêt à rouler avec un départ modifiable.
<b>Scénario principal</b>	
Utilisateur	Système Simudémie
L'utilisateur ajoute un scénario avec le [+]	Affiche la fenêtre de création de scénarios
L'utilisateur entre le nom du scénario, il choisit la carte à utiliser ainsi que la maladie.	Affichage visuel des éléments sélectionnés
L'utilisateur clique sur [modifier départ]	Affiche la fenêtre de modification de départ
L'utilisateur saisit les éléments souhaités	Affichage des éléments saisis par l'utilisateur
Enregistrer le scénario pour générer la sauvegarde.	Génère la sauvegarde du scénario
<b>Scénario alternatif</b>	
L'utilisateur saisit de mauvais paramètres	Message d'erreur demandant une nouvelle saisie

<b>Cas d'utilisation</b>	Modification des éléments de départ d'un scénario
<b>Système</b>	Simudémie — Création d'un scénario
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire reprendre le scénario qu'un collègue lui a envoyé Programmeur : Fournis l'option d'intégrer un scénario externe dans la plage locale de l'utilisateur
<b>Préconditions</b>	Le scénario à importer doit être un scénario valide créé par la même application
<b>Garantie de succès</b>	L'utilisateur peut utiliser le scénario comme s'il était le sien
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur ajoute un scénario avec le [+]	Affiche la fenêtre de création de scénarios
L'utilisateur clique sur l'icône d'importation	Affiche les répertoires de fichiers de l'utilisateur
L'utilisateur choisit le fichier de scénario désiré	Importe et génère les informations contenues dans le fichier du scénario
<b>Scénario alternatif</b>	
L'utilisateur choisit un fichier corrompu	Message d'erreur demandant de choisir un nouveau fichier

---

Cas d'utilisation	Supprimer un scénario
Acteur(s)	Utilisateur
Type	Secondaire
Description	L'utilisateur veut supprimer un scénario existant. Il sélectionne le scénario et clique sur la poubelle afin de le supprimer.

**Simulation**

<b>Cas d'utilisation</b>	Créer/ajouter des mesures
<b>Système</b>	Simudémie — Simulation
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire ajouter une mesure à la simulation Programmeur : Va représenter les effets de la mesure dans la simulation (ajuster les taux, les liens, un vaccin, etc.)
<b>Préconditions</b>	La simulation doit être en cours
<b>Garantie de succès</b>	Les mesures seront présentes et prêtent à l'utilisation.
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
Appuie sur le bouton [+]	Affiche la liste des mesures disponibles à l'ajout dans la simulation
L'utilisateur choisit de créer une nouvelle mesure en appuyant sur [nouvelle +]	Affiche la fenêtre de configuration d'une mesure
Il saisit les paramètres de la mesure et appuie sur le bouton appliquer	Affiche la saisie des paramètres, enregistre ces derniers et applique ceux-ci à la simulation
<b>Scénario alternatif</b>	
Appuie sur le bouton [+]	Affiche la liste des mesures disponibles à l'ajout dans la simulation
L'utilisateur choisit la mesure désirée établie par défaut	Applique cette mesure à la simulation présente
<b>Scénario alternatif</b>	
L'utilisateur saisit de mauvais paramètres	Affiche un message d'erreur demandant de recommencer la saisie

<b>Cas d'utilisation</b>	Mettre le scénario sur pause/en direct
<b>Système</b>	Simudémie — Simulation
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Peut mettre la simulation sur pause pour effectuer des actions la tête tranquille
<b>Préconditions</b>	La simulation doit exister
<b>Garantie de succès</b>	L'utilisateur arrive à mettre sur pause la simulation sans rien et arrive à la redémarrer sans rien briser
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
Lorsque l'utilisateur est dans la simulation, il peut cliquer sur le bouton rouge/vert pour arrêter ou continuer la simulation	Mets sur pause ou en direct la simulation
<b>Scénario alternatif</b>	
Aucun	

<b>Cas d'utilisation</b>	Avancer ou reculer le temps de la simulation
<b>Système</b>	Simudémie — Simulation
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire avancer ou reculer la simulation Programmeur : Offre la possibilité d'avancer ou de reculer la simulation à l'aide d'une bande glissante
<b>Préconditions</b>	La simulation doit exister
<b>Garantie de succès</b>	La simulation a bel et bien avancé ou reculé dans le temps
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
Lorsque l'utilisateur est dans la simulation en pause en cliquant sur le bouton vert.	Mets la simulation sur pause
Il glisse la barre de jours à gauche et à droite afin de reculer ou avancer, respectivement.	Affiche et génère les jours où la barre de jour atterrit
Glisse à gauche complètement, il retournera au départ de la simulation.	Affiche le jour de départ de la simulation
<b>Scénario alternatif</b>	
Aucun	

Cas d'utilisation	Sauter le temps de la simulation
Système	Simudémie — Simulation
Acteur	Utilisateur
Parties prenantes et intérêts	Utilisateur : Désire sauter le temps de la simulation d'un certain nombre de jours, afin de voir les effets de ses mesures ou vaccins. Programmeur : Permet à l'utilisateur de sauter un nombre de jours désirés et observer les effets de ce saut rapidement
Préconditions	La simulation doit exister
Garantie de succès	La simulation a bel et bien sauté les jours demandés et généré les changements de ce saut temporel
<b>Scénario principal</b>	
Utilisateur	Système Simudémie
Lorsque l'utilisateur est dans la simulation en pause en cliquant sur le bouton vert.	Mets la simulation sur pause
L'utilisateur appuie sur le bouton de saut dans le temps	Affiche la fenêtre de configuration de saut temporel
Saisis le nombre de jours souhaité et applique le saut	Applique le saut et génère les changements de ce saut temporel
<b>Scénario alternatif</b>	
L'utilisateur saisit un nombre de jours trop élevé	Affiche un message d'erreur et demande de saisir un nombre moins élevé

<b>Cas d'utilisation</b>	Observer les statistiques en survolant le pays/région
<b>Système</b>	Simudémie — Simulation
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : Désire avoir de l'information sur une zone Programmeur : Offre la possibilité de consulter les statistiques de ces zones
<b>Préconditions</b>	La simulation doit être en cours
<b>Garantie de succès</b>	Les informations d'un pays/région sont bien affichées
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur survole avec le curseur sur un pays/région	Affiche les statistiques du pays/région survolé sur une fenêtre contextuelle
L'utilisateur clique sur le bouton des vues [bouton de l'œil] et prend la vue désirée	Change la vue des statistiques sur la carte et ajuste les taux et les couleurs en fonction de ce qui est demandé par l'utilisateur
<b>Scénario alternatif</b>	
Aucun	

Cas d'utilisation	Enregistrer simulation
Acteur(s)	Utilisateur
Type	Secondaire
Description	L'utilisateur peut sauvegarder toutes les données relatives à la simulation pour la reprendre plus tard.

Cas d'utilisation	Afficher les statistiques et les données de la simulation	
Système	Simudémie — Simulation	
Acteur	Utilisateur	
Parties prenantes et intérêts	Utilisateur : veut visionner les statistiques de la simulation Programmeur : offre une interface pour visionner les statistiques pour les jours voulus	
Préconditions	La simulation doit en cours	
Garantie de succès	L'interface de statistiques s'affiche bien et montre les bonnes données	
Scénario principal		
Utilisateur		Système Simudémie
L'utilisateur clique sur l'icône statistique		Affiche la fenêtre des statistiques de la simulation
L'utilisateur saisit les informations désirant être affichées sur le graphique		Ajuste les informations saisies par l'utilisateur sur le graphique
Scénario alternatif		
Aucun		

<b>Cas d'utilisation</b>	Exporter les statistiques et les données de la simulation
<b>Système</b>	Simudémie — Simulation
<b>Acteur</b>	Utilisateur
<b>Parties prenantes et intérêts</b>	Utilisateur : veut pouvoir exporter les statistiques de la simulation Programmeur : donne la possibilité de le faire avec un clic et une interface pour choisir où déposer le fichier
<b>Préconditions</b>	La simulation doit être en cours
<b>Garantie de succès</b>	Un fichier de statistiques est créé et déposé dans un lieu au choix de l'utilisateur
<b>Scénario principal</b>	
<b>Utilisateur</b>	<b>Système Simudémie</b>
L'utilisateur clique sur l'icône statistique	Affiche la fenêtre des statistiques de la simulation
L'utilisateur clique sur le bouton exporter	Ouvre la fenêtre de répertoire de fichiers
L'utilisateur choisit et confirme la destination d'export	Génère un fichier statistique à la destination désirée
<b>Scénario alternatif</b>	
Aucun	

