8 octobre 2013

PDG

Cahier des charges

Groupe 3

Decorvet Grégoire
Froger Hadrien
Jaquier Kevin
Schweizer Thomas
Sinniger Marcel

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Table des matières

Première partie

Introduction

1 Cadre du projet

Le projet figure au programme de 3ème année de la formation Bachelor TIC-IL de la HEIG-VD. Il s'agit d'un projet de groupe de 5 ou 6 personnes se déroulant du vendredi 20 septembre au vendredi 17 janvier 2013 (date de rendu du projet).

Les objectifs du projet sont de spécifier, réaliser et tester une application informatique d'une certaine ampleur, de gérer la problématique du travail en équipe, rédiger un rapport et présenter le projet.

Le projet correspond à 75 heures effectives de travail au total par personnes, dont 3 périodes hebdomadaires en classe.

Une présentation de l'état d'avancement sera faite le 29 novembre.

2 Sujet

Il s'agit de concevoir et réaliser un moteur de simulation de propagation, autrement dit, un programme permettant de simuler la propagation d'un élément quelconque (information, maladie, ...) au sein d'une population quelconque (humaine, animale, ...). Il incombe à l'utilisateur de paramétrer le programme afin qu'il réalise la simulation souhaitée. Pour ce faire, il fournit au programme la définition de l'élément propagé, de la population simulée et son comportement, ainsi que les règles régissant la propagation et éventuellement les spécificités du "monde virtuel" utilisé par la simulation.

Le nom choisi pour l'application est Propagation Simulator.

3 Exemple

Pour illustrer les explications de ce document, nous prendrons, en guise d'application possible de notre programme, l'exemple vulgarisé d'une simulation d'épidémie de Malaria.

L'utilisateur devrait être en mesure de paramétrer le programme de façon à ce que la simulation prenne en compte les aspects suivants :

Une population de départ est infectée à un certain degré. On sait que les moustiques sont les vecteurs de propagation et qu'ils aiment les plaines vallonnées mais pas les endroits où il fait froid. Ils infectent les humains, ce qui les rend malades. Les symptômes sont la fièvre et la diarrhée. Un humain malade peut éventuellement mourir, selon sa morphologie, son âge et son milieu social. La maladie peut être guérie, et sa propagation peut être freinée, voire arrêtée. Il y a une probabilité qu'un ancien infecté retombe malade. La Malaria ne touche que des personnes d'une certaine tranche d'âge. On attrape la maladie si l'on se fait piquer par un moustique infecté, ou par contamination.

Deuxième partie

Méthode de travail

1 Gestion de projet

Pour ce projet nous allons travailler selon une approche agile suite à la proposition d'un membre du projet qui connait et applique cette méthode. La méthode agile dont nous allons nous inspirer est la démarche "SCRUM". L'objectif principal est de répondre à des besoins d'investisseur, comme des délais et des fonctionnalités promises tout en appliquant un procédé agile.

Une présentation de Scrum est disponible sous : `https://www.scrum.org`.

Étant donné que nous ne sommes pas assez nombreux, et que nous ne sommes pas assez expérimentés dans Scrum, nous n'allons pas implémenter la méthode complètement. Le but est d'utiliser certains outils de Scrum qui nous paraissent cohérents avec le projet. Nous suivons donc les recommandations de Scrum à propos des rendez-vous de groupes, de la planification, des normes de développement et de la gestion des tâches.

Comme le temps pour ce projet ne permet pas de faire de courtes itérations (équivalent des release dans le cadre de SCRUM), nous allons faire trois releases majeures qui marqueront chacune une phase de développement et de fonctionnalités. À la fin de chaque release, nous aurons un programme fonctionnel ainsi que la documentation associée (rapport, manuel utilisateur et manuel d'installation). Le but est de pouvoir théoriquement stopper le développement à la fin d'une release et avoir un produit prêt à être distribué.

À la fin de chaque release, une séance de debriefing sera consacrée pour faire le bilan de la release précédente et pourra conduire à effectuer une replanification ou un changement de méthode de travail si besoin.

Des séances auront lieu au moins une fois par semaine le vendredi entre les participants pour parler de l'avancement. Ces séances seront documentées, et la progression sera illustrée par des graphiques. Le journal de travail se construira donc après chaque séance, et le chef de projet harmonisera le document.

2 Qualité

Afin d'éviter de disperser nos efforts, nous avons utilisé le modèle Qualité de McCall [1] afin de sélectionner les aspects les importants de la réalisation de notre programme.

Les facteurs que nous avons considéré comme essentiels sont les suivants :

- Efficacité
- Maintenabilité
- Flexibilité

En tenant compte de ces facteurs, nous avons sélectionné les critères suivants comme étant centraux pour notre application :

- Traçabilité
- Cohérence
- Simplicité
- Lisibilité

1. Source : cours de Qualité du logiciel 2013, HEIG-VD TIC

Troisième partie

Réalisation

1 Technologies

Le programme sera développé avec le langage Python 2.7. Il comportera une interface utilisateur graphique qui sera réalisée à l'aide de la librairie Qt.

Lien vers Python : `http://www.python.org/`.

Lien vers Qt : `http://qt-project.org/`.

D'autres librairies seront éventuellement nécessaires pour certains parties du programme, comme NumPy et SciPy pour le calcul scientifique, par exemple. Les librairies standards du langage seront également exploitées.

