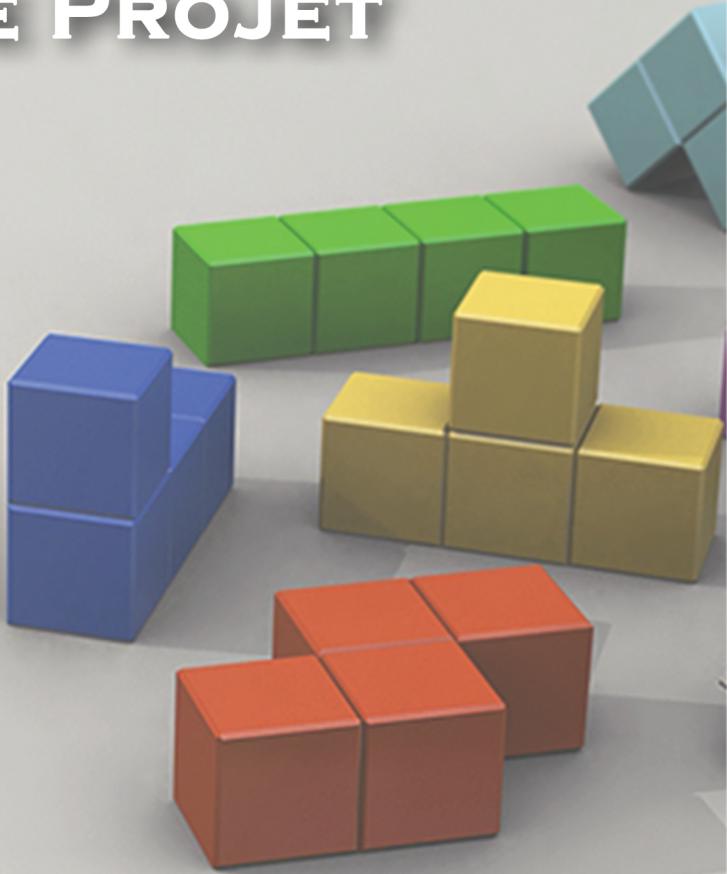


TETRA WORD

RAPPORT DE PROJET

Groupe

Bruno Crönert
Jérôme Fillette
Marie Huertas



SOMMAIRE

1. Introduction	2
2. Présentation de l'application	2
2.1. Résumé.....	2
2.2. Diagramme de classes.....	3
3. Architecture de l'application	4
3.1. Le projet	4
3.2. Les packages.....	4
4. Les différents modes de jeu	5
4.1. Le mode Tetris.....	5
4.2. Le mode Anagramme	8
4.3. Le mode Worddle.....	9
4.4. Le jeu principal	9
4.5. Les modificateurs	13
4.6. Le mode Multijoueur	14
4.6.1. Contre un ami	14
4.6.2. Contre l'ordinateur (Intelligence Artificielle).....	14
4.7. Les Options.....	16
4.7.1. Options de base	16
4.7.2. Éditeur de pièces.....	17
5. Difficultés rencontrées.....	18
6. Conclusion	18

1. Introduction

Débutants en Java pour certains, un peu plus connaisseurs pour d'autres, le Tetra Word est notre **premier projet de Java** à l'IMAC. Ce projet nous a permis de mettre en application les connaissances que nous avons acquises à la fois en cours et en TD mais aussi d'apprendre de **nouvelles techniques de programmation**.

Pour réaliser ce projet, nous étions un groupe de trois étudiants en deuxième année à l'IMAC :

- **Bruno Crönert** : chef de projet, développeur
- **Jérôme Fillette** : développeur
- **Marie Huertas** : développeur

2. Présentation de l'application

2.1. Résumé

Le **Tetra Word**, comme son nom peut l'indiquer est une **extension du célèbre jeu Tetris**. Comme son ancêtre, le but principal du jeu est de **placer correctement des pièces** qui tombent afin de faire des lignes et ainsi se débarrasser des briques de la grille. Mais la différence dans cette nouvelle version, c'est que chaque brique du jeu possède **une lettre de l'alphabet**. Il ne suffit donc plus de faire juste une ligne, mais il faudra en plus **composer un mot** en utilisant les lettres présentes sur la ligne complétée comme ci-dessous :



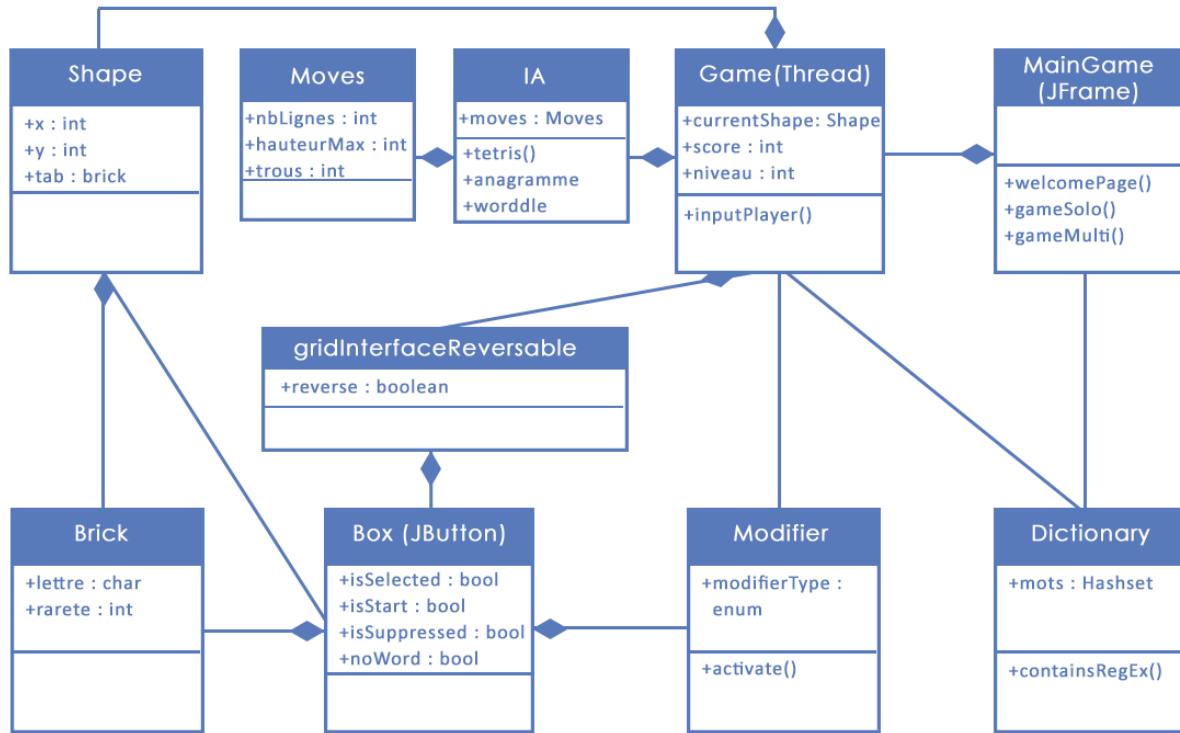
On peut faire le mot "soirée" ou "trouées"