## 2.3.3 — Diagramme de séquence système (DSS)

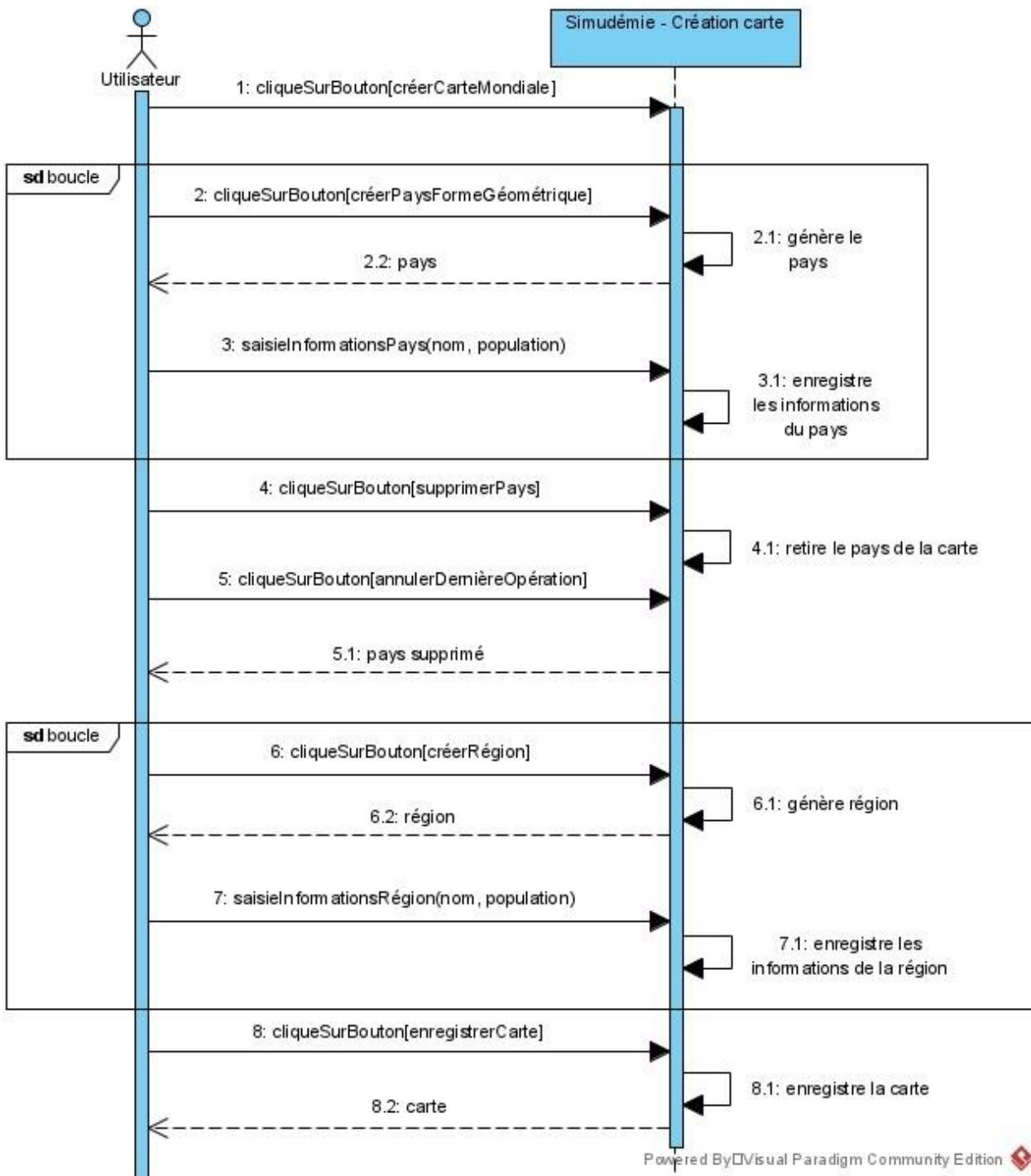


Diagramme 3.1 — DSS — Crédit à la carte

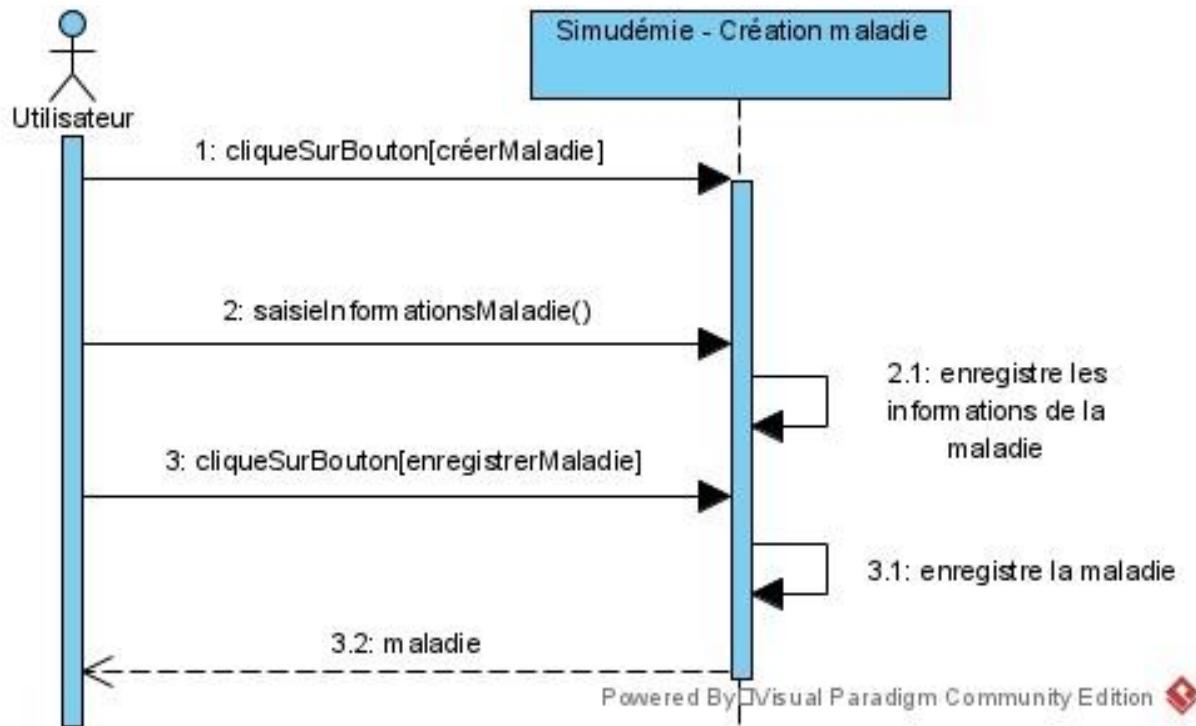


Diagramme 3.2 — DSS — Crédit Maladie

