1. Table des matières

Table des figures

GLOSSAIRE

Application
D'après dicofr.com, "les applications sont les outils qui nous permettent de tout faire sur un ordinateur. Les traitements de texte, les tableurs, les navigateurs Web sont des applications".

Discord
Discord est un logiciel gratuit de communication qui permet d'effectuer des discussions écrites et vocales à plusieurs. Skype est un logiciel similaire à Discord mais avec une cible d'utilisateurs différente.

Gestionnaire de version
D'après syloe.com, "c'est un outil (logiciel) permettant d'enregistrer, de suivre et de gérer plusieurs versions d'un fichier ou d'un code source".

Git
Git est un logiciel de gestion de version gratuit et décentralisé.

GitHub
Github est une entreprise qui développe notamment la plateforme GitHub permettant la gestion de version avec plusieurs collaborateurs, en ligne.

Google Drive
D'après Wikipédia, "Google Drive est un service de stockage et de partage de fichiers dans le cloud".

IDE
Integrated Development Environment (environnement de développement). D'après mobizel.com, l'IDE "est un logiciel qui rassemble des outils permettant de développer d'autres logiciels tels que des applications".

Eclipse
Eclipse est un IDE.

Java
Java est un langage de programmation.

2. CAHIER DES CHARGES

1. LE PRODUIT

1. Objectifs du produit

La conception de ce produit a plusieurs objectifs.

Premièrement, ce produit a un objectif pédagogique et d'apprentissage. En effet, étant un projet tutoré étudiant, il a pour but d'avoir un apport en connaissances tant au niveau technique (développement, création, rédaction…) qu'au niveau gestion de projet (distribution des rôles, organisation, méthode agile, planning…).

Deuxièmement, ce produit a aussi pour objectif d'être accessible à qui le souhaite. Il a pour but de servir de divertissement et de donner la possibilité à quiconque de jouer au Scrabble sur ordinateur. De plus, il a pour objectif de donner une vision différente du Scrabble "classique" en le dématérialisant sur une application et en proposant un environnement graphique / un univers différent, plus proche du jeu vidéo et moins traditionnel.

2. Fonctionnalités du produit

La fonction principale de ce produit est de divertir. En effet, l'objectif est de donner la possibilité à l'utilisateur de faire plusieurs parties du jeu de Scrabble gratuitement et facilement sans avoir besoin de posséder le jeu physique.

3. Utilisateurs du produit

Nous souhaitons rajeunir l'image du jeu de Scrabble à travers cette application grâce à un design plus proche des jeux vidéo modernes avec une touche rétrogaming qui lui donnerait un charme.
Ainsi, si tout utilisateur peut utiliser notre produit, nous visons une cible plus jeune, entre 15 et 30 ans.

Nous souhaitons, cependant, que l'application soit facilement compréhensible et accessible à tout le monde.

4. Contraintes du produit

Ce projet a une date limite : le 31 mai 2018. Il doit donc être effectué en 4 mois.

2. LES BESOINS

1. Besoins-exigences fonctionnels

1. Besoins communs

Plusieurs besoins communs sont à spécifier :

- Cette application doit suivre les règles du jeu de Scrabble traditionnel. Ces règles sont disponibles en annexe du document.
- Le langage de cette application doit être l'anglais pour améliorer les connaissances en anglais des utilisateurs et des étudiants de ce projet et, deuxièmement, élargir la cible de l'application.

2. Description des besoins exigence

Plusieurs besoins exigence sont à spécifier :

- Lorsque l'application s'ouvre, un écran d'accueil doit être visible. C'est l'écran d'entrée du jeu. Un bouton "Start" doit être disponible pour permettre à l'utilisateur, en cliquant dessus, de commencer une partie. Le nom du jeu ainsi que le logo doivent être présents sur cet écran.

- L'écran de jeu doit disposer de plusieurs éléments :
    - Un plateau de jeu de Scrabble
    - Un espace jeu du joueur (les lettres qu'il a à sa disposition) : le même pour le joueur 1 et le joueur 2
    - Une pioche
    - Une zone où est inscrit le score respectif des deux joueurs
    - Un bouton "Menu" qui permet à l'utilisateur de recommencer une partie ou de quitter l'application

2. Besoins-exigences non-fonctionnels

1. Exigences en termes de qualité

- Utilisabilité

Le produit doit être compréhensible tout de suite par l'utilisateur et, surtout, intuitif. L'application doit être simple d'utilisation.
Cependant, pour une meilleure utilisation de celle-ci et pour des doutes possibles de l'utilisateur, les règles du jeu ainsi que des précisions sur le fonctionnement de l'application seront fournies dans un document de texte avec l'application. Ce document est disponible en annexe

- Fiabilité

L'application doit répondre rapidement aux demandes de l'utilisateur et doit avoir un temps de chargement minimal.

- Performance

L'utilisation de la mémoire par l'application doit être minimale et fonctionner sur un maximum de machines.

L'application ne pourra être ouverte qu'une seule fois sur une même machine et non plusieurs fois en simultané.

L'application ne supporte pas la présence que d'un seul utilisateur d'un point de vue technique. Cependant, celle-ci sera construite de manière à ce que, devant la machine, les deux joueurs jouent à tour de rôle.

- Supportabilité

Le jeu sera disponible sur une clé USB et pourra être transféré sur machine de cette façon à l'aide d'un dossier compressé.
Cependant, l'installation du langage java sera nécessaire sur la machine pour que l'application puisse fonctionner.

- Documentation et aide en ligne

En plus de règles du jeu et du document qui donne des précisions sur l'utilisation de l'application, nous donnons accès à un tutoriel pour l'installation du langage java sur la machine.

2. Autres exigences

- Contraintes de conception

Cette application sera développée entièrement à l'aide du langage java.
Les maquettes design seront créées à l'aide de la suite Adobe et, plus particulièrement, à l'aide du logiciel Illustrator.

- Les composants non développés

Certaines images / certains icônes seront récupérés dans des banques d'images gratuites et libres de droit.
Mise à part cela, tout sera développé par les membres du projet.

- Les interfaces utilisateurs

→ Écran d'accueil

L'écran d'accueil est celui qui s'affiche lors de l'ouverture de l'application. Il permet de commencer une partie de jeu grâce au bouton "Start".
Cet écran possède le nom du jeu et un environnement design en accord avec celui du jeu : jeu vidéo rétro gaming avec un thème "Espace et Aliens".

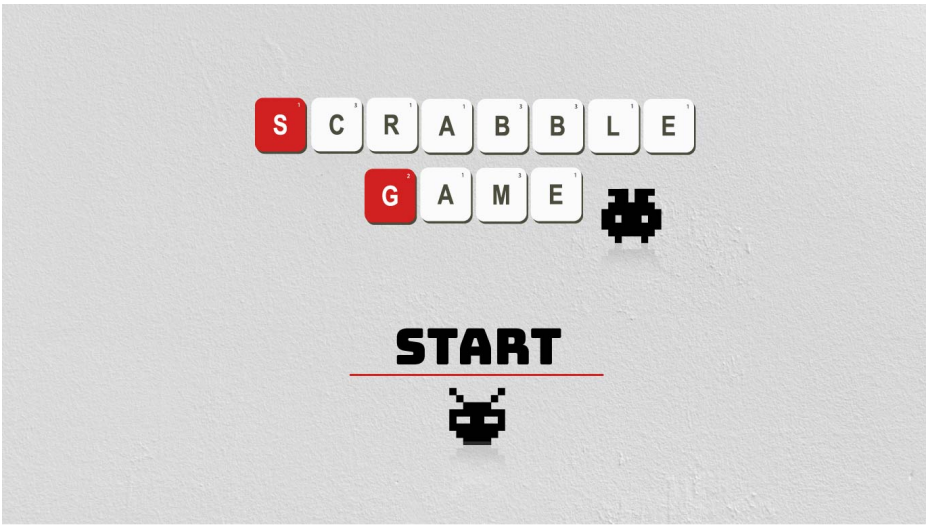

Figure 1 - Interface Ecran d'Accueil

→ Ecran de jeu

L'écran de jeu possède :

- Un plateau
- Un espace "jeu" qui est composé de la main du joueur : joueur 1 ou joueur 2 suivant le tour
- Une pioche
- Une zone score où est affiché le score des deux joueurs au fur et à mesure des parties : le joueur 1 est nommé "Human" et le joueur 2 est nommé "Alien"
- Un bouton Menu

Lorsque le joueur est en rouge, cela signifie que c'est à lui de jouer.

