**Rapport de projet**

**de programmation**

PI4 - 2022/2023

ALSA ALIZE

Groupe SB1c

Membres :

* Romain Houard
* Melissa Bouloufa
* Malone Sayarath
* Jules Cotrez

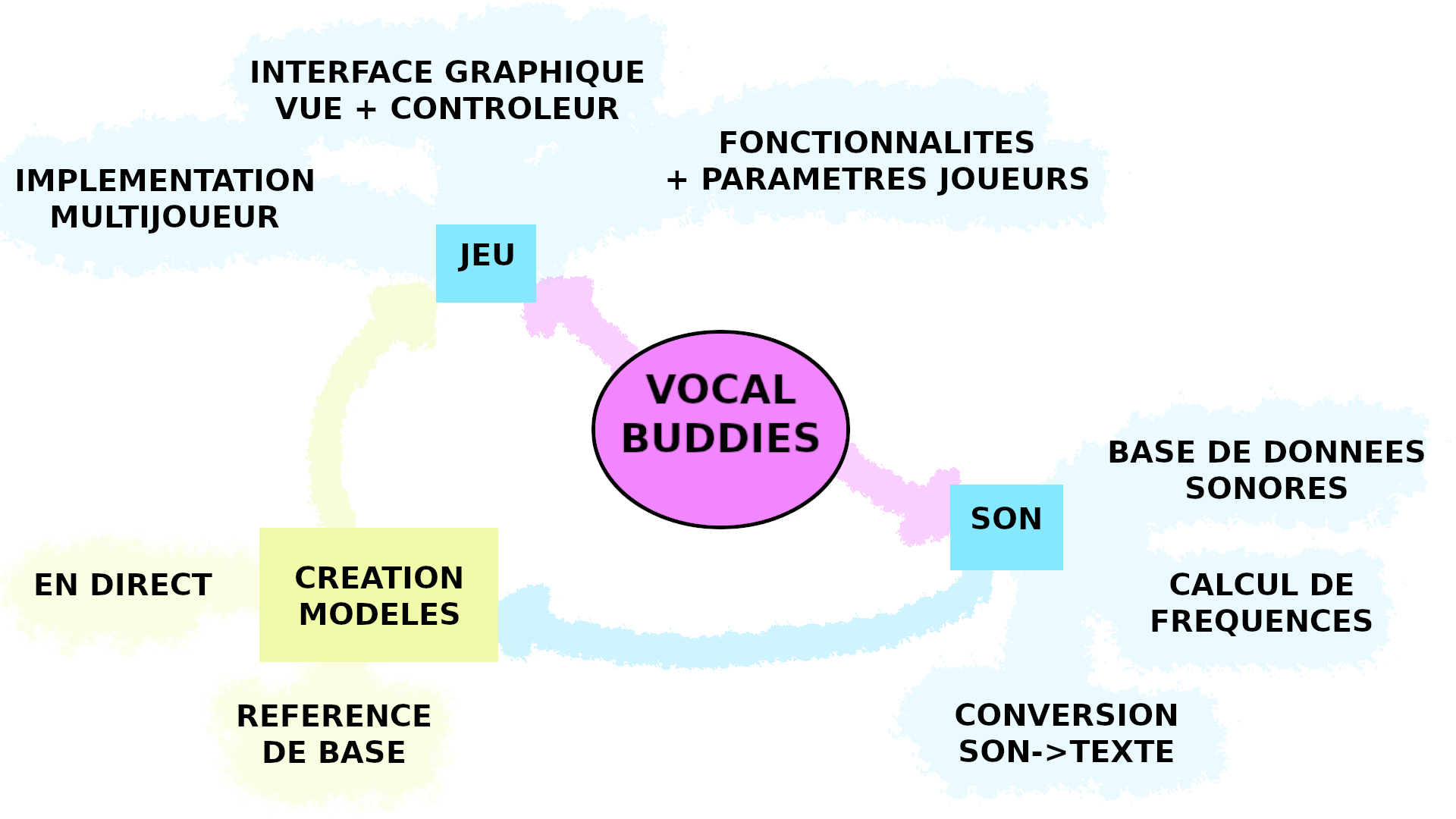


Diagramme représentatif de l’architecture du projet

**GENERALITES :**

1. Objectif :

L’objectif principal de notre projet est de permettre une partie de jeu à un ou plusieurs joueur(s) où le but est de sortir d’un labyrinthe en déplaçant son personnage grâce à la reconnaissance vocale et la reconnaissance du locuteur.

