TELECOM Nancy
Université de Lorraine
Rapport de PIDR

Encadrants universitaire : Martin Quinson & Gérald Oster

Programmation pour débutants grâce à la plate-forme JLM

Julien BASTIAN - IL
Geoffrey HUMBERT - TRS

Nancy, *l*e 23 mai 2013

Sommaire

I. Introduction 1
1. Objectif du module 1
2. Contexte 1
3. Enjeux 1
II. Organisation du projet 2
1. Environnement et organisation 2
2. Répartition des tâches et feuille des temps 3
III. Travail effectué 4
1. Réalisation d'univers dans la JLM 4
2. Réparation de l'univers des tris 9
3. Ajouts divers dans l'univers des labyrinthes 10
4. Amélioration du support des contrôles interactifs 11
5. Amélioration de la traduction en français de la JLM 14
6. Divers 15
V. Discussion 16
1. Problèmes rencontrés 16
2. Documentation 16
3. Extension des mondes des crêpes et du baseball 16
4. Conclusion 16
A. Références 17

I. Introduction

1.1. Objectifs du module

Le *Projet Interdisciplinaire* ou *Découverte de la Recherche* est un projet de 6 mois servant d'introduction aux élèves de deuxième année à l'univers de la recherche.

Dans ce module, contrairement aux projets standards, les élèves sont confrontés à un problème pour lequel il n'existe pas encore de solution explicite. Ils doivent donc explorer des pistes afin de trouver la bonne approche pour résoudre efficacement le problème.

1.2 Contexte

Java Learning Machine

Java Learning Machine (JLM) est un outil pédagogique développé par Martin Quinson et Gérald OSTER permettant aux étudiants d'apprendre les concepts de base de la programmation.

Il est utilisé à TELECOM Nancy dans les modules Préparation Informatique et TOP (Techniques et Outils pour Programmer) enseignés en première année. La plate-forme est composée de nombreux exercices à difficulté progressive, couvrant l'ensemble des notions de bases de la programmation (introduction aux variables de données, aux instructions de boucle, aux déclarations et appels de méthodes, etc).

Les exercices sont construits de la manière suivante : une introduction avec un petit scénario, une leçon, et un objectif pour mettre en pratique ce qui a été appris.

L'utilisateur acquiert donc pas à pas de nouvelles notions. La plateforme sauvegarde le code de l'étudiant, lui permettant ainsi de revoir ce qu'il a fait, et de potentiellement améliorer ce qu'il a fait.

Ainsi, Java Learning Machine introduit les élèves à des programmations de type procédural et impératif, ce qui constitue déjà une partie importante de la programmation. Cependant l'outil ne possède ni leçons ni exercices sur la programmation orientée objet.

Sciences Manuelles du Numérique

Le projet des Sciences Manuelles du Numérique propose un ensemble d'activités sans ordinateur permettant d'introduire les notions centrales de la science informatique. Cela évite les écueils qui rebutent classiquement les débutants, comme la difficulté de dialogue avec l'ordinateur, la syntaxe rigide ou les problèmes techniques.

La principale activité existante commence par introduire la notion simple d'algorithme, mais aborde également les preuves de correction d'algorithme, les heuristiques et les classes de complexité algorithmique (comme P et NP).

SMN permet ainsi de dédramatiser l'informatique auprès des débutants, et de montrer que les fondements de notre science sont bien dé-corrélés de la technologie des ordinateurs. Mais la transition à l'usage de l'informatique sur ordinateur reste très difficile.

1.3 Enjeux

L'enjeu principal du projet est d'implémenter dans la JLM différentes activités réalisées au cours du projet SMN afin de proposer une passerelle entre l'activité manuelle et l'ordinateur sans pour autant laisser l'utilisateur nager en eaux troubles grâce à l'utilisation de la JLM et de ses avantages.

II. Organisation du projet

2.1 Environnement et organisation

Environnement

Nous avons utilisé nos machines personnelles avec l'environnement de développement Eclipse (version Juno).

Le projet JLM est hébergé sur la forge GitHub, qui utilise le gestionnaire de version Git. Ainsi, chaque développeur envoie ses contributions sur le dépôt et peut récupérer celles des autres.

Nous avons créé une copie (« fork ») du projet actuel depuis la copie de M. Oster, qui a pu être modifiée indépendamment du projet original par la suite.

Les modifications apportées peuvent facilement être fusionnées avec le projet original via une « pull request ». Une fois la requête acceptée, la fusion peut se faire de façon automatique s'il n'y a pas eu de modification concurrente de fichiers. Sinon, il faudra résoudre les fichiers en conflits manuellement.

- Dépôt pour le code de JLM : github.com/g-humbert/JLM

Organisation

Pour que les enseignants puissent suivre notre progression, nous avons effectué chaque semaine un « reporting » par mail qui recensait le travail effectué au cours de la semaine, les objectifs à plus ou moins long terme et les questions éventuelles.

Tout le travail effectué sur le projet a été réalisé en anglais : code source, commentaires, messages de commit au dépôt et documentation.

En effet, JLM étant un logiciel libre, il doit donc accessible et modifiable par tout le monde. Ainsi, n'importe qui dans le monde doit être en mesure d'apporter une contribution ou d'utiliser le logiciel.

Enfin, nous avons utilisé l'outil Javadoc pour documenter notre code.

2.2 Répartition des tâches et feuille des temps

Répartition des tâches

Le travail s'est tout d'abord effectué en commun, selon une méthode de programmation de type agile.

Cependant, des divergences dans la méthode de travail étant apparue à partir du mois de février, cette organisation a été abandonnée pour passer à une organisation plus séparée où chacun se consacrait à une tâche distincte.

