Rapport de Projet Java, CY-PUZZLE

Groupe 13

Maxime CLEMENT
Rayan SAIL
Nael AHMED
Fares BEN MABROUK
Mehdi BOULAICH

25 mai 2025

Table des matières

1 Organisation de l'équipe **2**

2 Problèmes rencontrés et solutions **3**

3 Limitations fonctionnelles du projet **4**

4 Chronologie du projet **4**

1 Organisation de l'équipe

L'organisation de notre équipe s'est articulée autour de plusieurs rôles clés afin d'assurer une répartition efficace des tâches. Chaque membre avait une responsabilité principale :

— **Coordinateur des réunions de suivi** – coordination générale, suivi du projet, planification.

— **Développeurs** – développement des fonctionnalités, tests unitaires.

— **Responsable documentation** – rédaction de la documentation technique et du rapport final.

— **Responsable qualité** – relecture du code, conformité avec les spécifications.

Répartition des rôles au sein de l'équipe :

— Fares BEN MABROUK — Coordinateur des réunions

— Maxime CLEMENT, Fares BEN MABROUK, Nael AHMED, Mehdi BOULAICH, Rayan SAIL — Développeurs

— Rayan SAIL — Responsable documentation

— Maxime CLEMENT, Rayan SAIL, Mehdi BOULAICH, Nael AHMED, Fares Ben Mabrouk — Responsables qualité

Ce projet a demandé, au-delà du simple codage, beaucoup de réflexion sur les méthodes utilisées (analyse des pièces, comparaison, élaboration d'un algorithme de résolution). Ainsi, nous avons d'abord réalisé une interface graphique basique, puis chaque membre a tenté de développer sa propre idée pour résoudre le puzzle de manière algorithmique. Chacun a travaillé individuellement sur son propre algorithme. Cependant, pour garder une cohérence et réellement travailler en équipe, nous nous sommes réunis presque tous les jours (tous les un à deux jours) pour faire un point sur l'avancement de chacun et prendre les décisions ensemble. Plus précisément, nous avons utilisé un outil de gestion des tâches (JIRA) comme illustré ci-dessous.

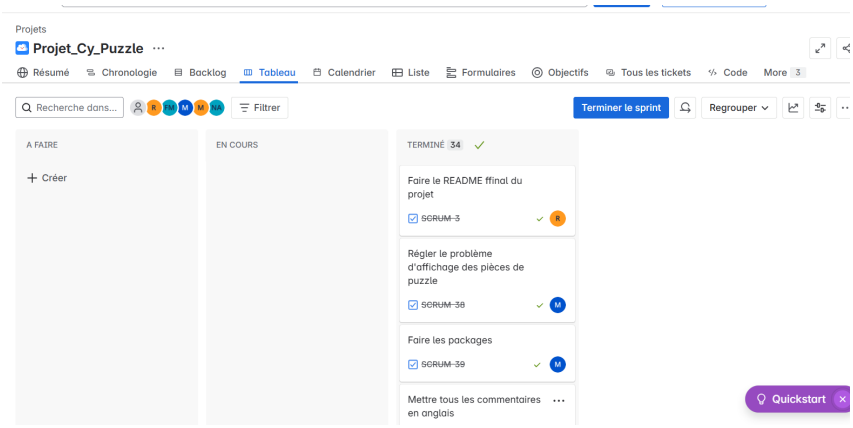

FIGURE 1 – ImageJIRA

Pour donner un exemple concret, après plusieurs réunions, nous avons unanimement décidé que l'avancement de Maxime sur l'algorithme de résolution était suffisamment

poussé. Le groupe a donc commencé à travailler ensemble sur l'affichage du puzzle résolu (assemblage correct des pièces et affichage dans l'interface graphique).

De plus, chaque semaine, nous avions une réunion avec notre encadrante de projet : Mme HAWARI.

2 Problèmes rencontrés et solutions

Lors du développement, nous avons été confrontés à plusieurs difficultés :

