Technologie Objet - Projet long
Développement d'une application Unlock
Rapport itération 3

Nom des auteurs
Aupetit Lucien
Cazes Noa
Duthoit Thomas
Georget Lucas
James Christopher
Paoli Baptiste
Winterberger Ilona

Département Sciences du Numérique - Première année
2019-2020

Table des matières

Table des figures

1 Objectif général du projet

Le but est de recréer le jeu « Unlock » sans l'aspect physique de la boite et de son contenu. Le ou les joueur(s) pourra/pourront jouer sur une ou plusieurs énigmes proposées par l'interface graphique.
Ce produit est destiné à tout public qui souhaite se distraire avec un jeu ludique et qui nécessite réflexion, et à la différence du jeu traditionnel, il est possible de jouer n'importe où au jeu Unlock.

2 Les principales fonctionnalités et leur état d'avancement

Les fonctionnalités souhaitées ont été fragmentées volontairement en sous-fonctionnalités simples à réaliser ce qui a permis de mettre en avant des besoins sous-jacents.

Dans le cadre de la méthode Agile, on se doit de créer le backlog produit. Un backlog produit initial a ainsi été créé lors de la rédaction du rapport sur les fonctionnalités. Ainsi des exigences logiciel (User Stories) et les fonctionnalités essentielles (Features) ont été définies. Ce backlog initial a été complété au cours des différentes itérations, notamment par le raffinage des principales fonctionnalités.

2.1 Remise en contexte : les différents types de carte présents

Dans le jeu Unlock, les utilisateurs jouent une partie et interagissent avec différentes cartes. Il existe cinq types de cartes, chaque type ayant une particularité différente :

1. Histoire
   C'est la première carte (le dos d'une pièce), on la lit pour comprendre l'intrigue puis la retourner déclenche le timer,
2. Lieu
   Les cartes lieux indiquent plusieurs numéros et invitent les joueurs à fouiller le paquet afin de dévoiler les éléments à étudier,
3. Puzzle
   Les cartes puzzles se combinent entre elles pour dévoiler une nouvelle carte (une rouge avec une bleue),
4. Machine
   Les cartes machines sont des mini-jeux qui permettent d'obtenir une nouvelle carte,
5. Code
   Les cartes codes sont des énigmes et permettent d'avancer dans l'histoire une fois résolues.

2.2 Première étape : les User Stories

1. Plateau
   En tant que joueur, je veux pouvoir visualiser les cartes à ma disposition et celles qui restent à découvrir, afin de choisir aisément sur quel élément me pencher.
2. Barre de saisie
   En tant que joueur, je veux qu'il y ait une barre de saisie à ma disposition, afin de rentrer les numéros de cartes que j'ai envie de découvrir ou les codes .
3. Carte puzzle
   En tant que créateur, je veux que l'aventure soit composée d'éléments logiques à combiner, afin de proposer aux joueurs un défi.

   En tout que joueur, je veux être en mesure de relier deux éléments logiques sommant les numéros des cartes sélectionnées, afin d'obtenir un nouvel élément pour pouvoir poursuivre l'aventure.

4. Carte lieu
   En tant que joueur, je veux accéder rapidement aux éléments découverts dans un lieu pour y avoir accès, afin de ne pas passer trop de temps sur les phases de fouille.
5. Carte machine
   En tant que joueur, je veux que les expériences proposées soient variées (sous forme d'énigmes, de casse-tête, ... à choix multiples), afin de réaliser des mini-défis intéressants.
6. Carte Code
   En tant que joueur, je veux pouvoir utiliser les éléments présents dans l ?aventure pour trouver un code adéquat, afin de clôturer la partie.
7. Affichage en grand
   En tant que joueur, je veux que lors d'un clic sur une carte celle-ci s'affiche en grand, afin de pouvoir observer minutieusement les détails ou me concentrer pleinement dessus.
8. Chronomètre
   En tant que joueur, je veux avoir à ma disposition un chronomètre et le garder sous les yeux, afin de pouvoir contrôler mon avancée dans la partie.
   En tant que joueur, je veux avoir un chronomètre que je peux stopper et relancer, afin de pouvoir mettre sur pause le jeu et le reprendre.
9. Musique
   En tant que joueur, je veux que de la musique soit jouée, puisse être mise en pause et puisse être reprise, afin de me plonger dans l'ambiance de l'aventure.