Feuille des temps

	Julien BASTIAN	Geoffrey HUMBERT
Monde des crêpes	31h	7h
Monde du base-ball	36h	8h
Monde des tris	16h	-
Monde des labyrinthes	11h	-
Contrôles interactifs	13h	-
Traduction française	13h	-
Débogage	18h	-
Git + Reporting	6h	-
Divers travaux	41h	5h
Rédaction du rapport	11h	-
Total	196h	20h

III. Travail effectué

3.1. Réalisation d'univers dans la JLM

3.1.1 Méthode générale

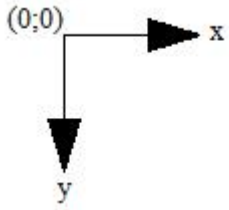
Pour réaliser un univers dans la JLM, nous avons toujours suivi les mêmes étapes :

1) Réalisation des classes internes qui serviront à représenter le problème

2) Réalisation du monde (*World*) : il contient les données du problème et utilise les classes internes, c'est l'équivalent du Modèle dans le patron Modèle-Vue-Contrôleur.

3) Réalisation de l'entité (*Entity*) : elle est responsable de modifier les données du monde, c'est l'équivalent du Contrôleur dans le patron Modèle-Vue-Contrôleur.
Elle est aussi chargée d'effectuer la démonstration de la résolution du problème.

4) Réalisation de la vue du monde (*WorldView*) : elle est responsable de la représentation graphique du monde, c'est l'équivalent de la Vue dans le patron Modèle-Vue-Contrôleur.
Pour représenter notre monde, il faut utiliser les méthodes de Java AWT (Abstract Windows Toolkit) qui nous permettent de dessiner au pixel près.
Il faut tout de même noter que les dessins se font dans le repère suivant :



5) Facultatif : réalisation d'un panneau de contrôle interactif (*EntityControlPanel*). Ce panneau permet à l'utilisateur d'interagir directement avec l'univers juste en cliquant sur les boutons ou en sélectionnant des valeurs dans des listes déroulantes.

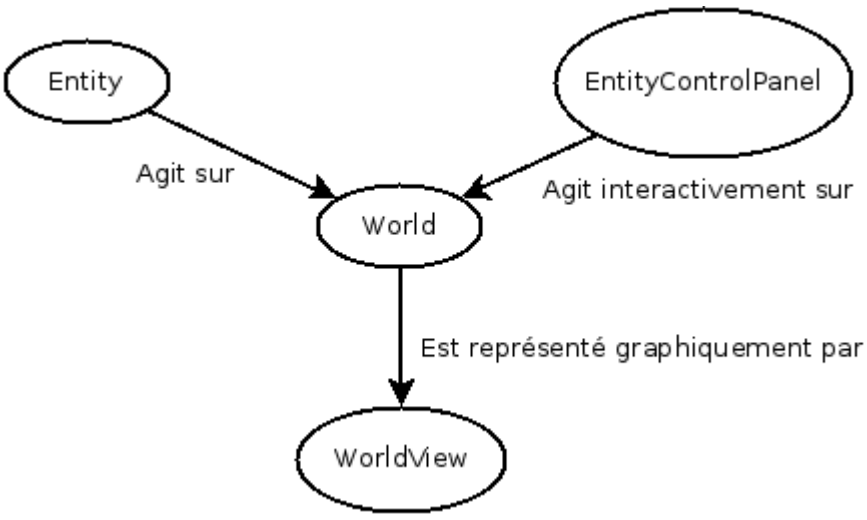


Figure 1: Structure de la JLM

3.1.2 L'univers du crêpier psychorigide

Énoncé du problème

A la fin de sa journée, un crêpier dispose d'une pile de crêpes désordonnée. Le crêpier étant « un peu » psychorigide, il décide de ranger sa pile de crêpes, de la plus grande (en bas) à la plus petite (en haut), avec le coté brûlé caché.

Pour cette tâche, le crêpier peut faire une seule action : glisser sa spatule entre deux crêpes et retourner le haut de la pile.

Comment doit-il procéder pour trier toute la pile ?

Réalisation

Nous avons scindé ce problème en deux par rapport à la version originale comme présentée dans la brochure des Sciences Manuelles du Numérique:

→ L'un où on doit se contenter de trier le tas de crêpes

→ L'autre où, en plus de trier le tas de crêpe, il faut aussi tenir compte de l'orientation du côté brûlé.

