

PROJET : SCRABBLE

Description

Dans ce projet nous devons réaliser un jeu de SCRABBLE avec les spécificités suivantes :

l'ordinateur ne joue pas mais contrôle le jeu;

il comptabilise les points des joueurs,

pas de base de données,

sauvegarde de la partie (pas fait),

respect des règles du jeu,

avec interface graphique.

Les joueurs jouent à tour de rôle. Lorsque c'est son tour(c'est à dire son panneau d'affichage est au vert),le programme initialise le rack à 7 lettres piochées dans le sac au hasard(random), puis il doit poser un mot (valide) sur le plateau en utilisant une ou plusieurs de ses lettres. Une fois le mot posé, les points de ce mot sont ajoutés à son score , et la main passe au joueur suivant.

Au début de la partie, le mot placé par le premier joueur doit obligatoirement passer par la case centrale, marquée d'une étoile. Cette case faisant office de case "mot compte double", la valeur du mot posé est multipliée par deux.

Ensuite, chaque mot placé par un joueur doit obligatoirement se raccorder sur une ou plusieurs lettres déjà posées sur la grille. Si un mot placé par un joueur forme d'autres mots, ceux-ci doivent également être des mots valides.

Présentation des membres

Amadou DJIGO(Chef de projet)

Alpha Ibrahima BARRY

Eric MAROVELO

Hammady CHERIF

Décomposition du PROJET SCRABBLE

Work Package n°	Intitulé	Leader	Participants	Début	Fin
WP 1	Project Management	DJIGO	Tout le groupe	3/10	22/11
WP 2	Modélisation	BARRY	Tout le groupe	11/10	29/10
WP 3	View	MAROVELO	Tout le groupe	15/10	5/11
WP 4	Controller	CHERIF	Tout le groupe	1/11	12/11
WP 5	Test	DJIGO	Tout le groupe	11/10	20/11

Description des WP

Work package 1: PROJECT MANAGEMENT

Objectives:

L'objectif de cet workpackage est:

-D'abord, nous devons nous rappeler comment jouer au Scrabble -> lire la documentation.

-Quels sont les objets du jeu de Scrabble? Comment allons-nous les traduire en classes Java?

-Comment les différentes classes interagiront les unes avec les autres? Quelles sont leurs méthodes ?

-Comment va être le design graphique du jeu.

Tâches :

- rédaction cahier des charges

- documentation

- présentation finale

Work package 2: MODEL

Leader : BARRY

Ce WP a pour but de faire une première modélisation du jeu en commençant par le trousseau où se trouvent tous les lettres, le plateau du jeu, le dictionnaire, le langage utilisé pour le jeu.

Le modèle d'architecture MVC impose la séparation entre les données, la présentation et les traitements, ce qui donne trois parties fondamentales dans l'application finale: le modèle, la vue et le contrôleur.

Dans ce workpackage nous allons concevoir le model du jeu :

traitements des données, interactions avec la base de données, etc. Ce model va décrire ou contenir les données manipulées par l'application. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient. Le modèle offre des méthodes pour mettre à jour ces données (insertion , suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ses données. Les résultats renvoyés par le modèle sont dénués de toute présentation. Dans le cas de données importantes, le modèle peut autoriser plusieurs vues partielles des données.

Tâches :

- **NewGameEvent** : crée un nouveau jeu.
- **BadWordEvent** : crée l'événement «you can't put the word here »

- **BoardEvent** : lorsque vous cliquez sur une case du plateau de jeu on a une fenêtre « AddWord » qui demande à l'utilisateur de proposer un mot.
- **PlayerEvent** : événement qui attribue le tour de chaque joueur (la main passe).
- **ScoreEvent** : événement pour le score de chaque joueur.
- **EndGameEvent** : événement pour la fin du jeu.
- **Letter**: classe qui s'occupe des lettres et mots « compte double » et « compte triple ».
- **Word** : classe qui s'occupe des lettres simples.
- **Player** : classe qui représente le joueur avec son nom, score et banc de lettres.
- **Board**: classe qui représente la grille.
- **Rack** : classe qui représente le rack.
- **Bag** : classe qui représente le bag.
- **Dictionnaire** : classe qui représente le dictionnaire.
- **Lang** : pour changer de langues : EN pour anglais ou FR pour français.
- **LanguageEnglish** : pour chaque lettre de l'alphabet sa valeur correspondante (point) et le nombre d'occurrences.

