TP2 : Détection de visage dans un flux vidéo.

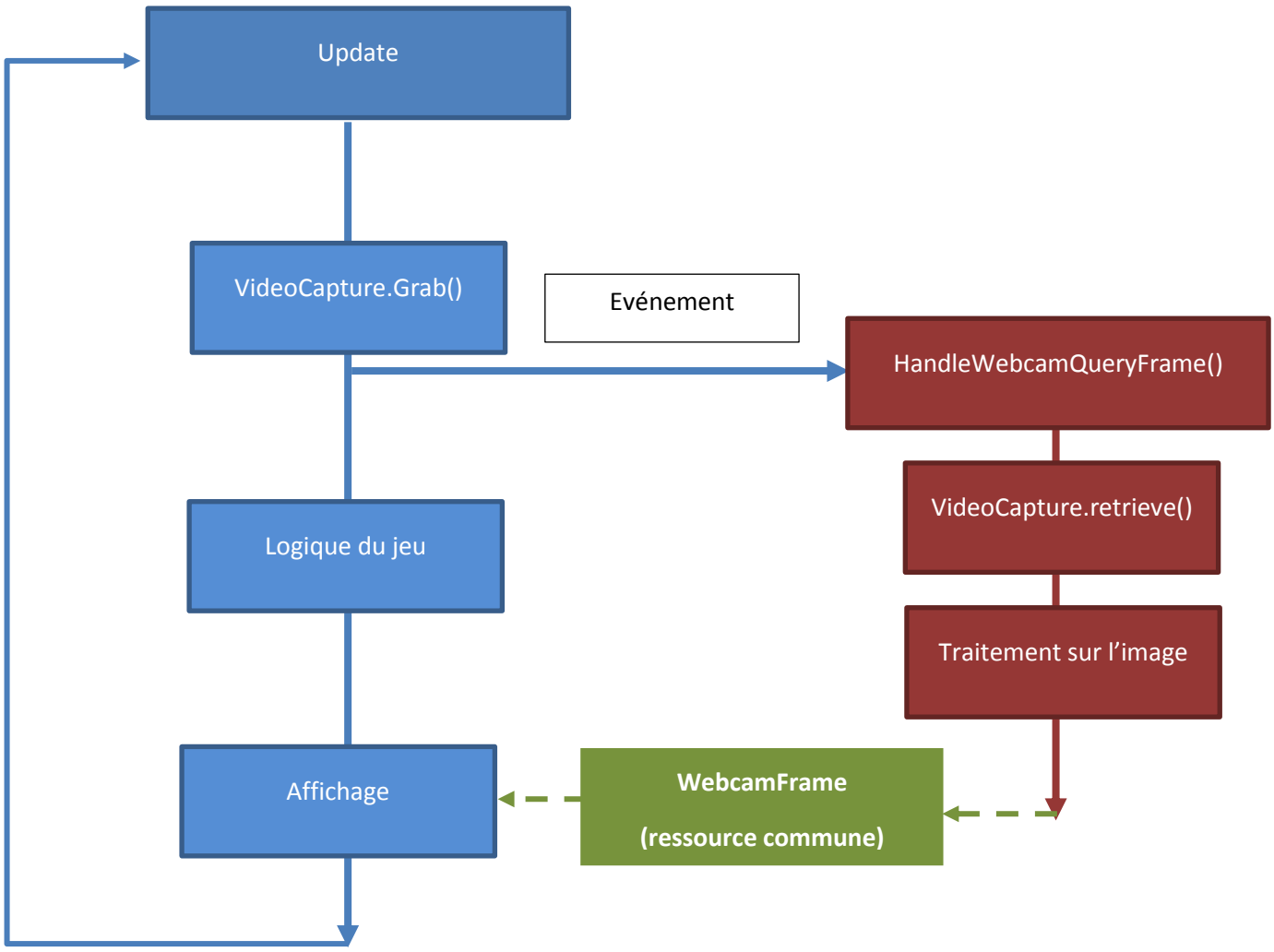
1 Introduction

Dans ce TP nous allons utiliser EmguCV et Unity. Nous allons voir comment récupérer un flux vidéo avec EmguCv et comment l'afficher sur un Canvas dans Unity.

La fonction CvInvoke.Imhsow() utilisée au TP précédent, n'est pas adaptée pour l'affichage d'une vidéo dans un jeu. Nous allons cette fois ci voir une méthode qui permet de déléguer l'acquisition d'un frame et les traitements de détections dans un thread dédié. Cela permet au frame rate du jeu de ne pas dépendre de ces tâches.

Nous mettrons en place une cascade de Haar qui permet de détecter des visages avec une bonne précision.

Pour cela, nous allons utiliser de la programmation évènementielle. Le digramme de séquence de l'application est le suivant :



Pour ce TP nous aurons 5 fonctions à coder :

Start() : Cette fonction servira à initialiser la webcam et le filtre de cascade de Haar, à associer la fonction HandleWebcamQueryFrame() à l'évènement d'acquisition d'une image et enfin à récupérer le gameObjet sur lequel on va afficher notre webcam dans le jeu.