Ce programme ne nécessitera pas de base de données. Les données seront organisées et stockées sous forme de fichiers texte.

2 Conventions

Le code sera écrit en anglais, et les commentaires en français. Cette combinaison permettra de rester cohérent avec la terminologie des librairies tierces, tout en permettant de maintenir une documentation claire et concise car nous maîtrisons mieux la langue française.

La convention de codage qui sera utilisée sera celle définie par le guide de référence de style [2] du langage Python.

2. `http://www.python.org/dev/peps/pep-0008/`

3 Glossaire

Le vocabulaire propre à l'application est défini ci-dessous. Ces mots et leur définition seront utilisés dans le reste du document, ainsi que pour la suite du projet.

Français	Anglais	Description
Entité	Entity	Définition de l'ensemble des caractéristiques communes à des individus. Exemple : Humain, Moustique
Individu	Individual	Instance d'une entité. Exemple : chaque moustique est une instance de l'entité moustique.
Population	Population	Ensemble des individus d'une entité. Exemple : la population de moustique qui va infecter une population d'humains.
Monde	World	Environnement de la simulation. Grille dans laquelle sont placées et évoluent les entités durant la simulation.
Carte	Map	Synonyme pour "Monde".
Terrain	Terrain	Ensemble des éléments du monde ayant des interactions avec les entités. Exemple : Montagne, Forêt.
Action	Action	Action pouvant être déclenchée aléatoirement et spontanément par une entité et n'affectant qu'elle. Exemple : Déplacement.
Interaction	Interaction	Action réciproque pouvant survenir entre deux entités durant la simulation. Modifie - en principe - leur état respectif. Exemple : Infection, Contamination.
Itération	Iteration	Unité de temps de la simulation. Correspond à une étape de mise à jour complète de l'état de tous les éléments de la simulation.
Release	Release	Version intermédiaire fonctionnelle, documentée et déployable de l'application.

À des fins de coordination et de vérification de l'état d'avancement effectif du projet, une définition du mot "fait" (ou "réalisé") sera effectuée, afin de ne pas considérer comme "terminées" des choses qui nécessitent encore du travail.

Quatrième partie

Objectifs

1 Projet

Le but de ce projet est de réaliser une application en Python afin d'apprendre ce langage et d'étendre ainsi nos compétences en programmation. L'intérêt sera aussi de découvrir et utiliser la librairie Qt, qui est très utilisée dans l'industrie.

L'autre objectif est de créer un programme évolutif, qui soit suffisamment flexible pour pouvoir s'adapter à plusieurs applications différentes. Ces applications ont en commun en la simulation d'une propagation (d'information, de maladie, etc.) au sein d'une ou plusieurs population (humaine, animale, etc.). Le défi sera de concevoir le programme de façon à ce qu'il puisse servir de base à ces applications et pouvoir donc être paramétré suffisamment en détail pour répondre aux besoins spécifiques de chaque simulation.

2 Simulateur

Notre simulateur s'adresse à des personnes ayant des notions avancées en informatiques, capables d'éditer des fichiers de configuration et d'écrire des programmes sous forme de scripts. En effet, le seul moyen d'adapter l'application à un simulation donnée, de façon à pouvoir reproduire les résultats et en gardant une interface utilisateur simple, sera de décrire dans des fichiers de configuration avec une syntaxe précise le comportement des entités de la simulation entre elles. L'utilisateur aura aussi la possibilité d'indiquer dans ces fichiers quelles données devront être enregistrées afin de permettre l'exportation des données générées par la simulation et ainsi pouvoir les utiliser dans un cadre statistique.

L'utilisateur pourra voir en temps réel l'évolution de la simulation grâce à un affichage graphique mis à jour à intervalle régulier pendant son exécution.

Cinquième partie

Description fonctionnelle

1 Application

Le programme n'est pas une simulation à proprement parler, car il ne simule rien de concret. Il s'agit plutôt d'un moteur de simulation, que l'on programme pour réaliser la simulation souhaitée.

Il n'est pas non plus destiné à être un simulateur universel, mais est limité à la simulation de propagation au sein d'une population. À l'utilisateur de définir l'élément propagé et la population concernée.

Pour ce faire, il convient de donner une représentation abstraite de ce qui sera simulé, afin que l'utilisateur définisse sa simulation par cette représentation.