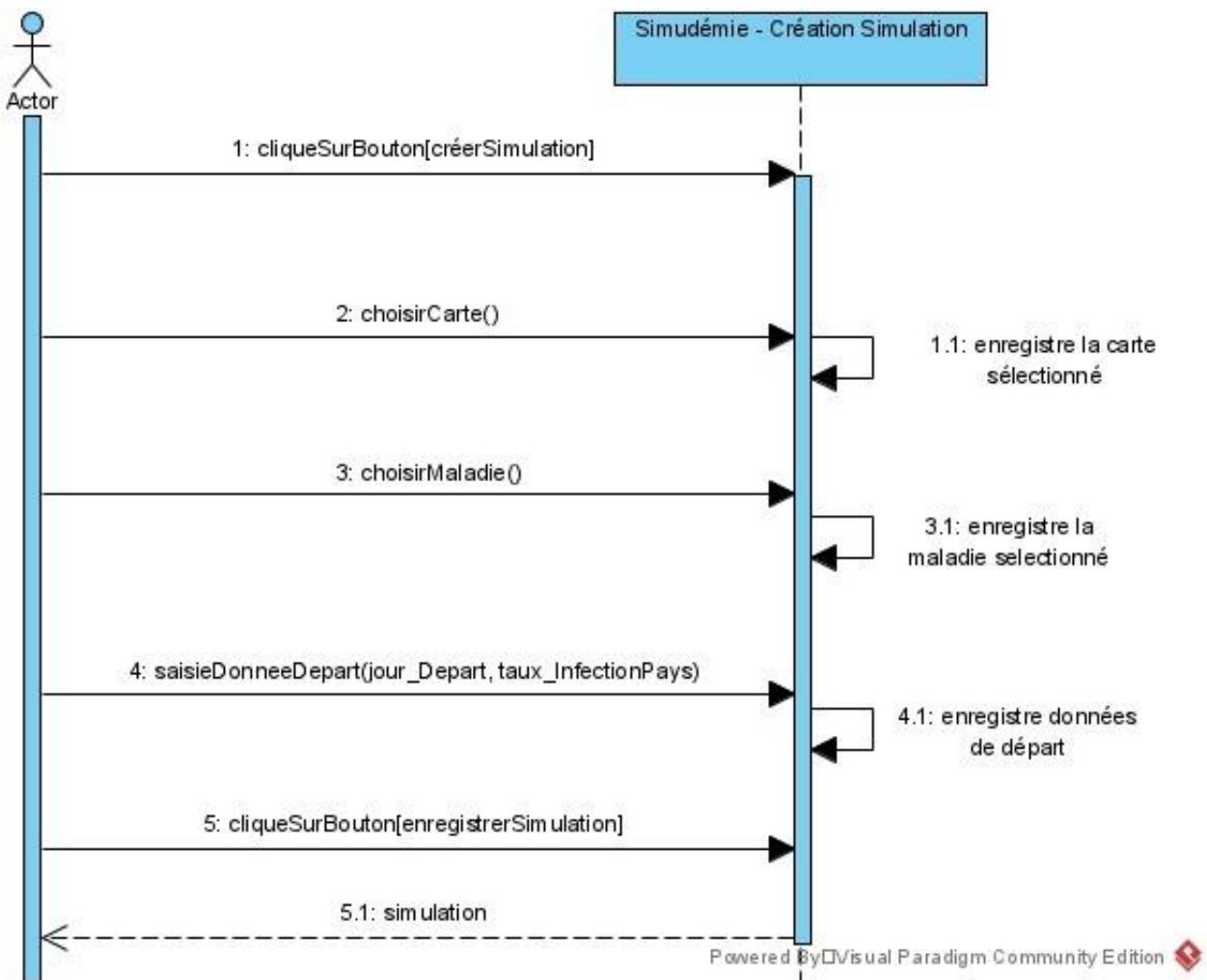


Diagramme 3.3 — DSS — Crédit Scénario

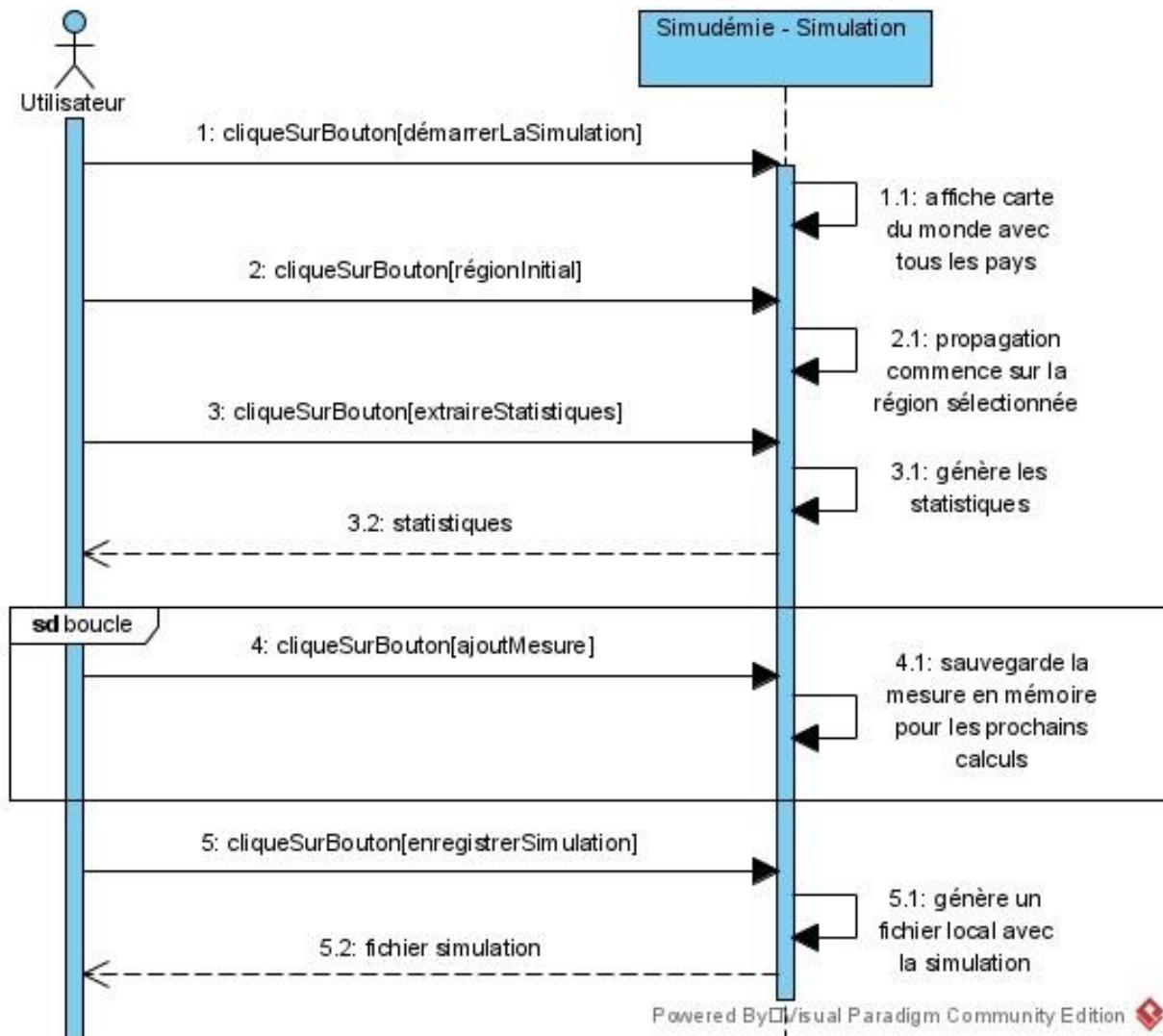
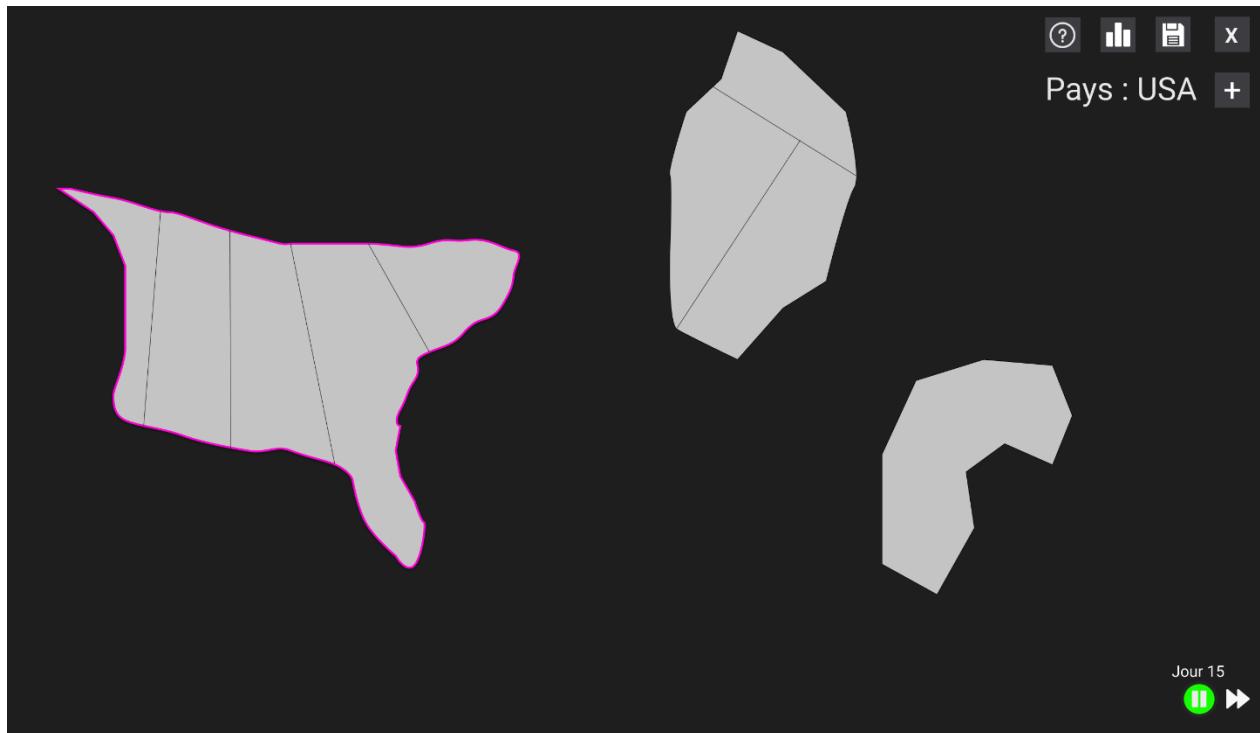
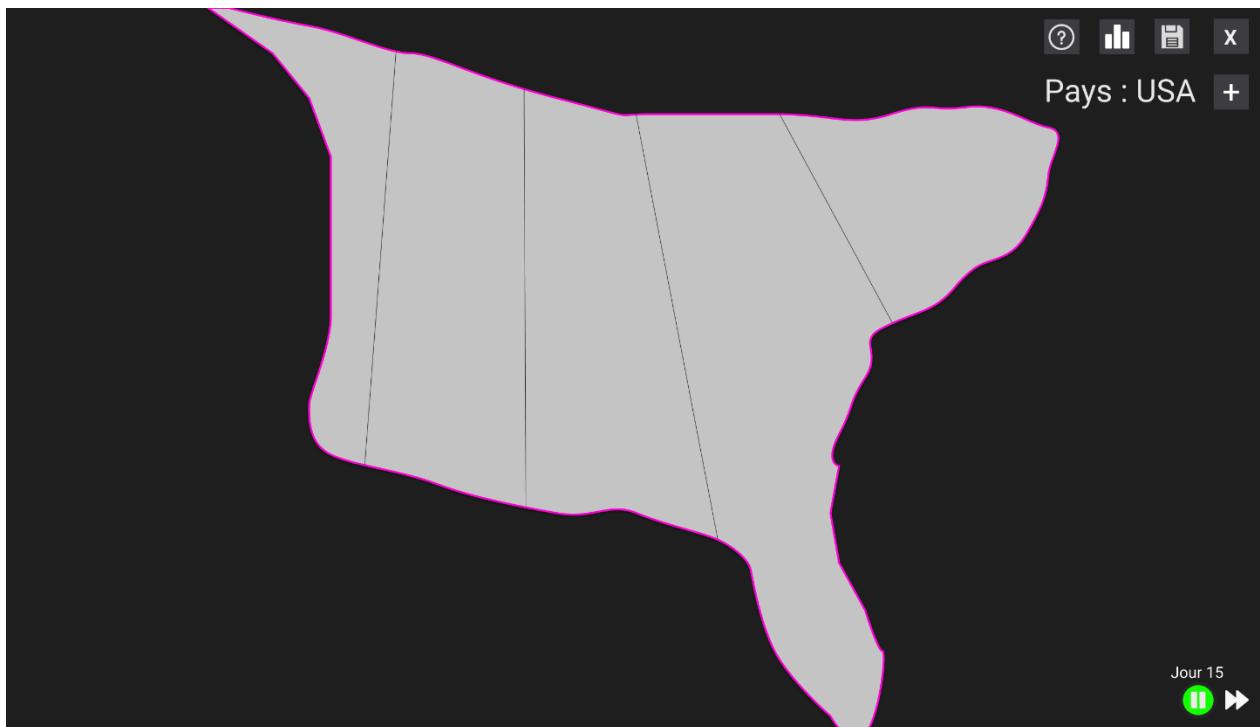