A cela, s'ajoute plusieurs options tels que :

- **le mode "Worddle"** qui permet de se débarrasser de briques en composant un mot qui commence par une lettre choisie aléatoirement par l'ordinateur et qui constitue un chemin avec les lettres des briques adjacentes
- **les modificateurs** qui corsent le jeu en modifiant certains de ses aspects : accélération, explosion, disparition, renversement du jeu, échange de la grille avec l'autre joueur
- **le mode "multijoueur"** qui permet de jouer soit contre un ami soit contre l'ordinateur

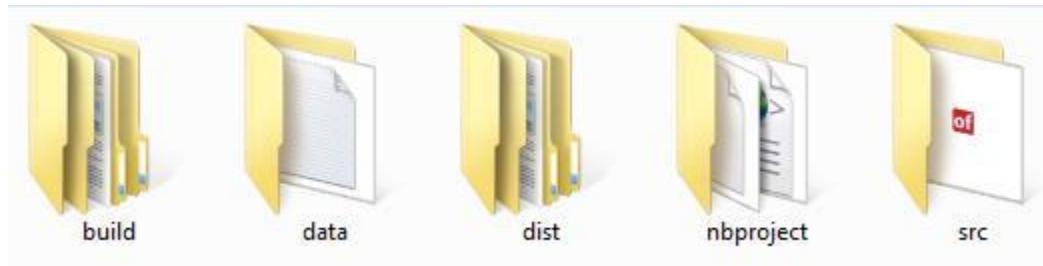
2.2. Diagramme de classes

Nous avons schématisé le **diagramme de classe** du Tetra Word de la manière suivante :



3. Architecture de l'application

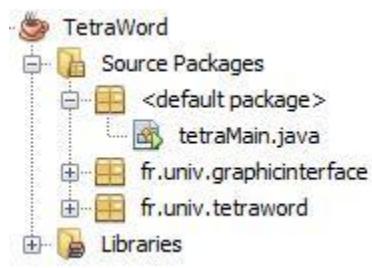
Pour concevoir notre application, nous avons utilisé l'environnement de développement **NetBeans 8.0** et nous avons découpé notre projet de la manière suivante :



3.1. Le projet

build	contient les fichiers compilés (.class)
data	contient les données utiles au jeu (dictionnaire, shapes)
dist	contient la javadoc
src	contient les fichiers sources (.java)

3.2. Les packages

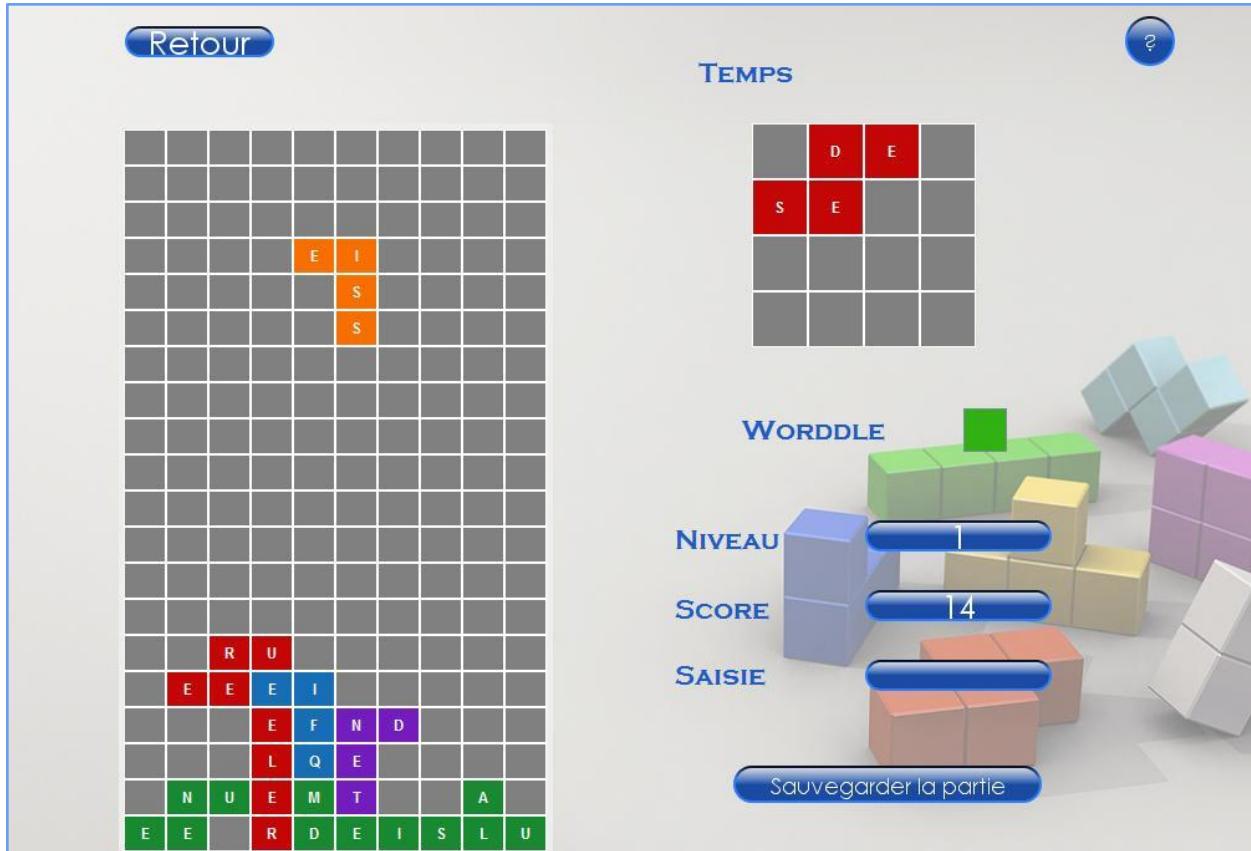


default package	contient le programme principal
fr.univ.graphicinterface	contient les classes liées à l'interface graphique
fr.univ.tetraword	contient les classes liées au jeu

4. Les différents modes de jeu

4.1. Le mode Tetris

En premier lieu, nous avons codé **les éléments de base** du Tetris, à savoir les briques, les pièces, la gestion des mouvements, du score et l'affichage de tout ces éléments.