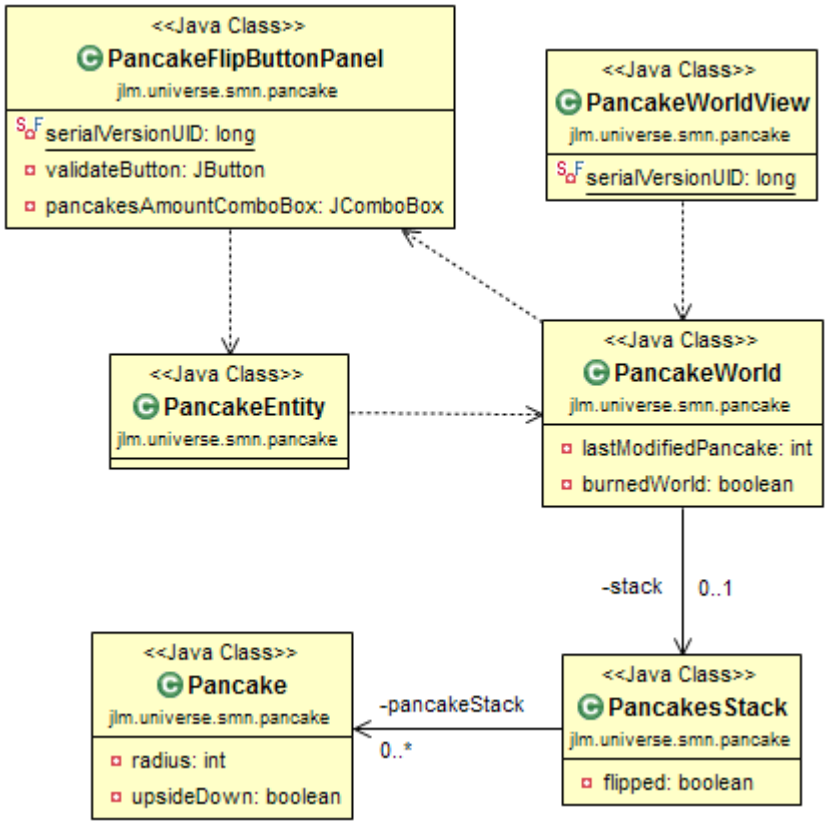


Figure 2: Diagramme de classes partiel de l'univers des crêpes

Les méthodes disponibles pour l'utilisateur lui permettent de connaître la taille du tas de crêpes, les caractéristiques d'une crêpe (son rayon et si la face brûlée est vers le haut ou non) en fonction de son numéro dans la pile. Il peut en outre savoir si le tas est trié correctement au sens du crêpier psychorigide et retourner un nombre quelconque de crêpes en partant du haut du tas.

Rendu graphique

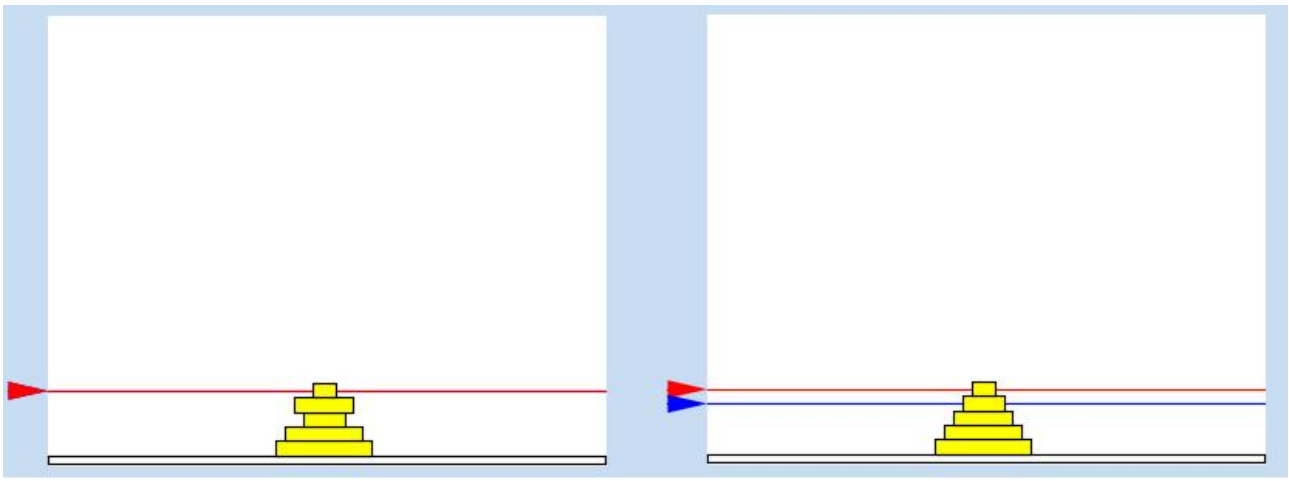


Figure 3: Rendu graphique de l'univers des crêpes non brûlées

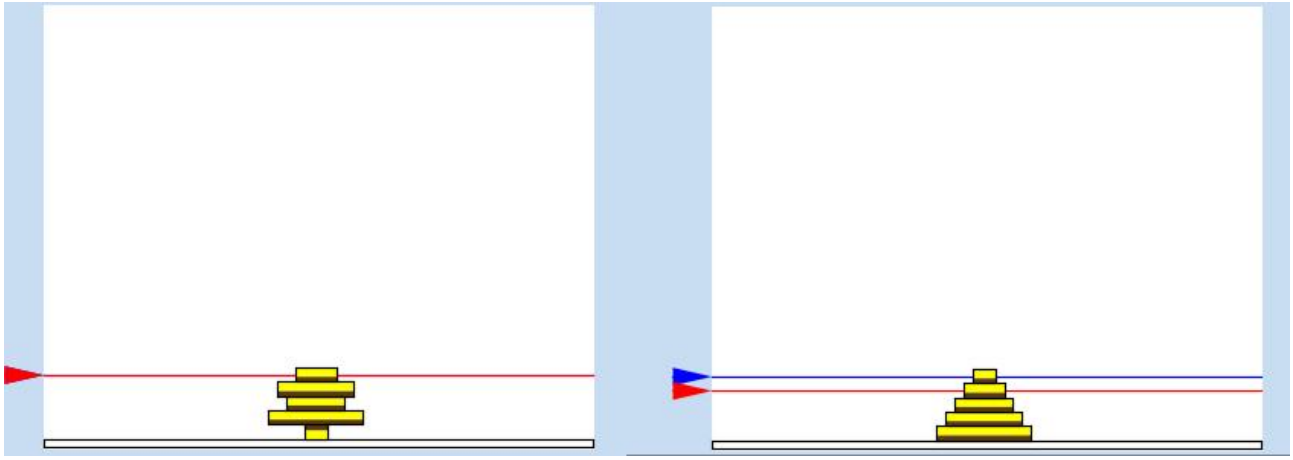


Figure 4: Rendu graphique de l'univers des crêpes brûlées

3.1.3 L'univers du base-ball multicolore

Énoncé du problème

On dispose de quatre bases de couleurs différentes, et deux joueurs associés à chaque base. Le but du jeu est de déplacer les joueurs afin d'amener chaque joueur sur la base correspondant à sa couleur.

Il y a cependant trois contraintes :

• les bases sont disposées en cercle, et un joueur ne peut se déplacer que vers les deux bases voisines (il ne peut pas traverser le terrain) ;
• on ne peut déplacer qu'un joueur à la fois ;
• chaque base a deux places, et un joueur ne peut se déplacer vers une base que si elle possède une place libre

Réalisation

Nous avons réalisé des classes internes pour un joueur de base-ball, la base, le terrain de jeu (formé par l'ensemble des bases) et le dernier déplacement effectué sur le terrain. Cette dernière classe a été réalisée afin d'améliorer la compréhensibilité et la visibilité du dernier déplacement de joueur.

