HUSSENET Augustin - MOATAZ Aymane

**Document technique**

**Réalité Augmenté et Dyslexie**

**Septembre 2019 – Juin 2020**

***Mines Nancy***

Professeur encadrant : M. Lamiroy Bart

En relation avec Mme Gagelmans Sofie (ANRT)

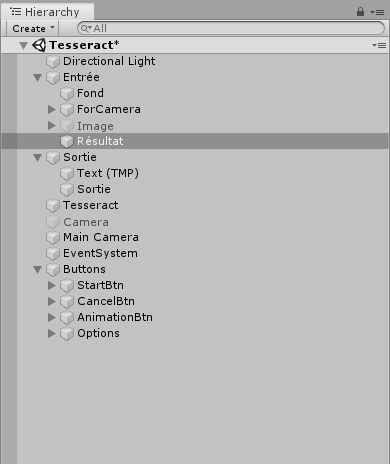
Table des matières

**1) Organisation de la scène principale** **3**

**2) Déroulement du processus** **4**

**3) Organisation des dossiers** **5**

1. **Organisation de la scène principale :**



Voici l’organisation de la scène principale Tesseract de notre projet Unity (cette scène se trouve dans le dossier « scene » du projet) :

* Entrée : contient tous les éléments de design de l’application. Ce Canvas correspond à ce qui est affiché en permanence à l’écran.
* Fond : correspond à l’image de fond de l’application
* ForCamera : contient l’image capturée par la caméra du téléphone.
* Résultat : affiche l’homographie de l’image capté par la caméra qui filme le Canvas Sortie, ce qui permet de bien replacer le texte dans l’image d’origine à la fin du processus.
* Sortie : contient tous les éléments associés au étapes intermédiaires du processus
* Sortie contient l’image extraite de l’image initial par l’algorithme OpenCV

Au cours du processus, tous les GameObjects contenant le texte extrait sont stockés dans cette image et sont affichés devant.

* Tesseract est un GameObject qui contient l’algorithme d’extraction du texte d’une image
* Camera : Caméra qui filme le canvas de sortie, c’est-à-dire l’image extraite de l’image principale ainsi que le texte reconnu placé devant à la bonne position.

Cette caméra contient également un script Homography qui permet d’appliquer une transformation homographique à l’image filmé par la caméra

* Main Camera : Filme le Canvas Entree ainsi que tous les éléments qu’il contient et affiche l’image capturé dans l’écran du téléphone. Cet objet contient également l’algorithme principal de l’application.
* Buttons : Canvas contenant les différents boutons de l’application :
* StartBtn : Bouton permettant de lancer le processus principal de l’application
* CancelBtn : Bouton permettant d’arrêter le processus pour le recommencer sur une nouvelle image
* AnimationBtn : Permet de lancer ou d’arrêter l’animation dyslexique du texte.
* Options : Permet de choisir l’effet dyslexique que l’on souhaite appliquer au texte

1. **Déroulement de l’algorithme principal de l’application**

L’algorithme principal de l’application se trouve dans la classe C# Camerascript. Voici une brève explication de son déroulement :