Les pièces du jeu (Shape.java)

Les attributs d'une Shape sont les suivants :

Brick bricks[][]	Les briques stockées dans un tableau de taille fixe
int x,y	Les coordonnées de la pièce dans la grille
int couleur	La couleur de la pièce
int width	La largeur de la pièce (nombre de cases -1)
int height	La hauteur de la pièce (nombre de cases -1)

Les fonctions essentielles d'une Shape :

rotateShape()	Cette fonction permet la rotation de la pièce L'algorithme consiste à faire tourner les briques qui composent la Shape dans le sens des aiguilles d'une montre
refreshShape()	Cette fonction permet de rafraîchir la pièce Si la première ligne est vide, on remonte tout. Si la première colonne est vide, on décale la pièce à gauche. On met les attributs largeur et hauteur à jour.
getRandomShape()	Cette fonction permet d'obtenir une pièce aléatoire On utilise la fonction random() sur le ShapeType

Les briques d'une pièce (Brick.java)

Citées précédemment, les briques composent une pièce. Elles contiennent :

char lettre	La lettre qui est sur la brique
double rarity	La rareté de cette lettre

Les fonctions essentielles d'une Brick

Brick(char lettre, double rarity)	Constructeur avec paramètres
setBrick(Brick b)	Équivalent d'un constructeur par recopie

Les cases de la grille de jeu (Box.java)

Les cases de la grille, appelées Box, de jeu sont contenus dans un tableau à 2 dimensions. Les Box représentent **le contenant** alors que les briques et les modificateurs sont **le contenu** d'une case de la grille.

Les attributs d'une Box sont les suivants :

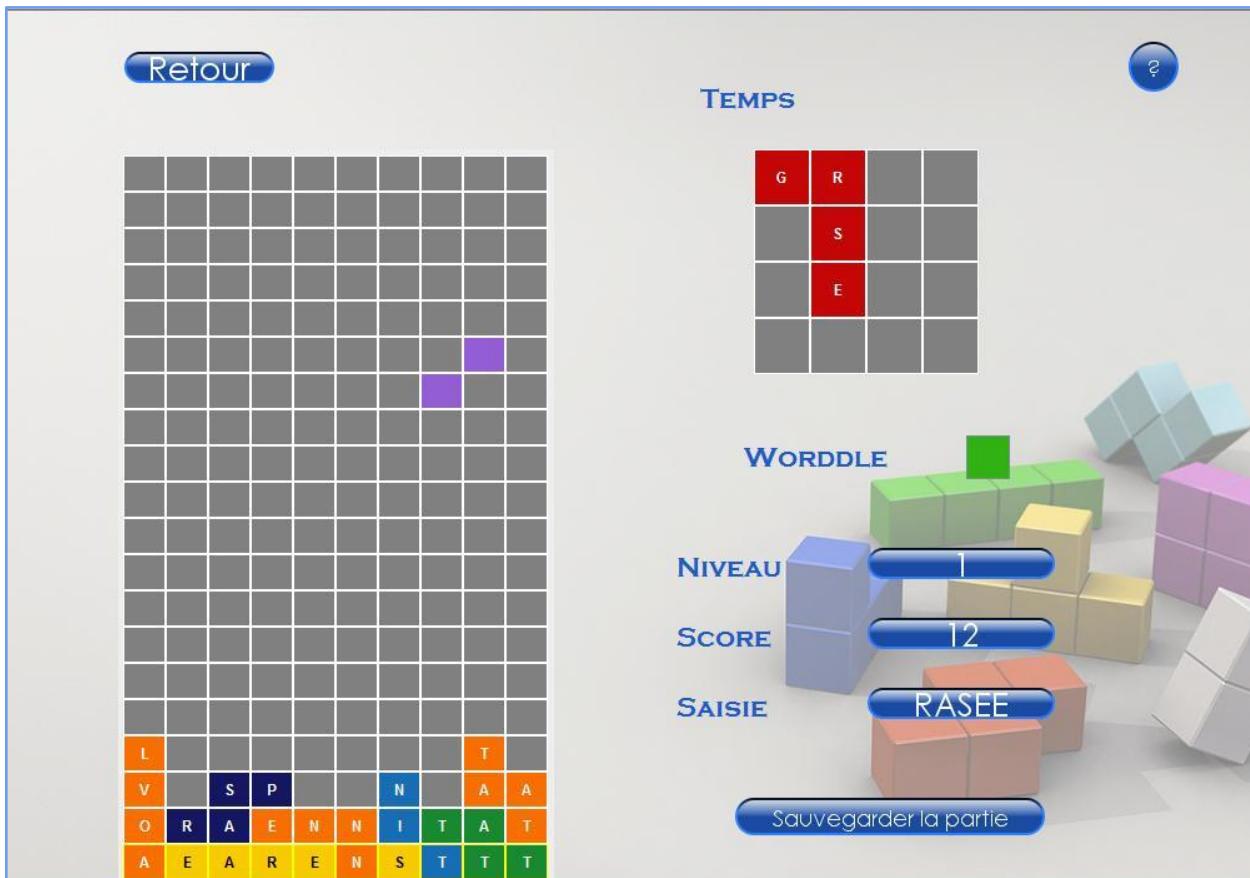
Shape shape	Shape contenue dans la case (s'il y en a une)
Brick brick	Brick contenue dans la case (s'il y a une Shape)
Modifier modifier	Un modificateur (s'il y en a un)
boolean isSelected	Booléen qui indique si la case a été cliquée ou pas
boolean isSuppressed	Booléen qui indique s'il faut supprimer la brique contenue dans la Box, après avoir utilisé le mode Worddle
boolean isStart	Booléen qui indique si on peut commencer de cette brique en mode Worddle
boolean noWord	Booléen , utile juste pour l'intelligence artificielle, qui indique si l'on peut former un mot à partir de cette case ou pas
Color temporaryEffect	Couleur d'un effet temporaire (modificateur), qui ne s'affichera que pendant une seconde

Les fonctions essentielles d'une Box

public Box(Shape shape, Brick brick)	Constructeur avec paramètres
public void rafraichir()	Permet de mettre à jour les composants Swing de la Box
public void boxChange (Box bo)	Permet d'échanger la Box courante avec une autre Box existante

4.2. Le mode Anagramme

Le mode anagramme s'active lorsque l'utilisateur a complété une ligne entière. Il doit alors composer un mot en fonction des **lettres présentes sur la ligne**.



