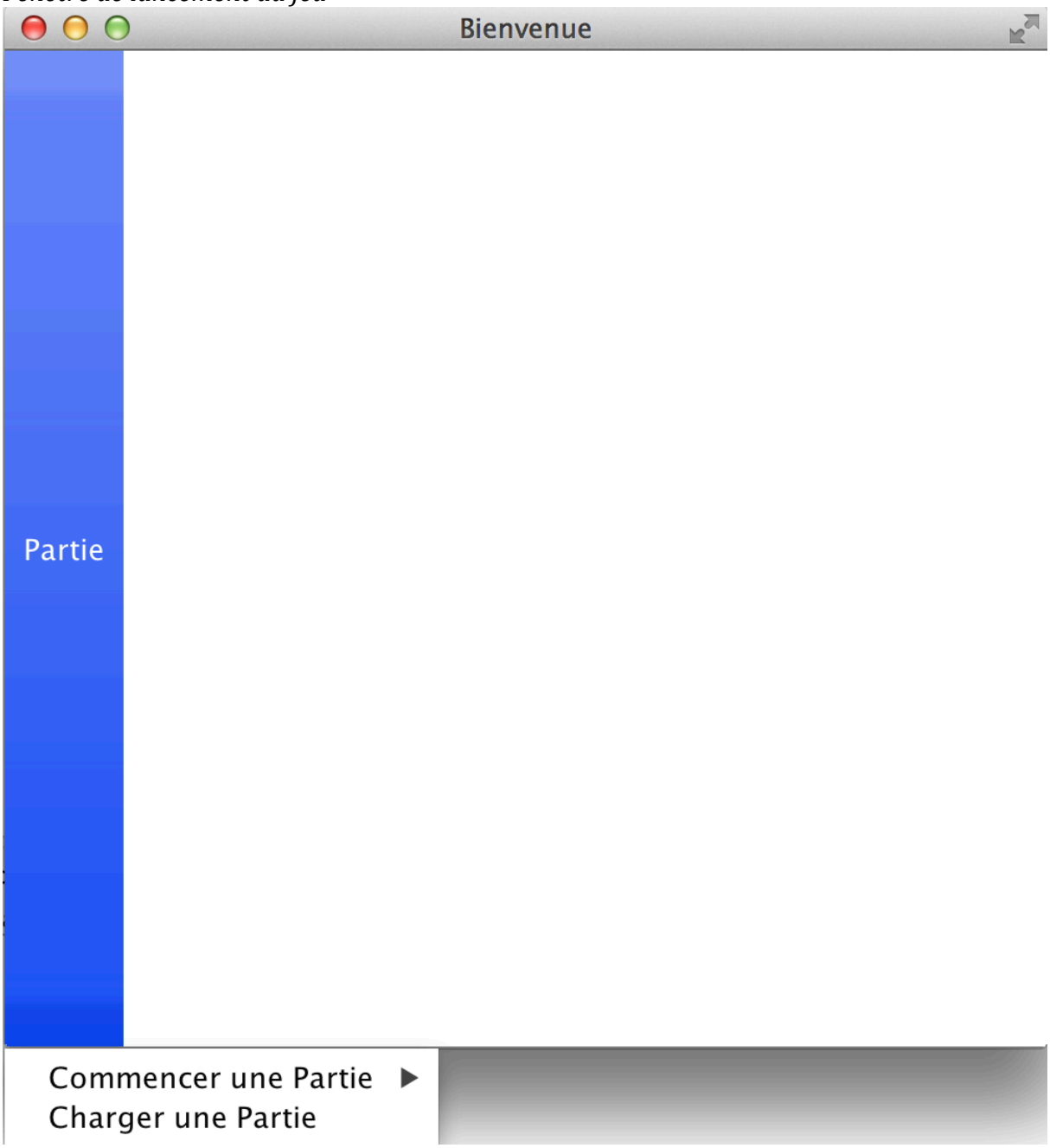
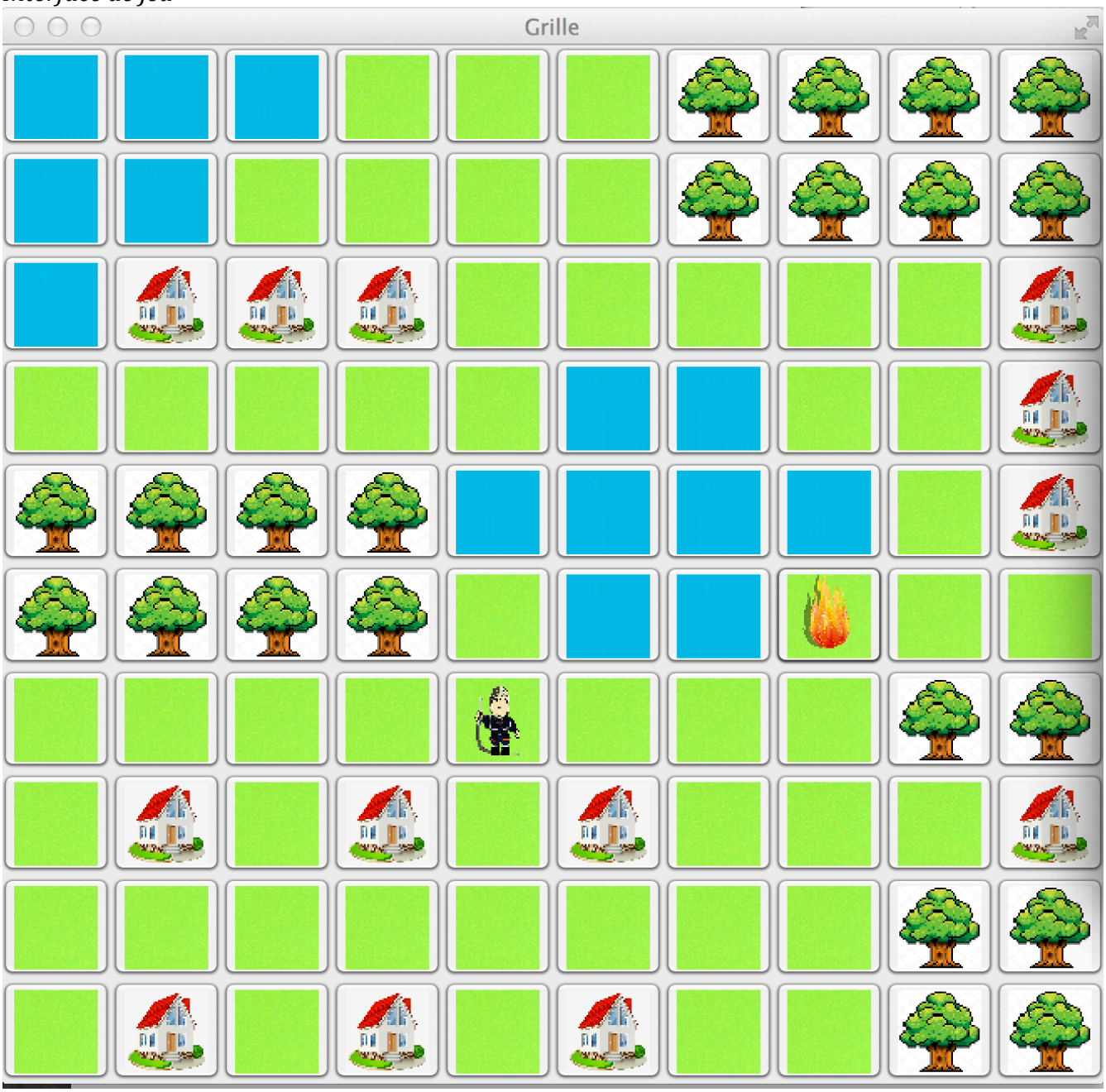
# Simulation de la propagation d'un incendie