2.3 Deuxième étape : Features

Les différentes fonctionnalités ainsi que leur état d'avancement sont les suivantes :

1. créer une interface d'accueil : l'interface d'accueil réalisée est une fenêtre contenant un bouton "Démarrer une partie" qui mène au plateau du jeu (itérations 2 et 3),
2. créer un menu pour sélectionner l'histoire souhaitée sur la page d'accueil : non réalisé car il a été décidé de mettre en place le tutoriel du jeu,
3. créer une interface correspondant au plateau de cartes (voir le paquet de cartes face cachées et de visualiser les cartes déjà retournées afin de se situer dans l'avancement de la résolution des énigmes) : interface réalisée, les cartes sont présentes sur le plateau, la possibilité d'agrandir une carte en cliquant dessus a été rajoutée (itérations 1, 2 et 3),
4. saisir des codes via l'item d'un menu : réalisé (itérations 2 et 3),
5. quitter le jeu via l'item d'un menu : réalisé (itération 1),
6. créer un chronomètre : chronomètre réalisé ainsi que son interface graphique (itérations 2 et 3),
7. avoir le chronomètre constamment sous les yeux : le chronomètre a été positionné au premier plan (itération 3),
8. mettre du son pour le chronomètre : non réalisé,
9. mettre sur pause le jeu : possibilité de stopper le chronomètre (itérations 2 et 3),
10. reprendre le jeu : possibilité de relancer le chronomètre (itérations 2 et 3),
11. quitter le jeu si le chronomètre est fermé par le joueur : réalisé (itération 3),
12. réaliser notre propre carte machine (créer une énigme) : réalisé (itération 3),
13. demander un indice : non réalisé,
14. saisir une pénalité reçue par une carte et éventuellement automatiser ce processus de sorte que l'utilisateur n'ait pas à saisir ses propres pénalités : non réalisé,
15. obtenir de l'aide sur un objet caché : non réalisé,
16. visualiser les indices reçus : réalisé par le biais de l'interface graphique (itérations 1 et 2),

17. désactiver ou activer la musique d'ambiance : réalisé via un item dans le menu de l'application (itération 3),
18. changer la langue de l'application : non réalisé,
19. créer notre propre histoire : non réalisé.

3 Le découpage de l'application en sous-systèmes

L'application a été découpée en sous-sytèmes selon l'architecture suivante :

1. Le jeu en lui-même
   - Carte.java
     Classe qui permet de définir tous les types de cartes présentés précèdement par le biais de différents constructeurs.
   - Plateau.java
     Interface qui permet de mettre en place les différentes actions possibles sur le plateau avec les cartes,
     - Jeu.java
       Une réalisation de Plateau.java qui permet de définir un jeu quelconque,
     - Tuto.java
       Une réalisation de Plateau.java qui permet de définir le tutoriel de ce jeu,
   - Musique.java
     La classe qui permet de définir les méthodes permettant de lancer et stopper une musique.
   - L'exception : CaseOccupeeException.java
     Est levée lorsqu'une carte essaye d'être posée sur une case du plateau déjà prise.
2. L'interface graphique (dépendante des classes précédentes)
   - Les classes principales
     - UnlockSwingTuto.java
       La classe qui permet la représentation du plateau de cartes du jeu avec les cartes du tutoriel,
     - FenetreAccueil.java
       La classe qui permet la représentation de la fenêtre d'accueil,
     - FenetreChrono.java
       La classe qui permet la représentation du chronomètre,
     - FenetreCarte.java
       La classe qui permet d'ouvrir une fenêtre avec la carte voulue en grand.
   - Les actions associées à des évènements
     - ActionQuitter.java
       Permet de définir l'action engendrée lors de l'évènement "appuie sur l'item Quitter du menu",
     - ActionSaisirCode.java
       Permet de définir l'action engendrée lors de l'évènement "appuie sur l'item Saisir code du menu".
     - ActionStopperMusique.java
       Permet de définir l'action engendrée lors de l'évènement "appuie sur l'item Stopper musique du menu",
     - ActionRedemarreMusique.java
       Permet de définir l'action engendrée lors de l'évènement "appuie sur l'item Redemarrer musique du menu",
     - CliquerCase.java
       Permet de définir l'action réalisée lorsque l'on clique sur une carte du plateau (soit on la retourne, soit rien ne se passe si on a pas l'autorisation de la retourner, soit on l'affiche en grand),

