Lost Voice

Cahier Des Charges

Remerciement :

Nous aimerions exprimer notre profonde gratitude à plusieurs personnes qui ont contribué à la réussite de notre projet.

Tout d'abord, nous tenons à remercier Monsieur Yann Jurski pour ses précieuses directives qui ont guidé notre travail dès le départ. Ses conseils judicieux nous ont permis de rester sur la bonne voie et de mener à bien notre projet.

Un remerciement spécial est également adressé à notre professeur, Sami BOUTAMINE. Votre accompagnement tout au long du parcours a été essentiel. Les connaissances que vous avez partagées et l'aide que vous avez fournie ont été inestimables pour parfaire notre projet. Nous sommes reconnaissants pour l'investissement personnel que vous avez démontré en nous aidant à surmonter les défis que nous avons rencontrés.

Enfin, nous aimerions remercier l'ensemble de nos camarades. L'esprit d'équipe et la collaboration dont nous avons tous fait preuve ont grandement contribué à l'aboutissement de ce projet. Votre soutien, votre motivation et vos idées ont été une source d'inspiration pour nous tous.

Encore une fois, merci à tous pour votre contribution précieuse à la réussite de ce projet. Nous sommes fiers de ce que nous avons accompli ensemble et nous sommes impatients de voir ce que l'avenir nous réserve.

Lost Voice

Sommaire

Lost Voice 

I. Présentation Du Projet

Initialement , notre projet consiste en une application qui permet de faire bouger un personnage dans un labyrinthe via l'identification du genre de locuteurs parlants dans un enregistrement audio , grâce à la bibliothèque Lium SpkDiarization .

En gardant ce sujet principal , nous vous proposons un jeu avec quelques extensions supplémentaires ! Dans lequel un personnage évolue dans un labyrinthe généré aléatoirement via des algorithmes bien spécifiques , les joueurs devront parler , afin de gagner du temps de parole selon leur genre dans le but de faire avancer le personnage avec des instructions données oralement dans un deuxième enregistrement !

II. Description fonctionnelle Des Besoin

I. Fonctionnalités Principales

Des fonctionnalités élémentaires indispensables au fonctionnement de notre application.

1. Enregistrement du son

1.1. Objectif :

Récupérer le flux de donnés audio transmises par les locuteurs et les enregistrer dans un fichier WAV

1.2. Description :

En utilisant Javax.Sound , il est possible de récupérer le flux de données audio via une TargetDataLine .

A l'aide d'un AudioInputStream , il est possible d'écrire ces flux dans un fichier WAV qu'on aura créé . Plus de détails sont dans l'État De L'Art.

1.3. Contraintes :

Il est impossible d'effectuer cette opération dans le Thread actuel , car celà va interrompre l'exécution d'autres tâches en parallèle .

Impossibilité de lancer plusieurs enregistrements en même temps pour cause de nombreux bugs

1.4. Niveau de priorité :

Priorité Haute

Lost Voice

2. Labyrinthe

2.1. Objectif :

Générer aléatoirement un labyrinthe qui sera utilisé pour le modèle.

2.2. Description :

Générer aléatoirement un labyrinthe en fonction d'une taille donnée. Pour ce faire, on utilisera l'algorithme de Prim qui est utilisé en théorie des graphes.

Pour chaque génération de labyrinthe il doit y avoir une entrée ainsi qu'une sortie. Chaque case du labyrinthe représente un mur ou bien un passage.

Par défaut, dans un labyrinthe il y a un joueur. Ce joueur est placé au point départ lorsque le labyrinthe est généré.

Ajouter des méthodes permettant le déplacement du joueur dans le labyrinthe. Ces méthodes doivent vérifier si le déplacement du joueur est valide.

2.3. Contraintes :

Aucune.

2.4. Niveau de priorité :

Priorité Haute

3. Traitement de la voix avec Lium

3.1. Objectif :

Analyser les enregistrements audio et de séparer les différents locuteurs, d'identifier le genre et la bande passante de chaque locuteur et de faciliter la transcription automatique de la parole.

3.2. Description :

LIUM SpkDiarization est un outil développé par le Laboratoire d'Informatique de l'Université du Maine (LIUM) en France. Il utilise des techniques telles que le Critère d'information bayésien (BIC) pour la segmentation et le clustering hiérarchique agglomératif, le décodage Viterbi pour ajuster les limites des segments, et la détection de la parole pour éliminer les éléments non parlés. L'outil détecte également le genre et la bande passante des locuteurs à l'aide de modèles de mélange gaussien

(GMM) et effectue un clustering basé sur GMM pour obtenir une relation un-à-un entre les clusters et les locuteurs.