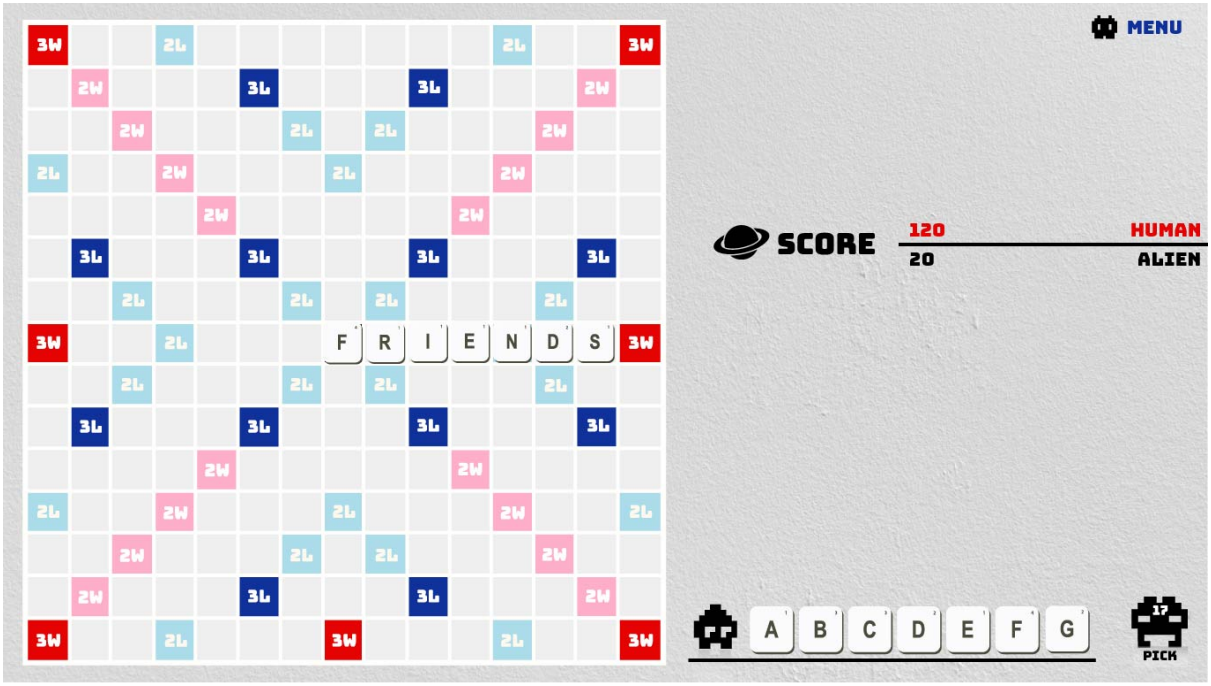

Figure 2 - Interface Ecran de jeu

Lorsque l'utilisateur clique sur le bouton menu, une fenêtre pop-up s'affiche et offre deux options :

- Recommencer une partie
- Quitter l'application (ferme l'application)

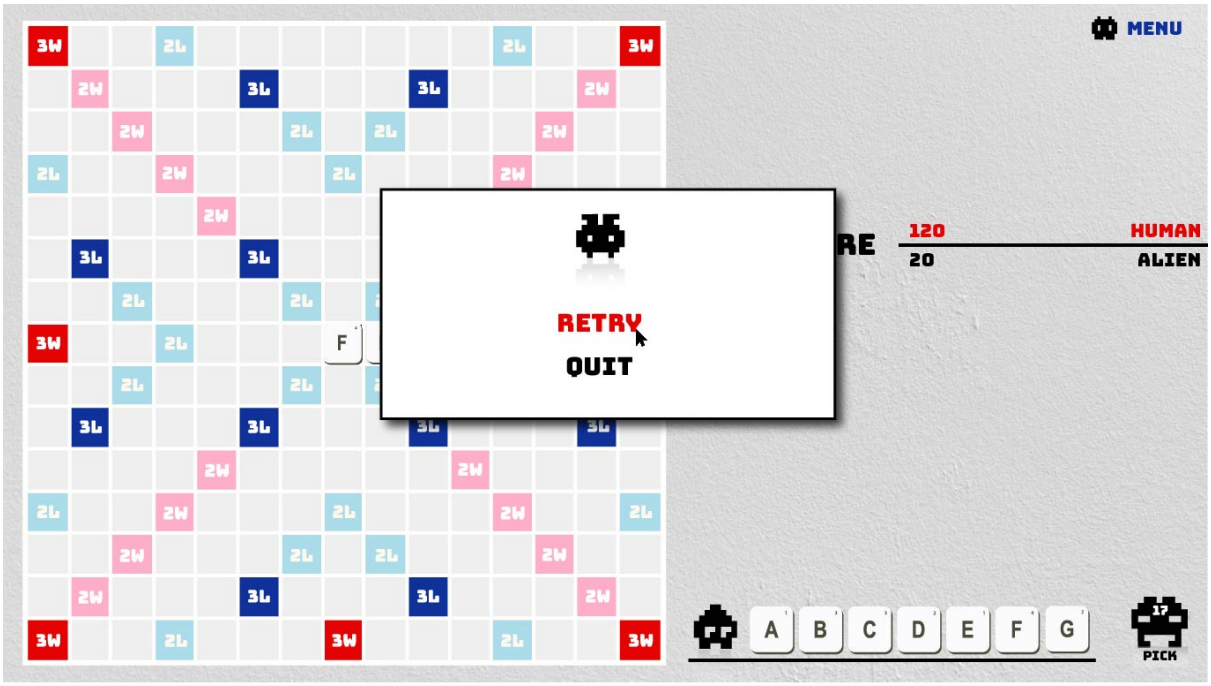

Figure 3 - Interface Fenêtre Menu

Pour jouer un mot, l'utilisateur doit effectuer les étapes dans l'ordre suivant :

- Sélectionner les lettres dans l'ordre où il souhaite les poser
- Sélectionner une case de départ pour son mot
- Sélectionner une direction (en bas ou à droite) à l'aide de flèches qui s'affichent une fois la case de départ sélectionnée

Si la case de départ n'est pas valide, un message "Invalid square" s'affiche et le joueur doit cliquer sur le bouton "OK" de cette fenêtre pour sélectionner une autre case.

Si la direction n'est pas valide, un message "Invalid way" s'affiche et le joueur doit cliquer sur le bouton "OK" de cette fenêtre pour sélectionner une autre direction ou une autre case de départ ou désélectionner les lettres choisies et en utiliser d'autres.

Si le mot n'est pas valide, un message "Invalid word" s'affiche et le joueur doit cliquer sur le bouton "OK" de cette fenêtre pour pouvoir retenter. Il doit tout recommencer, c'est à dire re-sélectionner des lettres dans son jeu etc. car tout a été désélectionné, réinitialisé.