1. Réalisation :

La réalisation de notre projet a nécessité plusieurs étapes que ce soit en premier lieu l’implémentation du labyrinthe, l’implémentation de l’interface graphique, la compréhension du logiciel ALIZE et des modèles de reconnaissance du locuteur ou l’implémentation de celui-ci. On a également réfléchi aux différents outils de retranscription audio et on en a testé quelques-uns tels que ALEXA.

Dans ce rapport/état de l'art, nous présentons les recherches et le travail réalisés pour accomplir ces différentes étapes, ainsi que les résultats obtenus.

**OUTILS VOCAUX UTILISES :**

1. Enregistrement :

* Pour les enregistrements vocaux, nous avons tout d'abord utilisé Alsa, un outil assez simple à exploiter pour créer les différents audios utiles à la création du modèle.
* Pour l'enregistrement dans notre jeu, nous utilisons Java Sound API, car c'est un outil directement inclus dans Java, ce qui facilite grandement les choses.
* Paramètre acoustique utilisé : 16000KHZ 16 bits mono canal.

1. Reconnaissance vocale :

Pour la reconnaissance du locuteur, nous avons décidé d'utiliser le logiciel Alize, car il est le plus adapté pour notre jeu.

Q1 : *C’est quoi la reconnaissance du locuteur ?*

La reconnaissance du locuteur consiste à identifier une personne à partir de ses caractéristiques vocales. Dans notre jeu, nous utilisons la reconnaissance du locuteur pour identifier les joueurs et les ajouter au jeu lorsqu'ils sont bien reconnus par le modèle.

Cependant, il est important de noter que la reconnaissance du locuteur n'est pas infaillible et peut présenter des défauts. Il est possible que le système reconnaisse un locuteur qui ne fait pas partie du modèle, ou au contraire, ne parvienne pas à reconnaître un locuteur qui est pourtant présent dans le modèle.

Ainsi, pour être efficace, un modèle de reconnaissance vocale doit être basé sur un grand nombre d'enregistrements vocaux provenant de différentes personnes. Dans notre cas, nous disposons de 30 minutes d'enregistrement par personne, ce qui peut être suffisant pour notre utilisation mais peux causer un certain taux d’erreur notamment sur les utilisateurs n’appartenant pas au modèle.

Q2 : *C’est quoi ALIZE ?*

ALIZE est un projet de recherche open-source visant à développer des technologies de reconnaissance vocale et de reconnaissance du locuteur. L'objectif principal du projet ALIZE est de fournir des outils pour la reconnaissance automatique de la parole en langage naturel et la reconnaissance du locuteur, avec des applications potentielles dans la sécurité, la biométrie, la transcription automatique de documents, la traduction automatique, et bien d'autres domaines encore. Le projet ALIZE utilise des algorithmes de traitement du signal pour extraire des caractéristiques acoustiques de la parole et des modèles statistiques pour identifier les locuteurs.

Il existe d’autres outils pour la reconnaissance de locuteur tel que LIUM qui lui se concentre sur la reconnaissance du nombre de locuteurs dans un enregistrement vocal.

1. Logiciel SPRO4 :

Afin de récupérer les caractéristiques acoustiques d’un enregistrement, nous utilisons le logiciel SPRO (Signal Processing Tools) qui est un logiciel d'analyse de signal audio développé par l'Institut Fraunhofer pour les circuits intégré (IIS) en Allemagne.

SPro est un ensemble d'outils de traitement du signal audio qui permettent d'extraire les paramètres acoustiques à partir d’enregistrements vocaux. Ces paramètres peuvent inclure des caractéristiques telles que la hauteur de la voix, la durée et la puissance des parties parlées, ainsi que des informations sur l'intonation et la mélodie de la parole.

Ce logiciel va donc nous permettre de récupérer les paramètres acoustiques d’un enregistrement qui seront stocké dans un fichier prm. Ces fichiers vont contenir des informations sur différentes caractéristiques de la voix telles que la fréquence fondamentale (F0), l'énergie et la durée des segments de parole, la fréquence des formants, etc.