3.3. ***Contraintes :***

- Format audio : L'audio doit être au format Sphere ou Wave (16 kHz / 16 bits PCM mono) pour être traité par LIUM SpkDiarization.
- Performance : La performance de la diarization dépend de la qualité de l'enregistrement audio et de la complexité des conversations.
- Ressources système : Le traitement nécessite une quantité suffisante de mémoire (par exemple, 2048 Mo pour traiter une émission d'une heure) et une machine équipée de Java.
- Limitations techniques : LIUM SpkDiarization peut ne pas être optimal pour certaines situations spécifiques, comme les conversations avec un bruit de fond important ou un grand nombre de locuteurs.
- Problème sous Windows : Il existe un problème non résolu avec le chargement des ressources (comme les GMM) sous Windows.

3.4. ***Taux de réussite et Précision des paramètres acoustiques***

3.4.1. Les essais et résultats

On note seulement les tests essentiels.

- Test 1 : Au premier essai, on a effectué un test avec 14 personnes dont 6 femmes et 8 hommes
  Moyenne de chaque segment pour Homme : ~7.125s

Moyenne de chaque segment pour Femme : ~7.2s
Pause entre chaque segment (2.30s au début et à la fin 8s) : ~5,65s
Longueur totale du cluster : 3min01

Résultat obtenu : 12 personnes en total dont 5 Hommes et 7 Femmes.

- Test 2 : Cette fois-ci, on a décidé d'augmenter la taille de chaque segment dont 10s. Notre cluster se compose de 11 personnes dont 5 Femmes et 6 Hommes

Moyenne de chaque segment pour Homme : ~10.2s
Moyenne de chaque segment pour Femme : ~10.2s
Pause entre chaque segment (2.30s au début et à la fin 8s) : ~5s
Longueur totale du cluster : 2min47s

Résultat obtenu sans les paramètres 16kHz : 5 personnes en total dont 3 Hommes et 2 Femmes
Résultat obtenu avec les paramètres 16kHz : 8 personnes dont 4 Hommes et 4 Femmes

- Test 3 :

Notre cluster se compose de 2 personnes dont 2 Hommes où il y a une personne qui va parler deux fois (A parle puis B puis A)
Moyenne de chaque segment pour Homme : ~20s
Pause entre chaque segment (2.30s au début et à la fin 8s) : ~5s
Longueur totale du cluster : 1min15s

Résultat obtenu : 2 personnes donc 2 hommes

- Test 4 :

Notre cluster se compose de 4 personnes dont 4 Hommes
Moyenne de chaque segment pour Homme : ~25,5s
Pause entre chaque segment (2.30s au début et à la fin 8s) : ~7s
Longueur totale du cluster : 2min10s

Résultat obtenu : 3 personnes dont 3 hommes

- Test 5 :

Notre cluster se compose de 3 personnes dont 2 Hommes et 1 Femme (1 homme parle 2 fois)
Moyenne de chaque segment pour Homme : ~25s
Moyenne de chaque segment pour Femme: ~28s
Pause entre chaque segment : ~5s
Longueur totale du cluster : 1min42s

Résultat obtenu : 3 personnes dont 2 Hommes et 1 Femme

- Test 6 :

Notre cluster se compose de 5 personnes dont 4 Hommes et 1 Femme
Moyenne de chaque segment pour Homme : ~25s
Moyenne de chaque segment pour Femme: ~28s
Pause entre chaque segment : ~5s
Longueur totale du cluster : 2min41s

Résultat obtenu : 4 personnes dont 3 Hommes et 1 Femme

3.4.2. Taux de réussite

- Test 1 : 12/14 personnes (85.7%)
- Test 2 (sans les paramètres 16kHz) : 5/11 personnes (45.5%)
- Test 2 (avec les paramètres 16kHz) : 8/11 personnes (72.7%)
- Test 3 : 2/2 personnes (100%)
- Test 4 : 3/4 personnes (75%)

- Test 5 : 3/3 personnes (100%)
- Test 6 : 4/5 personnes (80%)

Taux de réussite : ~85,5%

3.4.3. Conclusion et Remarques

Lium spk Diarization a montré une performance variable dans ces tests. Dans certains cas, il a correctement identifié tous les locuteurs, tandis que dans d'autres, il a eu du mal à les différencier. Les paramètres 16kHz ont amélioré les résultats dans le test 2, suggérant que la qualité audio pourrait influencer la performance du modèle.