La couleur de la base a été représentée par un entier allant de 0 au nombre de base moins un. Le trou est représenté par -1.

Afin de pouvoir réaliser un exercice plus difficile, nous avons permis d'avoir plus de deux joueurs par base, ce qui a de plus l'intérêt pédagogique de montrer qu'il est parfois nécessaire de modifier son algorithme pour pouvoir gérer des cas plus généraux auxquels on ne pensait pas au moment d'écrire la première version de l'algorithme.

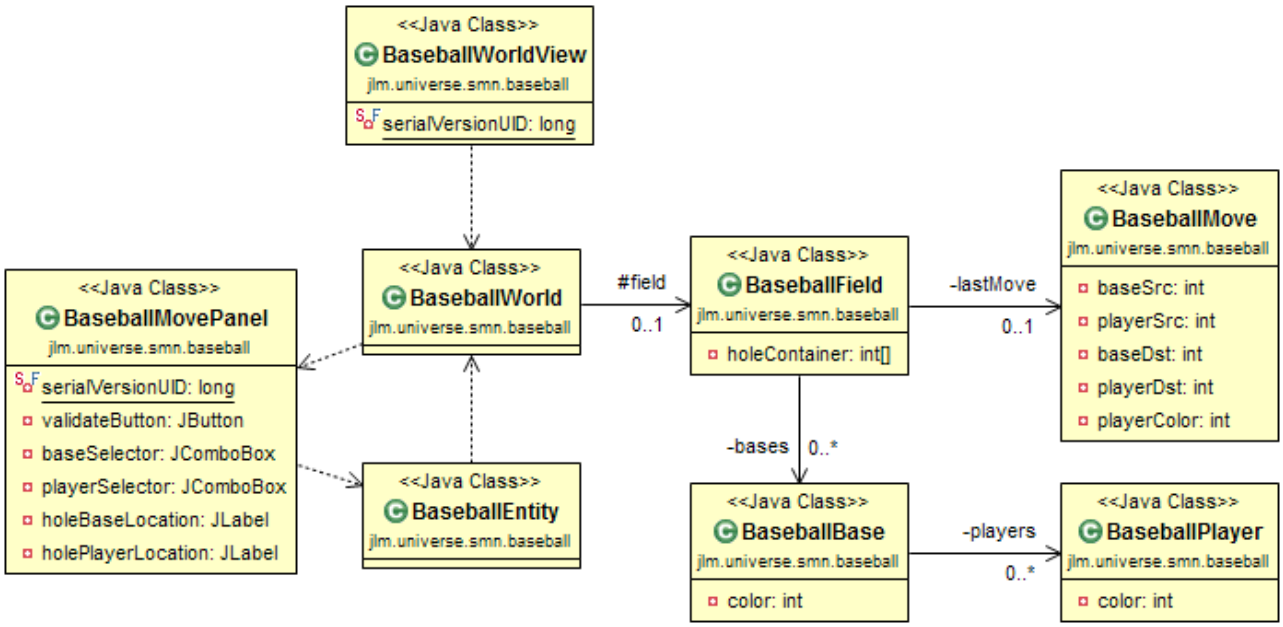
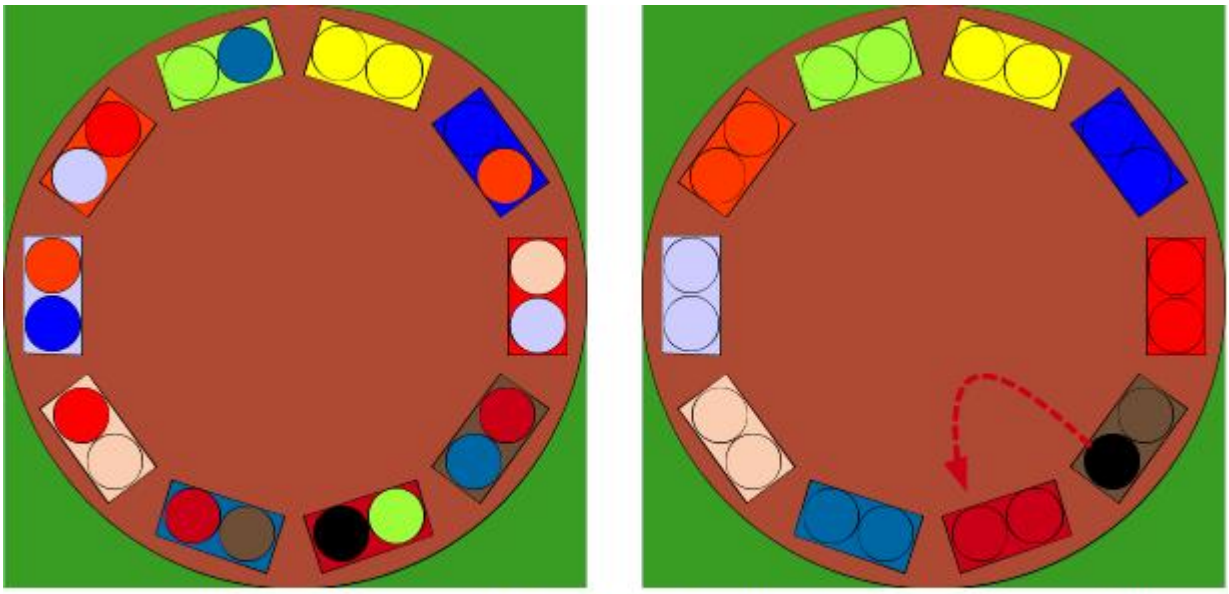


Figure 1: Diagramme de classes partiel de l'univers du base-ball multicolore

Les méthodes disponibles pour l'utilisateur lui permettent de connaître les caractéristiques du monde (nombre de bases sur le terrain, nombre de joueurs par base et localisation du trou), les caractéristiques d'une base précise (les joueurs qui sont dessus) et de déplacer un joueur d'une base adjacente au trou sur le trou.