Ces paramètres vont ainsi être utiles à la création du modèle. Le modèle va alors apprendre à reconnaître les différentes caractéristiques vocales associées à chaque locuteur, et sera capable d'identifier les locuteurs lorsqu'ils parleront dans notre jeu.

1. Les features :

Pour récupérer les paramètres acoustiques avec Spro nous avons besoin d’entrer les features qui sont les paramètres acoustiques souhaité. Il existe de nombreux paramètres acoustiques que ce soit LFCC (Linear Frequency Cepstral Coefficients) ou MFCC (Mel-Frequency Cepstral Coefficients) dans notre cas nous avons décidé d’utiliser 21LFCC comme paramètre acoustiques avec l’ajout de l’énergie. Elles permettent de capturer des informations sur la puissance du signal vocal c’est à dire l’intensité de la voix du locuteur. Ce qui est utiles dans la reconnaissance du locuteur afin de bien différencier les diverses intensités des voix des locuteurs.

1. Création du modèle :

Pour créer notre modèle à reconnaissance vocale, nous avons dû utiliser les différents outils mis à disposition par ALIZE.

Voici les différentes étapes pour créer un modèle à l’aide d’ALIZE :

1. ***EnergyDetector :*** L'étape de détection d'énergie a pour but de détecter les périodes de silence dans les fichiers audio. Cette étape produit un fichier d'étiquettes (.lbl) indiquant les zones avec signal. L'algorithme utilise trois distributions de Gauss sur le seul paramètre d'énergie.
2. ***Normfeat :*** L'étape de normalisation des paramètres acoustiques, appelée Normfeat, est utilisée pour normaliser les données acoustiques, Les nouveaux fichiers résultants sont enregistrés en format .norm .prm
3. ***TrainWorld* *:*** Durant l'étape de TrainWorld, un modèle de locuteur est créé en utilisant tous les fichiers disponibles dans la base de données. Cette étape permet la création d'un modèle de référence pour la reconnaissance de locuteur.
4. ***TrainTarget :*** Cette étape consiste à créer un modèle pour un utilisateur spécifique à partir du modèle monde créer avec TrainWorld.

Pour tester le modèle sur un locuteur nous avons utilisé la fonctionnalité *ComputeTest*, qui va calculer pour chaque fichier enregistrement sa vraisemblance par rapport à tous les modèles des utilisateurs.

1. Modèle GMM :

Dans notre projet nous utilisons des modèles GMM.

Un modèle GMM (modèles de mélange de distributions de Gauss) est une méthode statistique qui permet de regrouper des données similaires.

Dans notre cas, le modèle GMM va analyser les différentes caractéristiques vocales de chaque locuteur (hauteur de la voix, accents…).

Un modèle GMM est construit en utilisant un certain nombre de gaussiennes, qui sont des fonctions de densité de probabilité de variables aléatoires continues. Le modèle GMM utilise une combinaison linéaire de gaussiennes pour représenter la distribution de probabilité des données. Il faut savoir que le choix du nombre de gaussiennes est très important car celui-ci va déterminer la précision du modèle. Ainsi, si on choisit un nombre de distributions assez faible, cela ne pourra pas être suffisant pour capturer toutes les caractéristiques vocales. De même, si on choisit un nombre trop élevé, cela peut entraîner un effet de sur-apprentissage, c’est-à-dire qu’il y aura trop de données à traiter. Pour notre modèle, nous avons essayé plusieurs nombres de distributions de Gauss et nous avons décidé d'utiliser 128 distributions gaussiennes, ce qui nous permet d'obtenir une plus meilleure précision dans la reconnaissance des locuteurs.

Q :*Comment ça fonctionne ?*

Pour identifier un locuteur, on extrait d'abord ses caractéristiques acoustiques. Ensuite, on calcule la probabilité que ces caractéristiques correspondent à chaque modèle GMM. Le locuteur est alors identifié comme étant celui dont le modèle GMM présente la probabilité la plus élevée.

