Projet Informatique

Equipe 1

-

Hashiwokakero

Rapport de projet

Equipe :
AGOSTINHO DA SILVA Ninoh
HAMILCARO Joël
RALPH José
TU Jie

Encadrant :
NUNGE Arthur

Projet de programmation — L2 2018/2019

Table des matières

Résumé

Dans le cadre d'un projet universitaire, nous avons développé un jeu de Hashiwokakero en java. Ce document est le rapport de ce projet que nous avons réalisé pendant trois mois.

1 Retour sur les objectifs du cahier des charges

1.1 Objectifs principaux atteints

Les objectifs principaux de ce projet étaient les suivants :

1. Permettre à un joueur humain de jouer au jeu.
2. Écrire un algorithme de résolution d'une grille et proposer à l'utilisateur une fonction d'aide.
3. Générer de façon aléatoire des grilles jouables avec des niveaux de difficultés.
4. Étendre et personnaliser le jeu pour qu'il ne se limite pas à un jeu de Hashiwokakero classique.

Les quatre grands objectifs listés ci-dessus ont été atteints.

1.1.1 Jouabilité

Notre adaptation du jeu Hashiwokakero se joue exclusivement à la souris. Un tutoriel a été mis en place dans le jeu, pour réexpliquer les règles et lister de manière exhaustive les mécaniques de jeu de notre adaptation.

Concernant les fonctionnalités plus précises, nous avons implémenté comme prévu le chronométrage de la partie, les différentes manière de créer des ponts et la suppression des ponts avec le clic droit.

1.1.2 Solveur, générateur de plateaux et fonctions d'aide

Nous parlerons plus bas de l'algorithme de résolution et de la génération aléatoire de grilles. Pour les fonctions d'aide, nous avons opté pour le deuxième choix [1] que nous avons proposé dans notre cahier des charges. Le joueur a accès à différents joker. Chaque joker est utilisable une fois par partie. Le joker représente un type d'aide en particulier.

1.1.3 Extensions et personnalisation du jeu de Hashiwokakero

Nous avons choisi de créer un mode histoire avec un scénario et des éléments visuels supplémentaires pour remplir le contrat d'extension du jeu. L'esthétique du jeu (cinématique d'introduction, images, menus, musiques, animations) a été choisie de telle sorte qu'elle illustre l'univers de l'histoire.

Un fond aquatique animé (.gif) est activable et désactivable selon les préférences du joueur dans les paramètres. Ce paramètre concerne le phases de jeux (i.e. ce paramètre n'implique pas la suppression du fond aquatique des menus)

1.2 Ajouts supplémentaires non prévues dans le cahier des charges

Nous avons ajouté une fonction d'aide légère (non comptée dans les joker) ; une croix se trace sur les îles qui n'ont plus de ponts restants. Nous avons cinq tailles différentes de plateaux que nous pouvons générer aléatoirement, en un temps de chargement raisonnable (moins de 30 secondes), Il s'agit des plateaux 5x5, 6x6, 7x7, 8x8 et 9x9. Dans notre cahier des charges, nous avions prévu de ne proposer que trois tailles.

1. Le choix n°1 était le suivant :
le joueur a la possibilité d'activer ou de désactiver à sa guise des options d'aide (avant et pendant la partie).

1.3 Modifications et problèmes rencontrés

1.3.1 Méthode du "survol"

La méthode de glisser-déposer pour tracer les ponts faisant débat, nous avons créé la méthode du "survol", activable et désactivable dans les paramètres, comme alternative au clic en deux temps pour le tracé des ponts.

1.3.2 Algorithme de résolution

L'algorithme de résolution est sans aucun doute la fonctionnalité qui nous a pris le plus de temps. Il devait compléter notre algorithme de génération de plateaux qui créait des grilles valides, mais la solution n'était pas forcément unique. Sans entrer dans les détails du code, nous allons décrire comment il fonctionne, et comment nous l'avons optimisé.