— ActionChrono.java
Permet de définir l'action réalisée par le chronomètre (incrémentation),
— ActionChronoStartPause.java
Permet de définir l'action réalisée lorsque l'on appuie sur les boutons start, pause ou restart du chronomètre,
— Les images de cartes, de fonds

4 Les diagramme de classe

La figure 1 représente le diagramme de classe de notre application, réalisé uniquement avec les informations utiles pour comprendre l'architecture globale. Et la figure 2 représente le diagramme de classe lié aux réalisations de l'interface ActionListener.

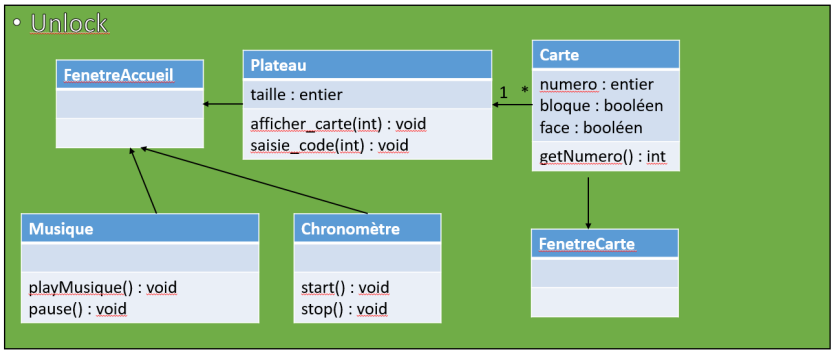

FIGURE 1 – Diagramme de classe 1

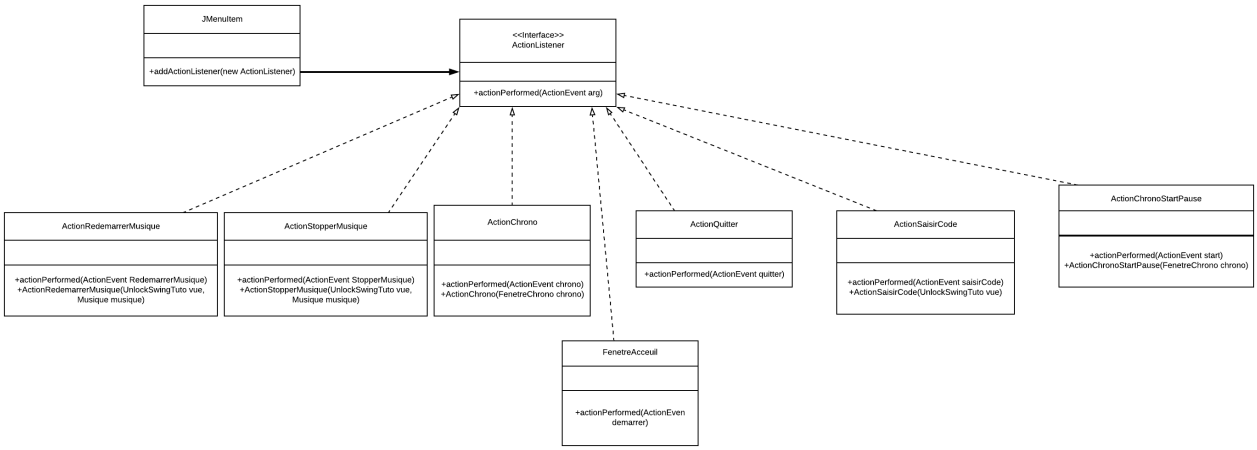

FIGURE 2 – Diagramme de classe 2

5 Les principaux choix de conception et de réalisation, les problèmes rencontrés et les solutions apportées

Le principaux choix de conception et de réalisation adoptés ont été les suivants :

