Projet Pirates 2024

Présentation oral du vendredi 17 mai

Infos présentation
pas de sommaire pour le diapo
premiere page = noms + titre du projet + date + on va expliquer le déroulé sans sommaire juste a l'oral
ensuite on part direct sur les chapitres (et on doit avoir le plan toujours visible avec celui sur lequel on est en couleur)
pas de souligne ou gras ou couleur
pas de phrase
le plus imagé possible / schema
tout doit etre lisible
tout en francais

Sommaire :

1. Présentation du jeu

Bonjour a tous aujourd'hui je me présente … et aujourd'hui ….
Partie 1 Nous allons vous présenter le jeu que nous avons développé pour ce projet de pirates. Pour rappel, notre client est venu à nous avec l'idée de créer un jeu de pirates se rapprochant du célèbre jeu de l'oie. Pour essayer d'apporter un peu d'originalité à ce projet, nous nous sommes demandé : *"Mais que font les pirates avec leur argent ?"*. C'est ainsi que nous est venue l'idée de baser notre jeu sur l'univers des casinos et des jeux d'argent.

Les pirates, après avoir vécu des aventures à la recherche de trésors pour accumuler de l'or, se retrouvent dans des bars et des casinos pour dépenser leur butin.

C'est pour cela que, dans notre jeu, nous ne jouons pas des pirates, mais nous sommes des pirates jouant sur une table de jeu de roulette représentant notre plateau, en misant de l'argent représenté par des jetons de poker qui sont notre vie. Les vulgaires dés à 6 faces classiques sont mis de côté pour laisser place à une machine à sous nous permettant de déterminer nos valeurs de déplacement. *Donnons donc un côté plus immersifs et originale quant au respect du sujet.*

Passons maintenant aux mécanismes du jeu :

Nous retrouvons sur notre plateau 4 cases spéciales :

- Case dégâts : enlève un jeton de poker/un point de vie au joueur.
- Case rejouer : permet au joueur de relancer la machine et de rejouer immédiatement.
- Case gambling : demande au joueur de lancer la machine pour obtenir une valeur supérieure à celle demandée pour ne pas perdre de vie.
- Case reculer : qui oblige le joueur à relancer la machine à sous pour reculer de case

Une mécanique de jeu permettrait à un joueur de miser un jeton de poker à n'importe quel moment de la partie pour infliger un malus à l'adversaire. Malheureusement, nous n'avons pas encore eu le temps d'implémenter cette mécanique, mais nous espérons le faire d'ici la prochaine présentation.

Partie 2 Nous allons maintenant entrer plus en détail dans le développement du projet. Tout d'abord, nous allons parler de l'organisation de l'équipe, qui fut le principal problème du développement de ce projet. En effet, durant les premières phases de développement du projet, nous n'avions pas de plan d'attaque ni d'organisation claire. Laissant chacun travailler à son rythme et faire du travail sans réel communication, ce qui nous a amenés à nous retrouver durant la dernière ligne droite avec un code fonctionnel, mais avec une architecture qui ne satisfasse pas les demandes du client.

C'est pour cela que durant la dernière phase de développement du projet nous nous somme retrouvé dans une période de rush, *et face à ce problème que nous avions, nous avons trouvé comme* ***solution*** *une méthodologie en 2 étapes* :

1. Nous avons décidé de tous nous investir en même temps sur la refactorisation de notre code pour obtenir une architecture ECB et mettre en pause le développement de toute la partie IHM, étant donné que nous avions une semaine de plus pour sa réalisation.
2. Une fois que nous avions l'architecture souhaitée, nous avons divisé les tâches en quatre catégories : les tests, le diagramme, l'UML et une personne dédiée à mettre à jour le code pour obtenir quelque chose de complet.

Grâce à cette méthode, nous avons réussi à rattraper notre retard et avoir un un code fonctionnel complet et respectant l'architecture que nous allons vous détailler par la suite. *Je laisse maintenant mes collègues vous parler de la suite et vous donner plus de détails sur le projet…*

Partie 4

Pour la connexion IHM de notre projet, nous avons utilisé la classe ‘Boundary’ comme liaison entre l’interface utilisateur et les différents contrôleurs du jeu.

Comme nous n’avons pas encore implémenté une interface graphique dédiée, nous avons utilisé temporairement la classe “boundary” pour afficher des messages sur l’invite de commande avec la méthode “afficher message” afin de pouvoir vérifier le bon fonctionnement de notre code.

L’architecture de “Boundary” est conçue pour être facilement adaptable à notre future interface graphique. Nous allons expliquer son fonctionnement :

1. Initialisation du jeu : Lorsque le jeu doit commencer, la méthode “start” de “Boundary” est appelée ce qui va lancer la boucle principale du jeu via “ControlJeu”

2. Lancer les dés : La méthode “spin” simule un lancer de dés, dans notre jeu ca sera un lancer de machine à sous, via “ControlSlotMachine” . Les valeurs obtenues sont affichées, et le résultat est utilisé pour déterminer le déplacement des pions.

3. Déplacement des pions : La méthode “deplacePion” utilise les valeurs obtenues précédemment dans “spin” et mets à jour les informations sur les pions de “PionRepository” via “ControlDeplacePion”, on va ensuite appeler la méthode “afficherEtatJeu” pour mettre à jour la position des pions dans notre jeu.

4. Affichage de l’État du jeu : “afficherEtatJeu” récupère les positions des pions via “PionRepository” mis à jour par “deplacePion” et affiche leur position dans l’invite de commande.

5. Notifications: “Boundary” implémente l’interface “INotificationService” ce qui lui permet de communiquer avec les controleurs A1

Notre Classe Boundary assure donc une interaction fluide entre l’utilisateur et le jeu, même sans interface graphique complète. Elle est prête à l’intégration d’une interface graphique, où les messages sur l’invite de commande seront remplacés par le contrôle d’éléments java Swing.

Partie 5

Pour la partie test de notre projet, principalement nous avons voulu assurer les formats et les effets de nos fonctions qui effectuent la connexion entre notre base de code et l'interface graphique même si elle n'est pas complète pour le moment.

Pour ça on a créé un système qui est composé de 3 parties :

Class Tester qui gardent toutes les méthodes de chaque contrôleur dans un hashmap et les exécutent, pour évaluer les résultats des parties de code qui gèrent les changements.

Class TestRes afin de garder les ressources nécessaires pour faciliter l'initialisation des différentes parties d'application

Class TestAll, pour exécuter les tests en totalité il ramasse les testeurs dans une List et les exécute pour un affichage qui donne plus de détails sur les étapes des méthodes.

Comme la structure a changé plusieurs fois pendant le projet il y avait des fois où nous avons dû changer la structure des tests pour éviter les problèmes au niveau d'accès aux services ou contrôleurs.

Pendant ces changements, ajout des attributs et des Méthodes pour les accéder et modifier était nécessaire ce qui nous a donné parfois la possibilité de tester les autres aspects de programme. Ce qui nous laisse dans une situation où on doit encore ajouter des cas de tests.

Finalement on a refactorisé les tests en éliminant les cases inutiles et fusionnant les cases qui peuvent renforcer le résultat des certains tests.