Dans sa version initiale, l'algorithme de résolution utilisait un backtracking qui testait toutes les configurations possibles (non absurde) afin de démontrer qu'une seule solution était possible. Cependant, même s'il ne testait pas les configurations absurdes (par exemple un 1 connecté avec un 1), il fallait quand même arriver au début du tour de boucle "for" pour qu'il s'en rende compte, et qu'il passe directement à une autre configuration. Cela implique qu'il faisait tout de même autant de tour de boucle que s'il devait tester des configurations absurde.

Dans sa version finale, nous faisons un calcul préalable précis pour connaître à l'avance le nombre de tour de boucles nécessaires et suffisantes à réaliser pour chaque île en prenant en compte différents paramètres sur l'état courant de ses voisins. Cela a grandement optimisé notre algorithme.

L'algorithme de résolution s'arrête si plusieurs solutions ont été trouvée ou si le nombre d'appels récursifs est trop élevé (dans ce cas on abandonne et on génère un nouveau plateau).

L'unicité des solutions nous permet de proposer des fonctions d'aide au joueur, en sachant qu'il ne peut pas trouver une solution alternative dans laquelle la fonction d'aide le pousserait à faire une erreur.

1.3.3 Idées abandonnées

Nous n'avons pas réussi à définir ce qu'était un niveau de difficulté pour les plateaux générés aléatoirement par notre algorithme. Par chance, un plateaux pourrait être considéré comme "facile" si l'algorithme avait directement commencé par faire les bons choix pour déterminer la solution. La difficulté déterminée par l'algorithme n'était donc pas significative. Nous avons donc laissé la catégorisation en tailles différentes, plutôt que par niveau de difficulté.

Nous avons également abandonné les idées suivantes qui étaient impossibles à mettre en place ou bien trop dépendantes de l'ordinateur utilisé :

— usage du multi-threading (pour générer des plateaux en continu pendant que le joueur puisse faire autre chose pendant le jeu).
— Lecture de vidéos qui nécessite d'installer des librairies externe dépendantes de la machine.

2 Modélisation du jeu

2.1 Descriptif du jeu

2.2 Choix de modélisation

Nous avons adopté une structure MVC (modèle-vue-contrôleur), illustrée par l'architecture du code source en trois packages. Le package modèle contient toute la logique du jeu et les éléments constants (comme les dialogues pour le mode histoire, les paramètres, et les chemins vers les fichiers ressources). Ce package est totalement indépendant de JavaFX. Les packages vues et controleur utilisent JavaFX. Le package controleur sert uniquement en cours de partie, pour gérer la synchronisation entre la vue du plateau et le modèle. Les vues hors-partie (comme les menus), ne nécessitent pas de contrôleur.

2.3 Diagramme UML

Le Diagramme UML qui représente la modélisation de notre code se trouve en annexe de ce rapport.

2.4 Documentation du code source

Notre code a été documenté, pour que chaque membre de l'équipe puisse comprendre les fonctions implémentés par les autres afin de les réutilisées.

Pour y accéder il faut ouvrir le fichier " index.html " dans le dossier " javadoc/ ".

3 Organisaton et repartition

Pour avoir la répartition détaillée des tâches au sein de l'équipe, il faut se référer aux crédits du jeu. En résumé, les tâches ont été répartie de la manière suivante :

- **Ninoh** : L'essentiel de l'interface graphique en cours de partie, programmation de l'intégralité du gameplay, des actions du joueur sur la vue, les animations du mode histoire, le design des plateaux, le design des éléments scénaristiques du mode histoire, le script du scénario, le script du tutoriel, l'animation des crédits, rédaction du rapport, la gestions de dépots GIT (depuis que nous sommes passés par SSH).
- **Joël** : L'interface graphique hors-partie (menus, cinématiques...), la gestions des passage menu/partie, la synchronisation modèle-vue, aide et correction du solveur, traducteur matrice-niveaux (qui permet de construire les plateaux stockés sous forme de matrice dans un fichier .txt), gestions des des sauvegardes, gestion de la mise en commun du code, architecture générale MVC.
- **José** : Programmation des plateaux, implémentation de la liste des niveaux, l'essentiel du solveur et tous les algorithmes liés à la résolution et la vérification de l'unicité de la solution d'un plateau, l'algorithme de backtracking, gestion et intégration des ressources externes (images, musiques...) dans le code, insertions des dialogues dans le modèle.
- **Jie** : La majorité du modèle, programmation des actions de jeu, de la logique du jeu, fonctions liés aux actions de jeu dans le modèle, l'intégralité du générateur de plateaux. (Plateaux valides sans forcément de solution unique), tous les algorithmes liés à la génération du plateau, version finale optimisée du solveur, essentiel de la documentation du code.