Diagramme 3.4 — DSS — Simulation

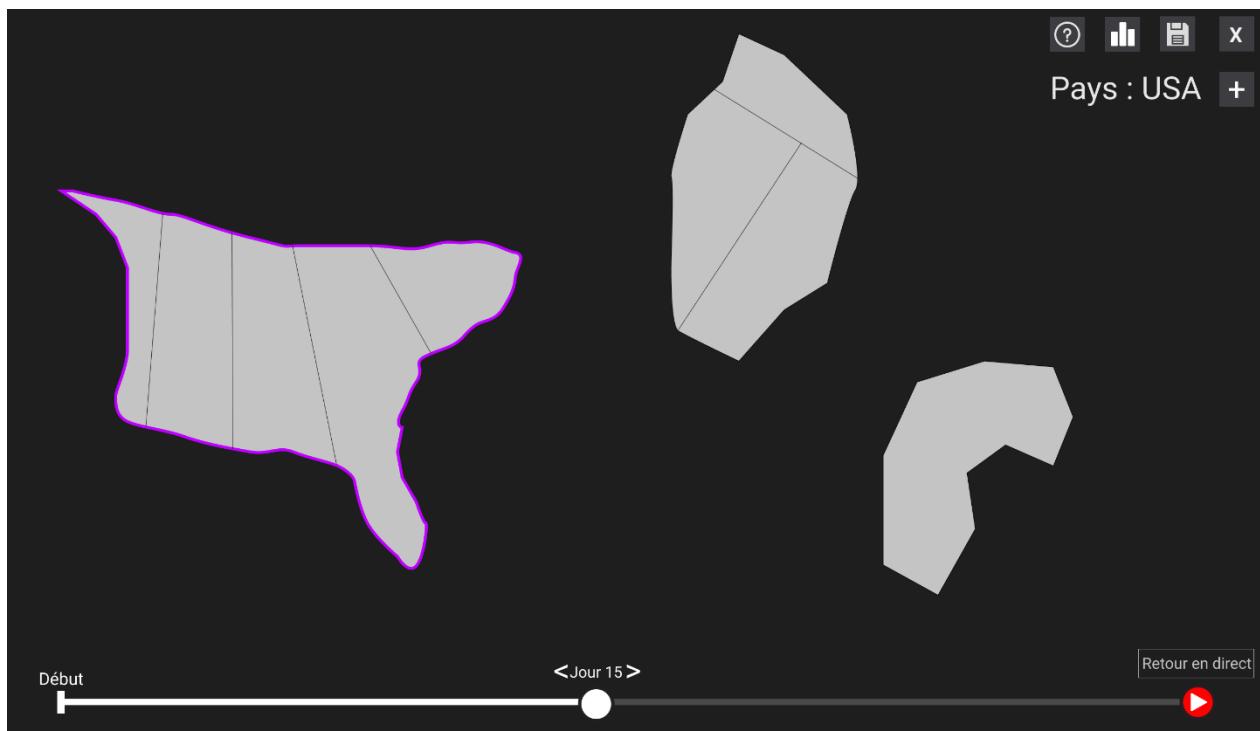
### 3.1 — Esquisses des interfaces utilisateur v1.0



*Esquisse 1.1 — Simulation — En Direct*



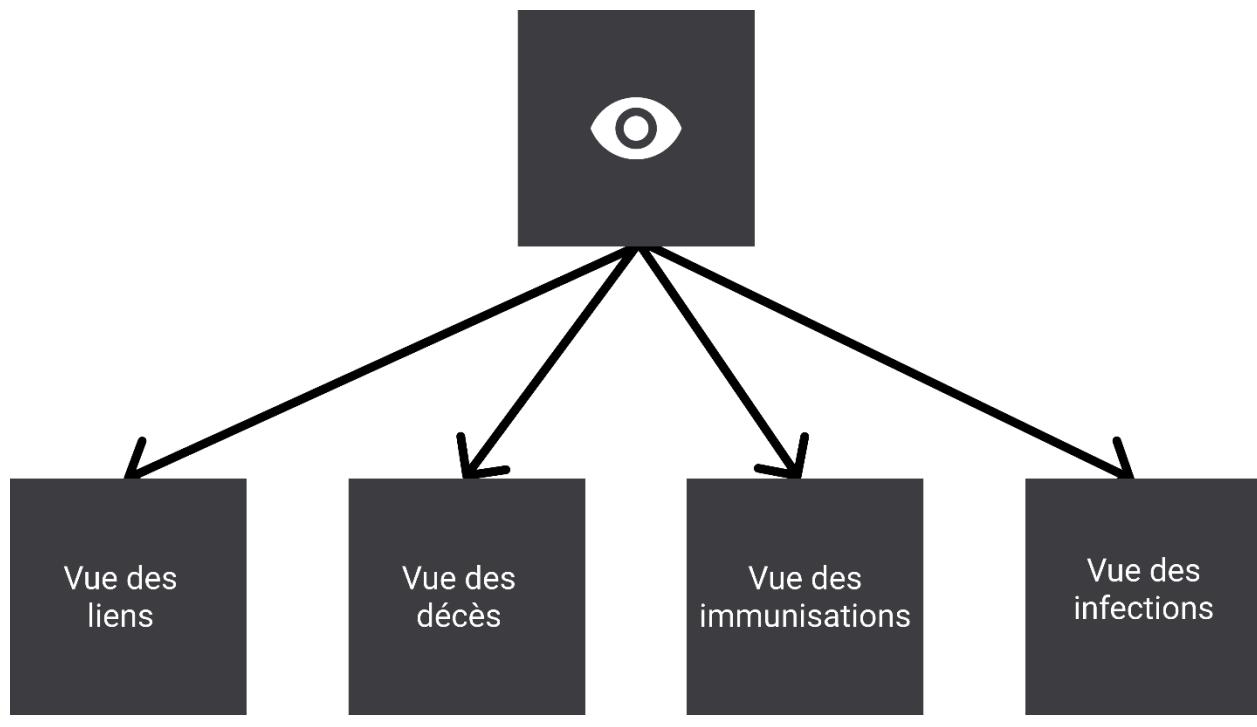
*Esquisse 1.2 — Simulation — En Direct + Zoom*



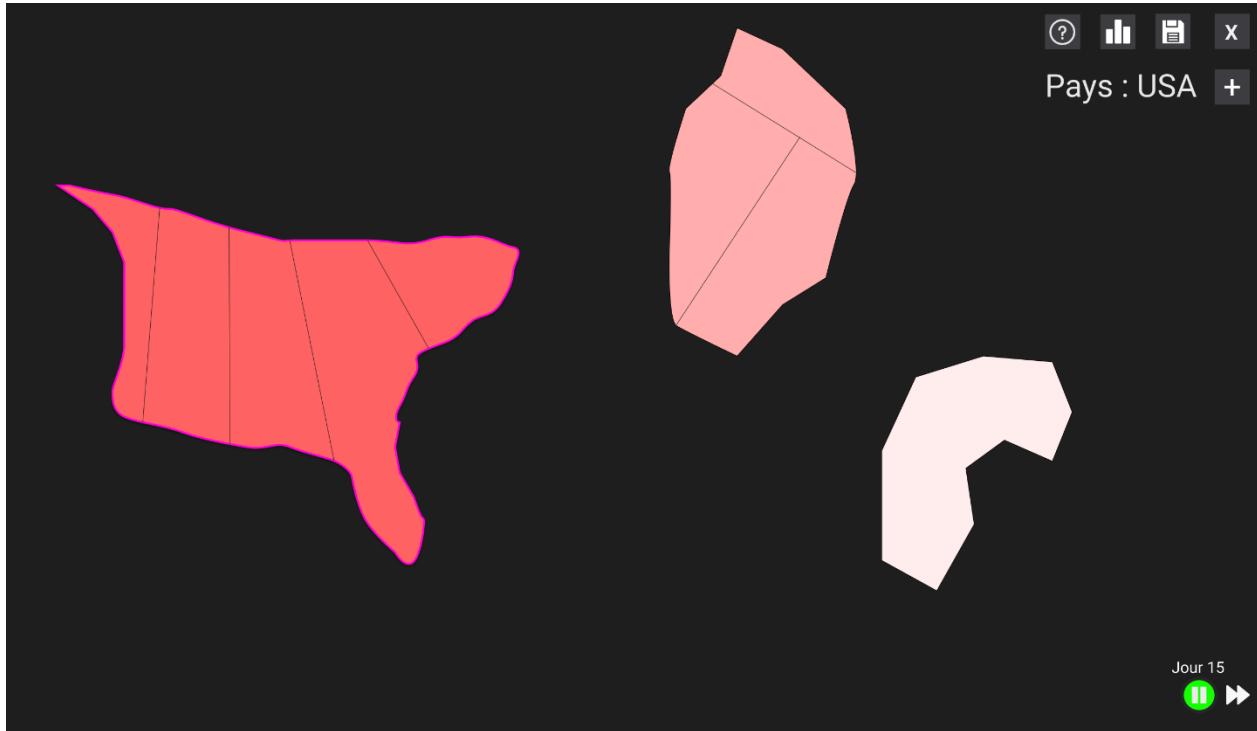
*Esquisse 1.3 — Simulation — En Pause*



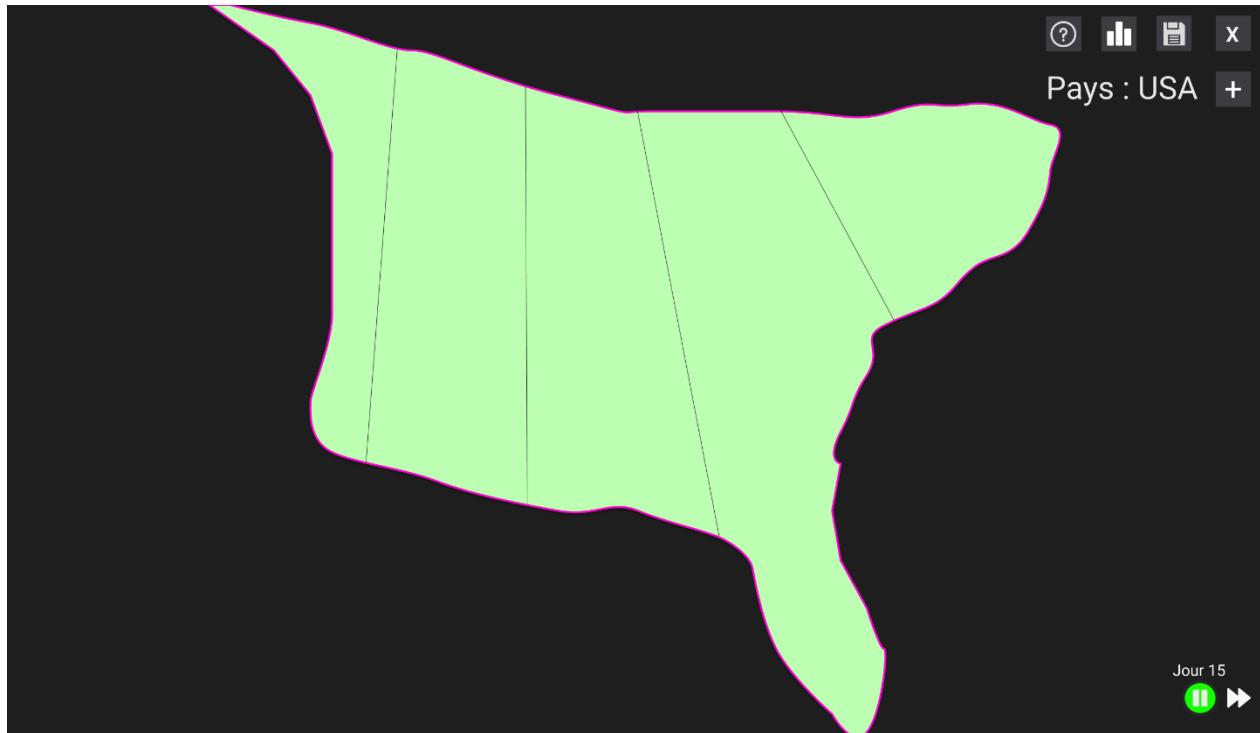
*Esquisse 1.4 — Simulation — En Pause + Zoom*