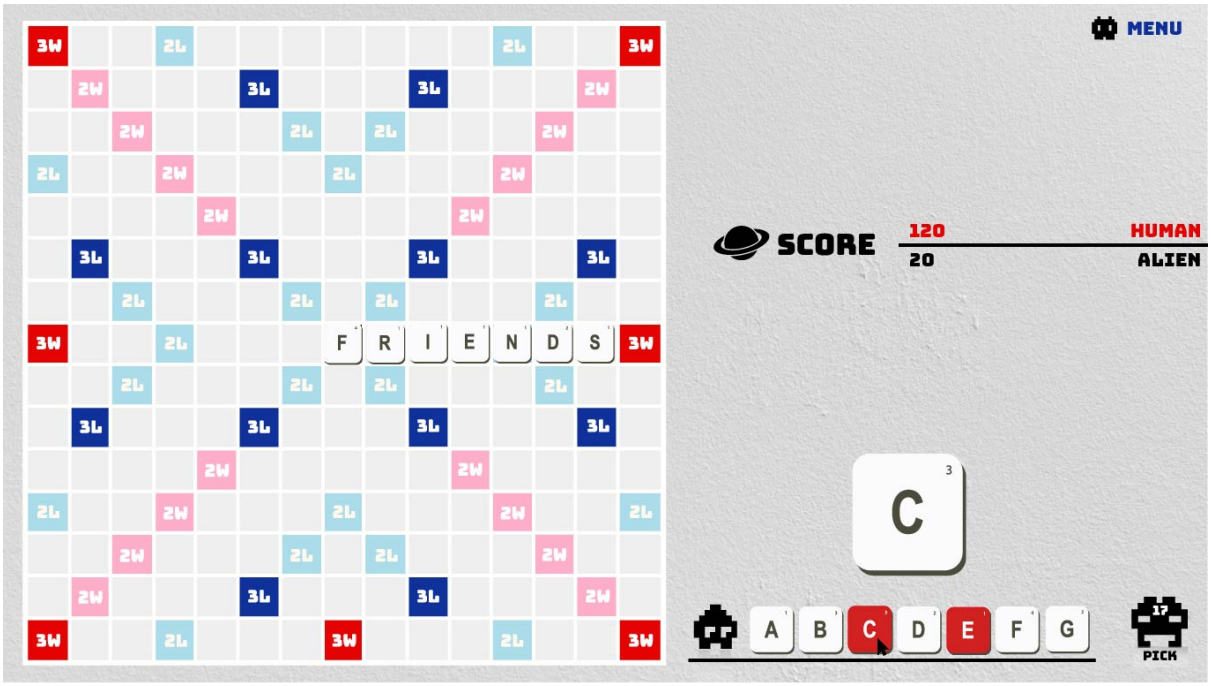

Figure 4 - Interface Sélection lettres jeu du joueur

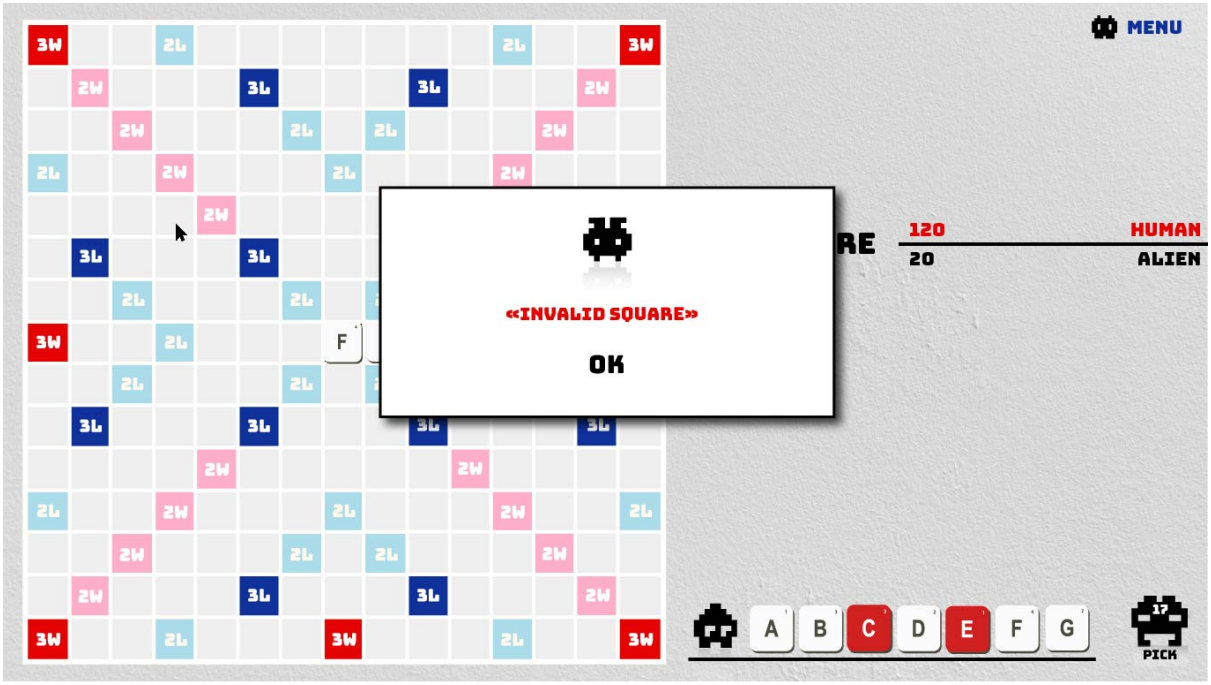

Figure 5 - Interface Message d'erreur n°1

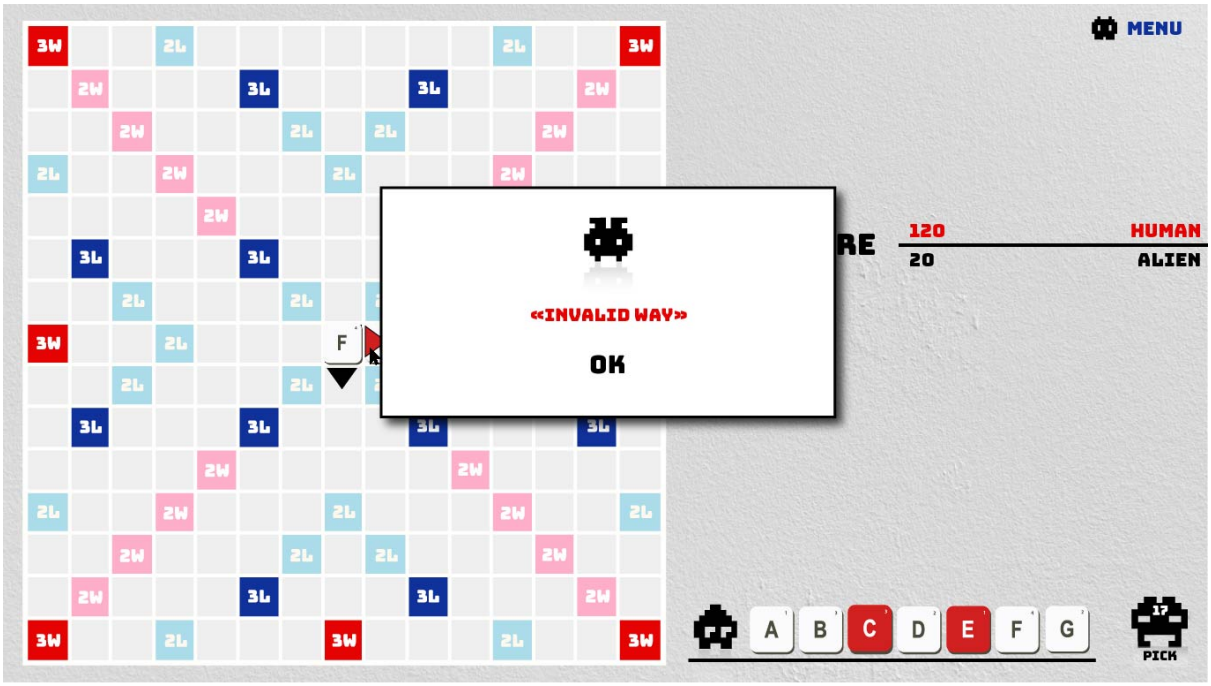

Figure 6 - Interface Message d'erreur n°2

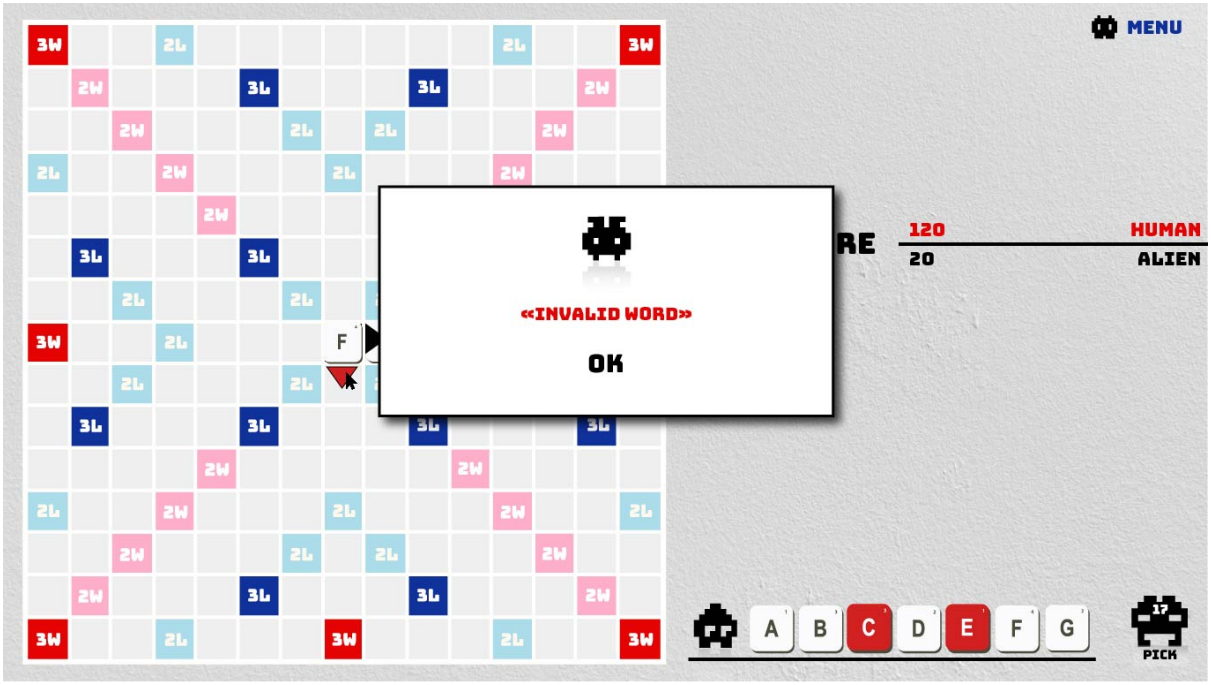

Figure 7 - Interface Message d'erreur n°3

3. RAPPORT TECHNIQUE

1. CONCEPTION

1. Technologies utilisées

Plusieurs technologies et logiciels sont utilisés pour la programmation de cette application.

- Langage Java

Ce choix de langage a été fait car Java permet une programmation événementielle et graphique intéressante. C'est aussi un langage que l'équipe de développement maîtrise le mieux.
Plus particulièrement, le Java orienté objet a été choisi, nous donnant plus de flexibilité.

- Logiciel Illustrator

Pour réaliser le design de l'application et avoir un rendu original, le logiciel Illustrator d'Adobe a été utilisé. C'est un logiciel donnant beaucoup de possibilités graphiques et permettant d'enregistrer des images sous divers formats.

- Eclipse

L'IDE Eclipse a été utilisé pour la programmation dans le langage Java. Cet IDE permet à l'équipe de développement de passer facilement d'un fichier à un autre et, ainsi, de les lier facilement afin qu'ils fonctionnent ensemble et donnent un résultat cohérent et fonctionnel.

2. Diagrammes UML

Ci-après, le « Use Diagram » de l'application. En annexe, vous trouverez deux diagrammes d'activités représentant une partie du programme.