A Annexes - Diagramme UML

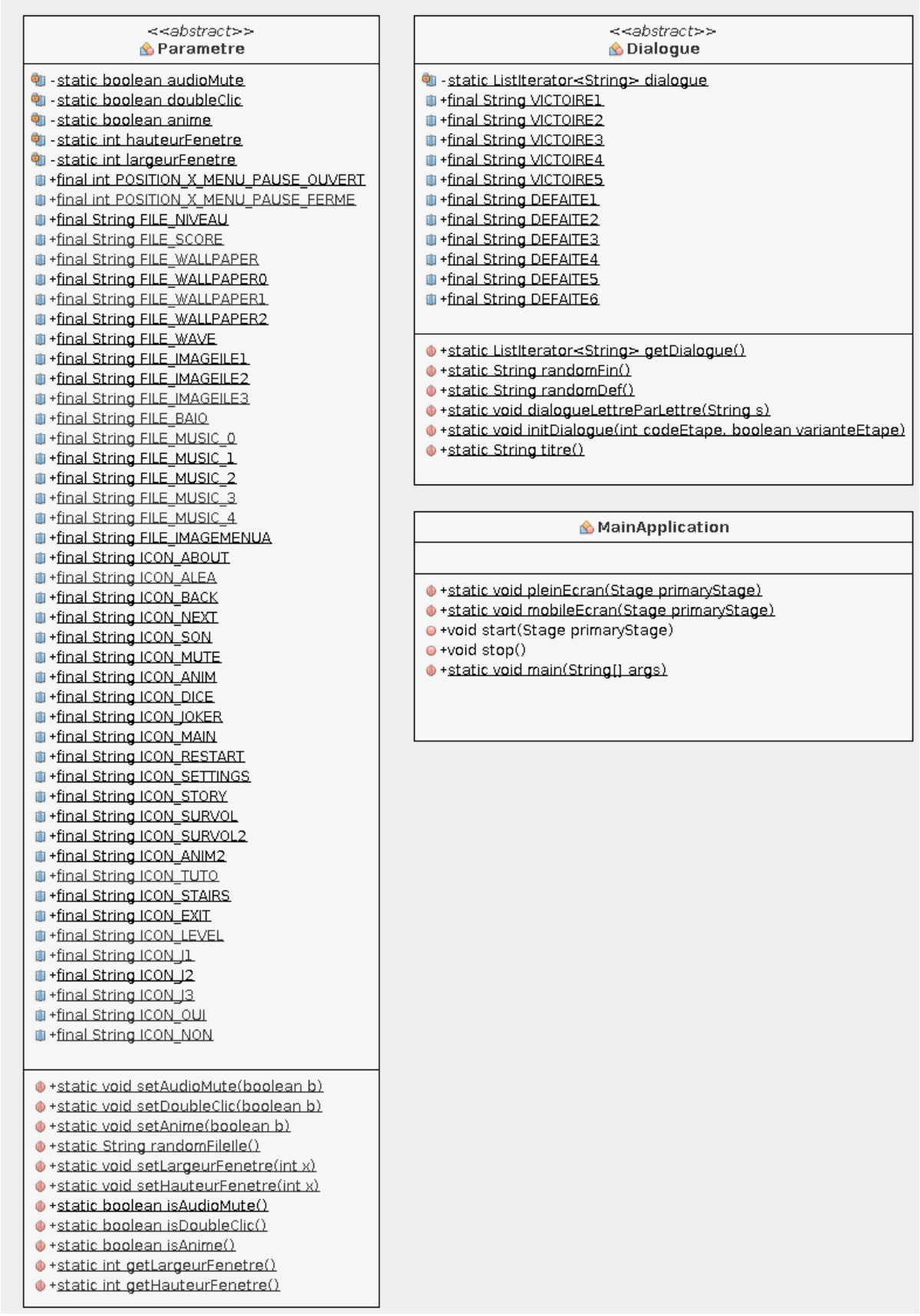

FIGURE 1 – Éléments constants du modèle (paramètres, dialogues) - Classe principale