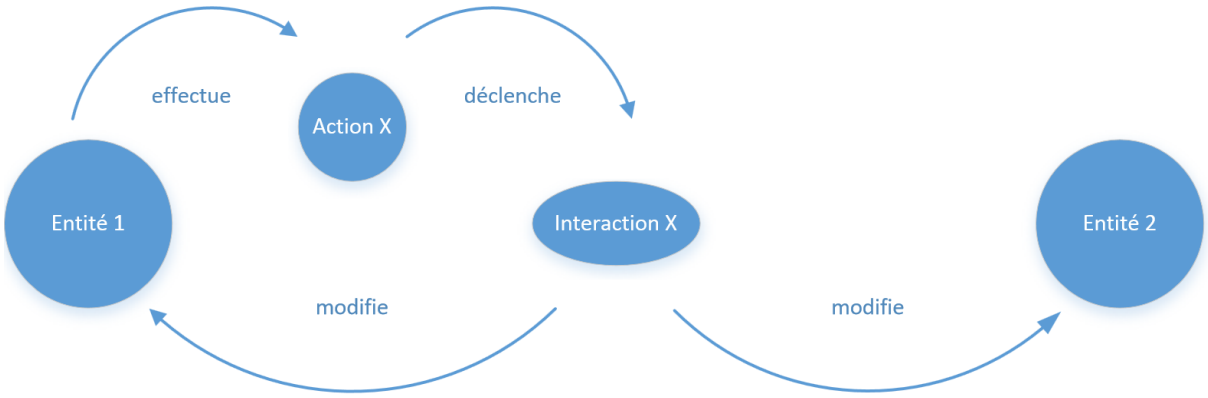

FIGURE 1 – Cas général

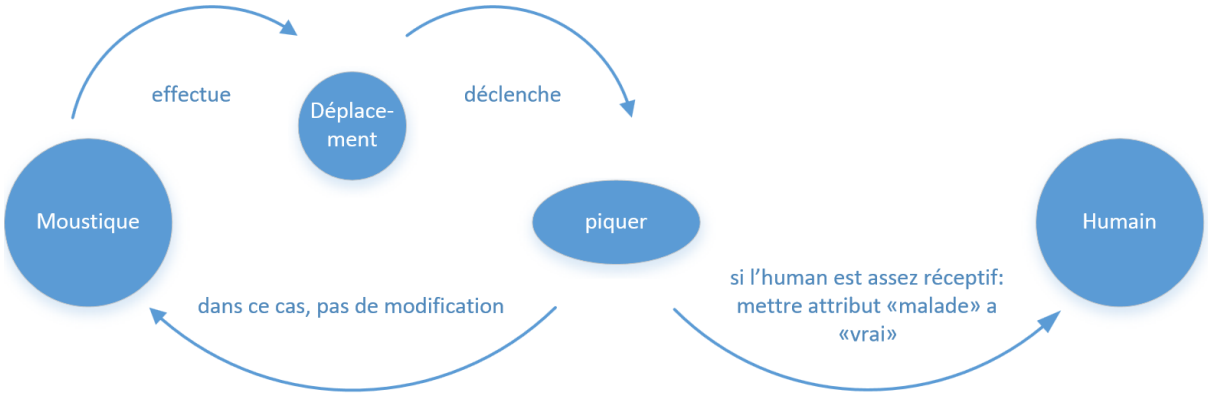

FIGURE 2 – Exemple : Moustiques - Humain

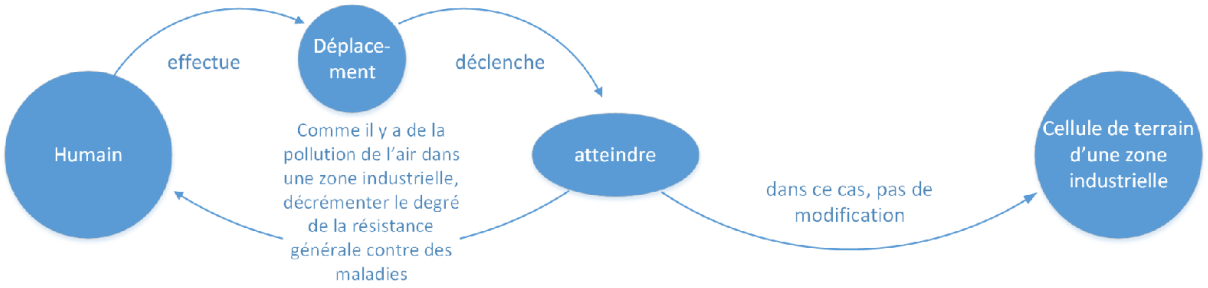

FIGURE 3 – Exemple : Humain - Terrain

1.1 Entités

Le programme simule l'évolution d'une ou plusieurs populations d'individus, définis selon une entité. Dans l'exemple de l'épidémie de Malaria, on simule une population d'individus de l'entité "Humain", et une population d'individus de l'entité "Moustique".

1.2 Monde

Les individus sont placés virtuellement sur une grille. Sur cette grille est généré un terrain aléatoire, défini en plusieurs zones, avec certaines caractéristiques paramétrables. Dans notre exemple, on peut imaginer une zone "Marais", où les moustiques sont plus nombreux que dans d'autres zones. On peut aussi ajouter une zone "Lac", où les déplacements des humains sont ralentis.

1.3 Actions

Les individus ne sont pas statiques. On a besoin de définir des événements qui sont déclenchés spontanément par les individus. Nous nommerons ces événements des actions. Les individus pourront, à chaque itération de la simulation, effectuer une ou plusieurs des actions définies dans leur configuration. C'est ensuite de leur responsabilité de réaliser l'action propre à leur situation, c'est à dire d'agir en fonction de leur position, du type de terrain ou des valeurs des attributs par exemple.

Une action d'un humain ou d'un moustique pourrait être, tout simplement, le déplacement.

1.4 Interactions

Il est nécessaire ensuite de définir la façon dont les individus interagissent ensemble. Pour cela on définit la notion d'interaction, qui est un événement généré aléatoirement sous certaines conditions par la simulation et qui affecte deux individus en changeant leur état.

À chaque itération, on passera en revue les interactions définies pour chaque entité, et on effectuera un test de leur conditions de déclenchement pour savoir si elles ont lieu.

Les interactions peuvent survenir à la suite d'une action. Une interaction peut également survenir entre une entité et le "terrain".

