

S.I.E.R.R.A. – Compte-rendu

Sujet n°4 : Simulation de la propagation d'un incendie

Henri Debray, Augustin Gagnon, Amaury Zarzelli

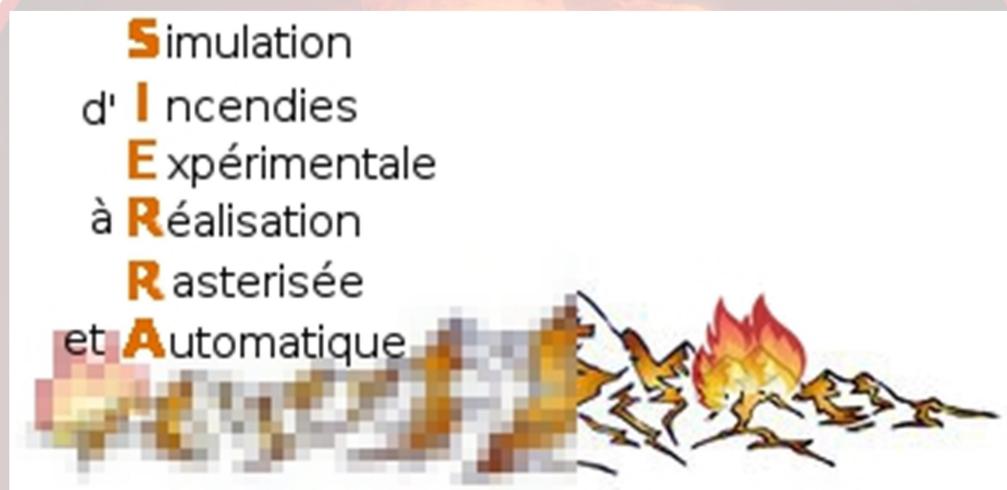


Table des matières

I. Compréhension du sujet	3
II. Analyse du sujet	3
A. Classes mises en jeu	4
B. États-transitions des cellules enflammées	5
C. Activité de la cellule	6
D. Activité du pompier	7
III. Réalisation	8
IV. Bilan & Perspectives	9

I. Compréhension du sujet

Le but de ce sujet était de réaliser une application permettant de simuler la propagation d'un incendie, et destiné à l'usage des collectivités publiques dans un but de prévention et de protection.

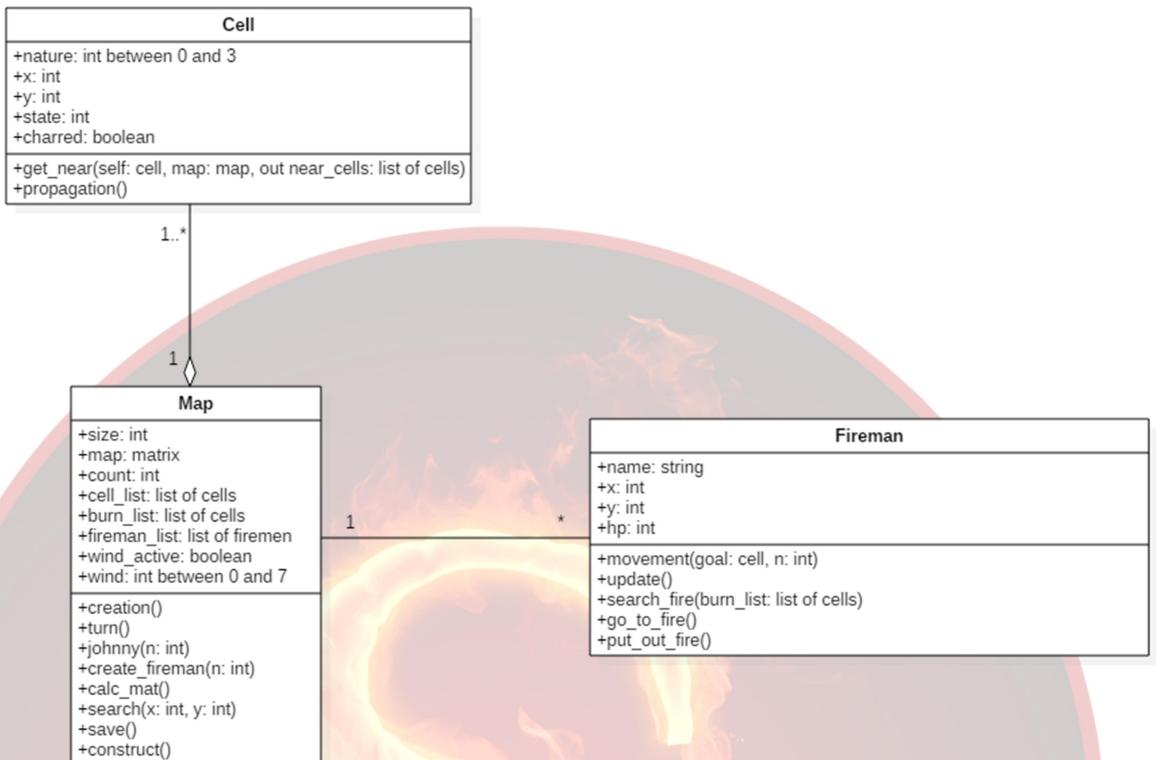
L'objectif final était donc d'obtenir une application fenêtrée et indépendante du langage de programmation (console Python), capable de réaliser une simulation modulable selon les options choisies par l'utilisateur. Il était aussi nécessaire de pouvoir sauvegarder cette simulation, et le format image a été choisi. Enfin, la simulation devait être réaliste, c'est pourquoi un travail plus soutenu a été fourni sur le déplacement des pompiers, la propagation du feu et la génération aléatoire du terrain.