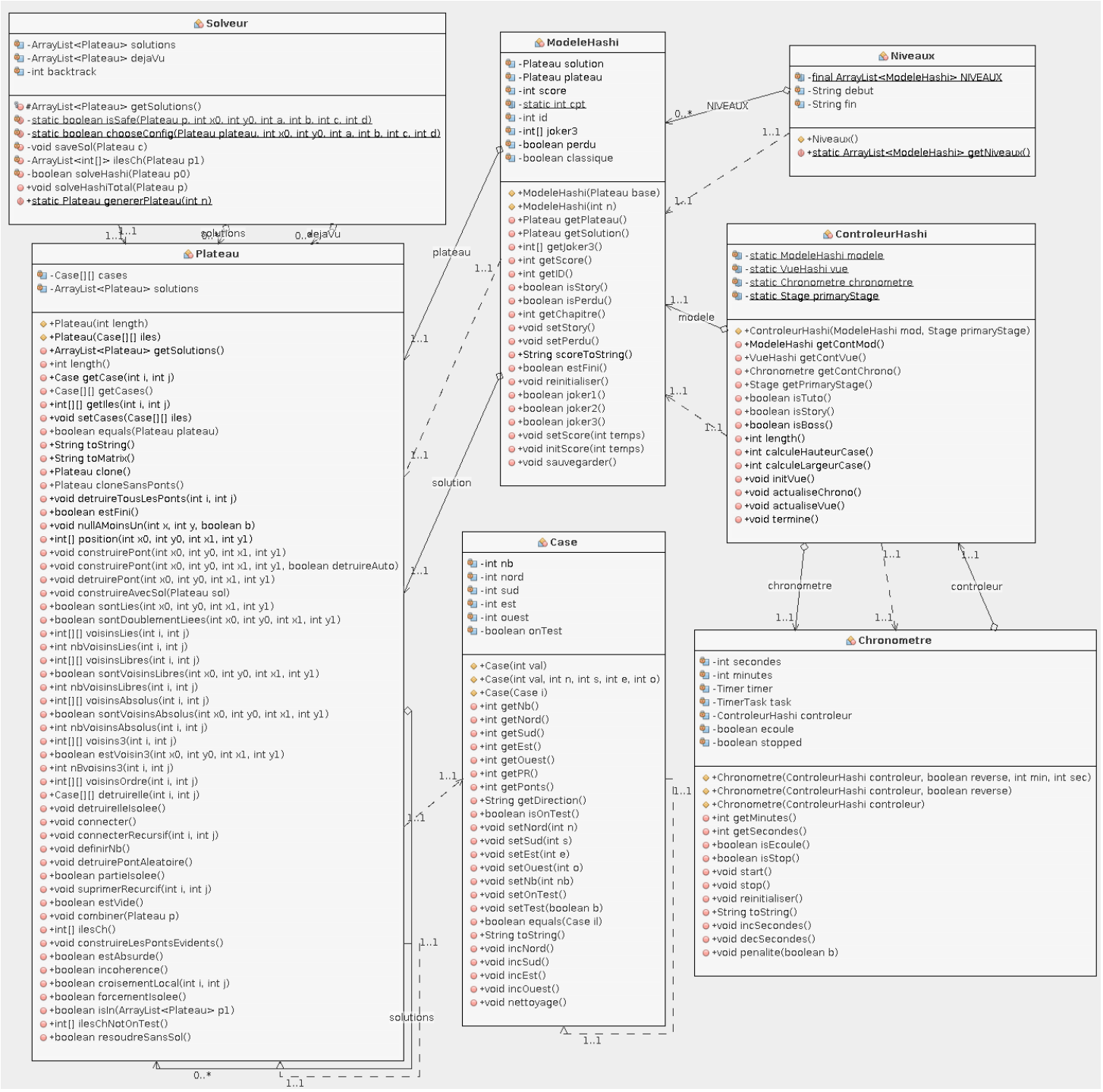

FIGURE 2 – Toute la logique du jeu - Lien avec le contrôleur