Dans notre exemple, une interaction pourrait être la piqûre d'un humain par un moustique. Il s'agirait d'une interaction du moustique vers l'humain, déclenchée lors d'un contact entre ces deux entités, selon une certaine probabilité. Un autre interaction pourra être ensuite définie pour représenter la contamination d'un humain par un autre. Un exemple d'interaction entre une entité et le terrain serait la zone "Lac" qui ralentit les déplacements des humains.

2 Fonctionnalités

Une fois que l'utilisateur a défini les entités et leurs actions, les interactions et la façon dont le monde est généré, il ne reste qu'à régler certains paramètres propres à la simulation en elle-même (comme le nombre d'itérations de la simulation), et le programme aura à disposition toutes informations nécessaire à son exécution.

Le programme est constitué de 3 parties :

- Paramètres : ensemble de fichiers de configuration permettant de définir la simulation.
- Simulateur : le cœur du programme, qui effectue la simulation selon les définitions et les règles données dans les paramètres.
- Interface graphique : permet de lancer la simulation, visualiser l'état du "monde" simulé, stopper la simulation et exporter les résultats.

2.1 Paramètres et définitions

Les paramètres des différents terrains et entités de la simulation, ainsi que de la simulation elle-même seront définis dans un ensemble de fichiers de configuration au format texte. Ainsi les actions et interactions des entités seront configurables dans des fichiers de configuration dédiés. Les paramètres généraux de la simulation, comme la taille de la carte, le nom de la simulation, les entités et le nombre d'entités générées pourront être définis dans les paramètres de simulation (fichier dédié lui aussi).

Chaque document de configuration sera localisé dans des dossiers spécifiques. Les différentes procédures de création de nouvelles configurations seront détaillées dans le manuel utilisateur. Le réglage des paramètres par interface graphique est prévu dans les fonctionnalités optionnelles.

2.1.1 Paramètres de simulation

Un fichier ou un ensemble de fichiers, permettra de définir tous les paramètres propres à une simulation.

Il s'agit notamment des paramètres suivants :

- Les éléments à importer dans la simulation : entités, actions, interactions. Ces éléments sont définis séparément.
- Le nombre d'entités et de terrains à générer
- Les paramètres nécessaires à la génération du monde et à sa reproduction, comme :
  — La méthode de génération du monde.
  Par exemple : aléatoire uniformément distribué, aléatoire par algorithme de bruit, manuelle...

  - La taille de la grille
  - La graine pour la génération aléatoire (seed), permettant la reproductibilité de la génération.
  - Les types de terrain, leurs propriétés, etc.
    Ils pourront utiliser les paramètres et comportements par défaut du simulateur.
- Les paramètres des entités, actions et interactions. Là aussi, ceux-ci pourront faire usage des paramètres déjà spécifiés lors de la définitions de ces éléments.
- Une éventuelle limite de la simulation (nombre d'itérations maximum ou durée d'exécution).
- La fréquence de rafraîchissement de l'aperçu graphique. En secondes.
- Le ou les paramètres à afficher et comment les représenter (par exemple, afficher en rouge les individus infectés, et les autres en vert).
- Les paramètres à enregistrer pour l'export des données.

2.1.2 Définition des entités

Un ensemble de fichiers de configuration sera dédié à la définition des entités.

L'utilisateur pourra donc définir pour une entité donnée :

- Ses attributs, par exemple pour l'entité "Humain" : sexe, âge, etc.
- Les actions qu'une entité peut réaliser, ainsi que leur définition.
- Certains paramètres additionnels, à savoir des variables qu'il aura à disposition.

Le programme mettra à disposition une définition par défaut l'entité "Humain".

2.1.3 Définition des interactions

Les interactions pourront également être définies au travers de fichiers de configuration. On pourra définir les conditions à réunir et la probabilité qu'une interaction soit déclenchée, ainsi que ses effets sur les entités concernées.

Le programme devrait donner la possibilité de définir une interaction dans un script qui sera exécuté au moment de la simulation, afin de donner un contrôle total à l'utilisateur sur les possibilités d'interactions entre entités.

Le programme mettra à disposition un ou plusieurs interactions par défaut pour l'entité "Humain".

2.1.4 Procédure de génération du monde

En plus des paramètres spécifiés dans les paramètres de la simulation, l'utilisateur devrait pouvoir personnaliser en profondeur la façon dont le monde est généré, en programmant son propre générateur dans un script qui sera exécuté au moment de la simulation.

Les paramètres du script personnalisé seraient définis dans un fichier de configuration annexe, permettant la réécriture des éventuels paramètres par défaut au niveau des paramètres de simulation.

2.1.5 Remarque sur la mise en pratique

Étant donné la difficulté d'estimer les possibilités, les limites et la complexité d'un tel système de définition et de paramétrage par fichiers de configuration, un prototype de test sera mis au

point en début de projet afin de valider la faisabilité de cet aspect. Suite à ces tests, le niveau de flexibilité dans les possibilités de paramétrage pour l'utilisateur est susceptible d'être modifié.

Il en va de même, dans une plus grande mesure encore, pour la fonctionnalité d'exécution de scripts, concernant la définition des interactions entre entités et de la méthode de génération de monde aléatoire. Il se peut qu'elle ne soient pas retenue car révélée trop complexe lors du prototypage.