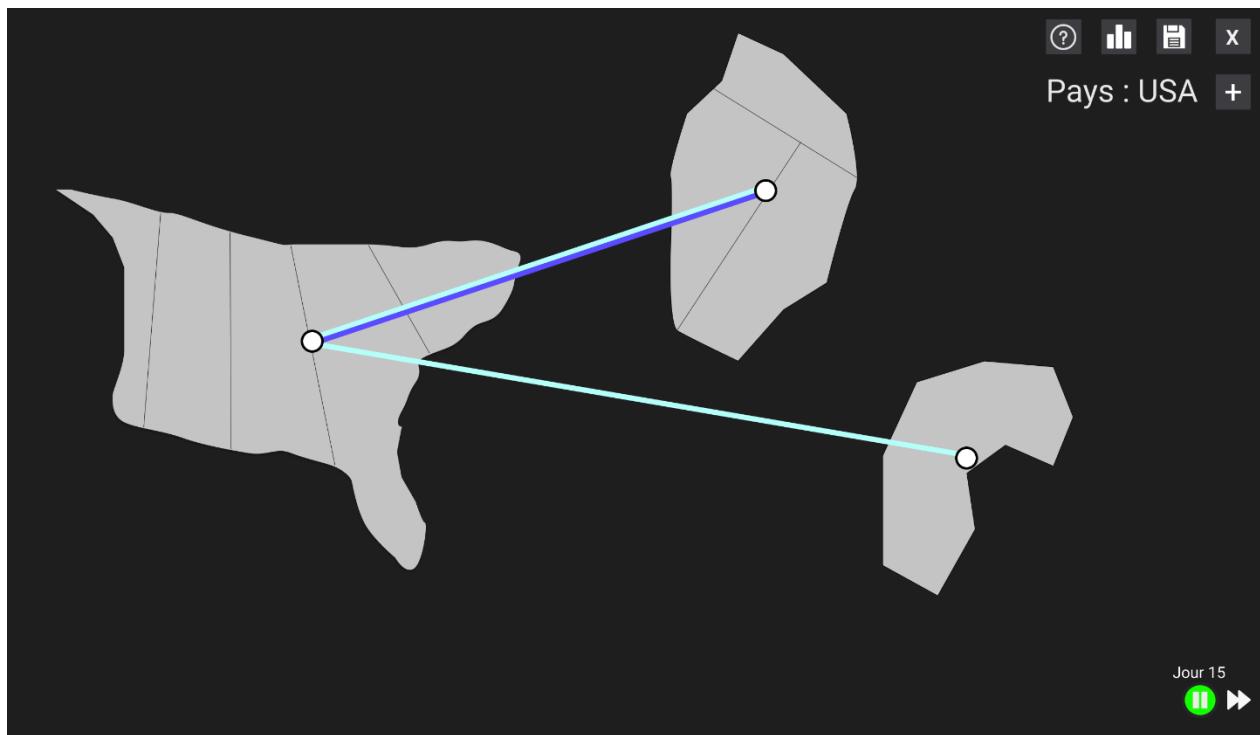
Esquisse 1,5 — Simulation — Bouton Vue



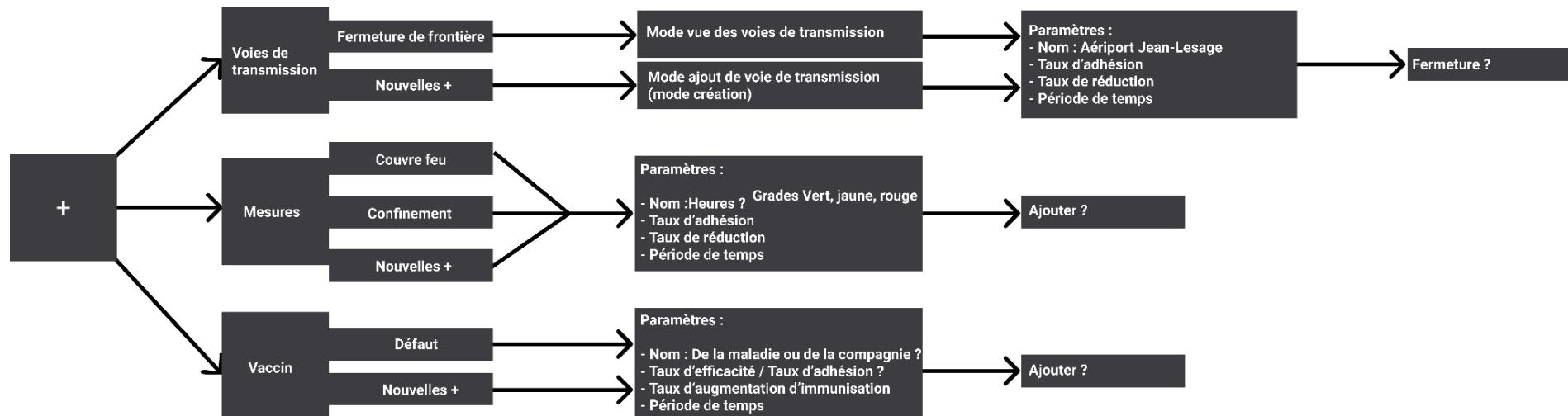
Esquisse 1,6 — Simulation — Vue Infection



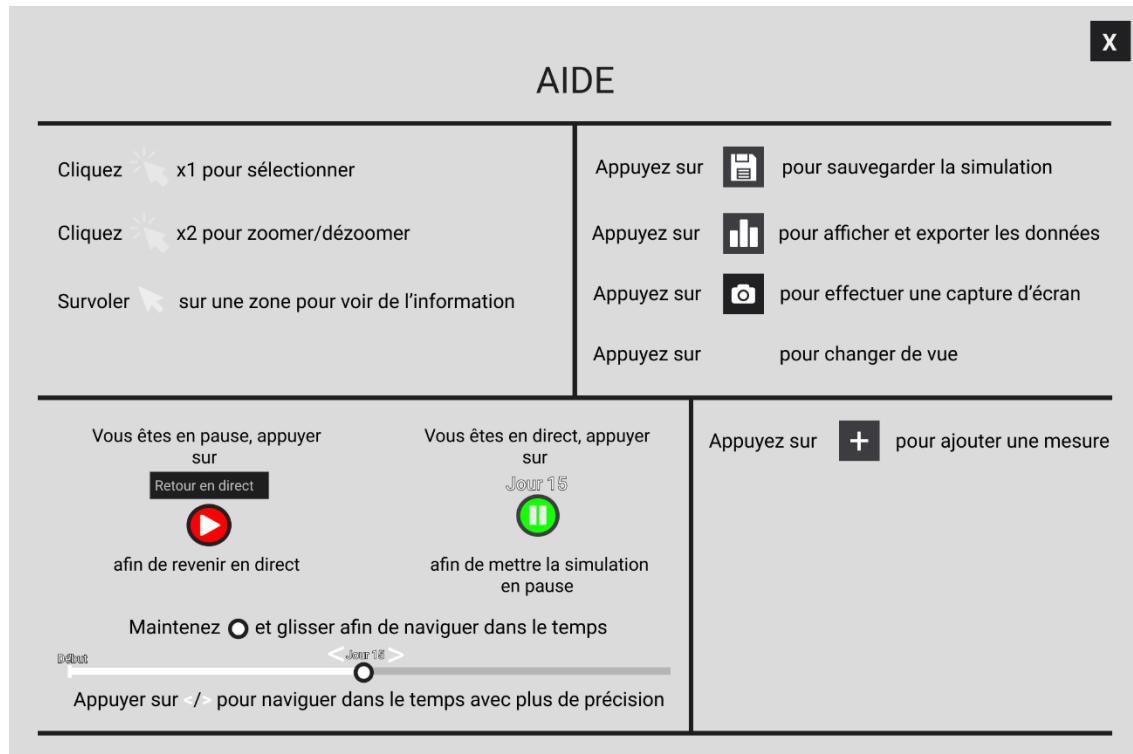
Esquisse 1,7 — Simulation — Vue Immunisation