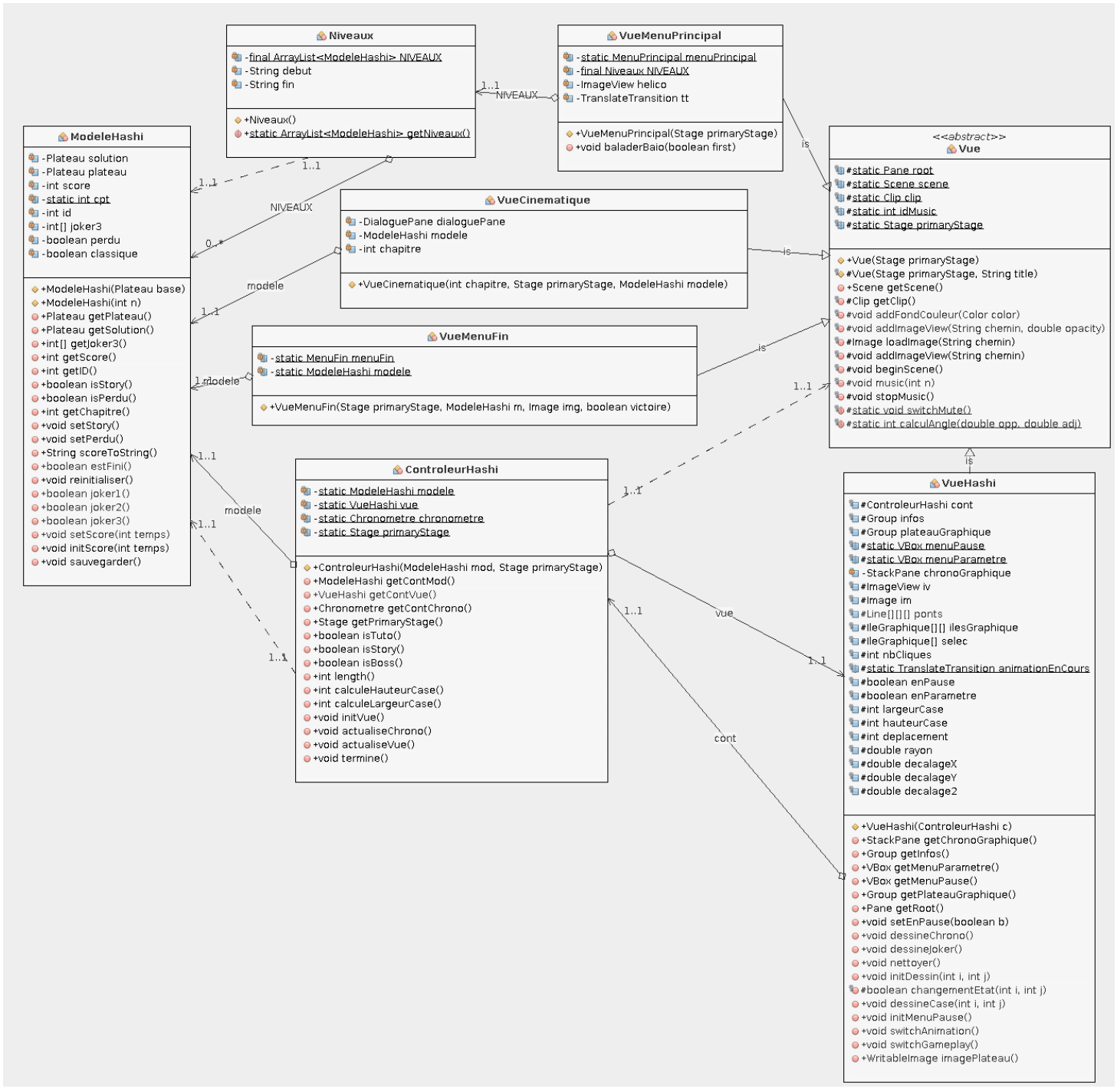

Figure 3 – Liens entre le modèle et les vues