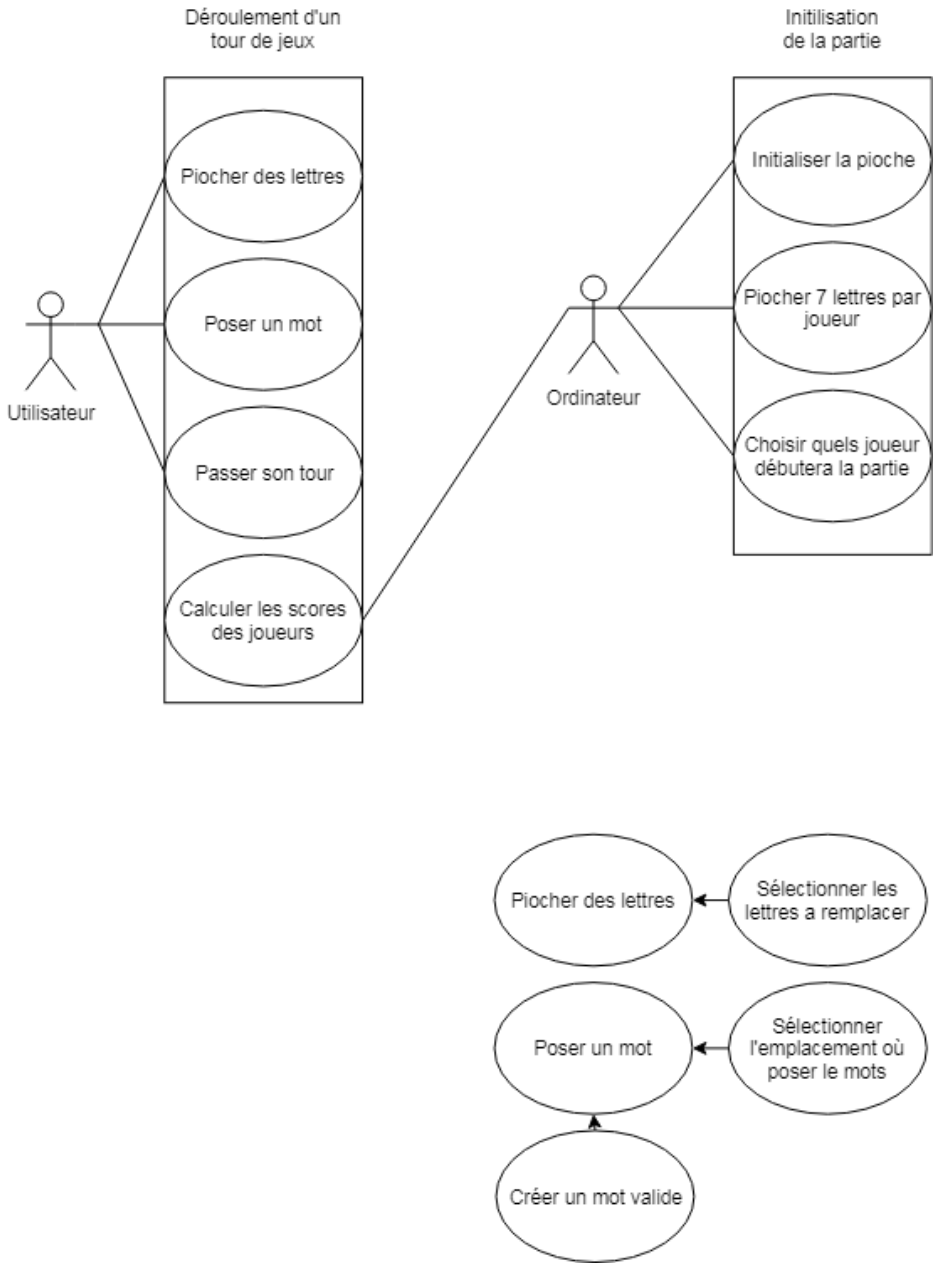

Figure 8 - Use Diagram

3. Algorithme

- Initialisation du jeu
    - Le premier joueur est choisi aléatoirement parmi les 2 joueurs

- La pioche est initialisée

- Tour de jeu d'un joueur
  - **Si** il reste assez de lettres dans la pioche **alors** n lettres sont ajoutées au jeu du joueur jusqu'à ce qu'il ai 7 lettres dans son jeu
    Si il ne reste pas assez de lettres **alors** toute la pioche est ajoutée au jeu du joueur
    Si il n'y a plus de lettres dans la pioche **alors** ne fait rien

Le joueur choisit de poser un mot →

- Le joueur choisit parmi ses lettres, dans l'ordre, le mot qu'il souhaite poser sur le plateau

- Le joueur choisit la case de départ de son mot sur le plateau

- Le joueur choisit la direction de son mot

- **Si** le mot existe dans la base de données **alors** le mot est posé sur le plateau
  - Le score est compté et est ajouté au score du joueur
  - Le tour du joueur suivant commence

  Si le mot n'existe pas dans la base de données **alors** la sélection des lettres du jeu du joueur est réinitialisée à 0 et le tour de jeu recommence

Le joueur choisit de piocher →

- Le joueur sélectionne le nombre de lettres qu'il souhaite se défausser et échanger avec la pioche
- Le joueur demande à la pioche de les échanger
- De nouvelles lettres remplacent les lettres précédemment sélectionnées qui sont, elles, ajoutées à la pioche
- Le tour du joueur suivant commence

- Fin de la partie

- **Si** il n'y a plus de lettres dans la pioche et si les deux joueurs signalent ne plus avoir de mots à jouer **alors** le joueur avec le plus gros score est affiché gagnant
- La partie se finit

2. REALISATION

1. Les classes

Le modèle choisit nécessite 4 classes différentes, chacune ayant ses propres attributs.

-**La classe Plateau** : elle correspond à la classe où nous allons stocker ce que contient le plateau.
Pour ce faire la classe dispose de trois attributs :

-Le tableau lettres, qui contient toutes les lettres de type “Jetons” (voir plus bas) dans un tableau.

-Le tableau remplissage, qui contient des booléens et qui nous permet de savoir si une case est libre ou pas, c’est un tableau de taille identique à “lettres”.

-Le tableau bonus, qui contient des entiers et qui contient les valeurs 0 si la case n’a pas de bonus, 1 pour une case lettre compte double, 2 pour une case lettre compte triple, 3 pour une case mot compte double et 4 pour une case mot compte triple.

-**La classe joueur** : Elle contient les informations inhérentes à chaque joueur à l’aide de deux attributs :

-Un entier qui correspond au score.

-Une ArrayList de “Jetons” (voir plus bas) qui contient les jetons présents dans la main du joueur.

-La classe Jetons : C’est la classe qui modélise les pièces que nous posons sur le plateau.

Chaque jeton à deux attributs :

-un char qui représente la lettre du jeton.
-un int qui correspond à sa valeur au moment du calcul des points.

-La classe pioche : Comme son nom l’indique, elle modélise la pioche.

Elle possède un attribut unique :

-une ArrayList de “Jetons”, qui correspond aux jetons disponibles dans la pioche.

Ces 4 classes correspondent au modèle de notre application, ce sont dans ces classes que l’on sauvegarde l’état de la partie et que l’on effectue le traitement sur nos données afin d’afficher à l’utilisateur des informations valides à l’aide de la classe Affichage.

Pour la partie “vue” de notre application Swing est utilisé ainsi que la classe Jpanel dans une classe unique dédiée à l’affichage.

-La classe affichage : Son rôle est d’afficher à l’écran l’état de la partie, pour ce faire elle dispose de très nombreux attributs :

-Un JFrame, correspond à la fenêtre à laquelle nous allons attacher notre JPanel.

-7 JButton pour afficher les lettres du joueur courant, de cette façon il est facile pour nous de savoir quelles lettres sont sélectionnées.

-7 JLabel pour pouvoir afficher les lettres jouées à chaque tour.

-Les ImageIcons et Images de tous éléments de la fenêtre (le plateau, les icônes, les lignes, le menu, les flèches).

Enfin, la partie “contrôleur” est réalisée par la classe Partie qui fait le lien entre les actions de l'utilisateur et le lancement de nos méthodes pour mettre à jour le modèle d’une part et de déclencher l’affiche d’autre part.

-La classe Partie :

Elle possède les attributs suivants :

-Elle prend chacune des classes précédentes en attributs.

-7 booléens, leurs valeurs sont modifiées par chacun des boutons représentant la main du joueur. Ils nous permettent de savoir quelles lettres le joueur désire jouer.

-3 booléens pour savoir si le joueur courant a déjà effectué une action pendant son tour de jeu (piocher, passer ou poser).

-1 une ArrayList dans laquelle nous stockons les lettres choisies par l’utilisateur afin de pouvoir les poser sur le plateau.

« Partie » implémente l’interface MouseListener qui nous permet d’écouter les cliques de la souris, dans notre cas nous utiliserons uniquement le clique sur le bouton gauche de la souris. Elle implémente également actionListener qui nous permet d’écouter les actions sur les boutons.

Chacune de ces classes disposent évidemment de méthodes, que nous allons à présent détailler.

2. Les Fonctions

Pour chacune des classes nous ne présenterons pas les getters et les setters, puisqu'ils ne présentent pas d'intérêt d'un point de vue algorithmique et qu'ils ne sont pas propres à la réalisation de notre Scrabble. Les méthodes des classes Pioches, Jetons et Joueur ne seront donc pas détaillés.

Fonctions de la classe Plateau :

Plateau dispose de trois fonctions principales, le calcul des scores, vérifier qu'un mot existe dans le dictionnaire et vérifier que le placement demandé par le joueur est possible.

- calculScore():

```
int calculScore(int idebut,int jdebut, int sens,ArrayList<Jetons> jetons){
 int bonusMot=0;
 int score=0;

 if(sens==1) {
        for(int j = 0; j < jetons.size(); j++) {
                if(bonus[idebut][j]==1 ||bonus[idebut][j]==2){
                score+=jetons.get(j).getValeur()*(bonus[idebut][j]+1);
                }
                else{
                        bonusMot*=(bonus[idebut][j]);
                }
        }
 }
        if(sens==0) {
                for(int i = 0; i <jetons.size(); i++) {
                        if(bonus[i][jdebut]==1 || bonus[i][jdebut]==2){
                                score+=jetons.get(i).getValeur()*(bonus[i][jdebut]+1);
                        }
                        else{
                                bonusMot*=(bonus[i][jdebut]);
                        }
                }
        }
                score=score*bonusMot;
                return score;
        }
```

Figure 9 - Fonction "calculScore"

Cette fonction prend en argument les coordonnées de la première lettre du mot, le sens, vertical ou horizontal, et enfin l'ensemble des lettres jouées dont il faut calculer le score.
La fonction récupère ensuite le bonus présent dans chaque case puis l'applique soit à la lettre soit au mot final.