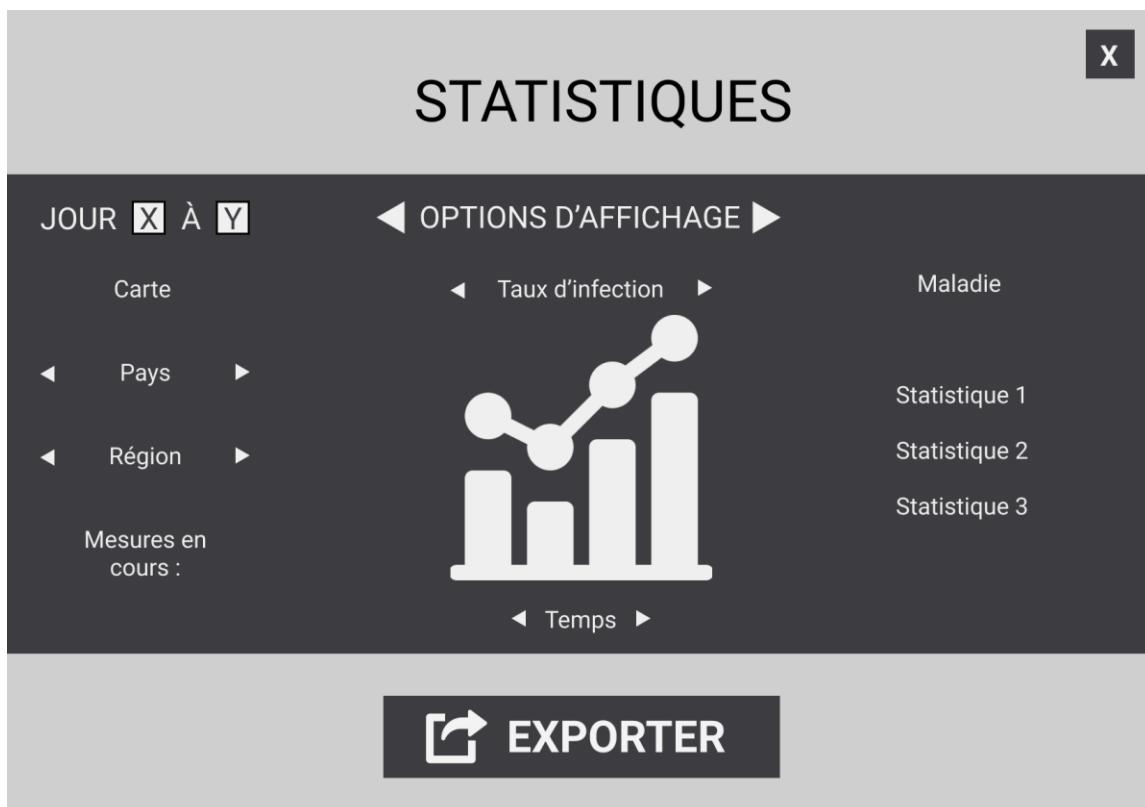
Esquisse 1,8 — Simulation — Vue Liens



Esquisse 1,9 — Simulation — Bouton Ajout



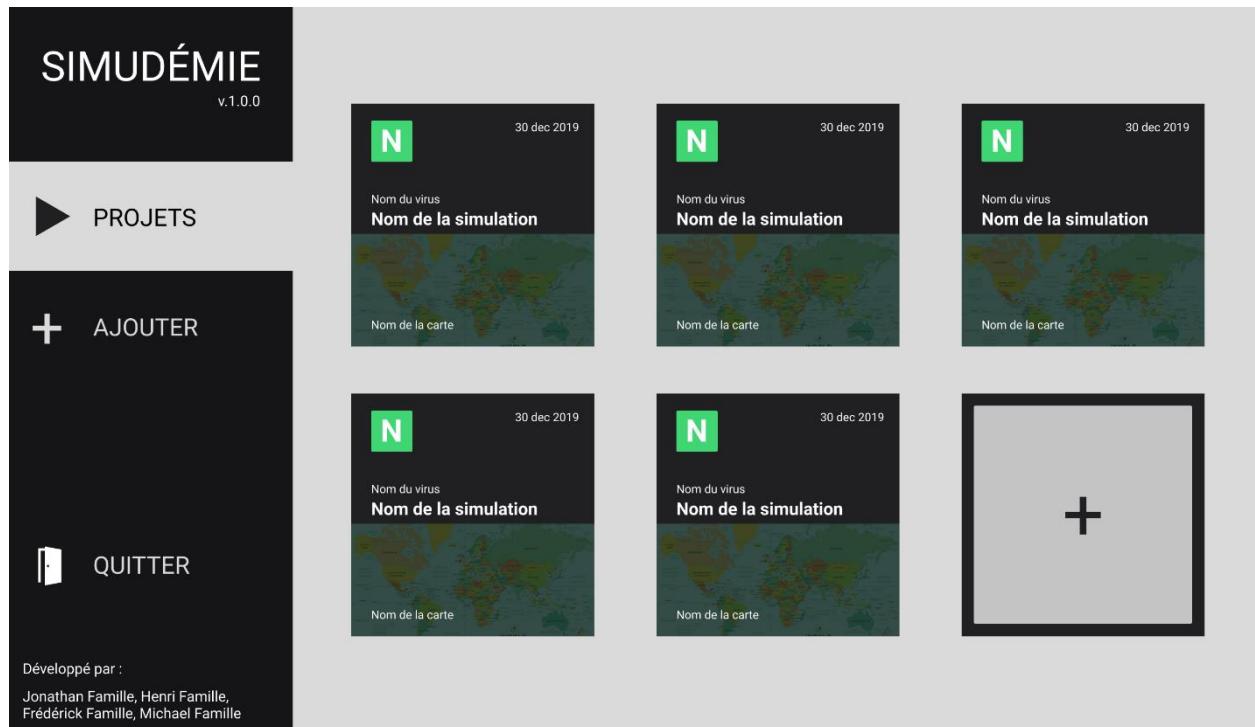
Esquisse 1,10 — Simulation — Aide



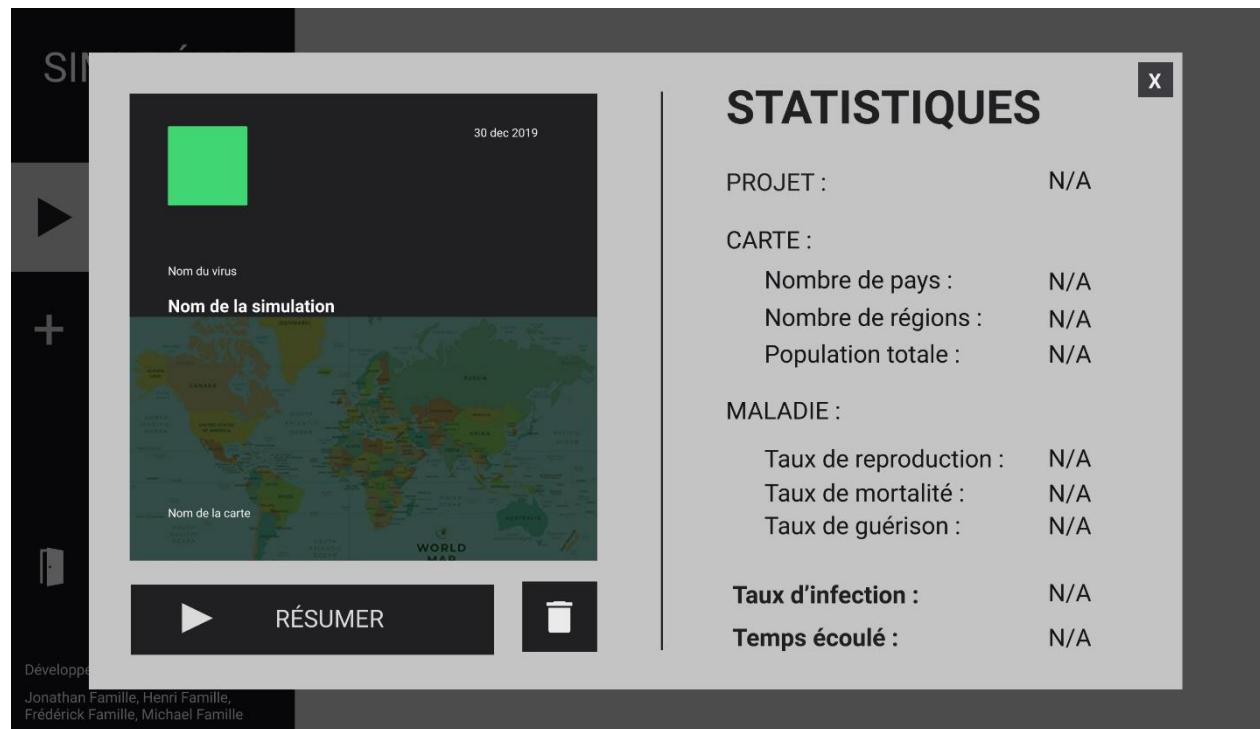
Esquisse 1,11 — Simulation — Statistiques



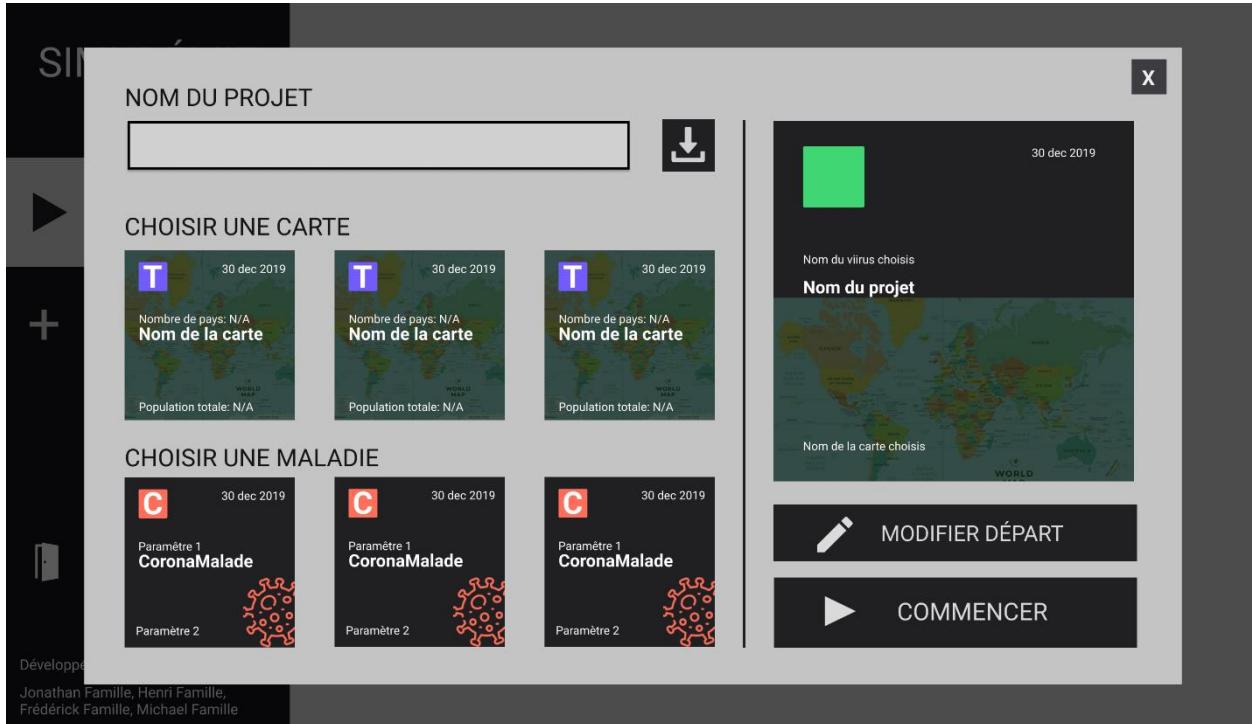
Esquisse 2.1 — Menus Scénarios — Logistique