Les contraintes du sujet étaient tout d'abord le langage de programmation, car il était impératif de construire l'application avec le langage Python. Cela implique qu'il est nécessaire d'installer Python sur un ordinateur avant de faire fonctionner l'application. Une deuxième contrainte a été la représentation en deux dimensions de l'ensemble de la simulation, ce qui a pu empêcher certains aspects de la propagation d'incendie d'être introduits dans la simulation.

II. Analyse du sujet

Nous avons modélisé le projet à l'aide du langage UML avec les diagrammes suivants. Nous n'avions au départ pas jugé utile la réalisation d'un diagramme de cas d'utilisation, car la solution initialement proposée ne comprenait qu'une simulation indépendante de l'utilisateur. Cependant, nous aurions pu rajouter des fonctionnalités « avancées » (en utilisant les modules argparse et npyscreen par exemple) qui auraient rendu pertinente la création d'un tel diagramme (*cf. Bilan*). En définitive, nous avons donc réalisé l'indispensable diagramme de classes, des diagrammes d'activité et des diagrammes d'états-transitions.

A. Classes mises en jeu



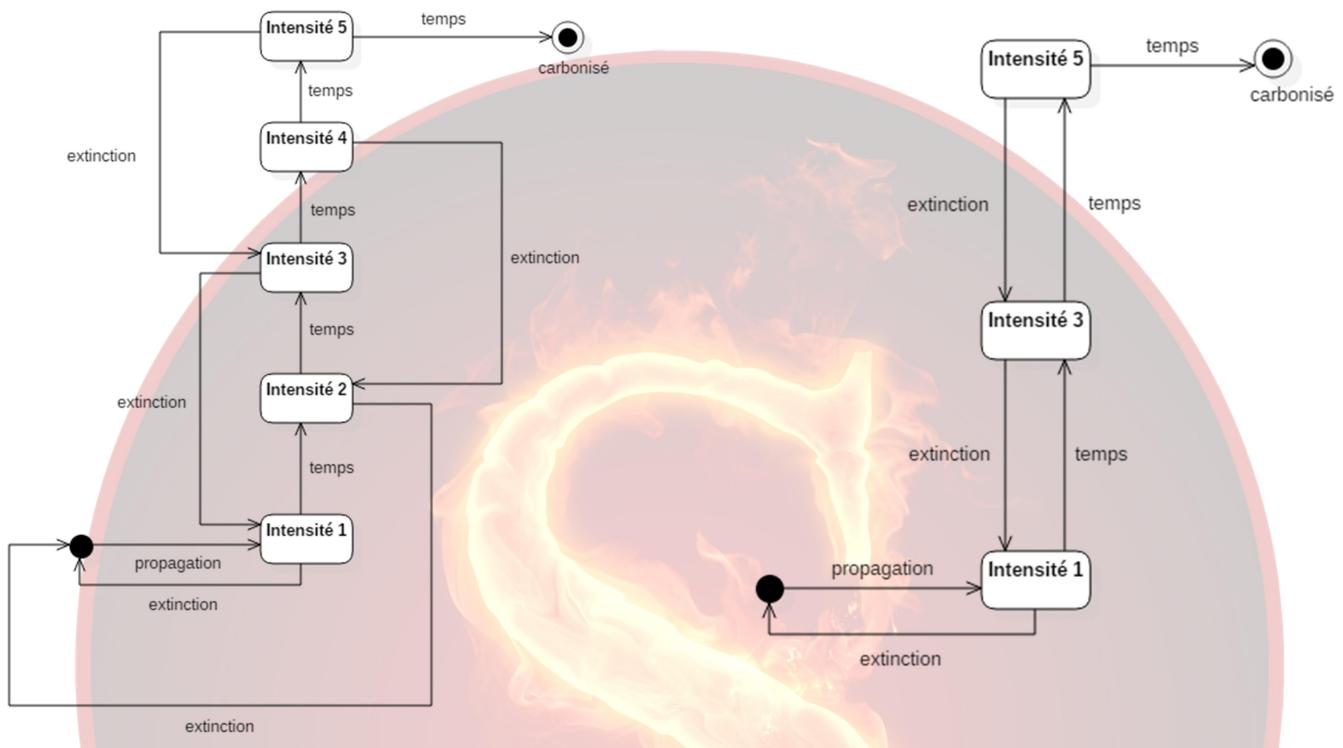
La simulation prend place dans une instance de la classe Map, définie par sa taille (size), son numéro d’itération (count), l’ensemble des cellules qui la constituent (cell_list), l’ensemble des cellules qui brûlent (burn_list), l’ensemble des pompiers présents (fireman_list), la présence ou non de vent (wind_active) et la direction de ce dernier (wind), stockée sous la forme