# **Description du projet - 1**

La spécificité du Master MIMO – certains diront sa force – réside dans son inclination à accueillir des étudiants en voie de réorientation. De fait, dans un monde de plus en plus spécialisé où se réveille peu à peu la conscience d’une nécessité à l’appréhender par des approches pluridisciplinaires, ce Master a fait le choix de croiser les compétences. Au sein de notre binôme, nous partageons par le prisme de notre formation initiale notre volonté première, qui était d’enseigner, de transmettre, partager. Notre connaissance fine du monde de l’Éducation nous permet d’en cibler les points perfectibles. Notre réorientation nous permet aujourd’hui, à travers la programmation en JAVA, de tâcher d’y répondre, en développant des outils pédagogiques. Ainsi, le projet JAVA constituait l’occasion parfaite de proposer un « jeu sérieux », pouvant être utilisé par des élèves en fonction des objectifs établis par l’enseignant. Nous pourrions l’imaginer en ouverture d’un cours, pour permettre une mise en route en mobilisant les connaissances et les compétences des élèves. Nous pourrions aussi l’imaginer manier par des utilisateurs quelconques, des honnêtes gens comme vous et moi, en quête de savoir, pour le plaisir d’apprendre.

Avant de décrire le projet JAVA, expliquons ce qu’est un jeu sérieux. La définition proposée par le Centre de Ressource et d’Informations sur le Multimédia pour l’Enseignement Supérieur (CERIMES) indique qu’il s’agit *«****d’applications développées*** *à partir des* ***technologies avancées du jeu vidéo****, faisant appel aux mêmes approches de design et savoir-faire que le jeu classique mais qui* ***dépassent la seule dimension du divertissement****».* Deux types d’activités en alternance ou en association se rejoignent : des phases ludiques et des moments d’apprentissage. Les thématiques peuvent être variées.

Dans notre cas, il s’agirait d’un jeu sérieux sur le thème de la géographie. Notre jeu, nous l’avons nommé Francle, et il tire son inspiration du jeu sérieux Worldle.

Francle propose la carte d’une région Française. L’objectif pour l’utilisateur est de trouver en 5 tentatives à partir de la carte, de quelle région il s’agit. Après chaque tentative échouée, des indices telles que la distance, un pourcentage e proximité et des spécificités de la région en question seront donnés. L’envie de recommencer une nouvelle carte après avoir nommé la précédente trouvera sa source dans la propension de l’utilisateur à rechercher le plaisir de la réussite, dans la satisfaction de connaissances tantôt acquises tantôt consolidées, ainsi que dans la prévision de la compétence globale acquise en terminant le jeu. Nous avons conscience que la méconnaissance totale des cartes régionales peut mener à une frustration de l’utilisateur, délétère à son apprentissage si elle le mène à quitter le jeu. Cet épilogue regrettable découlerait d’une curiosité insuffisante de l’utilisateur, qui à défaut de le mener à des recherches en vue d’obtenir la réponse, induirait l’abandon. C’est aussi pour cela que nous avons pensé ce jeu principalement pour une utilisation pédagogique.

Nous avons fait le choix de représenter dans un premier temps les régions dans leur configuration géographique depuis la réforme de 2016. C’est seulement dans un deuxième temps que l’utilisateur pourra deviner le nom des anciennes régions, plus nombreuses.

Nous avons utilisé Github pour réaliser ce projet. Il s’agit d’un service de Cloud Computing permettant aux développeurs de stocker et gérer l’évolution du code. Cet outil illustre la puissance du Cloud pour travailler à plusieurs sur un même projet en passant outre la limitante géographique – chacun peut contribuer au projet d’où qu’il soit à condition qu’il ait un accès à internet. Dans notre cas, Github nous a permis de travailler sur le fichier que nous avons nommé *main.Java*, dans lequel nous avons pu, par nos efforts combinés, suivre le code, le modifier, en observer l’avancement au travers les mises à jour du fichier. Ainsi, nous nous sommes familiarisé progressivement avec ce nouvel outil, et ce n’est que sur la fin du projet que nous nous sommes rendu compte qu’il y avait des fonctionnalités plus agiles telles que « pull request » et « forks » ; permettant de repérer exactement les changements dans le code. Nous pourrons optimiser notre temps en utilisant ces fonctionnalités sur un futur projet.