- motValide():

```
public boolean motAtourValide(int i, int j, int d) { //d est la direction 0 pour en bas, 1 pour à droite

        String motVerif = "";
        int curseur=0;
        int idep=i, jdep=j;

        //si la direction choisie est vers le bas, faire le déplacement vers le bas
        if(d==0) {
                while(getRemplissage(idep,j)==true) {

        //case de départ du mot joué (qu'on va appeler case d'origine) a une lettre voisine à gauche et pas à dr
                        if (getRemplissage(i,j-1)==true) {

                                //initialisation du curseur au début du mot voisin de gauche et du mot à vérifie
                                motVerif="";
                                curseur=j;
                                while(getRemplissage(i,curseur)==true) {
                                        curseur=curseur-1;
                                }

                                //parcours du mot jusqu'à la fin et construction du mot à vérifier
                                while(getRemplissage(i,curseur)==true) {
                                        motVerif = motVerif +  lettres[i][curseur].getLettre(); //lettreCase est
                                        curseur=curseur+1;
                                }

                                //si mot à vérifier faux, retourne faux
                                if(Test(motVerif)==false) {
                                        return false;
```

Figure 10 - Fonction "motAtourValide"

```
                }
        }

        //case d'origine a une lettre voisine à droite et pas à gauche
        if(getRemplissage(i,j-1)==true && getRemplissage(i,j+1)==false) {
                motVerif="";
                curseur=j;

                //parcours du mot jusqu'à la fin et construction du mot à vérifier
                while(getRemplissage(i,curseur)==false) {
                        motVerif=motVerif + lettres[i][curseur].getLettre();
                        curseur ++;
                }

                //si mot à vérifier faux, retourne faux
                if(Test(motVerif)==false) {
                        return false;
                }
        }

        //case d'origine a une lettre voisine en haut et pas en bas ou aussi en bas

        if(getRemplissage(i-1,j)==false) {
                motVerif="";
                curseur=i;
                while(getRemplissage(curseur,j)==false) {
                        curseur --;
                }

                while(getRemplissage(curseur,j)==false) {
                        motVerif=motVerif + lettres[curseur][j].getLettre();
                        curseur ++;
                }
                if(Test(motVerif)==false) {
                        return false;
                }
        }
```

Figure 11 - Remplissage plateau n°1

```
                return false;
            }
        }

        if(getRemplissage(i+1,j)==false) {
            motVerif="";
            curseur=i;

            while(getRemplissage(curseur,j)==false) {
                motVerif=motVerif + lettres[curseur][j].getLettre();
                curseur ++;
            }
            if(Test(motVerif)==false) {
                return false;
            }
        }
        idep ++;
    }
}
```

Figure 12 - Remplissage plateau n°2

Ici nous construisons tous les mots qui sont induits par le mot de base, dans le but de vérifier si la position est valide. Si tous les mots induits sont anglais et qu'il y a suffisamment de place pour poser les lettres alors la méthode retourne true.

- Test():

```
    }
public static boolean Test(String mot)
{
        boolean found =false;


    try {
        File fileDir = new File("src/images/ListeMots.txt");

        BufferedReader in = new BufferedReader(
           new InputStreamReader(
             new FileInputStream(fileDir), "UTF16"));

        String str;

        while ((str = in.readLine()) != null && found==false) {


       System.out.println(str);
      if (str.equals(mot)) {
                found=true;
      }
        }

        in.close();

        return found;
    }
```

Figure 13 - Fonction "Test"

Ici nous donnons à la fonction le mot que nous voulons tester sous forme de chaîne de caractères, puis la fonction cherche dans le fichier test si le mot est anglais. Si c'est le cas elle renvoie true, sinon false.

Fonctions de la classe Affichage :

- painComponent(Graphics g):

```
public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(backgroundimg, 0,0, null);
        g.drawImage(plateauimg, 0,0, null);
        g.drawImage(menuimg, 1190,0, null);
        g.drawImage(scoreimg,830,230, null);
        g.drawImage(scoretxtimg,905,240, null);
        g.drawImage(scorebarimg,1020,250, null);
        g.drawImage(gamebarimg,800,720, null);
        g.drawImage(pickimg,1200,650, null);


}
```

Figure 14 - Fonction "paintComponent"

C'est la méthode que nous utilisons après que le constructeur ait récupéré les ressources. Elle nous permet de placer les différents éléments dans l'espace. Tous ces éléments sont placés une seul fois au départ puisqu'ils sont statiques.

- affichagesLettresJoueursCourant(ArrayList<Jetons> jetons):

```
void affichageLettresJoueurCourant(ArrayList<Jetons> jetons) {
 lettre1.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(0).getLettre()+".png")));
 lettre2.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(1).getLettre()+".png")));
 lettre3.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(2).getLettre()+".png")));
 lettre4.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(3).getLettre()+".png")));
 lettre5.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(4).getLettre()+".png")));
 lettre6.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(5).getLettre()+".png")));
 lettre7.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(6).getLettre()+".png")));
```

Figure 15 - Fonction "affichageLettresJoueurCourant"

Cette méthode est appelée à chaque tour de jeu pour afficher les images des lettres dont dispose le joueur pour les placer sur les différents boutons.

- affichageLettresPlateau(ArrayList<Jetons> jetons,int x,int y,int sens):
-

```
if(jetons.size()>=1) {
        mot.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(0).getLettre()+".png")));
        mot.setBounds(x+(0*0)-48,y,48,48);

        this.add(mot);

        }
if(jetons.size()>=2) {
        mot1.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(1).getLettre()+".png")));
        mot1.setBounds(x+(1*48)-48,y,48,48);

        this.add(mot1);

        }
```

Figure 16 - Fonction "affichageLettresPlateau"

C'est un morceau de la méthode que l'on utilise pour placer les lettres sur le plateau. On y place chaque lettres les unes après les autres en fonction du nombre de lettres qui compose le mot. Puis on ajoute la lettre au JPanel.

- colorerLettre(int lettre,boolean couleur,ArrayList<Jetons> jetons):

```
if(jetons.size()>=1) {
        mot.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(0).getLettre()+".png")));
        mot.setBounds(x+(0*0)-48,y,48,48);

        this.add(mot);

        }
if(jetons.size()>=2) {
        mot1.setIcon(new ImageIcon(getClass().getResource("/images/Lettres/"+jetons.get(1).getLettre()+".png")));
        mot1.setBounds(x+(1*48)-48,y,48,48);

        this.add(mot1);

        }
```

Figure 17 - Fonction "colorerLettre"

Nous utilisons cette méthode pour change la couleur de boutons quand le joueur sélectionne une lettre, s'il est blanc il devient rouge et inversement.

Fonctions de la classe Partie :

La classe partie est composée de 4 fonctions principales, l'écoute du clique et d'une fonction pour chaque action principale (poser sur le plateau, passer son tour, ou piocher)

- mousePressed(MouseEvent e):

```
void mousePressed(MouseEvent e) {
  int x;
  int y;
  x=e.getX();
  y=e.getY();
  if(x<=730 && y<=730) {//gestion des cliques sur le plateau
          x=x/48;
          y=y/48;
          actionPlateau(x,y);



  }
  if(x>=1200 && x<=1264 && y>=650 && y<=736 && lettresPioches==false){//gestion du clique sur la pioche

          actionPioche();
  }

  if(motPose==true || lettresPioches==true){
          changementDeJoueur();


  }
```

Figure 18 - Fonction "mousePressed"

On regarde où le joueur clique pour lancer les actions adéquates, si le joueur vient d'effectuer une action valide on change de joueur.

- actionPlateau(int x, int y):

```
}
public boolean actionPlateau(int x, int y) {
        int nbjetons=inplay.size();
        int sens;
        if(flechesAffiches==false && inplay.size()!=0) {
                affichage.affichagefleches(x,y);
                flechesAffiches=true;
                xPrec=x;
                yPrec=y;

                System.out.println(plateau.getRemplissage(x, y));
                return true;

        }
        if(inplay.size()==0){
                JOptionPane jop1 = new JOptionPane();
                jop1.showMessageDialog(null, "Choisissez d'abord vos lettres !", "Information", JOptionPane.INF
                return true;
        }
        if(flechesAffiches==true) {
                if(xPrec==x-1)
                {sens=0;}
                else sens=1;
                affichage.affichageLettresPlateau(inplay, x, y,sens);
                jcourant.supprimerMainjoueur(inplay);
                for(int i=0;i<nbjetons;i++){
                        jcourant.ajoutJeton(pioche.piochejeton());
                }
                        plateau.setRemplissages(x, y, sens, inplay.size());
                        plateau.setLettres(x, y, sens, inplay);
                        jcourant.setScore(plateau.calculScore(x, y, sens, inplay));
```

Figure 19 - Fonction "actionPlateau"

Cette fonction a deux actions différentes, dans un premier temps elle demande au joueur de choisir la direction dans laquelle il souhaite poser son mot, puis il pose le mot du joueur.

- actionPioche():

```
public void actionPioche(){
        int nbjetons=inplay.size();

        jcourant.supprimerMainjoueur(inplay);
        pioche.remetplusieursjeton(inplay, inplay.size());
        inplay.clear();
        for(int i=0;i<nbjetons;i++){
        jcourant.ajoutJeton(pioche.piochejeton());
        }
        lettresPioches=true;
        affichage.affichageLettresJoueurCourant(jcourant.getJetonsjoueur());

}
```

Figure 20 - Fonction "actionPioche"

On fait piocher le joueur courant.

3. Diagramme de classe

Tout le fonctionnement de notre application est résumé dans le diagramme de classe ci-après.