d’un entier. La matrice map est une matrice d’entiers servant à l’affichage. La classe Map contient des fonctions d’initialisation : creation() qui génère aléatoirement une carte à partir d’une méthode de bruit de valeurs, johnny(n) qui allume n feux, create_fireman(n) qui crée n instances de la classe Fireman et les place aléatoirement sur la carte. La fonction turn() permet d’itérer la simulation, la fonction save() de sauver l’état de la Map à une itération donnée et la fonction construct() de construire une Map à partir d’une base de données SQLite.

La Map est constituée d’instances de la classe Cell, définies par leur nature (0 : eau, 1 : plaine, 2 : forêt, 3 : ville), leur position (x,y), leur état d’inflammation (intensité), et leur état carbonisé ou non (charred). Cette classe a pour méthodes une méthode de détection de voisinage (get_near()), et une méthode de propagation. Une cellule n’apparaît que dans une Map, et une Map est constituée d’au moins 1 cellule.

Sur la Map sont présents des instances de la classe Fireman, définies par leur nom, leur position, et leur nombre de points de vie. Les pompiers ont une méthode de mouvement pour se déplacer, une méthode update() qui itère sur le pompier en appelant les fonctions suivantes : search_fire(), qui recherche le feu le plus proche, go_to_fire(), qui utilise la méthode movement() pour se diriger vers

le feu le plus proche, et `put_out_fire()`, qui diminue l'intensité du feu adjacent si possible. Un pompier n'est présent que sur une Map, et une Map contient de 0 à n pompiers.