Work package 3: VIEW

Leader : MAROVELO

Description:

La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, etc). Ces différents événements sont envoyés au contrôleur puis le contrôleur va demandé au modèle si son état peut être changer, si oui le modele modifie son etat puis met à jour la

vue. Parfois la vue peut directement envoyé un message au modele sans passer par le contrôleur c'est à dire les événements qui ne nécessitent pas la modification du modele (ici par exemple pour verifier si un mot est belle est bien dans le dico). La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle. Plusieurs vues, partielles ou non, peuvent afficher des informations d'un même modèle. Ici on va créer l'interface graphique avec SWING, qui est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Foundation Classes (JFC), inclus dans J2SE. Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures.

Swing offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent, au prix de performances moindres qu'en utilisant Abstract Windows Toolkit (AWT). Il utilise le principe Modèle-Vue-Contrôleur (MVC, les composants Swing jouent en fait le rôle du contrôleur au sens du MVC) et dispose de plusieurs choix d'apparence (de vue) pour chacun des composants standards.

Tâches :

- **Grille** : affiche le plateau , si un mot est valide alors le modele envoie un message à la grille de se mettre à jour. La grille ici est un conteneur(JFrame) sur lequel on a mis des boutons (JGridButton).
- **JgridButton**: une class qui « extends » de la class JButton .
- **RackPanel**: affiche le rack du joueur sur l'interface graphique quand c'est à son tour.
- **ScorePanel** : affiche le score du joueur.
- **OptionsMenu** : pour avoir des menus de l'interface graphique.
- **LookFrame** : correspond à Menu Commands -> LookUp.
Vérifie si le mot proposé existe dans le dictionnaire.
- **HelpFrame** : une fenetre pour afficher « l'Aide ».
- **DistFrame** : fenetre qui affiche les différentes distributions de chaque lettre.

Ex : il reste 5 A dans le bag.

- **DebutFrame** : une fenetre qui s'ouvre et qui demande aux utilisateurs de mettre leur nom au debut du jeu.
- **FinFrame** : fenetre pour signaler la fin du jeu.
- **GameOverFrame** : fin du jeu avec la possibilité d'avoir des lettres.
- **NewGameListener** : écouteur sur l'événement NEWGAME.
- **BadWordListener** : écouteur pour les mots non valides i.e. les mots qui ne sont pas dans le dictionnaire ou bien les mots qui ne sont pas dans le rack.
- **BoardListener** : écouteur sur grille .
- **PlayerListener** : écouteur sur les joueurs (qui doit jouer ...)
- **ScoreListener** : écouteur sur le score de chaque joueur.
- **EndGameListener** : Ecouteur sur la fin du jeu.

Work package 4: CRONTROLER

Leader : CHERIF

Le controler prend en charge la gestion des événements de synchronisation à savoir la mise à jour de la vue ou du model après chaque traitement et aussi leur synchronisation quand cela s'avère nécessaire. En effet, le contrôleur n'effectue aucun traitement, ni ne modifie une donnée. Il analyse la requête du joueur et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.

Dans le cas de l'ajout d'un mot, le Controller demande au modèle de tester la validité du mot conformément aux lettres disponibles dans le rack et aussi son emplacement (valide ou pas); à son tour, le model informe la vue du changement (si intervenue) afin d'en tenir compte. Et le model s'occupe de réinitialiser le rack.

Tâches :

- **Controler** : à chaque fois il appelle des méthodes du modèle pour des évènements provoqués par l'utilisateur. C'est l'intermédiaire entre le MODEL et la VIEW.

- **ControlerInterface** : Classe qui implémente les prototypes des différentes méthodes utilisées par la class Controler.

Work package 5: Tests

Leader : DJIGO

Le but de faire des tests est de s'assurer que le jeu fonctionne comme prévu en essayant de minimiser les bugs. Nous commencerons par les tests unitaires de manière parallèle au codage pour vérifier que les fonctions s'exécutent correctement pour ensuite terminer par tester le jeu dans sa globalité.