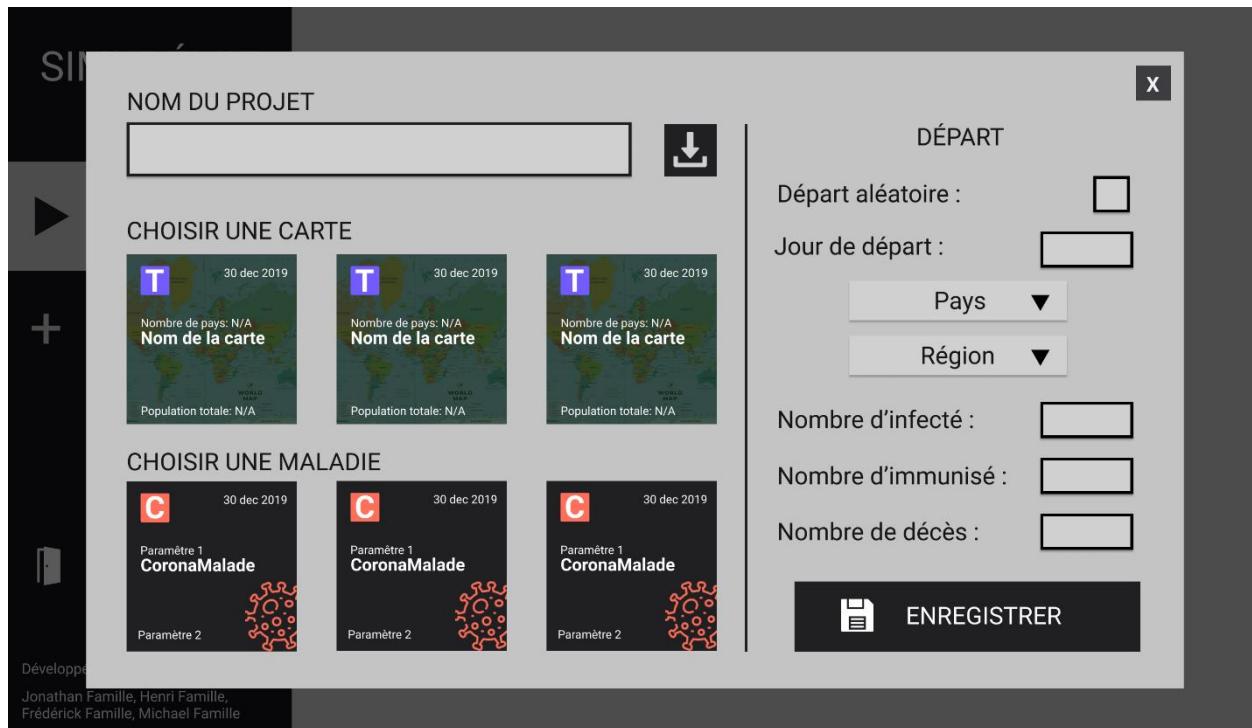
*Esquisse 2.2 — Menus Scénarios — Choix d'un scénario*



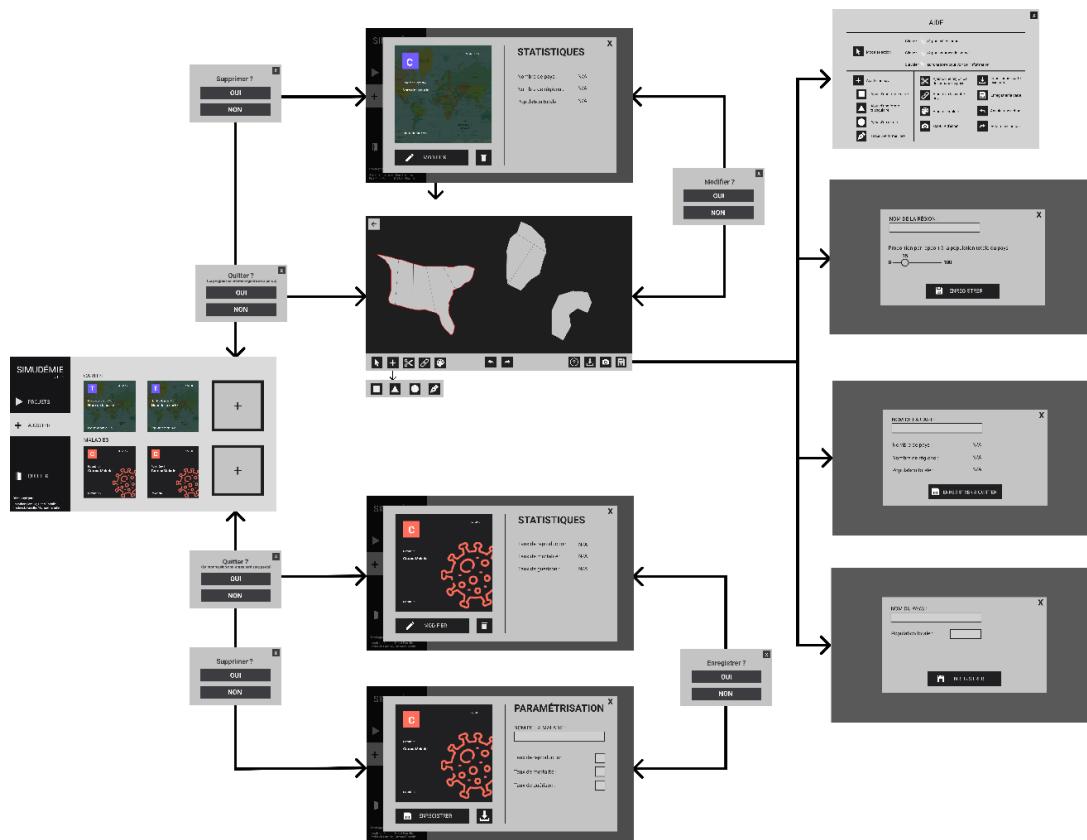
*Esquisse 2.3 — Menus Scénarios - Affiche Scénario Existant*



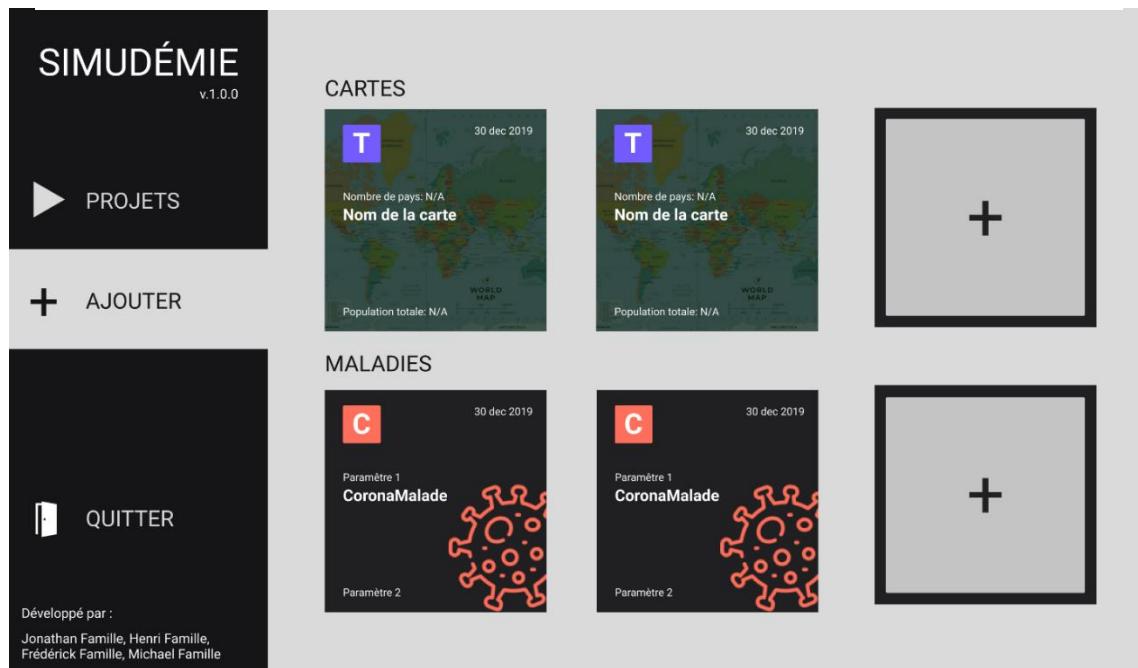
*Esquisse 2.4 — Menus Scénarios — Création Nouveau Scénario*



*Esquisse 2.5 — Menus Scénarios — Création Nouveau Scénario — Modification Départ*