1. ALIZE :
2. Implémentation :

Avant tout chose, nous avons eu besoin de créer les différents modèles. Pour cela, nous avons commencé par enregistrer chacun notre tour notre voix pendant 30 minutes, afin de collecter un ensemble de voix de référence.

Cet ensemble nous a permis de créer un modèle de base, ou un modèle monde, qui nous sert de référence pour la reconnaissance des locuteurs. Grâce à ce modèle, nous avons ensuite créé des modèles spécifiques pour chaque utilisateur. Ces modèles sont adaptés aux caractéristiques vocales de chacun. Ainsi, chaque utilisateur est censé être reconnu par son modèle.

Nous avons mis en place un système qui permet à l'utilisateur de créer son propre modèle et de l’ajouter pour pouvoir être reconnu par le jeu.

Ensuite, après avoir créé les différents modèles, nous avons implémenté la reconnaissance du locuteur dans notre jeu.

Avant le début de la partie, nous avons implémenté une page de test permettant aux joueurs de vérifier si leur voix est correctement reconnue. Si leur voix est reconnue avec succès, ils sont ajoutés automatiquement au jeu. Sinon, ils doivent relancer un test. Pour qu'un joueur soit considéré comme reconnu, la valeur de reconnaissance la plus élevée doit correspondre à son modèle vocal.

Pour réaliser ces tests, nous avons utilisé la fonctionnalité *computeTest* d'ALIZE expliquée précédemment.

1. Efficacité du modèle :

Afin d'évaluer l'efficacité de notre modèle, nous avons effectué plusieurs tests en enregistrant des échantillons de voix de différentes personnes. Nous avons calculé le taux de réussite en comptant le nombre de tests réussis et en le divisant par le nombre total de tests effectués. Nous avons obtenu un taux de réussite supérieur à 70%. Toutefois, cela n’est pas forcément représentatif de la réalité, car il est possible que même des personnes n'appartenant pas au modèle soient reconnues, et cela peut aussi dépendre de la qualité d’enregistrement du micro.

1. Difficultés rencontrées :

Nous avons rencontré une multitude de problèmes avec ALIZE :

1. Des erreurs lors de la compilation des différentes commandes présentées précédemment (EnergyDetector,etc)
2. Des difficultés à comprendre la documentation d’ALIZE ou bien à trouver des exemples sur internet (aucun)
3. Très peu voire aucune documentation autre que leur site officiel, ce qui nous a fortement ralentis à la compréhension du fonctionnement et de la résolution des erreurs obtenues
4. Des problèmes avec les différents fichiers de configurations

Mais grâce à notre patience et notre persévérance à comprendre comment ALIZE fonctionne et comment résoudre les différents problèmes, ainsi qu’en collaborant en groupe, nous avons finalement réussi à l’implémenter à notre jeu.

1. Reconnaissance vocale (reconnaissance des mots) :

Après avoir implémenté la reconnaissance du locuteur, nous devions implémenter la reconnaissance des mots par la parole afin de déplacer le personnage dans la direction voulue.

Pour cela, nous avons exploré plusieurs solutions :

* L'API Alexa d'Amazon et Google Cloud Speech-to-Text API, qui proposent des fonctionnalités avancées de reconnaissance vocale. Nous ne les avons pas choisies car elles sont assez complexes à implémenter pour notre projet.
* Nous nous sommes également renseignés sur différents outils de retranscription audio, tels que CMU Sphinx, qui était assez simple à utiliser mais peu efficace.
* Nous avons aussi eu l’idée d’utiliser ALIZE en utilisant différents modèles pour chaque direction et pareil que pour les locuteurs. Le modèle le plus reconnu correspond à la direction à laquelle le personnage doit se déplacer.
* En fin de compte, nous avons choisi l'API de Whisper AI pour la reconnaissance des mots dans notre application. C’est également un outil qui retranscrit l’audio en texte.

**WHISPER AI :**

Whisper AI est un système de reconnaissance automatique de la parole (ASR) qui utilise des techniques de deep learning pour transcrire des enregistrements audios en texte. Il a été entraîné sur une immense quantité de données audio enregistrées dans différentes langues à partir de sources diverses sur le web, ce qui lui permet ainsi de reconnaître efficacement les différents accents et prononciation.

Ainsi nous avons décidé d’utiliser whisper à la fois pour son efficacité et son utilisation peu complexe.