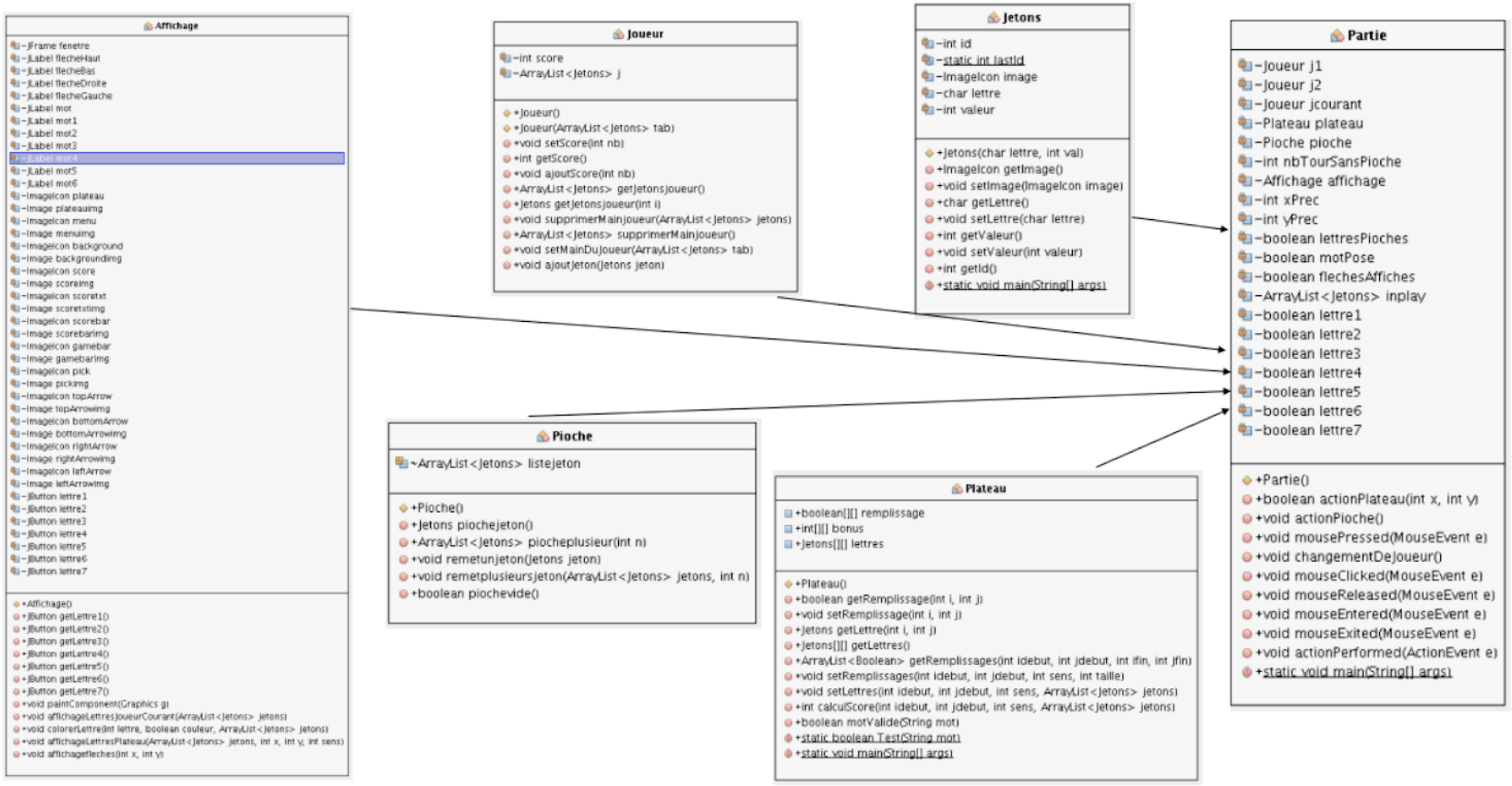

Figure 21 - Diagramme de classe

4. MANUEL D'UTILISATION

Au lancement chaque joueur reçoit 7 lettres dans sa main, ensuite un des 2 joueurs est choisi pour débuter la partie.

Lorsque que c'est son tour un joueur peut effectuer une de ces actions :

- Placer un mot : il doit sélectionner les lettres de son mot dans l'ordre avec sa souris (par exemple pour le mot « cat » il sélectionne d'abord la lettre c puis a, puis t). Ensuite il clique sur la case où il veut placer la première lettre, puis il clique sur une des flèches apparentes pour indiquer dans quel sens il souhaite écrire son mot. Si son mot est valide il sera placé sinon le joueur devra placer un autre mot ou effectuer une autre action.

- Échanger une/des lettres : il doit sélectionner les lettres dont il veut se débarrasser avec sa souris puis cliquer sur « Pick », il piochera ainsi de nouvelles lettres à la place de celles-ci.

-Passer son tour : il suffit de cliquer sur le bouton "Skip" avec sa souris.

Lors du premier mot placé, le mot doit avoir une lettre dans la case centrale du plateau. Ensuite, chaque mot placé par un joueur doit obligatoirement se raccorder sur une ou plusieurs lettres déjà posées sur la grille.

Lorsque la partie est terminée le gagnant est annoncé via une fenêtre apparente, les joueurs ont ainsi le choix de recommencer une partie ou de quitter le jeu, ce choix se fait à l'aide de la souris.

5. GESTION DE PROJET

1. DÉMARCHE PERSONNELLE

Chacun d'entre nous a choisi ce projet pour plusieurs raisons :

- La possibilité de développer cette application en Java et donc de s'améliorer dans ce langage
- Mettre en pratique la programmation d'application de jeu
- Mettre en pratique la création design

Effectuer un projet nous a aussi permis de s'améliorer sur plusieurs points :

- La programmation (Java, graphique, événementielle)
- La gestion d'équipe
- L'application d'un projet avec une méthode agile
- La planification

2. PLANIFICATION DES TÂCHES

1. Méthode Agile

Pour gérer au mieux notre projet et le planifier d'une façon qui correspond le mieux aux membres de notre équipe, nous avons choisi d'utiliser la méthode Agile.

Qu'est-ce que la méthode Agile ? D'après ideematic.com, elle se définit de la façon suivante : "Une méthode Agile est une approche itérative et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions.". En effet, elle permet, contrairement à d'autres méthodes qui planifient dès le début du projet l'ensemble des tâches, d'organiser des périodes de travail pour atteindre un objectif après l'autre et, ainsi, avancer le projet pas à pas en ayant la possibilité de s'adapter aux changements possibles de la demande.

Nous avons pensé que la méthode Agile était la meilleure à utiliser pour notre groupe, correspondant à la façon de travailler de chacun d'entre nous. De plus, nous étions d'accord sur le fait que c'ést la méthode la plus efficace pour gérer ce projet.

2. Différentes étapes

Nous avons commencé par la base du projet pour passer, étape par étape, à la tâche suivante. Le projet s'est donc déroulé en plusieurs périodes.

Première période (fin janv - février) - Analyse du projet :

Durant cette période, après avoir eu en notre possession la consigne, nous nous sommes renseignés sur différents sujets qui pouvaient nous aider à mieux comprendre et appréhender ce projet :

- Les règles du jeu Scrabble français et anglais : il était important de bien comprendre le jeu que nous allions développer pour réfléchir, par la suite, à une application ergonomique et fidèle.

- Les différentes applications de jeu de Scrabble existantes : nous avons regardé le développement de plusieurs applications de jeu de Scrabble pour comprendre comment nous pouvions nous y prendre et quelles possibilités nous avions. Cela nous a aussi permis de récolter plusieurs idées qui nous ont beaucoup servies par la suite.

- Les différents langages de programmation à notre disposition : il était important de choisir un langage qui nous permettait de développer une application optimale et qui s'adaptait à nos connaissances pour que l'on puisse arriver à un bon résultat. Notre choix s'est finalement porté sur le langage Java.

- La programmation graphique en Java : à ce stade de l'année nous n'avions pas encore eu de cours sur la programmation graphique et chacun des membres de l'équipe n'avait pas de connaissances à ce sujet. Nous avons donc fait des recherches pour se construire une idée de nos possibilités et de la structure de notre futur programme.

- La programmation événementielle en Java : cette recherche reprenait les mêmes objectifs que celle sur la programmation graphique.

Deuxième période (fin fev - mars) - Rédaction du cahier des charges :

Au cours de cette période, suite à des recherches et à une analyse du projet approfondies, nous avons décidé de rédiger un cahier des charges complet. Il était important pour nous de passer du temps sur cette étape pour que, par la suite, l'étape de développement se déroule sans accrocs par rapport à une mauvaise compréhension de la demande / consigne par un des membres de l'équipe.

Cette étape s'est accompagnée de trois autres :

- Construction de l'algorithme : c'était un premier pas pratique sur la technique qui a permis de clarifier la structure de notre futur programme.

- Construction des maquettes graphiques : toute la structure graphique a été effectuée lors de cette période en fonction de l'algorithme. Ces maquettes ont été construites grâce au logiciel Illustrator.

- Elaboration d'une première ébauche de diagramme de classe et d'utilisation : ceux-ci nous permettaient d'avoir un premier pas dans l'étape de développement, de l'appréhender avec plus de rigueur et de la faciliter au moment venu. Elle a permis aussi de clarifier un peu plus la structure de l'application en plus du cahier des charges.

Troisième période (avril - deb mai) - Développement de l'application :

Après avoir écrit et structuré les bases du projet à travers un algorithme, des diagrammes et des maquettes, nous sommes passés à l'étape supérieure : le développement de l'application. Cependant, celle-ci ne pouvait se faire que si les étapes précédentes avaient été effectuées.

Cette période s'est décomposée en plusieurs étapes :

- Développement des classes de base :
  - Pioche
  - Joueurs
  - Plateau
  - Jetons

- Développement des méthodes plus complexes de la classe Plateau (vérification de l'existence des mots posés sur le plateau dans le fichier .txt de mots)

- Programmation graphique :

Un premier affichage (incomplet) a été effectué. Il n'était pas complet, mais possédait volontairement assez de contenu pour passer à l'étape de la programmation événementielle.

- Programmation événementielle :

Des recherches approfondies sur l'événementiel ont été effectuées et les premières fonctionnalités ont été créées.

Quatrième période (mai) - Finalisation du développement et rédaction du rapport de projet :

Lors de cette période, nous avons finalisé le développement de l'application : notamment la programmation graphique et événementielle.

Nous avons, de plus, effectué les tests qui nous ont permis de vérifier si l'application était bien fonctionnelle.

Enfin, nous avons rédigé le rapport, élément final de notre projet.

Remarque - Planification réelle par rapport à la planification prévisionnelle :

Contrairement à la planification réelle décrite ci-dessus, nous avions prévu d'utiliser moins de temps pour la rédaction du cahier des charges. Cependant, la méthode agile nous a permis de gérer cet imprévu en nous permettant d'avancer tout autant sur notre projet tout en construisant un bon cahier des charges.