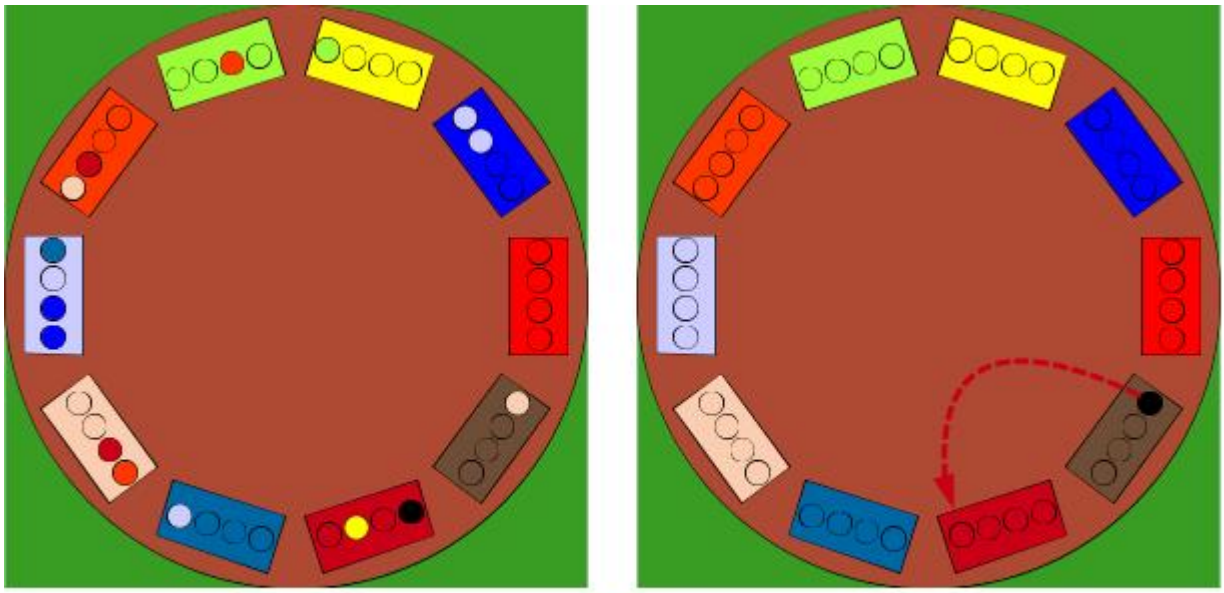
Rendu graphique



Une situation initiale possible, avec 10 bases et 2 joueurs par base

La situation finale obtenue après application de l'algorithme

Figure 2: Rendu graphique de l'univers du base-ball multicolore avec dix bases et deux joueurs par base



Une situation initiale possible, avec 10 bases et 4 joueurs par base

La situation finale obtenue après application de l'algorithme de résolution

Figure 3: Rendu graphique de l'univers du base-ball multicolore avec dix bases et quatre joueurs par base

3.2. Réparation de l'univers des tris

A la demande de M. Quinson, nous avons cherché à trouver la cause des problèmes empêchant le bon déroulement de cet univers en altérant au minimum les fonctionnalités présentes.

Ainsi, après lecture du code et différents essais de correctifs, la méthode la plus simple a été de séparer chaque algorithme de tri et de le mettre dans un exercice à part, causant néanmoins la perte de l'observation de l'exécution de tous les algorithmes en même temps.

De ce fait, il a été possible de se débarrasser de la capacité à gérer plusieurs entités sur le même monde et ainsi de simplifier, avec l'aide de M. Quinson, grandement le code sans altérer la composante graphique.