Tâches :

tests unitaires

ils permettent de vérifier chaque module séparément.

tests fonctionnels

ce sont les test en rapport avec l'affichage, la fiabilité, ...

tests beta

ils permettent de vérifier le jeu dans sa globalité par rapport aux attentes des utilisateurs.

tests de conformité

ils permettent de vérifier la conformité du logiciel par rapport à ses spécifications et sa conception.

Diagramme de GANTT avant

Developpement Scrabble.mvdx*
















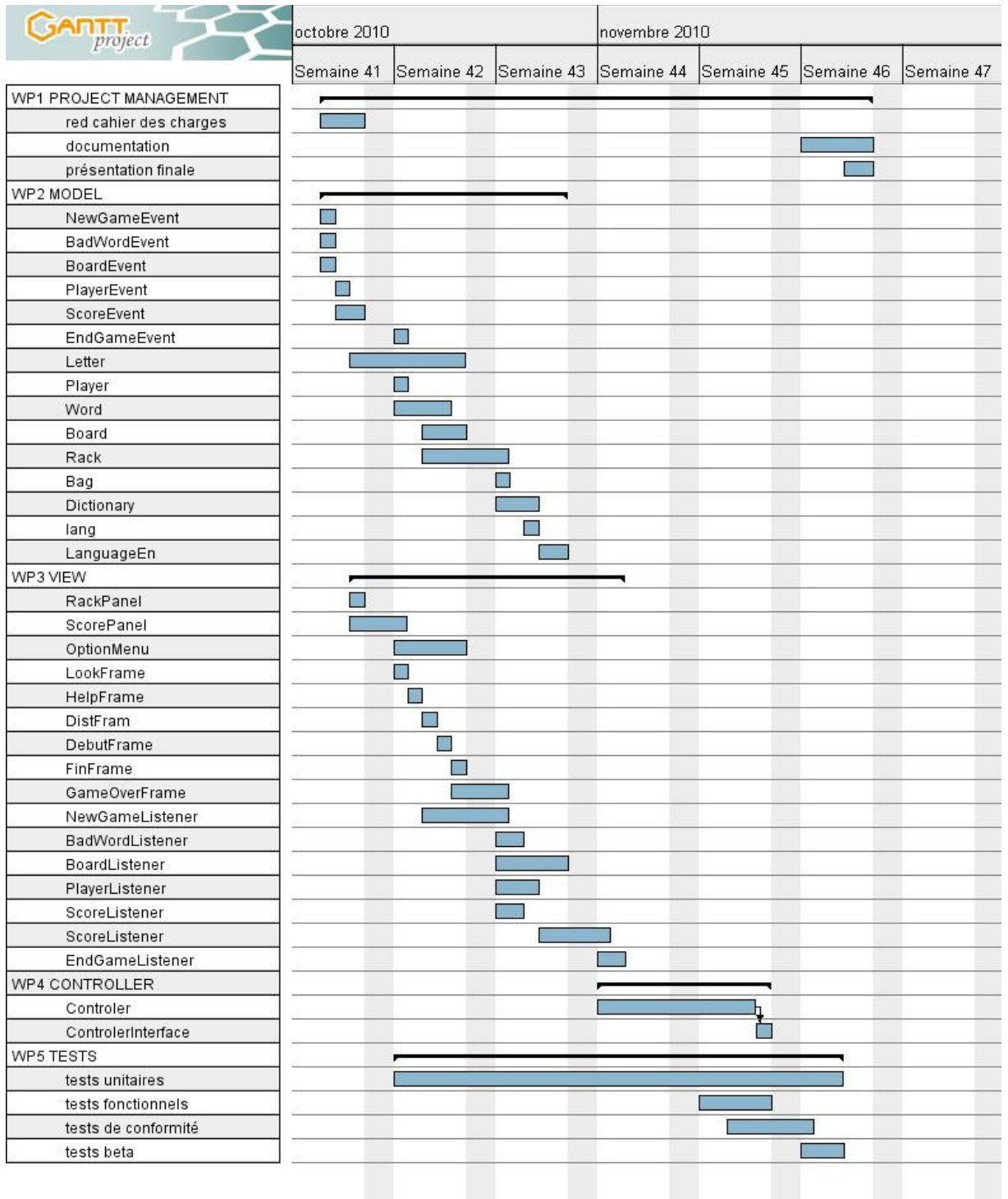
						11 oct. 10	18 oct. 10	25 oct. 10	1 nov. 10	8 nov. 10	15 nov. 10	22 nov. 10
						M M J V S D	L M M J V S D	L M M J V S D	L M M J V S D	L M M J V S D	L M M J V S D	L M M J
1		Nom de tâche	Durée	Début	Fin							
1		<u>Developpeme...</u>	29 jours	13/10/2010	22/11/2010							
2		<u>Project Man...</u>	29 jours	13/10/2010	22/11/2010							
3		Task1.1	3 jours	13/10/2010	15/10/2010							
4		Task1.2	6 jours	15/11/2010	22/11/2010							
5		Task1.3	3 jours	17/11/2010	19/11/2010							
6		<u>Model</u>	13 jours	13/10/2010	29/10/2010							
7		Task2.1	6 jours	13/10/2010	20/10/2010							
8		Task2.2	5 jours	20/10/2010	26/10/2010							
9		Task2.3	5 jours	25/10/2010	29/10/2010							
10		<u>Vue</u>	16 jours	15/10/2010	05/11/2010							
11		Task3.1	7 jours	15/10/2010	25/10/2010							
12		Task3.2	6 jours	29/10/2010	05/11/2010							
13		<u>Controleur</u>	10 jours	01/11/2010	12/11/2010							
14		Task4.1	7 jours	01/11/2010	09/11/2010							
15		Task4.2	2 jours	11/11/2010	12/11/2010							
16		<u>Test</u>	28 jours	13/10/2010	19/11/2010							
17		Task5.1	4 jours	13/10/2010	18/10/2010							
18		Task5.2	4 jours	25/10/2010	28/10/2010							
19		Task5.3	5 jours	10/11/2010	16/11/2010							
20		Task5.4	5 jours	15/11/2010	19/11/2010							