Nous avons choisi le gestionnaire de version GitHub fusionné avec Git. Il nous a permis de partager régulièrement chaque ajout, modification ou suppression de code.

En effet, nous partagions quotidiennement le code de chacun afin d'optimiser notre temps de travail et, ainsi, éviter que deux personnes aient fait la même partie de code.

2. Google Drive

Nous avons choisi ce service de stockage et de partage afin de partager et créer ensemble les différents fichiers autres que les fichiers de code.

Google Drive nous a servi notamment à :

- Noter les comptes rendu de réunion
- Créer un planning avec les rôles de chacun
- Avancer le cahier des charges
- Partager les différents diagrammes UML
- etc…

5. Bilan critique

1. Avancée

D'après le temps imparti, nos connaissances et nos capacités, nous avons eu la possibilité d'effectuer la majorité du cahier des charges, soit de l'application :

- Le plateau - Fonctionnel
    - Affichage
    - Pose des mots
    - Comptage de points
    - Flèches de direction
- Le jeu du joueur - Fonctionnel
    - Sélection des lettres dans l'ordre
    - Coloration des lettres lors de la sélection
- La pioche - Fonctionnelle

    - Pioche automatique à chaque fin de jeu du joueur
    - Possibilité d'échanger des lettres
    - Soustraction des lettres piochées à la réserve

- Le menu - Fonctionnel

    - Affichage d'une fenêtre
    - Option "Retry" qui recommence une partie
    - Option "Quit" qui ferme l'application

- Le score - Fonctionnel

    - Augmentation du score
    - Affichage du score

L'ensemble des éléments de l'écran de jeu sont affichés.

2. A développer

Cependant des détails pour rendre l'application plus complexe et confortable d'utilisation restent à être développés :

- Création du premier écran d'affichage avec le nom de l'application et le bouton "Start" qui mène à l'écran de jeu
- Modification de l'affichage des fenêtres en les adaptant au design du jeu comme sur les interfaces pensées dans le cahier des charges
- Ajout de fonctionnalités / boutons pour faciliter l'utilisation de l'application

3. Conclusion

Certains besoins non fonctionnels (secondaires) restent à être développés comme le design, l'ajout de quelques fonctionnalités qui facilitent la prise en main du jeu etc. Cependant, nous avons réussi à développer l'essentiel de l'application, c'est à dire les besoins fonctionnels (primaires) !

6. CONCLUSION

L'objectif du projet était de coder un jeu de Scrabble joueur contre joueur. Après un semestre entier à travailler sur le projet nous pouvons dire que l'objectif a été atteint.
S'il reste encore des améliorations et des tests encore plus poussés nous pouvons dire que l'essentiel est fonctionnel.

Nous avons également des idées sur comment rendre le projet meilleur avec plus de temps disponible.
Tout d'abord, rendre l'interface plus vivante, avec la possibilité de bouger ses Jetons par exemple.
Ensuite, pour parler d'améliorations techniques plus que cosmétiques, nous pourrions envisager de créer une intelligence artificielle, et avoir la possibilité de jouer contre l'ordinateur ou encore donner la possibilité à deux joueurs de jouer en réseau sur deux machines différentes.

Ce projet était également le premier travail concret que nous devions mener de bout en bout et il nous a permis d'apprendre certaines choses.

Tout d'abord techniquement, cela nous a appris à utiliser de nouvelles techniques de programmation de manière autonome. Lorsque nous étions confrontés à une classe graphique que l'on ne savait pas utiliser, la seule solution à notre disposition était de faire une recherche dans la documentation de la classe pour résoudre notre problème, ou encore quand nous avons dû apprendre à utiliser GIT.

Un autre aspect, encore plus important, a été la gestion du travail en équipe. Nous nous sommes rendu compte de la difficulté, mais surtout de la rigueur que cela demande pour pouvoir répartir le travail efficacement et avoir quelque chose de fonctionnel au moment de la mise en commun. Cela nous a permis de comprendre l'intérêt d'un cahier des charges précis et de bien définir les besoins et les fonctionnalités avant même d'ouvrir un IDE.

Pour conclure, nous pouvons dire que nous sommes satisfaits de cette expérience, d'un point de vue technique mais surtout d'un point de vue gestion. Même si nous aurions aimés avoir le temps de faire plus, nous sommes fiers de notre travail sur cette application « Jeu du Scrabble ».

ANNEXES

Règles du jeu « Scrabble »

(Règles récupérées sur le site le 26/04/2018 : https://aide-scrabble.fr/regles/)

Début du jeu : Sac de Scrabble 4

5 Chaque joueur pioche à tour de rôle une lettre dans le sac. Celui qui obtient la lettre la plus proche de A (dans l'ordre alphabétique) joue en premier. Si plusieurs joueurs ont pioché la même lettre, ils posent leur lettre sur la table puis piochent chacun une nouvelle lettre dans le sac.

Si un joueur a pioché un joker (lettre blanche), il le pose sur la table puis pioche une nouvelle lettre dans le sac. Une fois le premier joueur déterminé, l'ordre des joueurs peut être déterminé au choix : – par la lettre piochée par chaque joueur – par l'ordre des aiguilles d'une montre

Déroulement d'une partie

Mot compte double : Les joueurs jouent à tour de rôle.

Lorsque c'est son tour, un joueur pioche 7 pions dans le sac, puis il doit poser un mot d'au moins 2 lettres sur le plateau en utilisant un ou plusieurs de ses pions.

Une fois le mot posé, les points de ce mot sont ajoutés à son score (voir plus bas, « Décompte des points »), et c'est au tour du 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.

Un mot est valide s'il figure dans l'édition en vigueur du dictionnaire « L'Officiel du Scrabble » (ODS), édité par Larousse.

A ce jour, l'édition est vigueur est l'ods4 depuis le 1er janvier La prochaine édition (ODS5) sortira à l'automne et entrera en vigueur le 1er janvier

Pour reconnaître une version d'ods, il suffit d'y vérifier la présence de quelques petits mots : – le mot ZUP est apparu dans l'ods4 (année 2004) – le mot AA est apparu dans l'ods3 (année 1999) – le mot EX est apparu dans l'ods2 (année 1994)

De manière générale, sont acceptés : – les pluriels : MAISONS, GENOUX, BARMEN… – les féminins : GRANDE, MOQUEUSE, CADETTE – les conjugaisons : PARTIRA, VOIS, MENACIEZ… De manière générale, sont refusés : – les expressions et mots composés : DEMI-FOND, AU REVOIR, N'IMPORTE – les sigles et abréviations : DVD, NASA… – les symboles chimiques : Fe, Az, Pt… – les noms propres : LYON, ANTOINE, RAMBO… En cas de litige sur un mot, c'est l'ods qui sert de référence. Validité d'un mot : L'Officiel du jeu Scrabble (ODS) 5

Décompte des points : Exemples de décomptes (partie à 3 joueurs)

Lorsqu'un mot est placé sur le plateau, sa valeur est calculée comme suit : – on additionne la somme des valeurs de chaque lettre, en tenant compte des éventuelles cases « lettre compte double » ou « lettre compte triple » sur lesquelles sont posées ces lettres, – puis on multiplie cette somme en tenant compte des éventuelles cases « mot compte double » ou « mot compte triple » sur lesquelles est posé ce mot.

Si le mot passe sur deux cases « mot compte double », sa valeur est multipliée par 4 (« quadruple »). Si le mot passe sur deux cases « mot compte triple », sa valeur est multipliée par 9 (« nonuple »).

Si le mot posé forme simultanément d'autres mots, la valeur de ces mots est également comptabilisée, puis on additionne la valeur de tous les mots formés. L'effet multiplicateur d'une case n'est pris en compte que pour le(s) mot(s) posé(s) en premier sur cette case.

Au coup suivant, seul la valeur de la lettre posée sur la case est prise en compte. Si un joueur utilise ses 7 pions pour former un mot, il a un bonus de 50 points. Chaque jeu de Scrabble possède 2 pions « blancs », ce sont les jokers.

Leur valeur est de 0 point, mais ils peuvent remplacer n'importe quelle lettre de l'alphabet, au choix du joueur. Ils permettent souvent de faire un « Scrabble » et d'obtenir ainsi la prime de 50 points.

Le premier joueur pose HANTEZ en H6 : Le Z est posé sur une case « lettre compte double », et le mot passe sur l'étoile centrale (case « mot compte double »).

On additionne la valeur des lettres de HANTEZ : *2 (Z lettre compte double) = 28 points. On multiplie cette valeur par 2 (mot compte double). Le score est donc de 56 points.

Le second joueur pose FAÇADES en 6F, transformant HANTEZ en CHANTEZ : On additionne la valeur des lettres de FAÇADES : 4*3 (F lettre compte triple) *3 (D lettre compte triple) = 25 points. On additionne la valeur des lettres de CHANTEZ : = 21 points. On fait la somme des valeurs des 2 mots : = 46 points.

On ajoute 50 points de bonus car le joueur a utilisé ses 7 lettres. Le score est donc de = 96 points. AERIEN Le troisième joueur pose AERIEN en G6, en « collante » au-dessus de CHANTEZ : Le mot ne comporte que des lettres à 1 point, et passe sur 2 cases « lettre compte double ». Il forme au passage dans l'autre sens 5 petits mots de 2 lettres, tous valides. AERIEN rapporte 8 points EH = 6 points, RA = 2 points, IN = 3 points, ET = 2 points, NE 6

7 = 2 points Le score est donc de = 23 points.