2.2 Simulateur

Le simulateur est la partie "non visible" du programme, mais la plus importante. En effet il doit être capable de mener à bien la simulation tout en prenant en compte des directives fournies par l'utilisateur.

2.2.1 Exécuter la simulation

Au démarrage de la simulation, le programme effectue les opérations suivantes :

- Charger les paramètres de la simulation.
- Charger les définitions importées dans ces paramètres.
- Charger la méthode de génération de monde définie dans ces paramètres.
- Générer le monde avec la méthode chargée.
- Placer les entités, initialisées selon les paramètres chargés.
- Lancer les itérations de la simulation.

La simulation continue les itérations jusqu'à interruption manuelle de l'utilisateur, ou lorsqu'une éventuelle limite définie dans les paramètres de simulation soit atteinte.

2.2.2 Générer un monde

Cet aspect permet de générer aléatoirement un monde virtuel servant de terrain pour la simulation, selon des paramètres d'entrée, à savoir :

- Les dimensions (largeur / hauteur) de celui-ci.
- Les types de terrains possibles.
- La taille moyenne de chaque type de terrain et le nombre de zones souhaitées.

2.3 Interface graphique

L'interface graphique du programme sera basique et proposera les fonctionnalités décrites dans ce chapitre.

Tous les textes et libellés des interfaces graphiques seront en anglais. Le manuel utilisateur sera cependant en français.

2.3.1 Contrôle de la simulation

L'utilisateur peut démarrer la simulation, puis mettre en pause, reprendre ou arrêter la simulation à tout moment une fois celle-ci lancée.

2.3.2 Représentation schématique

L'utilisateur pourra suivre le cours de la simulation en temps réel à l'aide d'un affichage graphique basique, représentant l'état à un moment donné de la simulation des entités et du terrain, par des formes de couleur.

2.3.3 Export des données

À la fin de la simulation, l'utilisateur pourra exporter les données brutes recueillies par la simulation au format texte. Ces données représenteront la valeur des attributs de chaque instance d'entité, à chaque itération de la simulation. Les fichiers de configurations permettront de sélectionner les attributs à enregistrer.

Sixième partie

Fonctionnalités additionnelles

1 Fonctionnalités optionnelles

Les fonctionnalités suivantes ne seront implémentées que si le temps le permet :

- Menu simplifiant l'accès aux fichiers de configuration.
- Configuration par interface graphique de la simulation.
- Affichage de courbes mises à jour en temps réel d'attributs choisi par l'utilisateur.
- Génération de tableaux croisés dynamiques.

2 Fonctionnalités non-implémentées

Les fonctionnalités suivantes ne seront pas prises en charge par le programme :

- Génération de graphiques statistiques sur les données résultantes de la simulation
- Modification des paramètres en cours de simulation

Septième partie

Planification

La planification est découpée en 5 parties majeures :

- Analyse
- Documentation
- Première release
- Deuxième release
- Troisième release
- Finalisation - Rendu