* A chaque image, tant que le bouton Start n’a pas été activé, on exécute grâce à la fonction « Process » l’algorithme d’openCV permettant de détecter une forme quadrilatère plus claire que le reste de l’image. Le contour de la forme détectée est alors dessiné en bleu sur l’image capturée par la caméra du téléphone.
* Lorsque l’on appuie sur le bouton Start de l’application, On exécute à nouveau l’algorithme d’OpenCV : cet algorithme détecte une forme quadrilatère plus claire que le reste de l’image, l’extrait de l’image et la projette sous forme rectangulaire dans le plan du téléphone portable grâce à une transformation homographique.
* Une fois l’image extraite et affichée dans le GameObject sortie de type RawImage, on lance l’algorithme Tesseract contenu dans le gameObject tesseract en lui passant en paramètre l’image dont on veut extraire le texte. Cet algorithme extrait le texte de l’image que l’on lui a transmise et créer un GameObject de type TextMeshPro pour chacune des lettres qu’il repositionne au bon endroit devant l’image dont le texte a été extrait.
* Les GameObjects contenant le texte reconnu par tesseract sont alors mis en mouvement de façon à reproduire l’effet dyslexique sélectionné.
* En parallèle, le bouton Start active aussi la caméra « camera » qui filme l’image extraite sur laquelle on superpose le texte extrait par tesseract. L’activation de cette caméra lance également le script Homography qui lui est attaché. Ce script capture l’image filmé par la caméra et lui applique une transformation homographique de manière à la repositionner correctement dans l’image initiale. Cette homographie est réalisée à chaque image, ce qui permet d’observer les mouvements de lettres provoqués par l’effet dyslexique dans l’image initiale.
* Finalement, une fois le processus lancé, il est possible de relancer avec une autre image en appuyant sur le bouton cancel ou d’arrêter l’animation du texte afin de choisir un nouvel effet dyslexique.

1. **Organisation des dossiers du projet :**

Dossier OpenCV + Unity :

Ce dossier correspond à l’importation de la bibliothèque OpenCV dans notre projet. Cette bibliothèque nous permet de récupérer le script OpenCV permettant d’extraire une partie de l’image initiale et de la projeter dans le plan du téléphone. Ce script se trouve dans le dossier Demo/Document\_Scanner. L’algorithme transforme les couleurs de l’image initiale afin de lui appliquer une nuance de gris. À la suite de cela, l’algorithme recherche les coordonnées du contour de forme quadrilatère de couleur différete tout en maximisant son aire. Il lui applique finalement une transformation homographique afin de projeter l’image extraite dans le plan du téléphone.

Dossier Prefabs :

Ce dossier contient une Prefab d’un GameObject TextMeshPro qui va nous permettre, au cours de l’exécution de l’algorithme de Tesseract, de créer les GameObject contenant le texte reconnu par Tesseract et de les positionner correctement devant l’image dont le texte a été extrait.

Dossier ressource :

Ce dossier contient différentes images sur lesquelles nous avons pu réaliser des tests tout au long du développement de notre application. Cela nous permet d’éviter d’avoir à installer l’application sur téléphone à chaque modification de code.

Le dossier contient également tous les objets de Unity nous permettant de garder certaines valeurs en mémoire lors du redémarrage de la scène (appuie sur le bouton cancel) ou de l’application ou encore de transférer des valeurs entre plusieurs scripts (par exemple afin de connaitre les positions du contour de l’image extraite lors de l’homographie finale).

Dossier Scene :

Ce dossier contient la scène principale de l’application.c

Dossier Scripts :

Comme nous venons de l’évoquer, le script principal du processus de l’application est le script : CameraScript depuis lequel on appelle la majorité des autres scripts utilisés dans le processus.

Les autres scripts contenus dans ce dossier sont associés à l’implémentation dans Unity de tesseract. Les deux scripts TesseractDemoScript et TesseractDriver sont majoritairement associé à l’implémentation de tesseract dans Unity, nous n’avons modifié ces codes que pour passer certains paramètres utiles au dernier : TesseractWrapper. Ce dernier script est celui qui réalise l’extraction du texte de l’image. Nous l’avons modifié afin de pouvoir recouvrir le texte initialement présent dans l’image et créer les GameObjects contenant le texte reconnu par tesseract.

Le sous dossier Structures contient des structures utilisées par le script de tesseract et le sous dossier scriptableobject contient des définitions d’objets qui nous permettent de garder certaines valeurs en mémoires (comme l’option sélectionnée par exemple) lors du redémarrage de la scène ou de l’application.

Dossier uHomography :

Ce dossier contient une importation d’un projet permettant d’appliquer une homographie à l’image capturé par une caméra que nous avons trouvé en ligne.

Voici le lien du dépôt github de ce projet : <https://github.com/hecomi/uHomography>