Le lien ci-dessous permet d’accéder au GitHub sur lequel nous avons travaillé. Il présente différents formats de documentation, les plus intéressants étant les suivants :

* L’ébauche du code : expliquant dans les grands axes l’algorithme utilisé
* La totalité du code : main.java
* Un dossier contenant les images au format png de chaque région
* Un document contenant les indices cités pour chaque région

https://github.com/deIbine/francle

# **Notice d’installation – comment faire fonctionner le jeu – 1**

Il n’y a pas d’installation spécifique à prendre en compte pour ce jeu. Lorsque l’utilisateur accède à Francle, un menu s’affiche. Il devra choisir le numéro 1 pour accéder au jeu, dans ce cas, il se trouve devant une carte de la région de France (les autres fonctionnalités du menu sont expliquées ci-dessous dans la partie 6).

Un message lui indique qu’il doit écrire le nom de la région correspondant, si le nom indiqué est incorrect, un message d’erreur apparaît. Sinon, plusieurs cas de figure se présentent :

* Si la réponse est fausse, un message lui indique qu’il peut recommencer. Il a alors plusieurs indices :
  + La distance, qui est calculée en fonction de la latitude et la longitude des deux préfectures de la région. Plus l’utilisateur est proche de la réponse, plus la distance sera courte. Elle est indiquée en kilomètre.
  + Un pourcentage permettant d’indiquer la proximité en fonction de la distance de la région choisi par l’utilisateur et la région cible. Plus l’utilisateur est proche de la réponse, plus le pourcentage est élevé.
  + Des indices supplémentaires sont apportés seulement pour les 3 dernières tentatives :
    - 3ème tentative : Il s’agit de la spécialité culinaire
    - 4ème tentative : La capitale régionale
    - 5ème tentative : Personnalité provenant de la région
* Si au bout de 5 tentatives, la réponse est toujours fausse, un message lui indiquant qu’il n’y a pas la possibilité d’effectuer une supposition supplémentaire apparaît en plus de la réponse.
* Si la réponse est correcte, en plus du message le félicitant, les messages suivants apparaissent :
  + La distance est de 0km
  + Le pourcentage indique 100%

# **Choix des librairies – 1**

Pour le codage de Francle, nous utilisons la librairie standard appelé JAVA SE (Standard Edition). Cette librairie est organisée en packages dans lesquelles on trouve des classes souvent utilisées dans ce langage de programmation telles que « *java.lang* », qui contient des classes de bases comme String, System, etc... ou par exemple « java.math » permettant d’effectuer des calculs.

L’utilisation de ces packages se fait à travers la déclaration d’import au début du programme, à l’extérieur de toutes les classes.

Dans notre cas, nous utilisons les packages suivants :