Implémentation de WHISPER AI :

Nous avons ajouté une commande afin que l’utilisateur puisse s’enregistrer et prononcer la direction à laquelle il veut déplacer son personnage. Ensuite, nous avons créé un script python utilisant les fonctionnalités de whisper afin d’analyser l’enregistrement et le retranscrire en texte. Puis, avec java, on analyse le fichier texte en vérifiant s’il contient un déplacement qu’on aura indiqué.

**Problèmes rencontrés avec WHISPER :**

* Parfois, whisper a du mal à reconnaître certains mots tels que « haut », « bas » etc. Ainsi, afin de faciliter la reconnaissance des mots, nous avons décidé d’utiliser des phrases pour déplacer le personnage tels que « va en haut. », « va en bas ». Ainsi, le système dispose d'un contexte plus clair pour identifier la commande.
* De plus, l'exécution du script Python utilisé pour analyser l'enregistrement vocal et le retranscrire en texte peut prendre un certain temps, ce qui peut ralentir grandement la durée du jeu. Alors, nous avons essayé de trouver des solutions afin de réduire le temps d’exécution tout en gardant une certaine efficacité dans la reconnaissance des mots telles que : utiliser un model de reconnaissance moins efficace mais plus rapide.

**INTERFACE GRAPHIQUE :**

1. Librairies Java SE :

Il existe plusieurs API en Java pour mettre en œuvre une interface graphique. Parmi elles se trouvent AWT, Swing et JavaFX.

*AWT :*

Introduite dès les premières versions de Java, cette bibliothèque propose tous les éléments nécessaires à l’implémentation d’une interface graphique. Bien que la fonctionnalité recherchée soit garantie, l’inconvénient de AWT reste que Java fait appel au système d'exploitation sous-jacent pour afficher ses composants. Pour cette raison, l'affichage de l’interface utilisateur d’un programme peut diverger et changer d’un système d’exploitation à l’autre.

*SWING :*

Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures. Quelle que soit la plate-forme utilisée, l’application fonctionne et se présente à l’identique. Cependant, une application codée en Swing consomme beaucoup plus de ressources (mémoire et CPU) qu’une application codée en AWT.

*JAVAFX :*

JavaFX est l'API d'interface graphique principale et officielle du langage Java. Dans sa conception, elle est plus moderne que Swing et permet de produire des interfaces graphiques pouvant facilement être utilisées sur différents types d'écrans (écran standard de PC, smartphone & tablettes et applications Web).