- **Problème technique** : Détection des vrais coins d'une pièce de puzzle.
  - *Solution* : Nous avons extrait le contour de la pièce, puis à partir de ce contour, en suivant un motif de pixels, nous avons identifié les 4 coins.
- **Problème technique** : Extraire les bonnes informations pour comparer deux bords (développement de l'algorithme de comparaison).
  - *Solution* : Nous avons extrait plusieurs informations comme la taille en pixels, les couleurs des bords, la profondeur, et une partie de la forme du creux ou du tenon.
- **Problème technique** – Difficulté à afficher les pièces visuellement. Notre méthode initiale consistait à assembler le puzzle ligne par ligne, puis à fusionner les lignes entre elles. Le puzzle affiché n'était pas bien aligné graphiquement, bien que le résultat logique soit correct. De plus, il y avait des problèmes récurrents d'alignement lors de l'assemblage visuel. L'alignement tenon-creux était l'un des plus gros défis.
  - *Solution* : Nous avons décidé de repartir complètement de zéro. L'ancienne méthode s'est avérée inadéquate, nous avons donc opté pour une stratégie pièce par pièce. Nous récupérons la largeur et la hauteur de l'image finale, puis alignons les coins en conséquence. Grâce à cette méthode, nous pouvons positionner précisément les pièces et les afficher correctement dans l'interface.
- **Problème technique** – Problème de bruit : Plus la résolution de l'image est faible (peu de pixels), plus la différence de couleur entre deux bords peut être importante. Il faut donc prendre en compte ce bruit (écart de couleur) qui dépend de la taille du puzzle. Cela rend les comparaisons plus difficiles, et peut provoquer de mauvais emboîtements.
  - *Solution* : Mise en place d'un paramètre de « tolérance », mais il doit être calibré pour CHAQUE puzzle, ce qui est compliqué à ajuster à chaque fois.
- **Problème technique** – L'un des défis majeurs rencontrés a été la résolution de puzzles avec BEAUCOUP de pièces. Le nombre élevé de pièces augmentait fortement la complexité algorithmique. Le processus d'appariement entre les pièces devenait lent, et l'usage mémoire augmentait, provoquant parfois des ralentissements voire des plantages. Le débogage et l'affichage devenaient aussi plus compliqués. Ces facteurs ont rendu difficile la montée en charge de notre solution.
  - *Solution* : Nous avons dû constamment améliorer l'algorithme de résolution tout au long du projet. Face aux nouveaux cas et aux difficultés rencontrées, des ajustements itératifs ont été nécessaires pour augmenter la précision et l'efficacité. Concrètement, nous avons utilisé un algorithme glouton en optimisant un maximum les comparaisons et les données stockées.

3 Limitations fonctionnelles du projet

Malgré nos efforts de développement, certaines limitations subsistent dans la version actuelle :

— Les pièces avec rotation ne sont pas encore prises en charge.

Nous avons passé trop de temps à essayer de résoudre correctement l'algorithme pour les pièces normales, ce qui nous a fait prendre du retard, et nous nous sommes rendu compte trop tard qu'il fallait aussi gérer les rotations. Nous avons voulu faire fonctionner l'algorithme PARFAITEMENT pour les pièces classiques (et il n'est pas encore totalement parfait), puis nous voulions passer à la gestion des rotations une fois tous les cas standards couverts.

De plus, lorsque les pièces de puzzle ont une très faible résolution (peu de pixels par bord), notre algorithme a du mal à faire les comparaisons. Parfois, plusieurs pièces semblent pouvoir s'emboîter, et notre algorithme glouton n'arrive pas à bien gérer ce cas.

4 Chronologie du projet

Au début du projet, nous avons divisé notre travail en plusieurs phases réparties sur plusieurs semaines, en suivant une planification que nous pensions adaptée pour mener à bien le projet :

1. **Phase de conception** (Semaines 1-1,5) : rédaction des spécifications, configuration de Jira, diagrammes de classes et de cas d'utilisation initiaux.
2. **Phase de développement** (Semaines 2-4) : mise en place de l'architecture, premières fonctionnalités et interface graphique, réflexion sur les algorithmes, validation des algorithmes de résolution, assemblage graphique du puzzle.
3. **Phase de finalisation** (Dernier jour) : corrections de bugs, JAVADOC, livraison du projet.

Cependant, en raison des examens, nous avons dû mettre le projet en pause, ce qui nous a quelque peu « déconnectés ». À la reprise, juste après cette période, nous avons rencontré quelques difficultés pour nous remettre dans le rythme. La charge et la difficulté des examens, ainsi que la fatigue accumulée, ont ralenti notre progression. Mais une fois relancés, l'équipe s'est montrée sérieuse et efficace. Malgré notre engagement constant tout au long du projet, il nous a malheureusement manqué un peu de temps pour livrer une version pleinement fonctionnelle.

Nous avons trouvé le projet bien plus difficile que ce que nous imaginions. Toutefois, il était très intéressant et en travaillant ensemble, nous avons surmonté de nombreuses difficultés imprévues. Nous sommes fiers de ce que nous avons accompli sur ce projet.