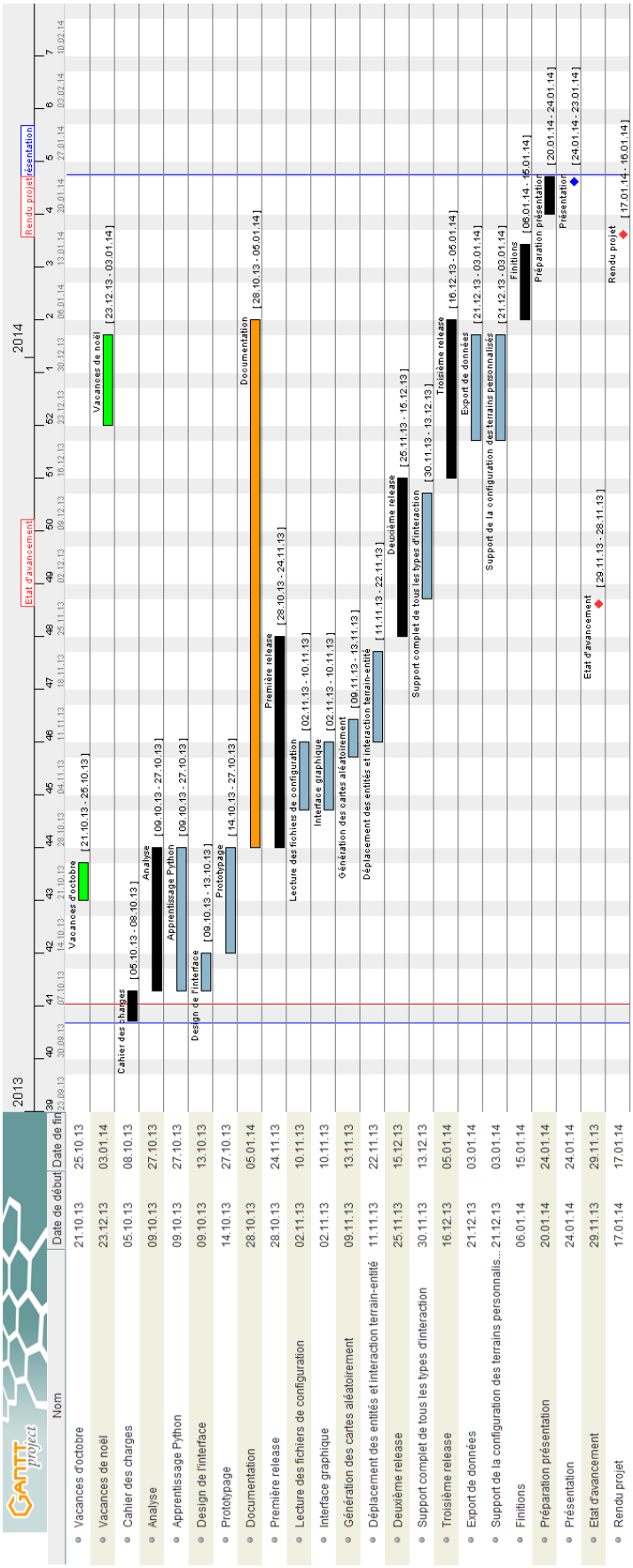

FIGURE 4 – Planification initiale

1 Analyse

Le but premier de cette partie est de préparer les documents d'analyse, notamment la structure de base du programme, le format des fichiers de configuration, l'interface graphique et les tests unitaires de la première release. La période d'analyse permettra aussi de commencer à apprendre le langage Python et d'utiliser les connaissances nouvellement acquises dans l'élaboration de prototypes du programme notamment au niveau de l'interface graphique et son affichage ainsi que la lecture des fichiers de configuration.

2 Documentation

Cette tâche représente l'effort de documentation qui doit être réalisé tout au long du projet. En pratique, cette tâche sera surtout effectuée lors des réunion hebdomadaire de groupe en debriefing pour le journal de travail ainsi qu'au début et à la fin d'une release.

3 Releases

Le but d'une release est de fixer une date pour rendre un produit à l'aspect fini. Cela permet d'avoir des commentaires rapides sur l'application et affiner les prochaines priorités des fonctionnalités.
La planification ne spécifie aucune tâche pour le début et la fin d'une release : ce temps sera occupé par des analyses propres à la release et aux fonctionnalités à développer, ainsi qu'à la rédaction des divers documents à produire (tests unitaires, manuel utilisateur, manuel d'installation, documentation d'analyse, etc...). À la fin de la release, il y aura également une critique constructive de l'itération écoulée afin de mettre en évidence ce qu'il faut changer pour la prochaine itération et les éventuelles replanifications nécessaires.

Le produit à la fin d'une release doit répondre à plusieurs critères :

- Manuel utilisateur prêt.
- Produit déployable, comprenant une procédure d'installation.
- Test unitaires et utilisateurs effectués avec succès.
- Documentation prête et complète.

Ces critères sont très exigent, c'est pourquoi nous allons devoir adopter un système d'intégration progressive afin de ne pas être sous pression lors du rendu des releases.
Les différentes releases sont décrites à l'avance par des test utilisateurs. Nous établissons, comme avec le Test Driven Developpement de la méthode XP, des tests utilisateurs précis que devra passer la release pour être considérée comme réussie. La présence du mandant lors de la démonstration peut être demandée, afin de l'avertir de l'avancement.

À la fin de chaque release, nous effectuons des tests utilisateurs très précis décrivant exactement les actions que pourra faire l'utilisateur. Il s'agit donc d'un exemple d'utilisation démontrant les fonctionnalités développées. Ceci n'est pas un test exhaustif, nous l'utilisons afin de mettre toute l'équipe au courant de la situation du projet.

En cas d'échec d'une release, nous allons suivre la procédure suivante :

1. Prendre tous les tests utilisateurs qui ont échoués.

2. Identifier les tâches nécessaires au fonctionnement du test échoué.
3. Réévaluer la priorité les fonctionnalités de la prochaine release.
4. Mettre à jour les tests utilisateurs de la prochaine release.
5. Avertir le mandant de la nouvelle planification.

Ces différentes releases ne remplissent ni ne testent l'ensemble du cahier des charges. Toutes les fonctionnalités supplémentaires seront implémentées par les développeurs comme tâches secondaires, avec une priorité plus basse. Les releases représentent le cœur de l'application, les fonctionnalités annexes sont de moindre importance et ne seront présentées que lors des différents meeting et non pas durant les démonstrations de release.

3.1 Première release

Le but de la première release est de mettre au point toute la structure de départ du programme. Peu de fonctionnalités pourront donc être intégrées.

Voici les objectifs visés :

- Génération de monde purement aléatoire pour avoir une base de travail rapidement.
- Affichage de l'interface graphique. Aucune configuration ne se fait dans l'interface. L'interface graphique sera composée : d'un bouton "play" pour lancer la simulation, d'un bouton "pause" pour la stopper et des informations sur la simulation (voir le prototype graphique de la release).
- Configurer une entité : exécution des actions. Mais pas des interactions entre entités.

L'utilisateur pourra effectuer les quelques tests suivants.
En tant qu'utilisateur je peux :

- Lancer l'application exécutable afin de pouvoir démarrer la simulation que j'ai configurée.
- Configurer un type de terrain "normal" où les entités puissent se déplacer.
- Configurer une entité "humain" afin qu'elle puisse se déplacer dans le monde.
- Configurer une entité "moustique" afin qu'elle puisse se déplacer dans le monde.
- Configurer un type de terrain "infranchissable" afin que les entités "moustique" et "humain" ne puissent pas entrer dans le terrain.
- Voir les entités se déplacer sur le monde, et je vois les différents paramètres de ma simulation (temps passé, nombre d'itérations, nombre d'itérations totales, quantité de chaque entité, et noms des zones différentes).