Le dictionnaire (*Dictionary.java*)

Afin de vérifier l'existence des mots composés par l'utilisateur, Jérôme a implémenté la classe “Dictionary”, dictionnaire contenant les mots autorisés.

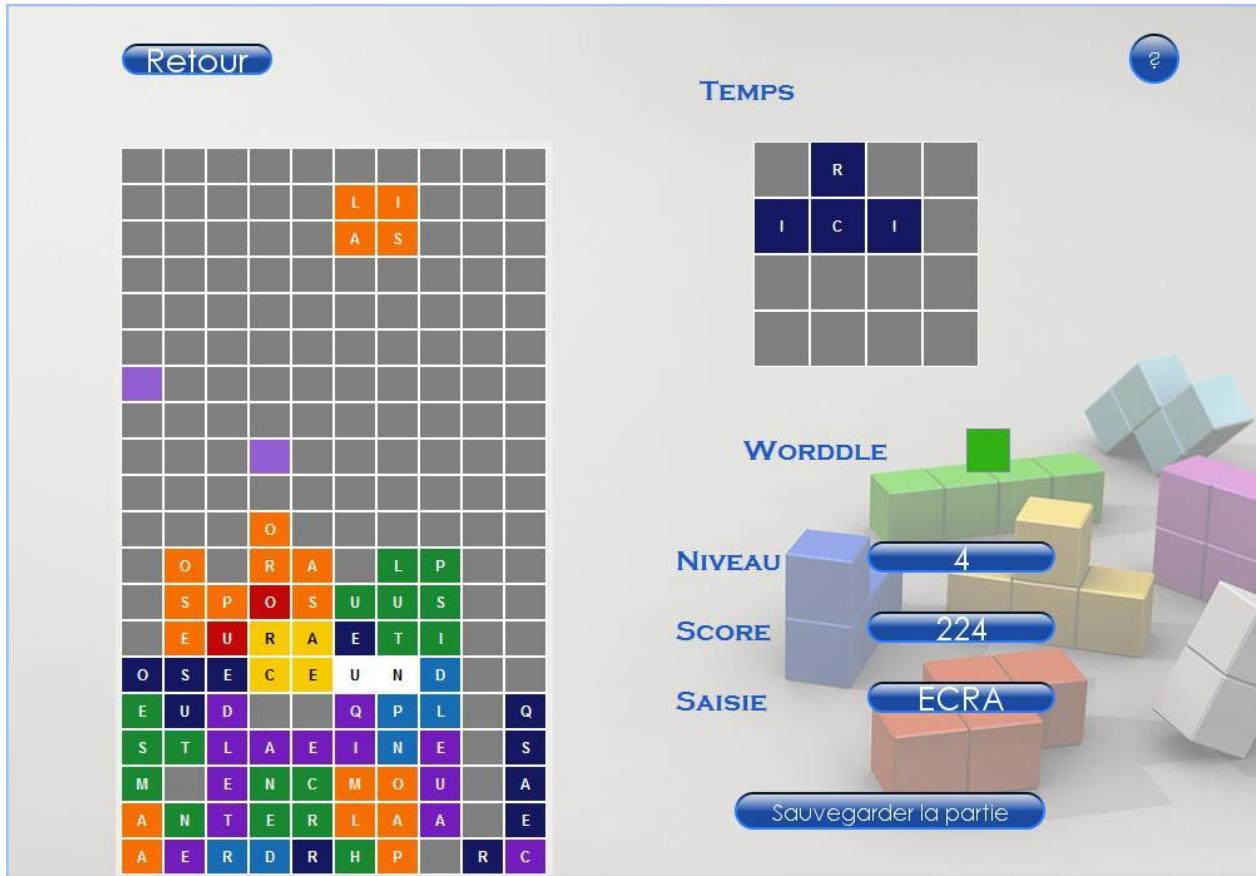
Les fonctions essentielles de Dictionary

Dictionary()	Constructeur par défaut Il ouvre le fichier <i>dictionary.txt</i> et stocke tous les mots dans un HashSet
containsRegEx (String regex)	Cette fonction permet de tester l'existence d'un mot selon une expression régulière

4.3. Le mode Worddle

Le mode Worddle s'active lorsque l'utilisateur clique sur **la touche "w"** ("m" pour le joueur 2) de son clavier. Le programme sélectionne alors une **brique aléatoire** présente sur la grille qui sera la première lettre du mot que le joueur devra composer.

Le joueur devra alors trouver un chemin en utilisant **les lettres des cases adjacentes** pour former un mot. Si le mot que l'utilisateur a composé est valide, celui-ci pourra continuer à composer des mots, en commençant par **une case du mot précédent**.



A la fin du temps imparti, les cases utilisées sont détruites et les pièces situées au-dessus retombent si elles le peuvent.

4.4. Le jeu principal

Classe essentielle pour le bon fonctionnement du jeu, le **Game** qui fait appel aux fonctions et aux classes décrites précédemment.