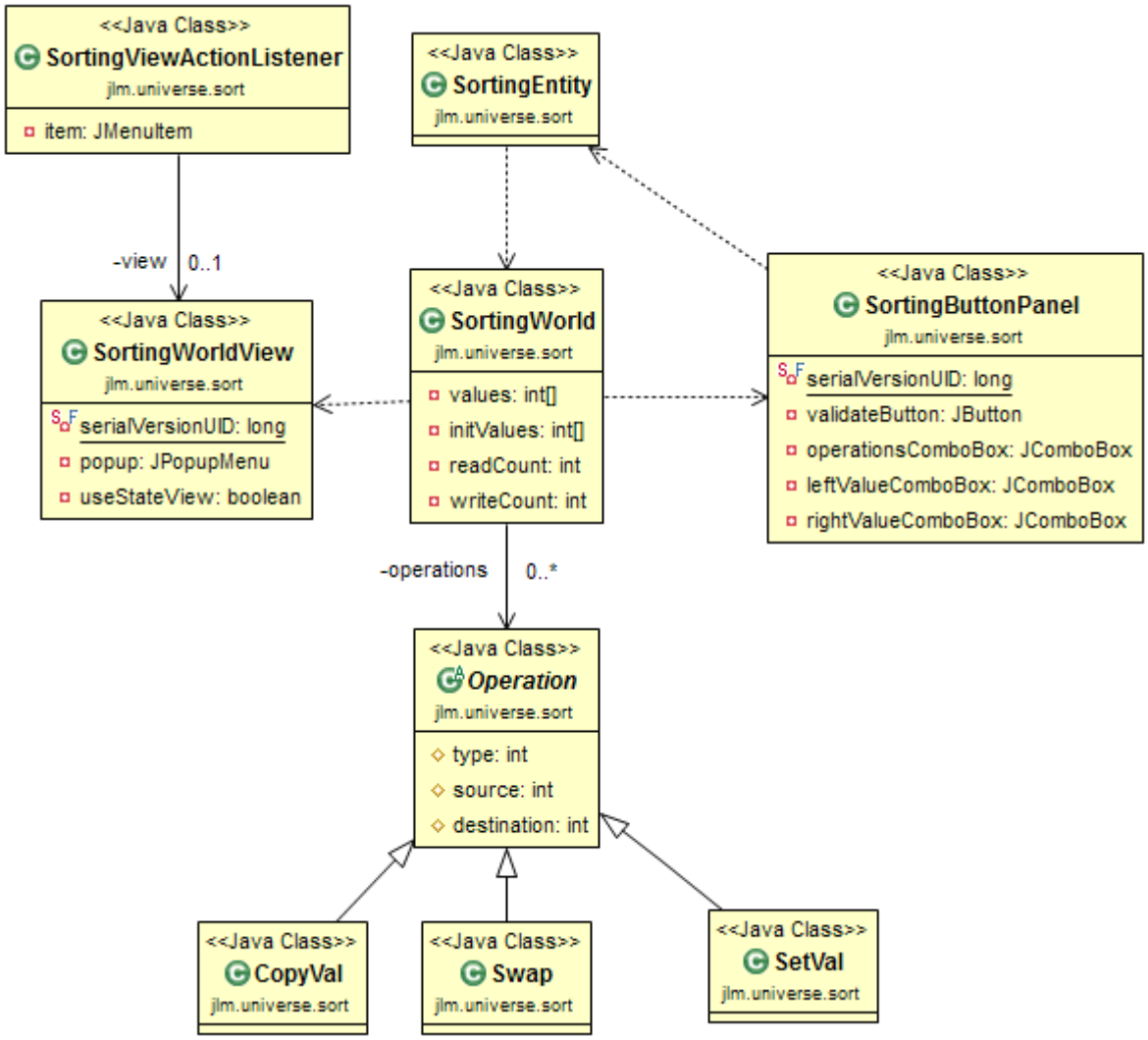


Figure 4: Diagramme de classes partiel de l'univers des tris

3.3. Ajouts divers dans l'univers des labyrinthes

Simplification de l'exercice *RandomMouseMaze*

L'énoncé de l'exercice était peu clair et l'exercice nécessitait d'écrire plusieurs fonctions pour pouvoir le réaliser alors que cet algorithme est sensé être le plus simple à mettre en place (mais aussi le moins efficace).

Nous avons donc simplifié cet exercice en faisant tel qu'il suffit de boucler tant que le Baggle n'a pas été trouver en prenant une direction aléatoire.

Correction des algorithmes de *PledgeMaze* et *IslandMaze*

La démonstration de ces exercices entraient dans une boucle infinie et ne parvenaient pas à atteindre le Baggle. Le vrai problème, au delà du fait que la démonstration ne marchait pas, était qu'il n'y avait pas de moyen de terminer l'exécution de la démonstration autrement qu'en tuant le processus de la JLM.

La solution a donc été de réécrire les algorithmes utilisés par ces entités.

Passage de tous les exercices à deux labyrinthes

Les exercices de cette univers disposaient tous (à l'exception de celui initiant à la recherche du plus cours chemin) d'un seul monde. Cela rendait possible de coder en dur la solution en écrivant par exemple : « forward(42);turnLeft() ; »

Le fait de rajouter des labyrinthes a permis de rendre si ce n'est impossible, au moins difficile de coder en dur la solution.

La création de ces labyrinthes c'est faite par le biais du *Buggle World Editor* qui permet de créer via une interface graphique des mondes pour les Buggles.

3.4 Amélioration du support des contrôles interactifs

Problématique

Les contrôles interactifs sont un moyen simple et efficace d'appréhender dans un premier temps un univers de la JLM et les méthodes qui sont à même de modifier un monde.

Néanmoins, au début du projet, ces contrôles étaient très peu développés car seul l'univers des Buggles en disposait mais de manière assez limitée, dans le sens où les seuls actions réalisables étaient de déplacer la Buggle et d'abaisser ou de remonter la brosse.

Univers des Buggles

La possibilité de changer changer les les couleurs de la brosse et de la Buggle se fait via deux listes déroulantes contenant les couleurs disponibles et les couleurs sélectionnées initialement étant celles de la Buggle et de la brosse à l'origine.

Pour des raison de clarté, un CellRenderer a été défini de tel sorte que la couleur de la liste soit représenté par un rectangle coloré plutôt que par le nom de la couleur, comme c'est le cas par défaut.

En outre, au lieu d'afficher un message d'erreur dans la console lorsque la Buggle rentrait dans un mur avec les contrôles interactifs, on affiche désormais une fenêtre de dialogue expliquant l'erreur.

Il a été toutefois nécessaire de modifier le moment où le panneau de contrôle interactif est rafraîchit afin de résoudre un problème d'initialisation de ces listes déroulantes.

Figure 5: Aperçu du panneau de contrôle des Buggles

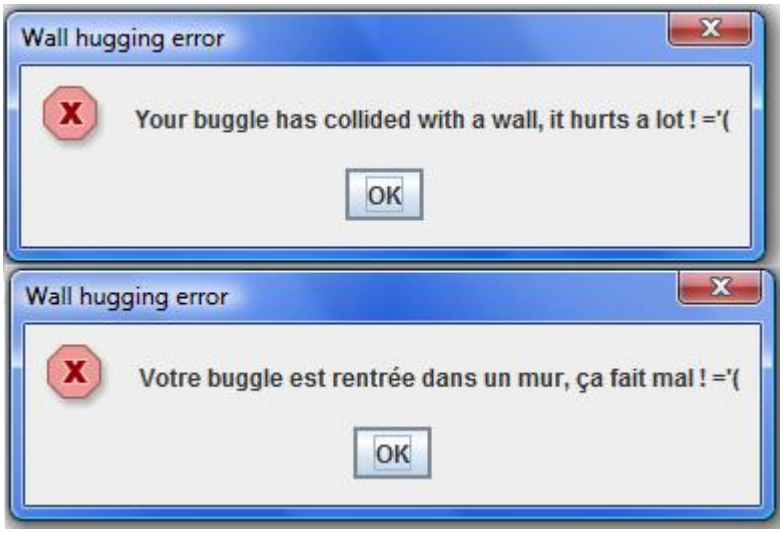


Figure 6: Aperçu des messages d'erreurs engendrés par la confrontation entre une Buggle et un mur via les contrôles interactifs

Univers des tris

En plus de la réparation et de la simplification du code de cet univers, des contrôles interactifs ont été rajoutés afin de permettre à l'utilisateur d'effectuer manuellement l'algorithme avant de le coder.