C'est au tour du premier joueur, qui pose JUNGLE en F10, en « pivot » au-dessus d'aerien : Le J (8 points) est posé sur une case « lettre compte triple ». Sa valeur monte donc à 24 points, et ce pour les 2 mots formés simultanément par le J : JUNGLE rapporte = 32 points JET rapporte = 26 points UNE rapporte = 3 points Le score est donc de = 61 points.

Passer son tour / Changer ses lettres

Chaque joueur a le droit de passer son tour.

Il peut en profiter pour changer une ou plusieurs de ses lettres.

Dans ce cas : il annonce qu'il veut changer des lettres. il pose les lettres à changer, face cachée, à la vue des autres joueurs. il pioche autant de nouvelles lettres dans le sac. Il place les lettres rejetées dans le sac. 2 Il n'est plus possible de changer des lettres en fin de partie, lorsqu'il reste moins de 7 lettres dans le sac.

Mot joué incorrect

Chaque joueur peut contester un mot posé par un autre joueur. Dans ce cas, on vérifie l'existence du mot litigieux dans l'ods.

Si ce mot n'est pas valide, le joueur reprend les lettres qu'il vient de poser, marque 0 point, et doit passer son tour sans pouvoir changer ses lettres. La contestation doit être faite avant que le joueur suivant ne place son mot.

Si un mot incorrect est placé sans avoir été contesté, il est bien sûr possible pour un autre joueur de s'appuyer ensuite sur une ou plusieurs lettres de ce mot. Ce joueur peut alors faire lui-même l'objet d'une contestation à la condition que celle-ci concerne uniquement l'un des mots formés par le joueur sur son tour.

Par exemple, si André a joué « Velour » (sans le S final obligatoire) sans avoir été contesté, et si ensuite Bernard joue un mot correct formant au passage « VELOURS », alors le coup de Bernard est parfaitement valide.

Par contre, s'il joue un mot correct mais qui forme au passage « Veloure », alors il peut faire l'objet d'une contestation et devoir passer son tour.

Diagrammes d’activités

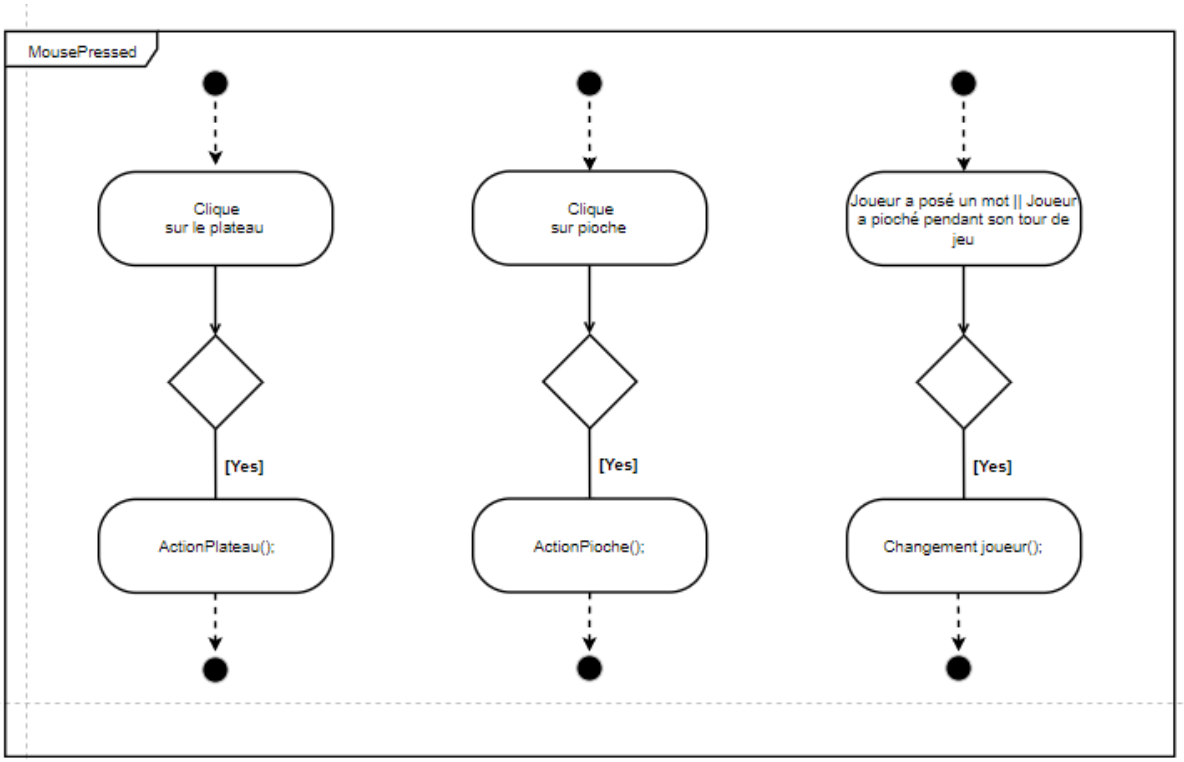

Figure 22 - Diagramme d'activité "Mouse Pressed"

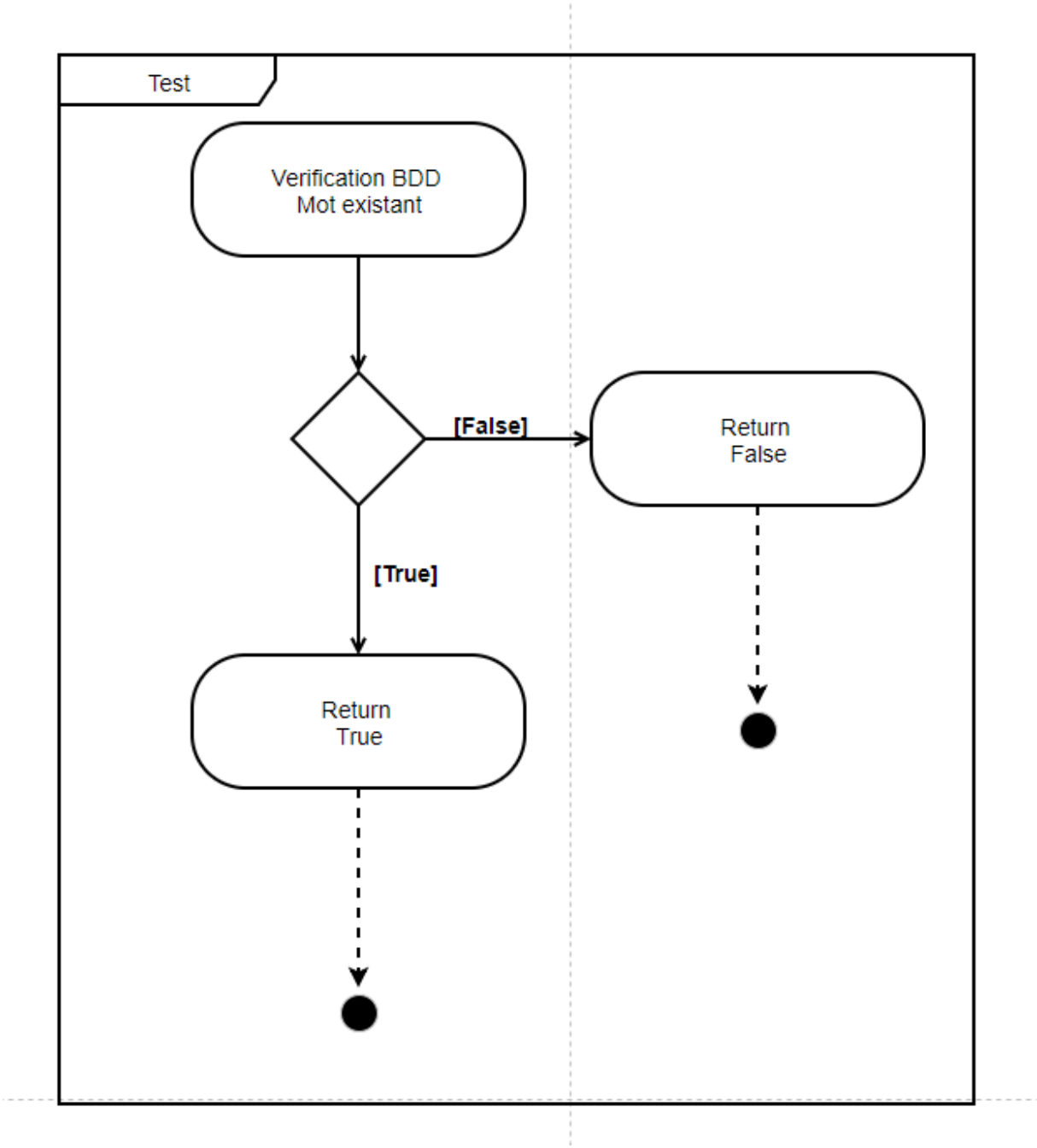

Figure 23 - Diagramme d'activité "Test"

RESUME

Le projet du jeu de « Scrabble » est une application pour ordinateur dont l'objectif est de permettre aux amateurs du jeu de société de jouer sur machine au travers un univers différent et particulier. Cet univers, plus ludique, fait référence aux jeux vidéo et, plus précisément, au rétrogaming et aux aliens.
Ce jeu de Scrabble revisité doit être accessible à toute personne possédant un ordinateur. Il doit être facile d'utilisation et ludique à utiliser. Il permet à 2 joueurs de s'affronter sur un même écran d'ordinateur.
L'application a été développée avec le langage Java.

Mots clés : Application, Java, Scrabble, jeu

This project of Scrabble game is a computer application. The objectives are to enable lovers of this board game to play with a machine troughout a different and special universe . This universe is more playful and refers to video games more specifically retrogaming and aliens.
This Scrabble game revisited must be accessible to every people who have a computer. It must be easy-to-use and fun. It makes it possible for two player to do battle with a same computers screen.
The application have been developped with Java langage.

Keywords : Application, Java, Scrabble, Game