Comme c'est cette classe qui régit l'ensemble, elle possède de **nombreux attributs** :

Shape currentShape	La pièce qui est en train de tomber et que le joueur doit placer
Shape nextShape	La prochaine pièce qui va tomber
double score	Le score du joueur
int level	Le niveau du jeu
Dictionary dictionary	Le dictionnaire du jeu
Box grid[][]	La grille du jeu principale
Box nextgrid[][]	La grille qui affiche la prochaine pièce
gridInterfaceReversible gridInterface	Le JPanel de la grille du Jeu (qui peut s'inverser)
JPanel nextInterface	Le JPanel de la grille de la prochaine pièce
int mode	Le mode de jeu (Tetris, Anagramme, Worddle)
JFrame window	La fenêtre dans laquelle le jeu sera affiché
String mot	Le mot qui est en train d'être composé
int anagLine	Numéro de la ligne qui est en mode anagramme
Vector<Integer> worddleStartX	Vecteur qui contient les positions en x des cases sur lesquelles on peut commencer le mode Worddle
Vector<Integer> worddleStartY	Vecteur qui contient les positions en y des cases sur lesquelles on peut commencer le mode Worddle
int worddleBoxPosX, worddleBoxPosY	Position de la dernière case sélectionnée en mode Worddle
long worddleTime	Le temps maximal pour composer un mot en mode Worddle
long worddleReload	Le temps de recharge du Worddle
long worddleLast	L'heure à laquelle l'utilisateur a fini le dernier Worddle
long anagTime	Le temps maximal pour composer un mot en mode Anagramme
long fallTime	Le temps de chute d'une pièce du jeu
int anagLettres	Le nombre minimum de lettres que doit contenir un mot

<code>HashMap<String, JComponent> composants</code>	Les composants Swing à mettre à jour (score, niveau, saisie, temps)
<code>IA intelligence</code>	L'intelligence artificielle du jeu (s'il y en a une)
<code>boolean pause</code>	Indique si le jeu est en pause ou pas
<code>long beginTime</code>	L'heure de début de l'anagramme
<code>Game other</code>	L'autre jeu qui se joue en parallèle (s'il y en a un)
<code>boolean gameOver</code>	Indique si le jeu est fini ou pas

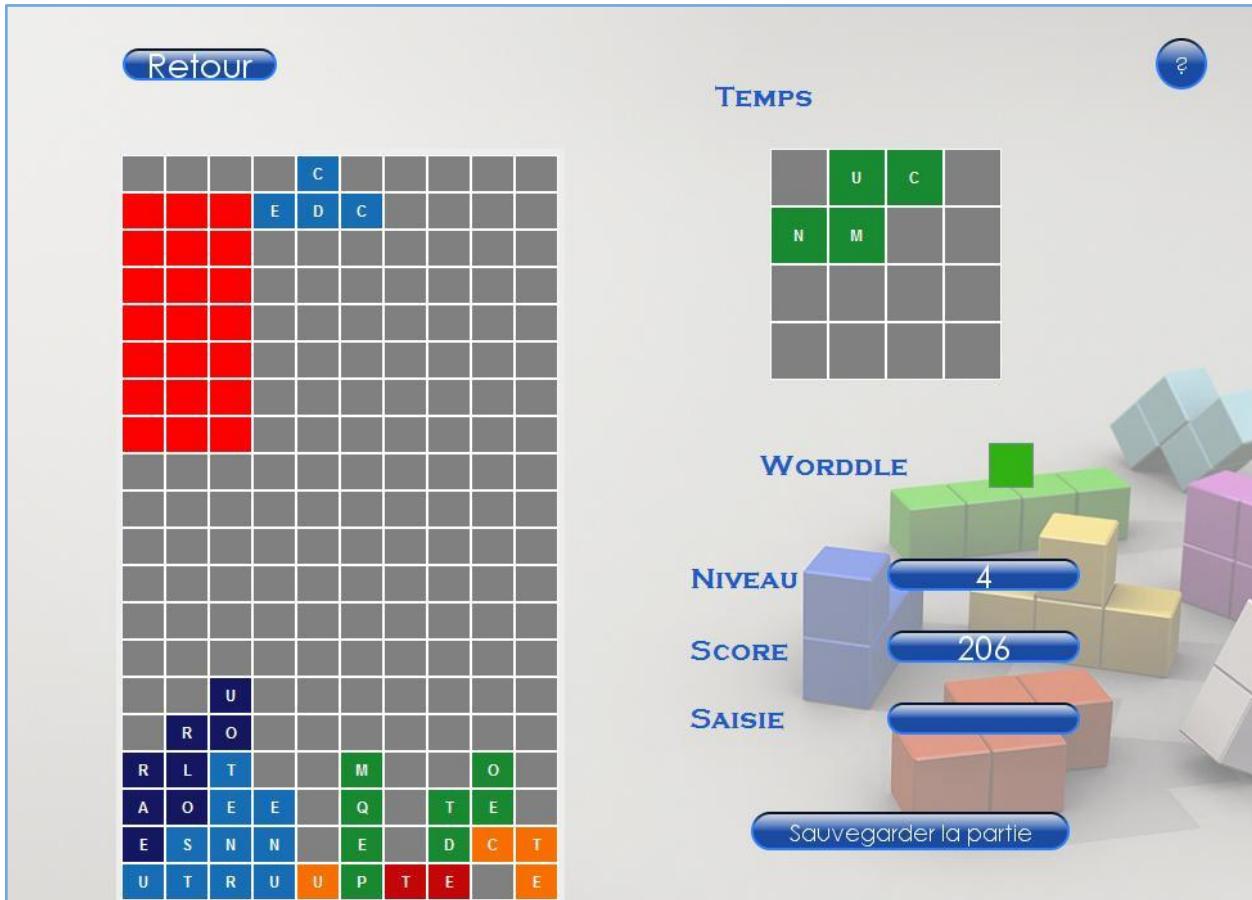
Les fonctions essentielles du Jeu Principal