Remarques :

- Les performances semblent être meilleures lorsque le nombre de locuteurs est plus faible. Cela pourrait être dû à une complexité accrue lorsque le nombre de locuteurs augmente.
- Les performances varient également en fonction de la durée des segments et des pauses entre eux. Il peut être utile d'explorer d'autres durées de segments et de pauses pour optimiser la performance du modèle.

3.5. Niveau de priorité :

Priorité Haute

4. Interface graphique

4.1. Objectif :

Avoir une interface afin que l'utilisateur ait une expérience visuelle et une interaction agréable avec le jeu.

4.2. Description :

Créer une interface visuelle et interactif pour l'utilisateur

- Menu
- Le jeu
- Pause
- et affichages divers

4.3. Contraintes :

Choix entre l'utilisation des JPanel et JFrame et de différentes variables permettant les affichages divers

4.4. Niveau de priorité :

Priorité Haute

5. **MVC**
   - ***5.1. Objectif :***
     Réaliser un Modele-View-Controller afin de faire fonctionner le jeu
   - ***5.2. Description :***
     Suivre le design pattern MVC , en créant trois classes qui feront chacune office de modèle (Labyrinthe) , Vue (Interface graphique) , Controller (Pont entre les deux)
   - ***5.3. Contraintes :***
     Relier chacun de ces trois éléments tout en s'assurant du bon fonctionnement de ces derniers
   - ***5.4. Niveau de priorité :***
     Priorité Moyenne

6. **Menus du jeu**
   - ***6.1. Objectif :***
     Rendre l'expérience de l'utilisateur moins confuse et simple à prendre en main
   - ***6.2. Description :***
     Réaliser des menus permettant à l'utilisateur de naviguer dans notre application
   - ***6.3. Contraintes :***
     Les Réaliser en SWING/AWT
   - ***6.4. Niveau de priorité :***
     Priorité Basse

II. Extensions

Quelques pistes dont l'implémentation rendrait le jeu plus intéressant et plaisant à jouer , visant à rendre l'expérience utilisateur le plus agréable que possible

1. **Transcription Vocale (Alexa)**
   - ***1.1. Objectif :***
     Obtenir les directions que les joueurs indiquent dans leurs vocaux afin de pouvoir bouger le personnage en conséquence
   - ***1.2. Description :***
     Utiliser les bibliothèques d'Amazon Transcribe pour réaliser la transcription de l'audio vers un texte pour y récupérer les commandes énoncées par le joueur.
   - ***1.3. Contraintes :***

Délais assez long (10 secondes) qui s'est vu à la baisse avec des conversions de fichiers.

1.4. Niveau de priorité :

Priorité Haute

2. Musique et ambiance de fond

2.1. Objectif :

Rendre l'expérience utilisateur plus agréable

2.2. Description :

Composer des musiques qui vont accompagner l'utilisateur lors de son utilisation de notre jeu

2.3. Contraintes :

Aucune

2.4. Niveau de priorité :

Priorité Moyenne

3. Mode de jeu

3.1. Objectif :

Varier l'expérience utilisateur

3.2. Description :

Proposer des modes de jeu différent, des règles, des interactions différentes (objets, téléportation…)

3.3. Contraintes :

Aucune

3.4. Niveau de priorité :

Priorité faible

4. Menu interaction

4.1. Objectif :

Menu pour interagir avec les joueurs

4.2. Description :

Un chat/dialogue pour présenter le jeu, donner des instructions dont la façon de jouer le jeu. Permettre aux joueurs de revérifier leurs déplacements, composition de leur équipe (combien d'Homme et de Femme).

4.3. Contraintes :

4.4. Niveau de priorité :

Priorité Moyenne

5. Multijoueur

5.1. Objectif :

Séparer plusieurs utilisateur sur le même jeu

5.2. ***Description :***

- Adapter le jeu pour pouvoir être joué sur la même machine avec plusieurs joueurs.

5.3. ***Contraintes :***

Gérer le fonctionnement du jeu et création de nouvelles règles liées au multijoueur

5.4. ***Niveau de priorité :***

Priorité Basse