1. Bibliothèques utilisées : *Java AWT et Swing*

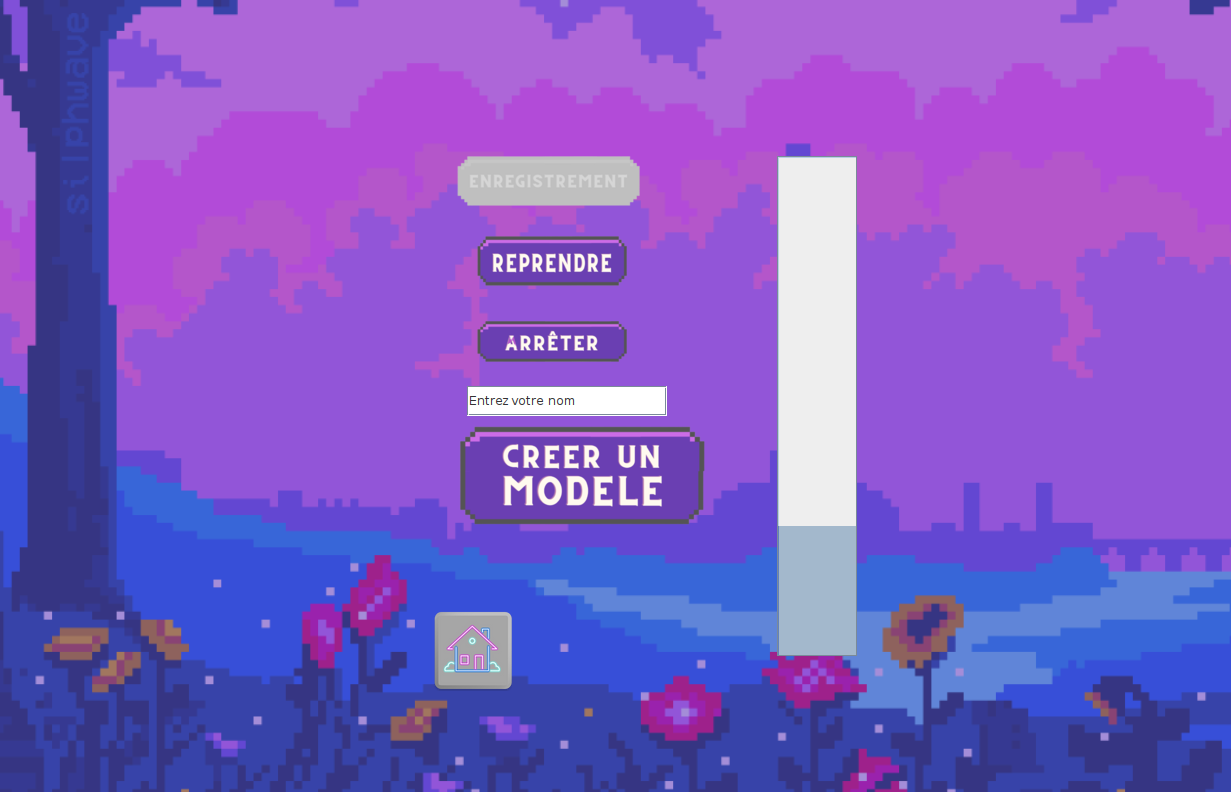
Nous avons opté pour ces deux bibliothèques en raison de leurs multiples avantages, tels que :

* Elles nous sont familières et nous les avons déjà beaucoup utilisées au cours de notre premier semestre
* Elles sont faciles à utiliser et offrent une API intuitive qui permet de créer des interfaces graphiques en quelques lignes de code
* Elles sont multi-plateformes : elles peuvent par exemple être utilisées sur Windows, Linux ou Mac OS
* Elles offrent des composants graphiques variés et hautement personnalisables, ce qui nous permet d’obtenir des résultats très satisfaisants et bien précis
* La bibliothèque Swing nous a été imposée

Voici le résultat du menu du jeu (composé de boutons et de textes) fait entièrement avec Java AWT et Swing (JPanel et JLabel) :



Et ici une page d’enregistrement vocal, où nous avons utilisé une jauge de progression (JProgressBar) et un champ de texte (JTextField) pour entrer le nom de l’utilisateur :

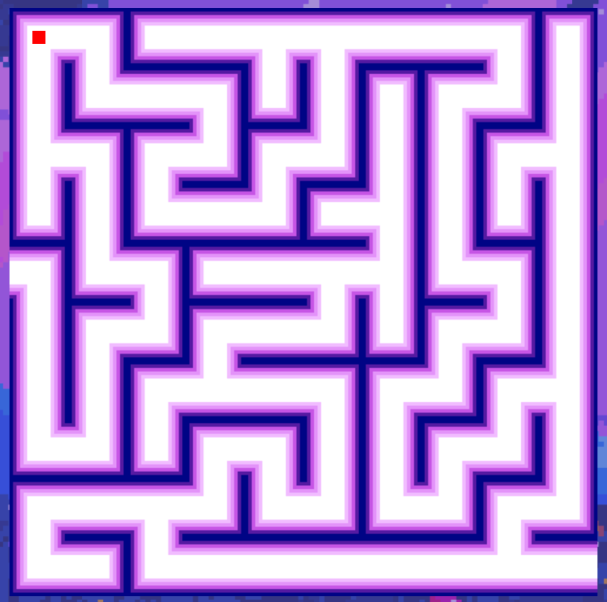


Nous avons également utilisé une liste déroulante (JComboBox) pour le choix de joueurs avant le début d’une partie de jeu.

1. Supports et images sources :

Image de fond : image libre de droit prise sur internet.

Boutons et labyrinthe : dessinés entièrement à l’aide du logiciel Gimp.



Chaque pièce étant comme ceci (exemples) :



Etc. qui une fois rassemblées, forment un labyrinthe complet.

**Fin.**