<code>public Game(JFrame window, Dictionary dictionary, HashMap<String, JComponent> composants, boolean ia)</code>	<p>Constructeur avec paramètres Il initialise tous les attributs cités précédemment</p> <ul style="list-style-type: none"> • temps pour faire un Worddle = 40s • temps de recharge du Worddle = 20s • temps pour faire un anagramme = 30s • temps de chute d'une pièce : 1s
<code>public void init()</code>	<p>Cette fonction permet d'initialiser les composants du jeu En particulier la grille principale et la grille de la prochaine pièce</p>
<code>public void rafraichir()</code>	<p>Cette fonction permet de mettre à jour les composants Swing A savoir les briques, le niveau, le score, le champ de saisie, etc</p>
<code>public void run()</code>	<p>Cette fonction est la boucle principale du jeu Elle initialise le jeu, définit aléatoirement la prochaine pièce, et exécute les différents modes de jeu (Tetris, Anagramme) appelés tant que l'utilisateur n'a pas perdu</p>
<code>public void suppressionWorddle()</code>	<p>Cette fonction supprime les cases qui ont été utilisé en mode Worddle S'il y a des trous, l'algorithme les comble</p>
<code>public void worddle()</code>	<p>Cette fonction active le mode Worddle L'algorithme sélectionne une brique au hasard et vérifie l'existence de briques adjacentes.</p>
<code>public long anagramme(int ligne)</code>	<p>Cette fonction active le mode Anagramme Si un autre joueur est en mode anagramme, il faut attendre que celui-ci est fini pour commencer à composer un mot. Le mode Anagramme est actif lorsque les briques concernées ont un</p>

	contour jaune.
<code>public void validate()</code>	Cette fonction permet de vérifier la validité d'un mot Si celui-ci existe dans le dictionnaire, on supprime les briques sélectionnées, sinon on désactive le mode en cours
<code>public boolean newShapeInGame()</code>	Cette fonction permet d'ajouter une forme dans le jeu Elle créer une forme aléatoire parmi celles existantes et met à jour la prochaine pièce sortante.
<code>public void moveShapeAside(int sens, boolean takeModifier)</code>	Cette fonction permet de bouger la pièce latéralement Elle se déplace à gauche si sens = -1 et droite si sens = 1 Avant de déplacer la pièce, on vérifie que le mouvement est possible grâce à la fonction canMoveAside(int sens)
<code>public boolean canMoveAside(int sens)</code>	Cette fonction vérifie si un mouvement latéral est possible Le mouvement ne sera pas possible s'il y a présence d'une bordure de la grille ou d'une autre pièce.
<code>public boolean canRotate()</code>	Cette fonction teste si la pièce courante peut effectuer une rotation En fonction des pièces qui l'entourent.
<code>public int shapeFall(Shape shape, boolean modifierTake)</code>	Cette fonction permet de faire tomber une pièce Tout en vérifiant la présence de modificateurs ou pas.
<code>public static void saveGame(Vector<Game> savedGame) throws IOException</code>	Cette fonction permet d'enregistrer un ou plusieurs jeux En écrivant toutes les données nécessaires à la réouverture du jeu dans un fichier "game.txt"
<code>public static Game[] readGame()</code>	Cette fonction permet d'ouvrir un jeu existant En récupérant les données présentes dans le fichier "game.txt"
<code>public void actionPerformed(java.awt.event.ActionEvent evt)</code>	Cette fonction permet de stocker les données des briques cliquées par l'utilisateur
<code>public void levelUp()</code>	Cette fonction permet d'augmenter le niveau du jeu Le niveau est incrémenté de 1 et les temps pour composer des mots en mode Worddle ou anagramme sont réduits.

4.5. Les modificateurs

Les modificateurs sont des objets (représentés par des carrés violets) que le joueur peut ramasser avec sa pièce et qui affecte le jeu selon **le type du modificateur ramassé**.



Le modificateur “explode” détruit tout sur son passage

Les types de modificateurs (`modifierType.java`)

L'énumération `modifierType` contient les différents types qu'un modificateur peut prendre :

- **speedFall** : Accélère la vitesse de chute
- **slowFall** : Ralenti la vitesse de chute
- **directFall** : La pièce tombe directement en bas de la grille
- **reverse** : La grille de jeu se retrouve à l'envers
- **switchGrid** : Échange la grille avec l'autre joueur
- **bonus** : Ajoute des points au score
- **malus** : Enlève des points

- **explode** : La pièce en cours explose et détruit les briques autour
- **allowWorddle** : Enlève le temps de recharge du mode Worddle
- **reverseOther** : inverse la grille de l'adversaire

Les Modificateurs (*Modifier.java*)

Les modificateurs apparaissent dans la grille aléatoirement **toutes les 3 pièces**. Ils sont représentés par un **carré violet**, quel que soit le type. On ne peut donc pas connaître à l'avance l'effet du modificateur. Certains modificateurs ne peuvent apparaître qu'en mode multijoueur, à savoir **switchGrid** et **reverseOther**.

L'utilisateur peut ramasser le modificateur avec la pièce qu'il dirige, et l'effet du modificateur se produira lorsque celui-ci sera ramassé.

4.6. Le mode Multijoueur

En mode multijoueur, l'utilisateur a la possibilité de jouer en simultané **contre un ami** ou **contre son ordinateur**. Le jeu comporte alors deux grilles côté à côté avec à gauche celle du joueur 1 et à droite celle de son adversaire.