Diagramme de GANTT après



Implication avant

	WP1	WP2	WP3	WP4	WP5	Total en h
DJIGO	20	5	5	5	20	55
MAROV ELO	8	15	20	7	4	54
BARRY	4	25	4	8	8	49
CHE RIF	5	4	10	20	8	47
TOTAL en h	37	48	44	50	37	

Implication après

	WP1	WP2	WP3	WP4	WP5	Total en h
DJIGO	22	4	7	9	22	64
MAROV ELO	12	15	22	7	4	60
BARRY	4	25	4	8	8	49
CHE RIF	3	9	14	20	3	49
TOTAL en h	41	53	47	44	40	

Dépendances

Work Package	Dépendances
WP 1 : Project Management	-
WP 2 : Modèle	WP1
WP 3 : View	WP1, WP2
WP 4 : Controler	WP1, WP2,
WP 5 : Test	WP1, WP2, WP3, WP5

--	--

Problèmes rencontrés -Solutions apportées :

Ajout d'un mot :

Un des premiers problèmes que nous avons rencontrés était le fait d'ajouter un mot au niveau de l'interface graphique.

Quand l'utilisateur ou le joueur proposait un mot, celui-ci n'apparaissait pas sur l'interface graphique même s'il était correct.

Par contre lorsque le joueur proposait un mot qui n'était pas valide on avait bel et bien le message d'erreur.

Solution apportée :

La solution de ce problème s'est fait en changeant de méthode d'implémentation. Syntaxiquement le code était correct. Il a fallu reconcevoir les classes en rapport avec l'interface et AjoutMot (int x, int y, Plateau Swing plateau).

Dans le package Vue ; Swing nous avons la classe AjoutMot qui va gérer l'ajout des mots. Cette classe est un JFrame.

Le listener (toutes les cases de la grille) écoute un événement (click droit sur un bouton du plateau de jeu) déclenché par un utilisateur. Ceci entraîne l'appel de la méthode PoserMot (.....) du contrôleur qui va demander si son état peut être modifié. La méthode PoserMot va aussi faire appel à la méthode AddWord de la classe Board qui se trouve dans le package MODEL. Si son état est validé alors le MODEL va mettre à jour la VUE.