* ***Java.lang.Math*** va permettre l’utilisation de fonctions mathématiques au sein du programme. Nous l’utilisons notamment à travers les fonctions trigonométriques.
* ***Java.util.Scanner*** est un outil permettant la simplification d’expression utilisée régulièrement dans le programme. Dans notre code, on associe le scanner à la méthode *nextInt,* pour extraire un nombre entier. On l’associe également à la méthode *nextLine* pour lire la ligne du texte jusqu’au prochain retour à la ligne.
* ***Javax.swing.\****et***Java.awt.\**** : sont des outils semblables permettant la création d’interface graphique. Ils peuvent servir pour faire apparaître la fenêtre par exemple.
* ***Java.awt.image.BufferedImage*** joue un rôle dans le traitement des images. Dans notre cas, il s’agit des différentes cartes régionales. Cet outil va gérer l’image qui se situe dans la mémoire, fournir des méthodes de stockage et permettre l’obtention des données en pixels.
* ***Java.io.File***permet la manipulation du fichier. Aussi, c’est un outil qui peut représenter un chemin sur le système.
* ***Java.io.IoException***comme son nom l’indique, il s’agit d’exception d’entrée et de sortie. Elle se produit lorsqu’une opération d’entrée ou de sortie échoue. Lorsqu’un fichier n’est pas reconnu lors de sa lecture une exception d’E/S est alors lancé.
* ***Java.util.Objects*** permet d’opérer sur des objets**:** pour calculer le code de hachage d’un objet, comparer 2 objets ou encore renvoyer une chaîne de caractères pour un objet.
* ***Java.awt.image.ImageObserver*** et ***Java.awt.image.ImageProducer*** : permettent le suivi de la procédure de chargement de l’image. Ils permettent de manipuler l’état de chargement de l’image en fonction de la demande.

# **Structures de données – 2**

Rappelons que Francle est un jeu dont le but pour l’utilisateur est de reconnaître une image, plus précisément la région de France à laquelle correspond la carte présentée. il a donc fallu inclure des images dans le code.

Pour permettre cela, les images ont dû être conçues. Nous avons dans un premier temps récupéré les images du site « France.comersis.com » qui propose des cartes libres de droit de la France. Dans un second temps, chaque région de la carte a été colorée et séparée à partir d’un logiciel de traitement d’image. Enfin, nous avons exporté les images au format PNG (Portable Network Graphics). Nous avons utilisé ce format pour des questions de qualité d’image. Effectivement, contrairement au format JPEG, l’image en PNG n’est pas compressée.

Même si cela peut paraître évident, notons que Francle inclut les régions d’outre-mer, trop souvent oubliées des cartes de France.

Expliquons maintenant la partie codage sous-jacente à la présentation des différentes cartes régionales à l’utilisateur.

Premièrement, focalisons notre attention sur la « *class* *representation* » :

* On se trouve devant une première méthode qui permet de charger l’image, notamment grâce à « *BufferdImage* » qui va créer une variable image. C’est cette partie du code qui va être utilisée pour faire la représentation d’image avec notre programme.
* On détermine ensuite une variable « *private* » image.
* On se trouve ensuite devant une autre méthode dans laquelle on prend la région qui a été choisie aléatoirement. Prenons l’exemple suivante. Si l’image qui apparaît s’agit de la Corse, alors le code permet de chercher dans le chemin d’accès l’image correspondante à cette région.

Ensuite, dans la « *class* *main* », plus précisément dans la partie du « *case 1* », nous créons une représentation qui va permettre d’accéder à l’interface graphique et ainsi dévoiler l’image de la région correspondante.

# **Manuel utilisateur – tout ce qu’on peut faire dans le jeu – 1**

Nous avons fait le choix d’une interface graphique simple, intuitive et ergonomique pour l’utilisateur. Comme indiqué dans la partie 2 de ce document, sur l’écran, l’utilisateur se trouve devant un menu avec plusieurs possibilités :

* **0** pour quitter le jeu.
* **1** pour commencer le jeu. Le joueur se trouve devant la carte de la région à deviner.
* **2** pour configurer les paramètres selon ses envies :
  + Il peut paramétrer **les types de régions** : choisir entre les régions avant ou après la réforme, mais aussi toutes les régions confondues (avant et après la réforme de 2016).
  + Il peut **modifier la difficulté du jeu** en choisissant par exemple pour les critères de difficulté de cacher l’image de la carte, pour les critères de simplicité d’afficher une liste indiquant le nom des régions.

**Reste à faire :**

* **Algorithme utilisé : UML – Explication globale de l’algo**
* **Partie Manuel utilisateur : ajouter une partie sur «  les avancés du projet »**
* **Documentation du code : Clément**