Nous avons développé notre projet sous la forme d'un jeu. Le programme simule un feu qui se propage selon les règles prédéfinies dans le rapport initial. Le but du joueur est de stopper le feu par l'action des pompiers au travers de l'interface.

*Fenêtre de lancement du jeu*



*Interface de jeu*



*Informations*


# La programmation

### *Les grands axes du code :*

- La classe Pompier permet au pompier de se déplacer et d'éteindre le feu.
- Les classes Forêt, Plaine, Eau et Maison permettent de construire des cases et de propager le feu selon leur nature.
- La classe grille permet de sauvegarder et de charger une partie.
- La classe affichage gère l'interface graphique.
- La classe Lancement_Simulation lance le jeu.

### *Organisation du programme :*

L'intégralité du jeu est gérée par la méthode afficher().
Au lancement du jeu, le joueur peut :

- Démarrer une nouvelle partie : La méthode initialisation () de la classe Grille créé une liste de couple <Case, JButton>. Cette grille initiale est unique.
- Chargé une partie déjà commencée : Un appel à la méthode charger() de la classe Grille renvoie une liste de couple <Case, JButton> en récupérant les cases préalablement enregistrées dans la BDD incendie (table cases) et en associant les JButton correspondants.

Rem : Si aucune partie n'a été enregistrée (table cases est vide), la méthode charger() fait appel à la méthode initialisation().

A chaque tour, le joueur peut choisir de :

- Se déplacer : La méthode deplacement() de la clase Pompier retourne la liste des positions accessibles.
- Réaliser 1 ou 2 fois l'action d'abaisser l'intensité d'une case en feu puis se déplacer : La méthode eteindreFeu() permet d'abaisser l'intensité d'une case (i,j) et est suivie de la méthode deplacement().
- Réaliser 3 fois l'action d'abaisser l'intensité d'une case en feu : 3 appels à la méthode eteindreFeu().

L'arrêt d'une partie se fait :

- Par défaite : Si 5 maisons passent à l'état carbonisé la méthode afficher() arrête la partie et affiche un message de défaite.
- Par victoire : Si le nombre de case en feu est nul, la méthode afficher() arrête la partie et affiche un message de victoire.
- Par manque de temps : Le joueur peut à chaque fin de tour enregistrer sa partie. La méthode sauvegarder() entre tout les attributs des cases en fin de tour dans une BDD.

### *Exécution du programme :*

Pour jouer, l'utilisateur doit

- ✓ ouvrir le package incendie dans Eclipse.
- ✓ Lancer le main de la classe Lancement_Simulation.
- ✓ Jouer.

## Bilan du projet

### *L'organisation*

Pour nous organiser, nous avons commencé par définir les objectifs minima du projet. Le but étant de privilégier un programme simple et fonctionnel et le perfectionner en fonction du temps. À partir de là, nous avons mis en place deux outils d'organisation :

- Les diagrammes de cas d'utilisation, de classe, de séquence et d'activité nous ont donné l'organisation fonctionnelle de notre projet.

- Le gant nous a donné l'organisation temporelle. Nous permettant Ainsi de perfectionner notre projet en fonction de notre avancement.

### *Les objectifs atteints*

Les objectifs que nous avons pu atteindre sont :

- Création d'une interface graphique.
- La propagation de l'incendie.
- Le déplacement et l'action des pompiers.
- Sauvegarde et chargement d'une partie via une BDD. Pour cela, nous avons du rajouter une classe ConnexionSingleton dans notre package Incendie.
- La cohérence de notre jeu de simulation.

### *Les objectifs non atteint*

- Création de grilles de départ aléatoires par manque de temps.
- Mise en place et gestion de plusieurs pompiers sur la grille par manque de temps.
- Il reste également des erreurs dans le programme telles que : la disparition du JButton coin haut-gauche sur l'interface, le feu initial qui ne change pas d'état tout au long de la partie, le saut du feu au cours de la partie qui doit venir d'un problème d'indice.

### *Les acquis du projet*

Dans ce projet nous avons utilisés une BDD et nous avons donc accrus notre maîtrise en la matière. Nous pouvons maintenant établir une connexion entre JAVA et une BDD et la gérer correctement dans notre code.
Nous avons également découvert toute la partie Java2d qui nous a permis de réaliser une interface graphique et qui constitue une grande partie de notre code.
Nous avons enfin acquis une plus grande maitrise de JAVA et bien entendu une plus grande maîtrise dans l'extinction des feux de forêt.

## Conclusion

Au travers d'un projet stimulant, nous avons pu appliquer concrètement des cours théoriques de l'année sur les diagramme UML ou le langage SQL et se perfectionner en programmation orienté objet.