Remarque : toutes les configurations se font dans des fichiers de configuration.

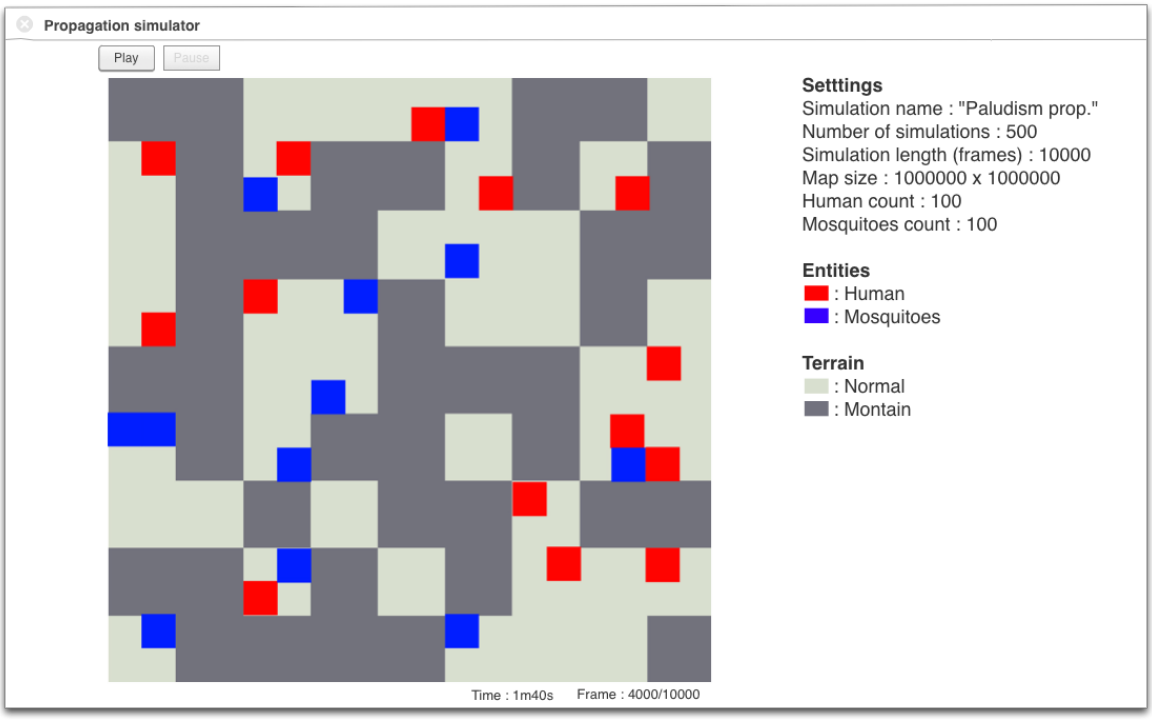

FIGURE 5 – Prototype graphique pour la première release

3.2 Deuxième release

Le but de la deuxième release est de finaliser tous les types d'interactions possibles. Ainsi nos différentes entités pourront définir plusieurs types d'interaction avec d'autres entités.

Objectifs visés :

- Les entités peuvent avoir des interactions entité-entité.
- Toutes les interactions sont configurables (on peut ajouter de nouvelles interactions entité-terrain ou entité-entité).

L'utilisateur pourra effectuer ces quelques tests. En tant qu'utilisateur je peux :

- Définir autant d'interactions que je veux entre les moustiques et les humains.
  Par exemple, un humain peut "écraser" un moustique, et un moustique peut "infecter" un humain.
- Configurer un type de terrain "à risque" où les moustiques peuvent "infecter" les humains.
- Configurer un type de terrain "normal" où les moustiques ne peuvent pas "infecter" les humains.
- Configurer un type de terrain "infranchissable" afin que les entités "moustiques" et "humains" ne puissent pas entrer dans le terrain.