Update() : Nous appelerons ici la fonction qui sert à déclancher les évènements d'acquisition de la webcam, ainsi que la fonction permettant de mettre à jour l'affichage de l'image sur le gameObject.

HandleWebcamQueryFrame() : Fonction appelée lors de l'évènement d'acquisition. C'est ici que se fera la récupération de la frame courante ainsi que la détection de visage.

DislayFrameOnPlane() : Permet de charger la frame courante dans une texture et de l'appliquer à notre gameObject.

OnDestroy() : Fermeture du flux de la camera.

2 Exercices

1) **Instancier une webcam dans la méthode Start et libérer ses ressources dans la méthode Destroy. La webcam est une donnée membre de la classe principale.**

   Classes et méthodes
   - VideoCapture.Start()
   - VideoCapture.Stop()
   - VideoCapture.Dispose()

2) **Créer une fonction `private void HandleWebcamQueryFrame(object sender, System.EventArgs e)` qui récupère une image et affiche sa taille dans la console unity. Ajouter cette fonction à la file des EventHandler de votre webcam.**

   Classes et méthodes
   - `Webcam.Retrieve()`
   - `Webcam.ImageGrabber`

3) **Dans la méthode update, vérifier que la caméra est ouverte, et appeler la méthode Grab() de votre webcam afin de provoquer un évenement. Lancer le programme dans Unity, la console devrait afficher la taille de l'image.**

   Classes et méthodes
   - VideoCapture.Grab()

4) **Dans Unity, ajouter un objet RawImage (Présent dans UI->RawImage). Renommer votre objet webcamScreen (sous-objet du canvas).**

5) **Créer la fonction DisplayFrameOnPlane(). Dans cette fonction créer une nouvelle texture de la taille de votre « webcamScreen » copier la frame courante dans la texture (cette copie nécessite un redimensionnement ainsi qu'une conversion de format en RGBA). Charger la texture sur votre « webcamScreen ».**

   Classes et méthodes

   - Texture2D
   - UnityEngine.UI.RawImage
   - CvInvoke.Resize
   - CvInvoke.CvtColor
   - CvInvoke.Flip
   - Texture2D.LoadRawTextureData()
   - Texture2D.Apply()
   - UnityEngine.UI.RawImage.texture

6) **Télécharger les cascades de haar du dossier data du dépôt Github d' OpenCV et les ajouter aux assets (https://github.com/opencv/opencv/tree/master/data/haarcascades)**

7) **Dans la fonction HandleWebcamQueryFrame, Ajouter la détection de visages. Celle-ci s'effectue sur une image en niveau de gris. Dessiner les boîtes englobantes des visages sur la frame courante.**

   Attention, la frame courante est une donnée partagée avec le Thread principal…Utiliser les mécanismes de partage de ressources sécurisés du C#.

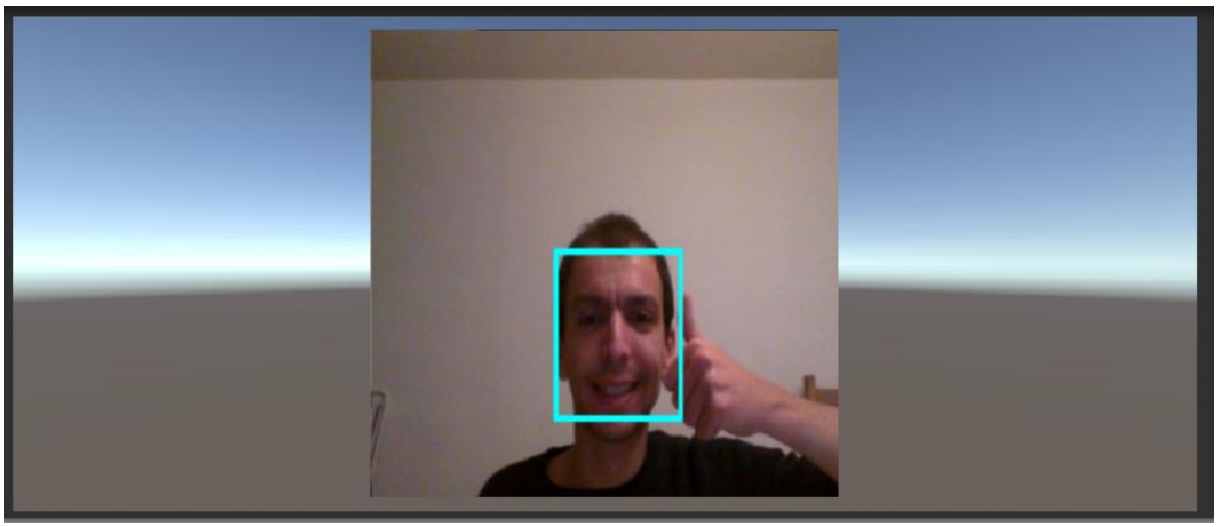


Figure 1 - Exemple de détection d'un visage heureux