Les contrôles interactifs lui laissent le choix complet de l'opération à effectuer sur le tableau, sans que celui-ci n'ait à craindre de dépasser les bornes du tableau.

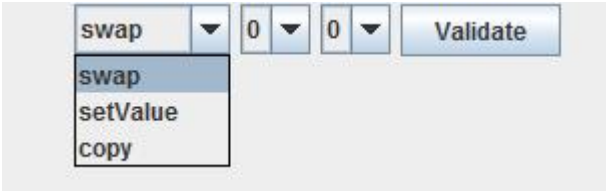


Figure 7: Aperçu du panneau de contrôle du monde des tris

Univers des crêpes

L'univers des crêpes étant une implémentation d'une des activités des Sciences Manuelles du Numériques, il s'adresse tout particulièrement à des débutants en informatique.

C'est avec ce constat que la nécessité de permettre les contrôles interactifs dans cet univers s'est fait voir au grand jour.

Le contrôle interactif est très simple est se résume à permettre de retourner un nombre donné des premières crêpes du tas.

Si l'on tient compte de l'orientation du côté brûlée des crêpes, alors la valeur minimale via le contrôle interactif est de 1 ; sinon elle est de 2.

La valeur maximale est toujours du nombre de crêpes dans le tas.

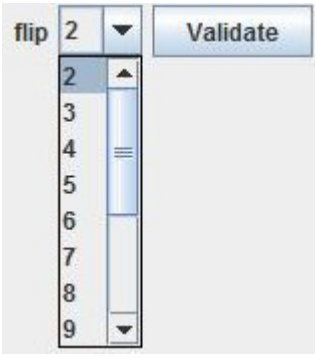


Figure 8: Aperçu du panneau de contrôle du monde des crêpes

Univers du base-ball

A l'instar de l'univers des crêpes, le monde du base-ball s'adresse à des débutants en informatique. Il était donc important d'ajouter la possibilité de modifier interactivement le monde afin d'aider à la bonne compréhension de la fonction « move ».

Le panneau de contrôle est composé de deux parties :

→ l'une permettant d'utiliser la méthode *move()* de manière similaire à la méthode *flip()* du monde des crêpes.

→ l'autre indiquant la localisation du trou dans le terrain.

Il est à noter que les mouvements invalides sont affichés dans une fenêtre de dialogue.

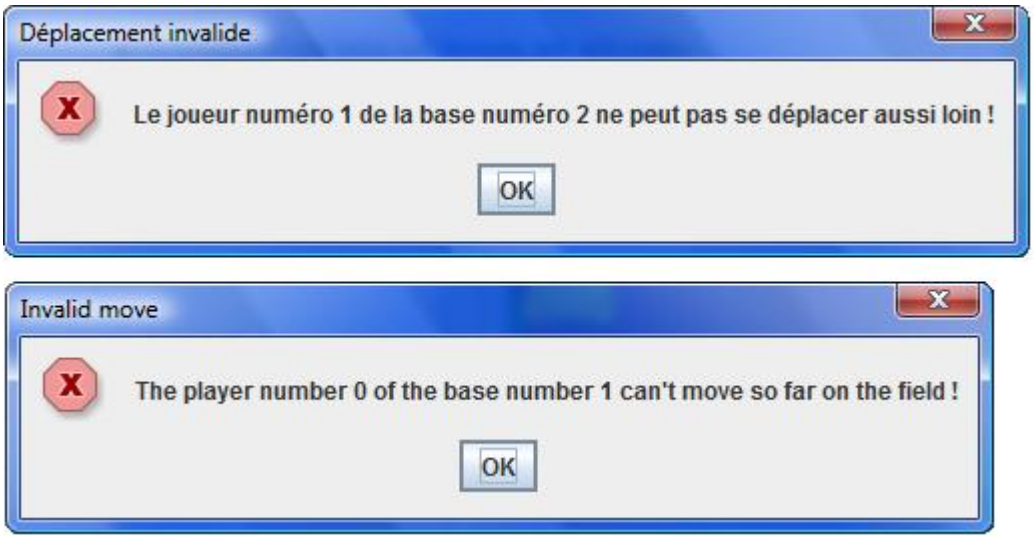


Figure 9: Aperçu des messages d'erreurs engendrés par les contrôles interactifs du monde du base-ball

Univers de Hanoï

L'univers de Hanoï implémente dans la JLM le problème des **tours de Hanoï**, imaginé par le mathématicien français Édouard Lucas, qui est un problème classique en algorithmique.

Bien qu'étant un problème classique, il est parfois difficile de visualiser directement la suite d'opération à effectuer afin de le résoudre.

C'est ici que les contrôles interactifs prennent toutes leurs importances étant donné qu'ils permettent d'effectuer manuellement l'algorithme et d'ainsi faciliter sa compréhension tout en affichant à l'écran les messages d'erreurs liés aux mouvements illicites que l'utilisateur a essayé de réaliser.

Figure 11: Aperçu des contrôles du monde de Hanoï

3.5 Amélioration de la traduction en français de la JLM

3.5.1 Présentation de po4a

L'objectif du projet po4a (PO for anything — PO pour tout) est de simplifier la traduction (et de façon plus intéressante, la maintenance des traductions) en utilisant les outils gettext dans des domaines pour lesquels ils n'étaient pas destinés, comme la documentation.