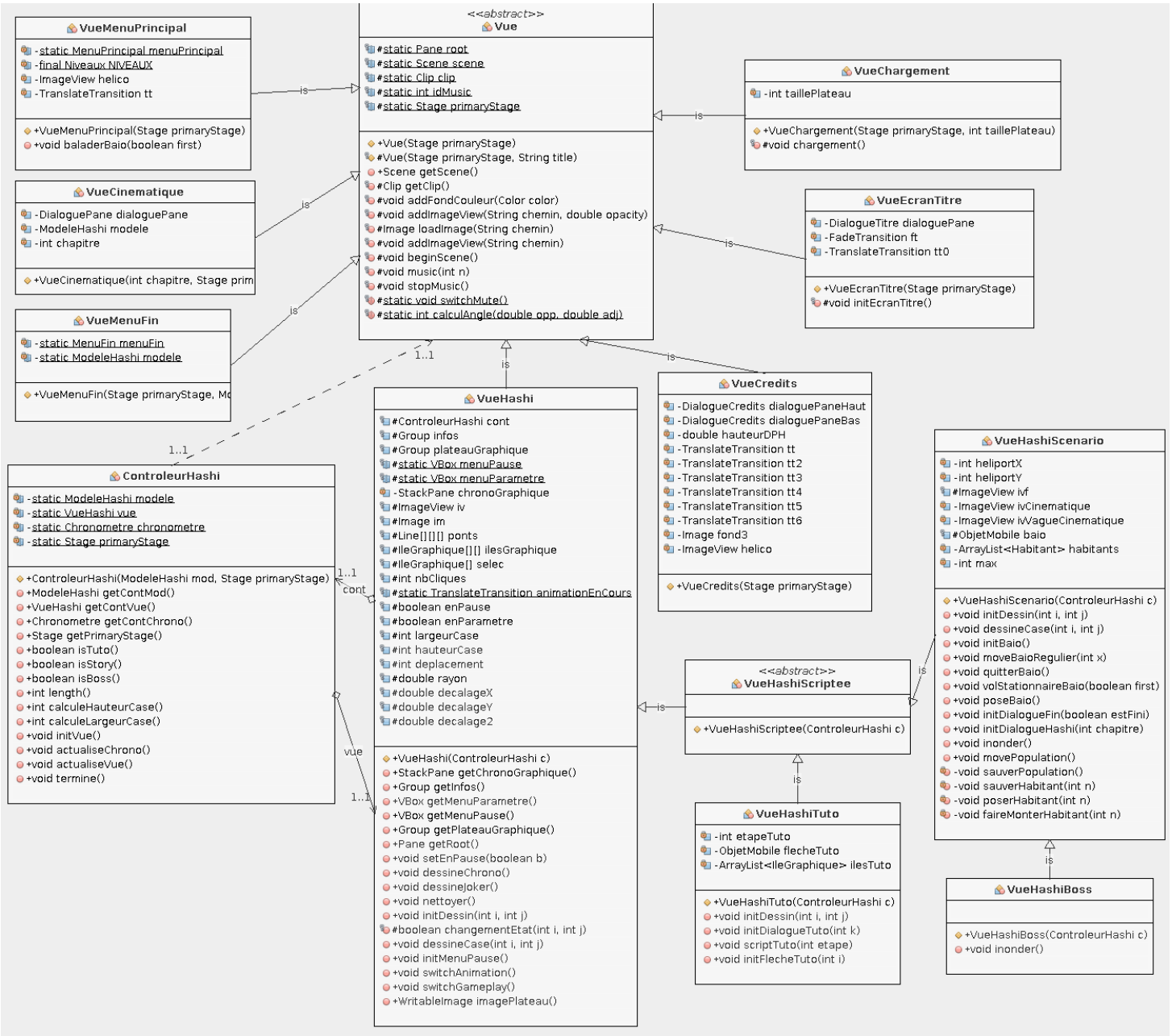

Figure 4 – Toute l'interface graphique - Lien avec le controleur