B. États-transitions des cellules enflammées

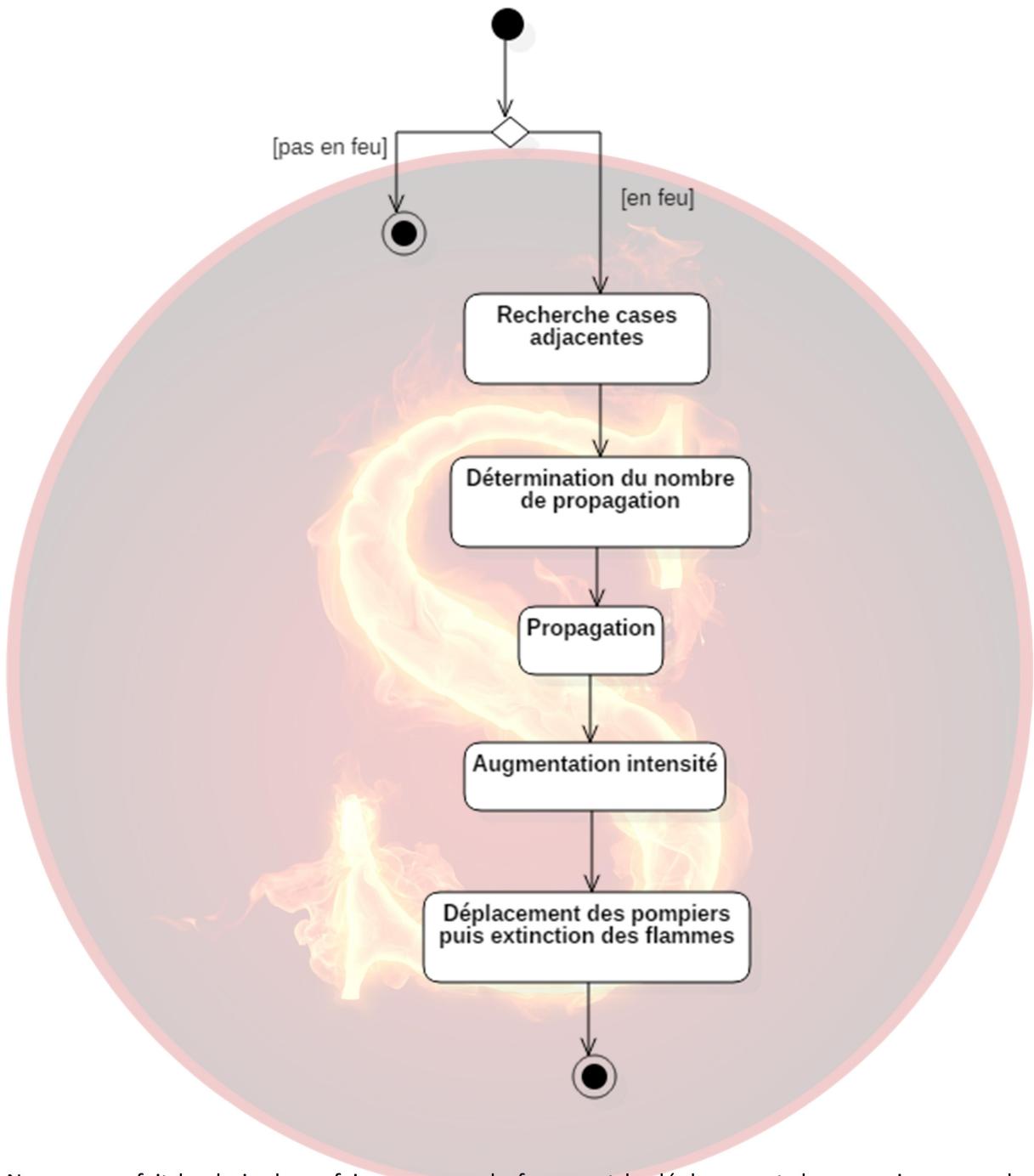


États-transitons d'une cellule en feu par défaut

États-transitons d'une cellule de type "ville" en feu

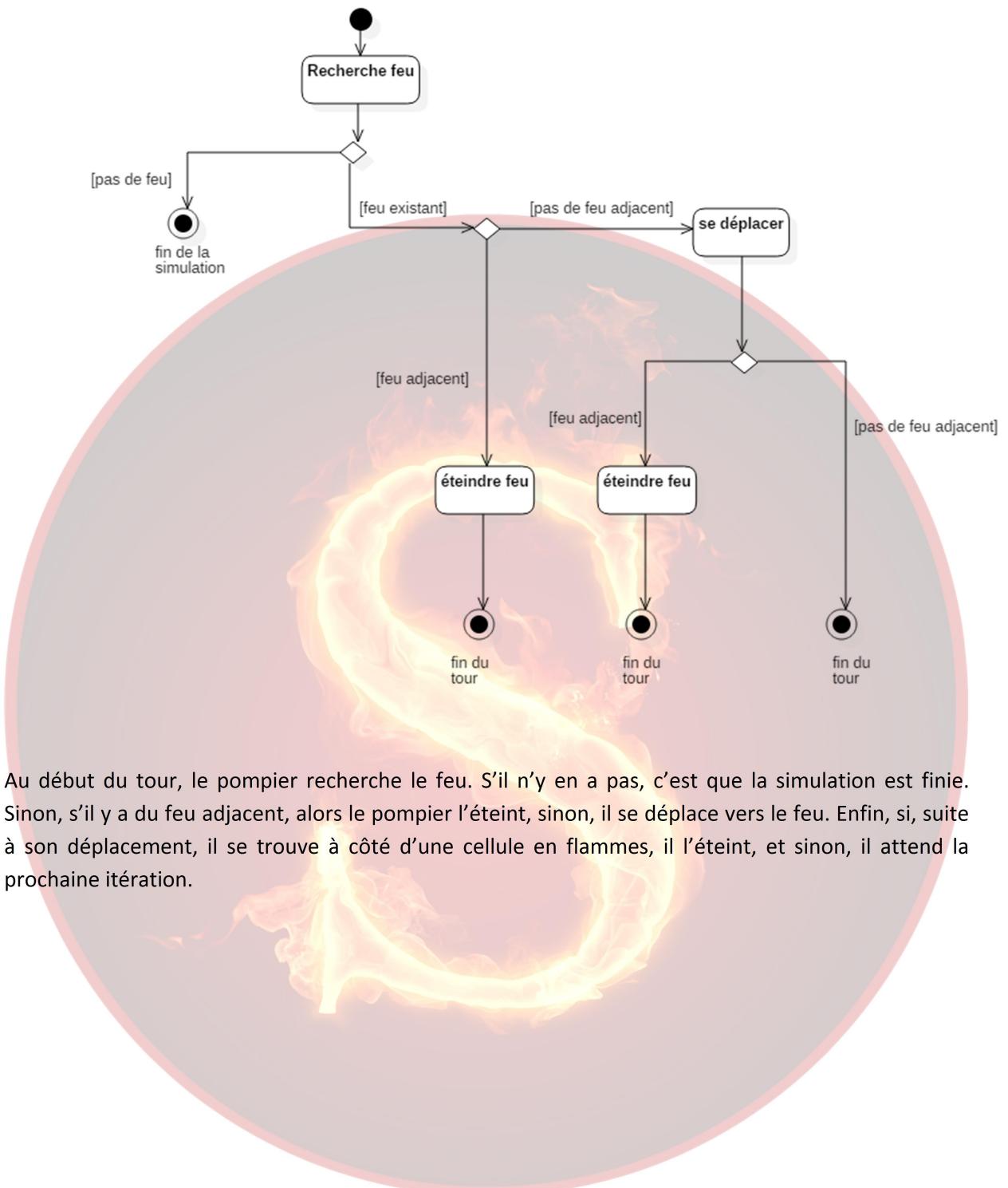
L'intensité des flammes d'une cellule en feu augmente de 1 à chaque tour (ou de 2 si la cellule est de type ville), et diminue de 2 si on pompier l'éteint. Si l'intensité devient supérieure à 5, la cellule est carbonisée et ne peut plus s'éteindre.

C. Activité de la cellule



Nous avons fait le choix de se faire propager le feu avant le déplacement des pompiers pour des raisons d'implémentation. La propagation se fait en suivant ces étapes : recherche des cellules adjacentes pouvant brûler, détermination aléatoire du nombre n de propagation, propagation (c'est-à-dire, changement d'état des cellules sur lesquelles le feu arrive), augmentation de l'intensité du feu sur la cellule. Ensuite, la cellule subit l'action des pompiers. Enfin, si la cellule ne brûle pas, elle est passive jusqu'à la prochaine itération.

D. Activité du pompier



Au début du tour, le pompier recherche le feu. S'il n'y en a pas, c'est que la simulation est finie. Sinon, s'il y a du feu adjacent, alors le pompier l'éteint, sinon, il se déplace vers le feu. Enfin, si, suite à son déplacement, il se trouve à côté d'une cellule en flammes, il l'éteint, et sinon, il attend la prochaine itération.

III. Réalisation

Il a été unanimement choisi d'attribuer à chaque personne du groupe une classe d'objet afin de commencer le développement. La répartition individuelle du travail a notamment permis à chaque membre du groupe de progresser indépendamment des heures réservées au projet, en utilisant l'outil GitHub. Ces heures spécifiques ont principalement servies à la mise en commun des idées, le débogage du programme, et la planification des futures tâches individuelles.