Le fonctionnement est simple, on crée un fichier .conf contenant le chemin absolu des fichiers à traduire, le type de chaque fichier, les langues dans lesquels il faut les traduire et l'emplacement où mettre chaque fichier traduit.

Po4a génère alors un fichier .po qui concentre les différentes « chaînes » de caractères à traduire.

Les fichiers traduits ne doivent jamais être modifiés à la main, po4a se charge de tout.

On peut donc se concentrer sur la traduction du ou des fichiers PO, que po4a met à jour automatiquement si et seulement si le fichier dans la langue source est modifié. On peut bien entendu forcer la mise à jour de ces fichiers.

3.5.2 Traduction effective

Terminer la traduction de la JLM a consisté à trouver les fichiers non traduits soit car ils n'étaient pas présents dans po4a.conf ou car leur traduction n'était pas réalisées dans fr.po

Cela s'est traduit par un rajout de 24 fichiers à po4a.conf et à la rédaction de 1314 lignes (d'une taille de soixante caractères environ) dans fr.po.

Cette tâche nous a permis de corriger différentes typographies réalisées dans les fichiers traduits comme dans les fichiers sources.

3.6 Divers

Intégration du langage Python

Étant donné que l'Éducation Nationale a nommé le langage python comme étant l'un des deux principaux langages de programmation recommandés pour l'apprentissage de l'algorithmique en classe de seconde. La JLM a donc développé sa capacité à gérer ce langage.

Le travail a donc consisté à traduire des entités du langages Java vers le langage Python.

Rangement de chaque exercice dans son propre répertoire

Afin d'améliorer l'organisation du code source de la JLM, il a été demandé de ranger chaque exercice dans son propre répertoire.

Demande de confirmation lorsqu'on veut quitter la JLM

Afin d'éviter de fermer accidentellement la JLM en cliquant par mégarde sur le bouton de croix de la fenêtre, une fenêtre de confirmation de la fermeture de l'application apparaît désormais.

Séparation de la chasse à la limace

Afin de simplifier l'exercice de la chasse à la limace (SlugHunting), ce dernier a été séparé en deux exercices, l'un se limitant à écrire la fonction *hunt()* en ayant accès à la méthode *isFacingTrail()* et l'autre se limitant à écrire *isFacingTrail()* en ayant la méthode *hunt()* accessible.

Intégration du lessonNavigator

Suite à un oubli, une partie du travail effectué l'an dernier par Marion LeBras n'avait pas été intégré à la JLM. Cela s'est donc traduit par récupéré son travail sur son dépôt Git et à rechercher les ajouts non intégrés et à les fusionner manuellement avec la JLM.

Uniformisation des entités

Afin de permettre un contrôle plus strict, il a été nécessaire de s'assurer que chaque entité Java disposait bien dans son code d'une paire /* BEGIN TEMPLATE */ et / * END TEMPLATE */. De plus, elle devait disposer soit d'une paire /* BEGIN SOLUTION */ et / * END SOLUTION */ soit d'une paire ou plusieures paires /* BEGIN HIDDEN */ et / * END HIDDEN */

Remontée de bugs et déboggage

La JLM étant un projet en cours de développement, il est normal que l'on découvre des bugs et autres comportements inattendus. Nous avons donc résolu différents bogues découverts au cours du temps, soit par nous même soit par d'autres personnes.

Si nous découvrions un bogue et que nous ne parvenions pas à le résoudre dans la foulée, nous ouvrions une « issue » sur Git afin de le faire connaître à l'ensemble des personnes travaillant sur la JLM dans le but d'augmenter les chances de le résoudre.

V. Discussion

5.1 Problèmes rencontrés

La plupart des difficultés et ralentissements rencontrés provenaient d'une méconnaissance de l'architecture de l'application, même si les rapports des PIDR des années précédentes nous ont fait gagné du temps.

Néanmoins, une documentation plus fournie nous aurait été d'un grand secours à certains moments du projet même si M. Quinson a toujours répondu à nos questions dans les plus brefs délais.

En outre, nous avons dû nous familiariser avec le gestionnaire de version Git.

5.2 Documentation

Comme nous avons pu le constater au cours du projet, une documentation claire et complète est cruciale pour développer de manière efficace dans une application dont nous ne sommes pas à l'origine.

Même si nous avons fait l'effort d'ajouter à notre code de la documentation Javadoc, il reste de nombreuses protions de code régissant les mécanismes internes de la JLM que nous n'avons aucunement documenté.

De plus, malgré toute notre bonne volonté, il se peut que notre contribution en terme de documentation ne soit pas aussi limpide que souhaitée.

C'est pourquoi le travail sur la documentation devra sans doute être le fil rouge des prochains groupes d'étudiants à travailler sur la JLM.

5.3 Extension des mondes des crêpes et du baseball

Ces deux mondes pourraient être modifiés afin de rajouter des exercices.

Le monde des crêpes pourrait contenir des exercices proposant d'autres algorithmes de résolution du problème du crêpier psychorigide, de manière similaire aux exercices proposés dans le monde des tris.

De même ajouter une vue contenant plus d'informations, a l'instar de la « Chronoview » du monde des tris serait une autre possibilité envisageable.

Ceci est repris dans les problèmes #80 et #79 du dépôt Git.

Pour le monde du baseball, on pourrait imaginer un dernier exercice où cette fois le nombre de joueur par base changerait en fonction de la base.

5.4 Conclusion

Ce projet interdisciplinaire a permis de développer une bonne base pour les Sciences Manuelles du Numérique tout en clarifiant la création d'un univers pour les prochains étudiants grâce à la Javadoc.

Il est important de se rappeler de l'importance du dépôt Git qui permet de centraliser les choses à faire ainsi que les bugs connus.

A. Références

[1] Guillaume Gérard & Adrien Siebert & Julien Guepin, *Rapport de PIDR 2012*, Mai 2012

[2] Marion Le bras, *Rapport de PIDR 2012*, Mai 2012