1. Utilisation de la bibliothèque Swing pour l'interface graphique,

2. Représenter tous les types de cartes grâce à une seule classe Carte.java, dotée de plusieurs constructeurs,
3. Les cartes ne sont pas désignées grâce à un élément d'une enumération (par exemple $enum\ Carte\ \{Un, Deux, Vide\}$) mais par des entiers,
4. Utilisation d'un booléen *retournee* pour savoir si une carte a déjà été retournée ou non (état sur l'interface graphique) et d'un autre *decouvCarte* pour savoir si une carte a déjà été découverte ou non (état au sens du droit d'être retournée),
5. L'ensemble des cartes présentes sur le plateau est stocké dans un tableau à deux entrées,
6. Dans l'interface graphique, pour saisir le code, cela se fait grâce à une option du menu de la fenêtre, et non pas par une zone de texte présente de façon permanente sur le plateau,
7. les cartes sont affichées en petite taille mais possibilité de les afficher en grande taille en cliquant dessus,
8. Définition de gestionnaires d'évènements spécifiques (chaque gestionnaire réalise une seule action) afin d'éviter la création d'un unique gestionnaire d'évènements qui impose une forte dépendance entre vue et contrôleur, ainsi qu'une série de conditionnelles (peu logique dans une approche objet),
9. L'énigme doit être réalisée en 10 minutes mais le chronomètre ne s'arrête pas à l'issue des 10 minutes, le joueur peut alors continuer l'escape game, même en ayant dépassé le temps imparti,
10. Choix de fonds d'accueil et de plateau différents des traditionnels jeux d'escape game.

Les problèmes rencontrés et les solutions apportées sont les suivants :

1. La taille des images a posé problème lors du placement de ces dernières sur une organisation de la fenêtre de type $GridLayout$: elles ont alors été redimensionnées,
2. Le passage de la représentation des cartes sous la forme d'un tableau dont les éléments sont de type énumération ou de type entier a posé problème : ce problème venait de l'utilisation du type *int* à certains endroits et $Integer$ à d'autres,
3. Problème lors de l'implémentation de CliquerCase.java, qui dans une de ces conditions modifiait le booléen *decouvCarte* : création d'un nouveau booléen *retournee*.
4. L'affichage des messages à propos des codes faux/vrais et du message de fin a posé problème, notamment sur la façon de faire un saut à la ligne dans un JLabel : utilisation d'html.

6 Organisation de l'équipe et mise en oeuvre des méthodes agiles

Tout d'abord, nous avons discuté ensemble des différentes fonctionnalités voulues pour notre application. Ensuite, la rédaction du rapport sur les fonctionnalités a permis de nous fixer des objectifs précis.

Il a été décidé que 4 personnes se chargeraient de la programmation du jeu en lui-même, 2 personnes de la programmation de l'interface graphique et une personne des user stories, des diagrammes UML, et du compte Trello. On a aussi mené des discussions au moment de faire des choix importants quant à la réalisation du code de façon à prendre en compte l'avis de chaque membre de l'équipe. Chaque "équipe" a pris soin de partitionner le travail en tâches élémentaires, de déterminer lesquelles étaient prioritaires tout en permettant de livrer tôt de la valeur métier (en estimant l'effort que représente chaque fonctionnalité du backlog), et enfin de se les répartir.

C'est ainsi que nous avons pu *livrer tôt de la valeur métier tout en limitant le risque* (en répondant à, d'abord de petites fonctionnalités, puis en implémentant des fonctionnalités plus importantes). On a alors affiné et modifié le backlog de notre produit au cours de l'avancement de notre projet, notamment en réajustant certaines fonctionnalités : nous n'avons pas hésité à remettre en cause notre travail en changeant de direction concernant les choix de conception.

7 Conclusion

A l'issue de cette itération nous avons réussi à régler les problèmes rencontrés au cours de l'itération 2 et ainsi à créer un jeu d'escape game... arriverez-vous à vous échapper de ce bureau dans le temps imparti ?