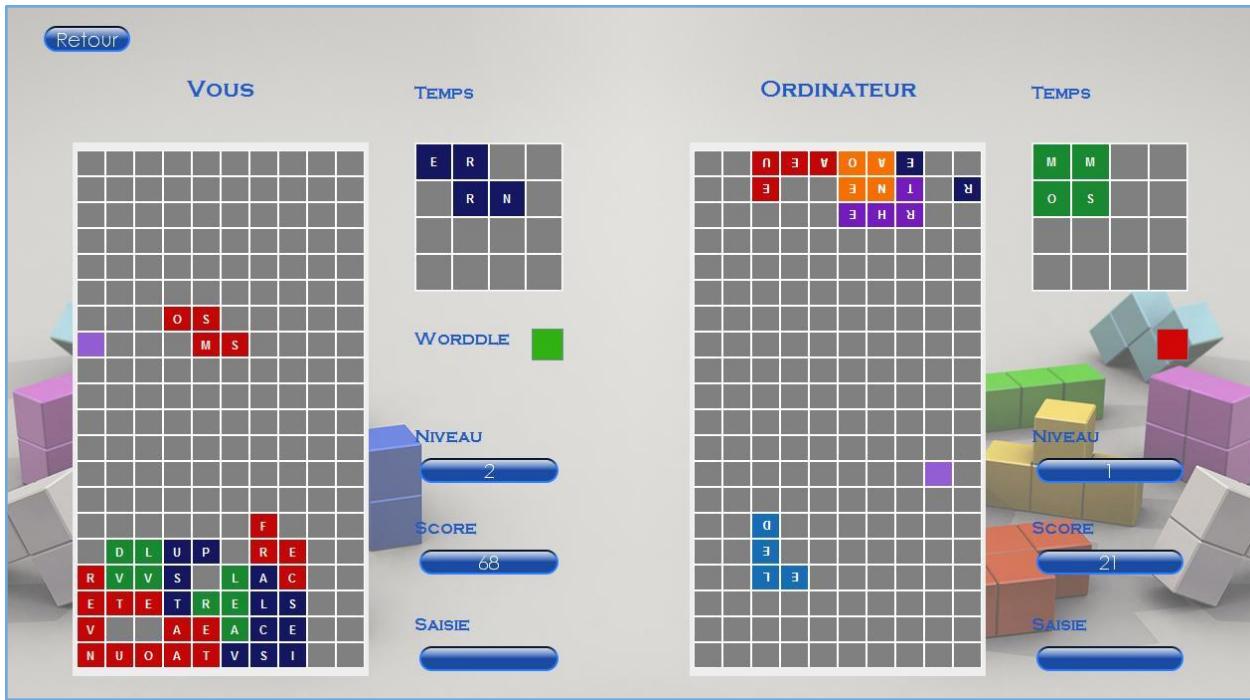
4.6.1. Contre un ami

Chaque joueur a des touches prédéfinies (Z, Q, S, D pour le joueur 1 et flèches directionnelles pour le joueur 2) et jouent au Tetra Word **en parallèle**. Ils peuvent ramasser des **modificateurs** pour gêner leur adversaire.

4.6.2. Contre l'ordinateur (Intelligence Artificielle)

Dans ce mode le joueur doit affronter **une intelligence artificielle** qui joue au Tetra Word. L'intelligence artificielle regroupe les algorithmes qui doivent permettre à l'ordinateur de jouer au Tetra Word tout seul. Ces algorithmes doivent répondre à 3 types de jeu différents :

- le mode anagramme
- le mode Worddle
- le mode Tetris



En mode Anagramme

En mode Anagramme, l'intelligence artificielle recherche **toutes les possibilités de mots** jusqu'à en trouver un existant.

En mode Worddle

En mode Worddle, l'intelligence artificielle teste **tous les chemins possibles** à partir de chaque case de départ Worddle. Il sort du mode Worddle lorsque toutes les cases de départ ne peuvent plus fournir de mots.

En mode Tetris

En mode Tetris, l'intelligence artificielle teste tous les chemins possibles qu'une pièce peut suivre dans la grille et choisit **le meilleur des mouvements**.

Les mouvements de l'IA (Moves.java)

Les mouvements sont les différentes actions que l'intelligence artificielle doit réaliser pour **arriver à la position finale**. Ils sont caractérisés par le nombre de rotations, et par son déplacement latéral.

De plus, pour savoir si un mouvement est intéressant à faire, on a associé à cette classe 3 attributs :

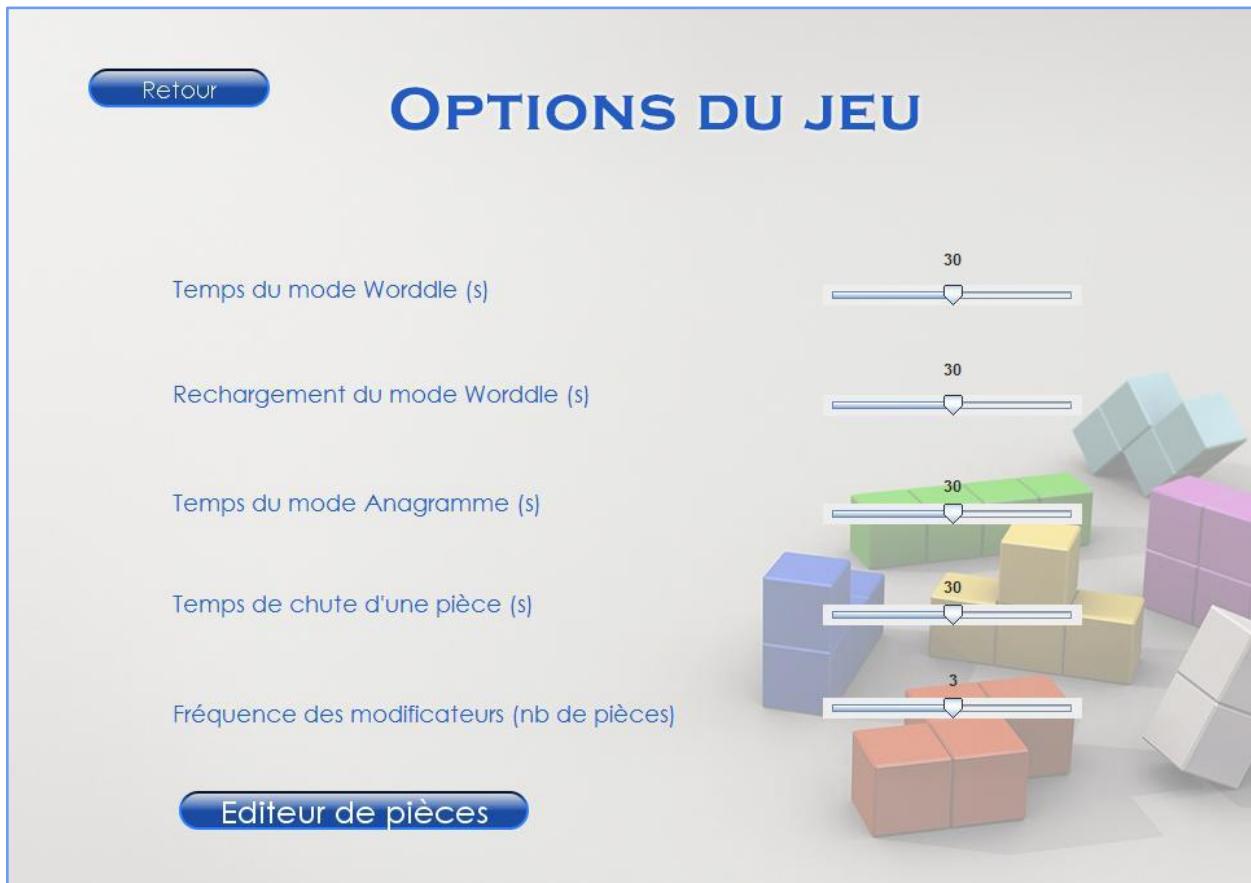
- **le nombre de lignes Tetris** que ce mouvement réalise
- **la hauteur maximale** des pièces après avoir réalisé ce mouvement
- **le nombres de trous** inatteignables laissés par la position de la nouvelle pièce

En ordre de priorité, l'intelligence artificielle choisit d'abord le mouvement qui fait le plus de lignes, puis celle qui fait le moins de trous possible et enfin celle qui monte le moins haut.