Une fois les fonctionnalités de base développées, le programme était capable de générer un terrain aléatoire et cohérent, et de calculer la simulation correspondante tout en sauvegardant chaque état sous forme d'image. Toutefois, le déplacement des pompiers imposé par le sujet limitait le réalisme de la simulation, et l'affichage du résultat était limité à la console Python. Nous avons donc concentré nos efforts sur l'accessibilité du programme, à la fois par le développement d'une interface homme-machine et par la traduction intégrale du programme en anglais, ainsi que l'amélioration du réalisme de la simulation en modifiant le comportement des pompiers.

Une fois que l'interface fut complète, nous avons pu commencer à introduire de nouvelles fonctionnalités : le vent qui modifie le déroulement de la simulation, la possibilité de compiler les images en une vidéo (en fait, une image GIF), la sauvegarde de la simulation dans une base de données, etc. En parallèle, nous avons continué à chercher un mode de déplacement intelligent pour les pompiers afin de proposer une simulation la plus réaliste possible.

Enfin, la dernière partie du développement a été consacrée à la documentation et aux tests unitaires du code, réalisés respectivement à l'aide de Doxygen et du module doctest. Les tests sont lancés à partir du fichier *run_tests.bat* dans le dossier *code* qui écrit le log des tests dans le dossier *code*.

Une fois le développement terminé, il fallait trouver un moyen plus « convivial » de lancer le programme. Tout d'abord, cela fut fait à l'aide d'un fichier batch (*run.bat*), permettant l'exécution de lignes de commandes Windows. Ensuite, nous avons utilisé pyinstaller afin de compiler le programme et de le fournir sous forme exécutable portable (donc sur un ordinateur non pourvu de Python). C'est l'objet du dossier *application* qui contient l'exécutable ainsi que les dossiers et dll nécessaires à son fonctionnement.

IV. Bilan & Perspectives

Au cours du projet nous avons réussi à réaliser une application de simulation de propagation d'incendie. Nous avons donc réalisé notre génération de terrain aléatoire mais cohérente et fait en sorte que l'état initial de notre terrain soit sauvegardé dans une base de données. Nous avons réussi à automatiser le déplacement de nos pompiers.

Outre le cahier des charges, nous avons rajouté quelques points supplémentaires dans notre application. Le public visé par notre application étant une collectivité territoriale, nous nous devions de respecter autant que possible (dans la mesure des délais et de la complexité attendue de notre application) le réalisme des caractères mis en jeux. Ainsi nous avons fait en sorte que nos pompiers ne soient pas immortels (*i.e.* quand ils sont trop exposés au feu, ils sont déclarés « morts »). Afin de rendre notre application plus conviviale et exploitable par tout type d'utilisateurs, nous avons élaboré une interface graphique utilisateur qui rendait compte visuellement de la propagation et du déplacement des pompiers. Toujours dans cette optique de réalisme de la simulation, nous avons proposé nos propres règles de propagation du feu (sens du vent) et de déplacement des pompiers (afin d'obtenir un déplacement efficace). Enfin dans l'optique d'une diffusion à tout type de collectivité territoriale, nous avons réalisé une application externe à Python, dans lequel notre code est compilé afin de ne pas nécessiter l'installation de Python sur la machine de l'utilisateur.

Compte tenu des délais, certaines de nos améliorations n'ont pas encore abouti. Nous comptions mettre au point un déplacement plus intelligent de nos pompiers, qui prendrait en compte chaque foyer et assignerait certains pompiers à un foyer plutôt qu'un autre en fonction de son éloignement et de la taille du foyer considéré. Par ailleurs nous avions commencé à développer une version de notre application qui serait exploitable à différentes échelles de complexité par différents utilisateurs (avec Argparse). Et enfin, bien qu'ayant développé certains tests unitaires de notre code, d'autres encore auraient pu être envisagés.

Ainsi cette application répond au cahier des charges initialement décrit par le client. Cependant nous avons souhaité lui apporter des modifications et améliorations. Celles-ci permettent de présenter notre projet comme l'ébauche d'une application à la fois : plus performante, plus adaptée aux besoins mis en jeu, plus propice à la diffusion.