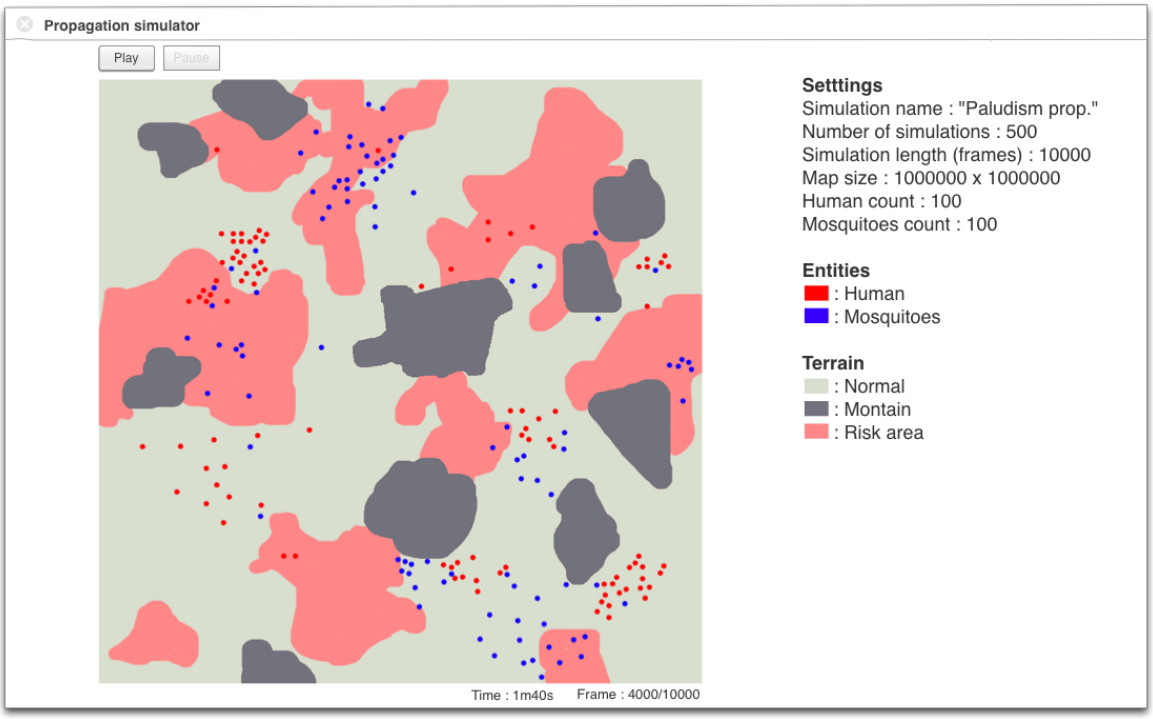

FIGURE 6 – Prototype graphique pour la deuxième release

3.3 Troisième release

La troisième release permet à l'utilisateur scientifique de réellement utiliser notre application. Une fonction basique d'export sera mis à disposition, ainsi qu'une configuration manuelle de carte.

Objectifs visés :

- Export des données.
- Map customisée ou map aléatoire. Configuration des maps dans le fichier de configuration de la simulation.

En tant qu'utilisateur je peux :

- Avoir accès à des données de simulation exploitable par la suite dans des logiciels statistiques (tels que R).
- Configurer la cadence d'export. Les exports contiendront tous les attributs des entités et leurs valeurs pour chaque capture statistique.
- Ajouter une carte que j'ai créé manuellement (chargée depuis un fichier contenant directement les données de la carte sous forme de texte).
- Configurer la génération de carte. afin de choisir un mode aléatoire ou manuel.

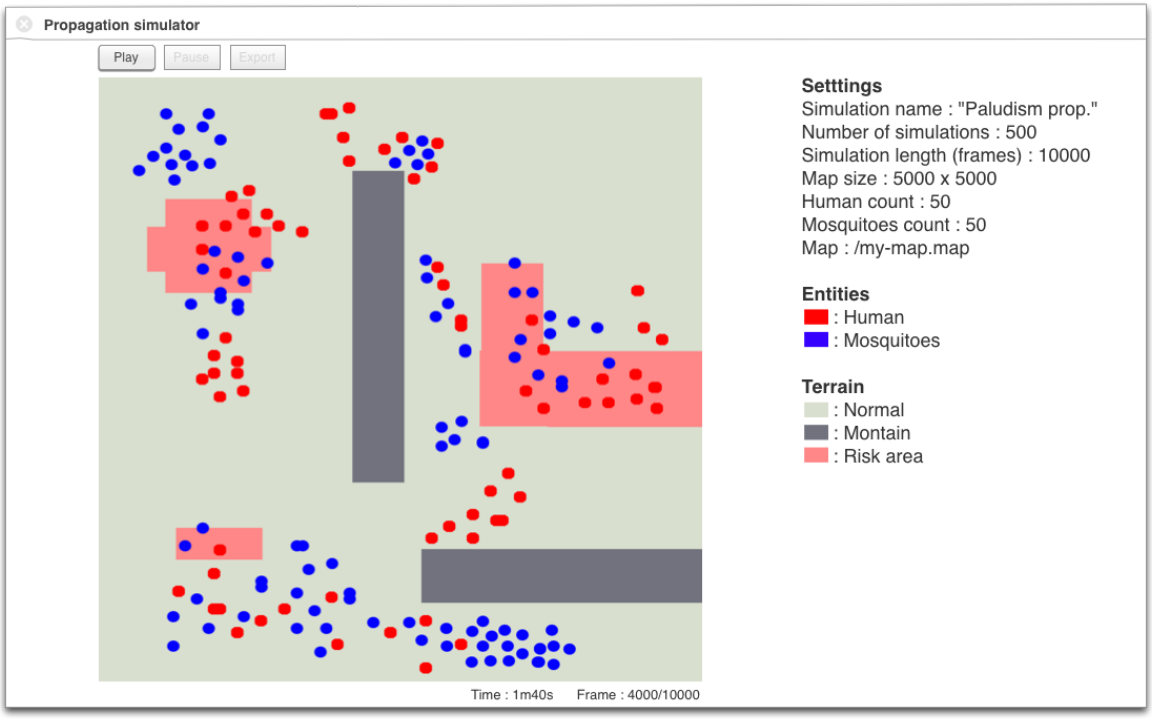

FIGURE 7 – Prototype graphique pour la troisième release. Les différents types de mondes peuvent être entre autres configurés

4 Finalisation - Rendu

Cette dernière étape a été planifiée sur un peu moins de deux semaines. Il s'agit essentiellement d'une zone de sécurité qui nous permet de déborder si nous constatons qu'une fonctionnalité n'a pas pu être développée complètement à la fin d'une release. Cette période servira également à mettre les dernières touches finales et harmonisations aux documents du projet ainsi qu'à leur relectures respectives.