Poser une seule lettre au début du jeu:

En faisant des tests de conformité et fonctionnels nous nous sommes rendu compte qu'un mot composé d'une lettre unique était accepté au début du jeu alors que dans les règles du jeu il est précisé que les lettres uniques ne sont acceptées que lorsqu'il n'y a plus de pions dans le bag.

Solution apportée :

Ce problème a été corrigé en modifiant le code de la méthode AddWord

Public boolean AddWord (Word word, Rack rack, Bag bag, Dictionary dico)

si la longueur du mot proposé est > 2 , se trouve dans le dictionnaire, ne dépasse pas les contours du plateau de jeu et chaque lettre du mot est dans le rack, alors cette méthode retourne VRAI et FAUX sinon

Placement :

Dans les règles du jeu de scrabble un nouveau mot doit forcément être relié aux restes des lettres posées sur le plateau de jeu. Par contre l'ordinateur validait les mots posés séparément sans aucune liaison avec le reste du plateau : problème non résolu.

Récupération du dictionnaire :

C'est un des problèmes que nous n'avions pas pensés en choisissant ce sujet. Nous étions tous convaincu qu'il n'y a pas de dictionnaire à fournir. Nous nous sommes rendu compte que ce jeu était beaucoup plus difficile à programmer que nous pensions.

Ceci a vraiment permis de comprendre le dur la réalité de la programmation dans la vraie vie (monde professionnel).

La manière dont nous voyons les choses et leur réalisation sont souvent différents pour ne pas dire carrément opposés

Résolution de ce problème :

Téléchargement dans un forum internet. Nous imaginons très bien que faire du génie logiciel face aux attentes des clients constitue un vrai challenge.

Pour qu'un mot soit validé, il faut soit dans le dictionnaire c'est-à-dire un fichier dans le quel se trouvent 151477 mots.

Par contre les verbes conjugués ne sont pas dans notre dictionnaire.

Google code :

Durant la réalisation de ce projet, une seule personne a réussi à utiliser Google code comme il se devait. Nous pensons que nous tous compris pourquoi on devait utiliser Google code. Par contre il n'y avait que deux membre qui possédaient une connexion internet chez soi et donc qui la possibilité de commiter.

Nous n'avons pas réussi à refaire l'installation de Plugin Subclipse et pourtant nous avons réussi à le faire sur le premier ordinateur.

Ainsi tous les codes ont été commiter à partir de l'ordinateur d'Eric Marovelo. Sur Google code il est l'auteur de tous les commits et donc de tout le travail. Pour changer l'auteur il fallait le faire avec Eclipse lors du commit car Google Code attribut automatiquement.

Chaque semaine nous nous réunissons pour donner nos modifications à Eric Marovelo qui était le seul à pouvoir commiter.

Avantage de ce problème :

Cette contrainte nous permis de se voir manière régulière du début jusqu'à la fin ce projet et d'effectuer un travail d'équipe.

Réflexion :

Certaines contraintes peuvent être très bénéfiques lors que vous travailler en groupe. (Tout le monde a besoin de tout le monde).

Bilan :

Pout être à la hauteur de nos attentes il a fallu réfléchir sur l'architecture du model MVC qui était nouveaux pour certain d'entre nous. Au moment de la rédaction de ce rapport et même après la soutenance l'application sera toujours la phase de développement et certaine amélioration seront apportée afin de parfaire le jeu dans no cadre d'utilisation personnelle et ou de projet réaliser à faire figurer dans nos CV. Ce projet est d'un intérêt très significatif pour nous c'est une expérience très important qui vient recadrer compléter ou confirmer l'intérêt de cette matière l'Analyse et Gestion de Projet aux seins de cursus universitaire .n ou savon une vision plus précise des gestions de projet et dans le monde de travail aux sens large du terme avec ses extrémités.

Au niveau fonctionnelle nous somme convaincue à travers le projet l'utilité absolue de formaliser un projet pour assurer sa viabilité et de son aboutissement réussi.

Les réunions et autres rencontres nous permis d'améliorer notre sens de contact .ce projet nous également au déroulement d'un projet intégrer a une équipe quatre personne dans ce contexte, savoir repartir les tache communiquer sur son travail gérer le temps et les aléas de la vie, être à l'écoute des autres membres de l'équipe...