Esquisse 3.1 — Menus Création — Logistique



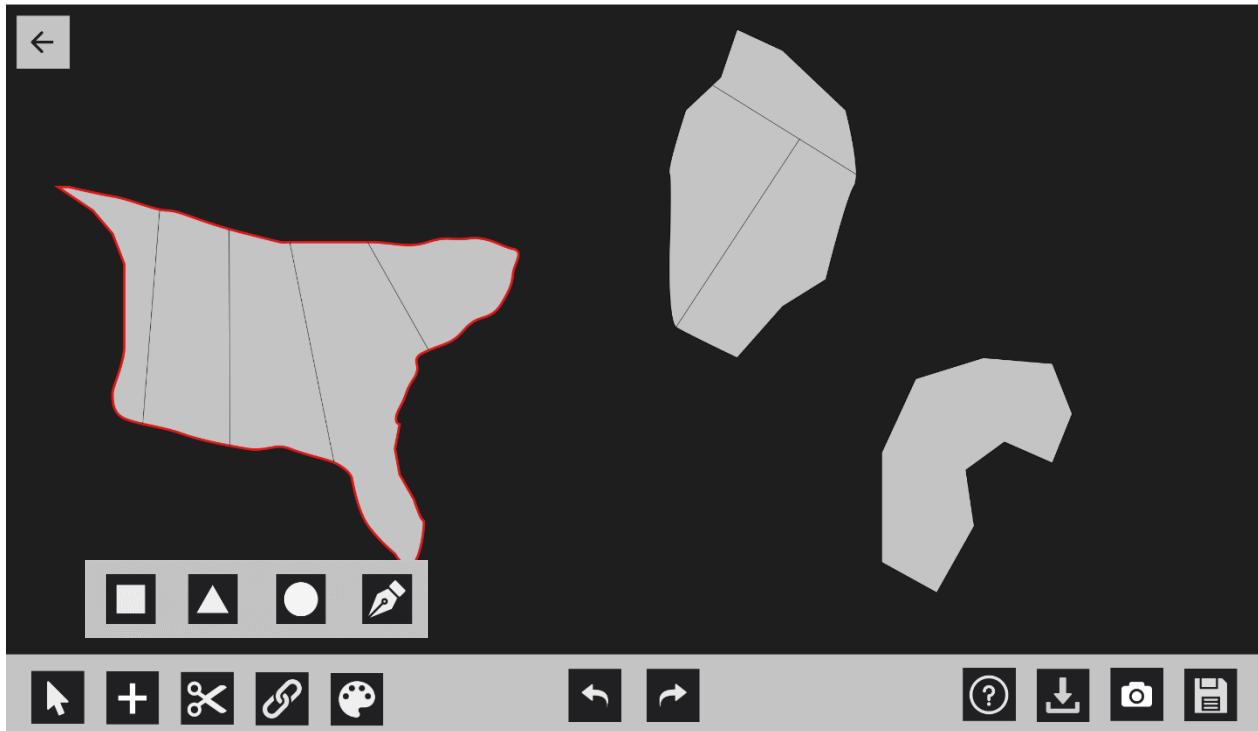
Esquisse 3.2 — Menus Création - Choix Carte & Maladie



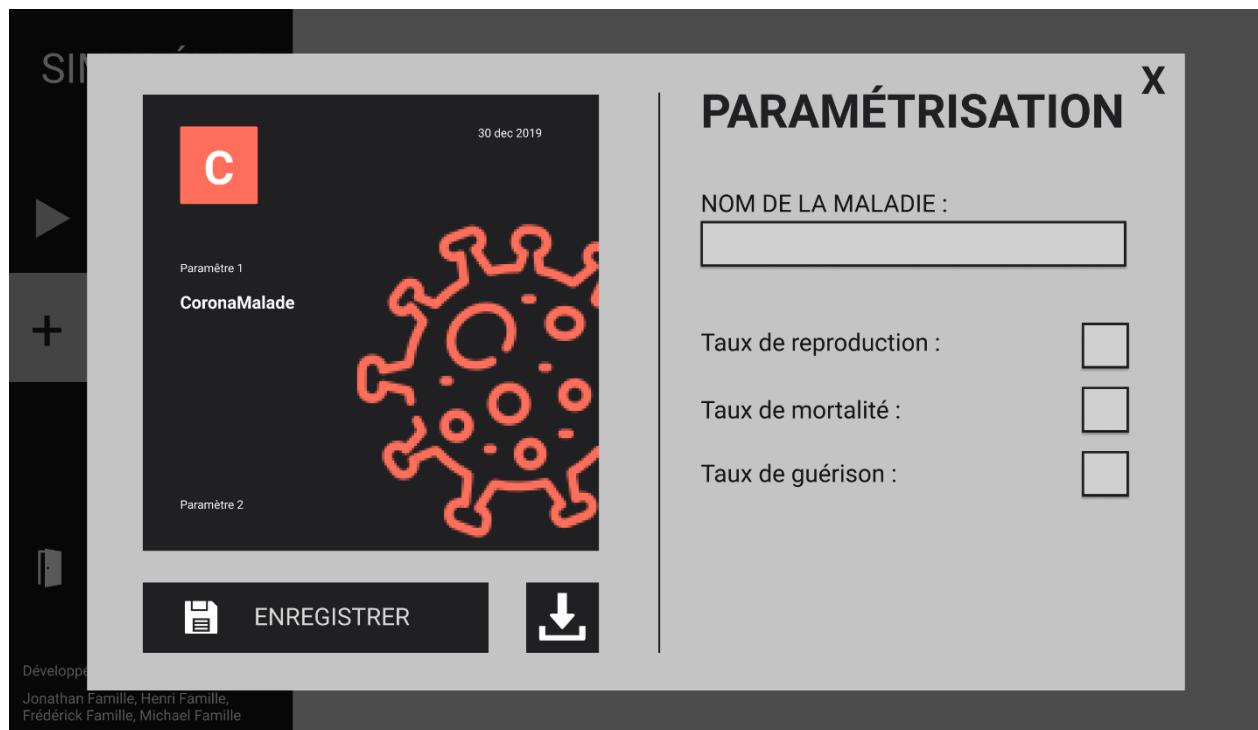
Esquisse 3.3 — Menus Création — Affiche Carte Existante



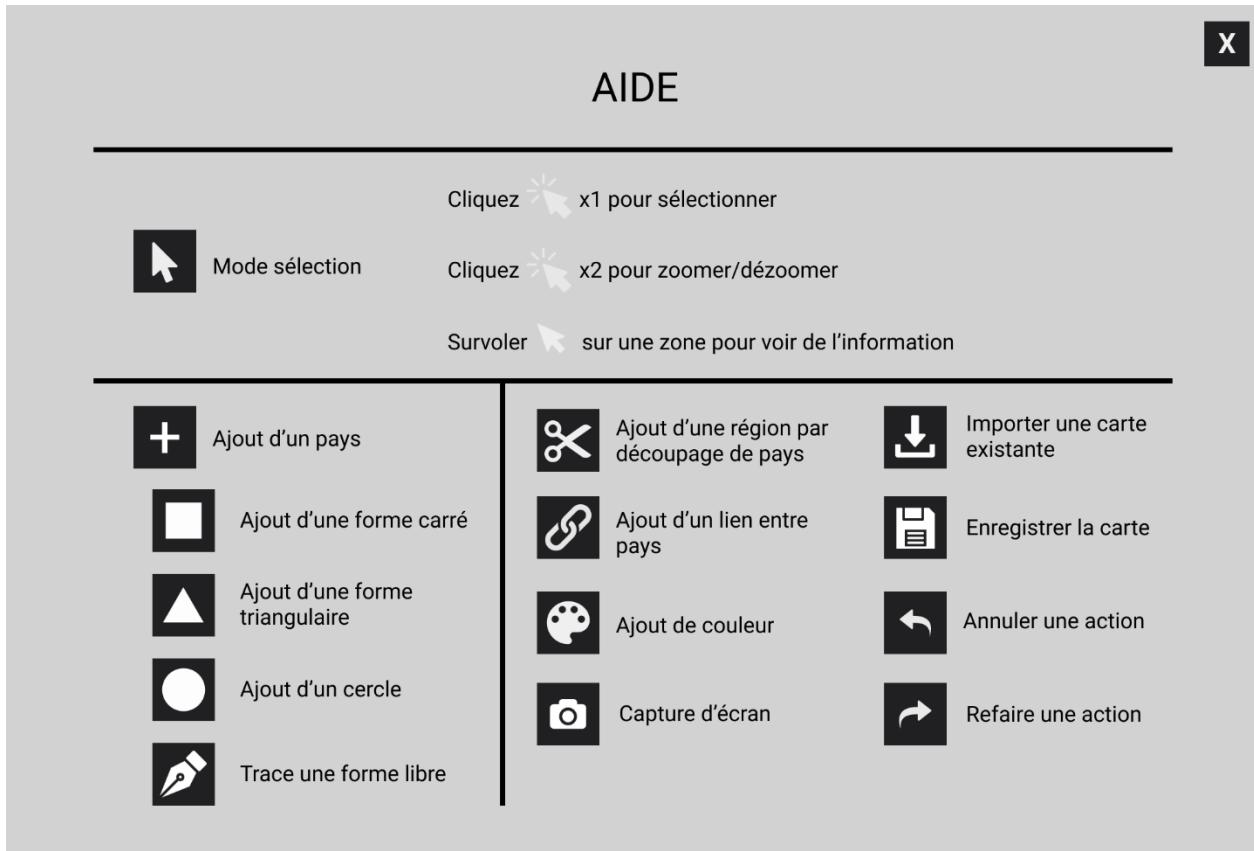
Esquisse 3.4 — Menus Création — Affiche Maladie Existante



Esquisse 3.5 — Menus Création — Crédit/Modification Carte Mondiale



Esquisse 3.6 — Menus Création — Crédit/Modification Maladie



Esquisse 3.7 — Menus Création — Crédit Carte Mondiale — Aide

NOM DU PAYS :

Population totale :

X

 ENREGISTRER

Esquisse 3.8 – Menus Création – Crédit Carte Mondiale – Crédit  
Pays

NOM DE LA RÉGION :

Proportion par rapport à la population totale du pays :

15  
0 ————— 100

X

 ENREGISTRER

Esquisse 3.9 – Menus Création – Crédit Carte Mondiale – Crédit  
Région

NOM DE LA CARTE :

Nombre de pays : N/A

Nombre de régions : N/A

Population totale : N/A

X

 ENREGISTRER & QUITTER

Esquisse 3.10 – Menus Création – Crédit Carte Mondiale – Enregistrer

### 3.2 — Thèmes & design v1.0