4.7. Les Options

4.7.1. Options de base

Afin que l'utilisateur puisse avoir une **marge de contrôle** sur les durées pour composer un mot ou la fréquence d'apparition des modificateurs, nous avons rajouté une fenêtre d'options :

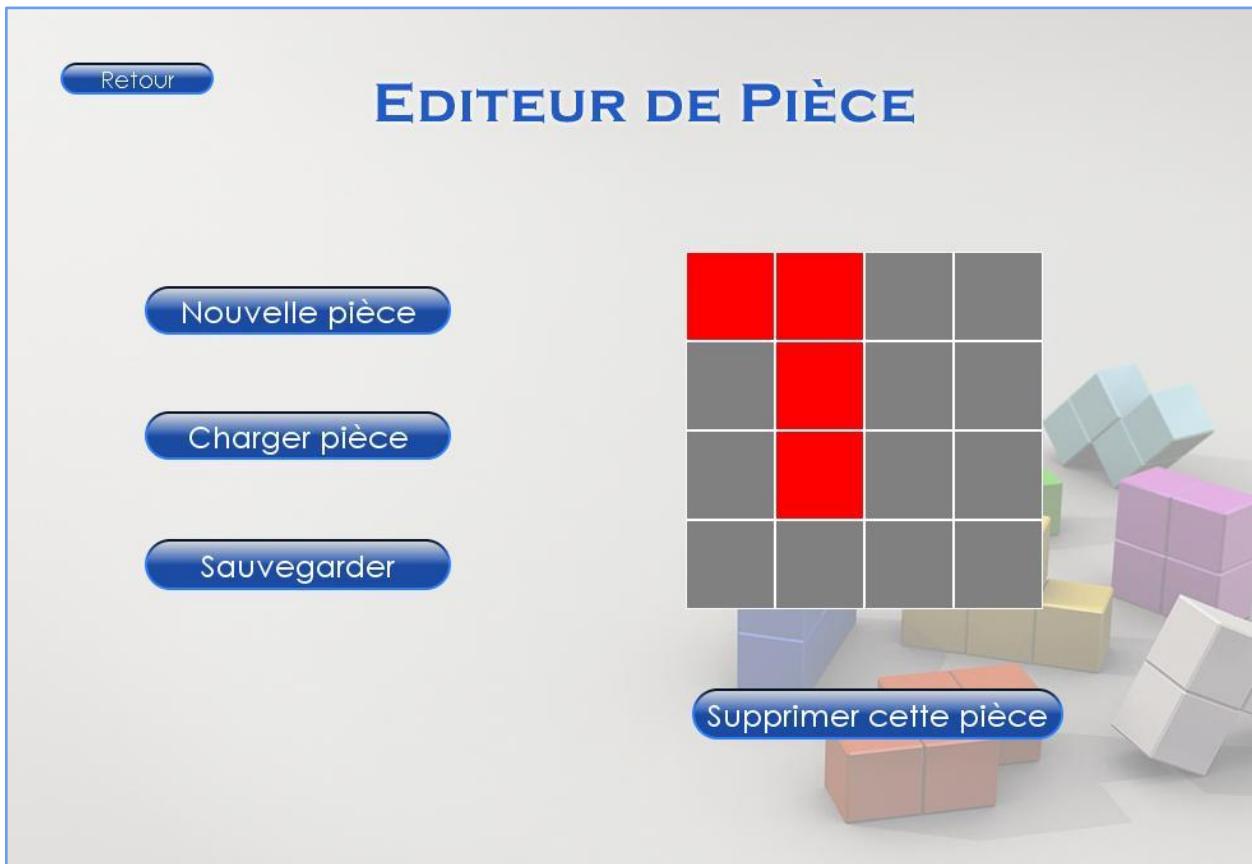


Ainsi, le joueur peut modifier :

- **le temps pour composer** un mot en mode Worddle
- **le temps de rechargement** du mode Worddle
- **le temps pour composer** un mot en mode Anagramme
- **le temps de chute** d'une pièce
- **la fréquence d'apparition** des modificateurs

4.7.2. Éditeur de pièces

Nous avons également intégré un **éditeur de pièces** afin que l'utilisateur puisse en créer de nouvelles ou en supprimer (excepté les formes de base) :



Dans cet éditeur, l'utilisateur peut :

- **créer** une nouvelle pièce
- **charger** une pièce existante et la modifier
- **sauvegarder** la pièce qu'il vient de créer
- **supprimer** une pièce (sauf les pièces de base du Tetris)

5. Difficultés rencontrées

Pour ce projet nous n'avons pas rencontré de difficultés particulières, si ce n'est en ce qui concerne **l'architecture globale du projet** ainsi que **l'intelligence artificielle**.

En effet, au début nous avions créé de **multiples classes** qui n'était pas forcément utiles au bon fonctionnement du programme.

Nous avons donc réfléchi ensemble à une architecture plus simple et logique que la précédente.

Pour l'intelligence artificielle nous nous sommes posés plusieurs questions, à savoir :

- comment dire à l'ordinateur de prendre un bon chemin ?
- comment placer la pièce ?

Nous avons recherché sur Internet quelle était la meilleure solution et nous avons finalement opté pour un **algorithme naïf** qui teste toutes les possibilités de chemin que la pièce pouvait prendre puis choisit le chemin optimale.

6. Conclusion

Ce projet nous a permis de mettre en application nos connaissances aussi bien en programmation **Java** qu'en **algorithmique avancée**. En effet, non seulement nous avons mis en pratique des notions déjà vues lors des TD mais nous avons également appris de nouvelles notions telles que la **création d'interface en Swing**, la réalisation d'une **intelligence artificielle**, la sauvegarde d'objets, la **création d'un exécutable**.

En plus de cela, il nous a permis de renforcer notre capacité à gérer un **projet conséquent** dans un temps imparti. Nous sommes fiers d'avoir réalisé une extension du jeu auquel nous avions tous **joués étant petits** et ne manqueront pas d'y rejouer à nos heures perdues.