GESTURE BASED USER INTERFACE PROJECT

Github Repo: https://github.com/jaymz95/GestureUIProject

Abstract

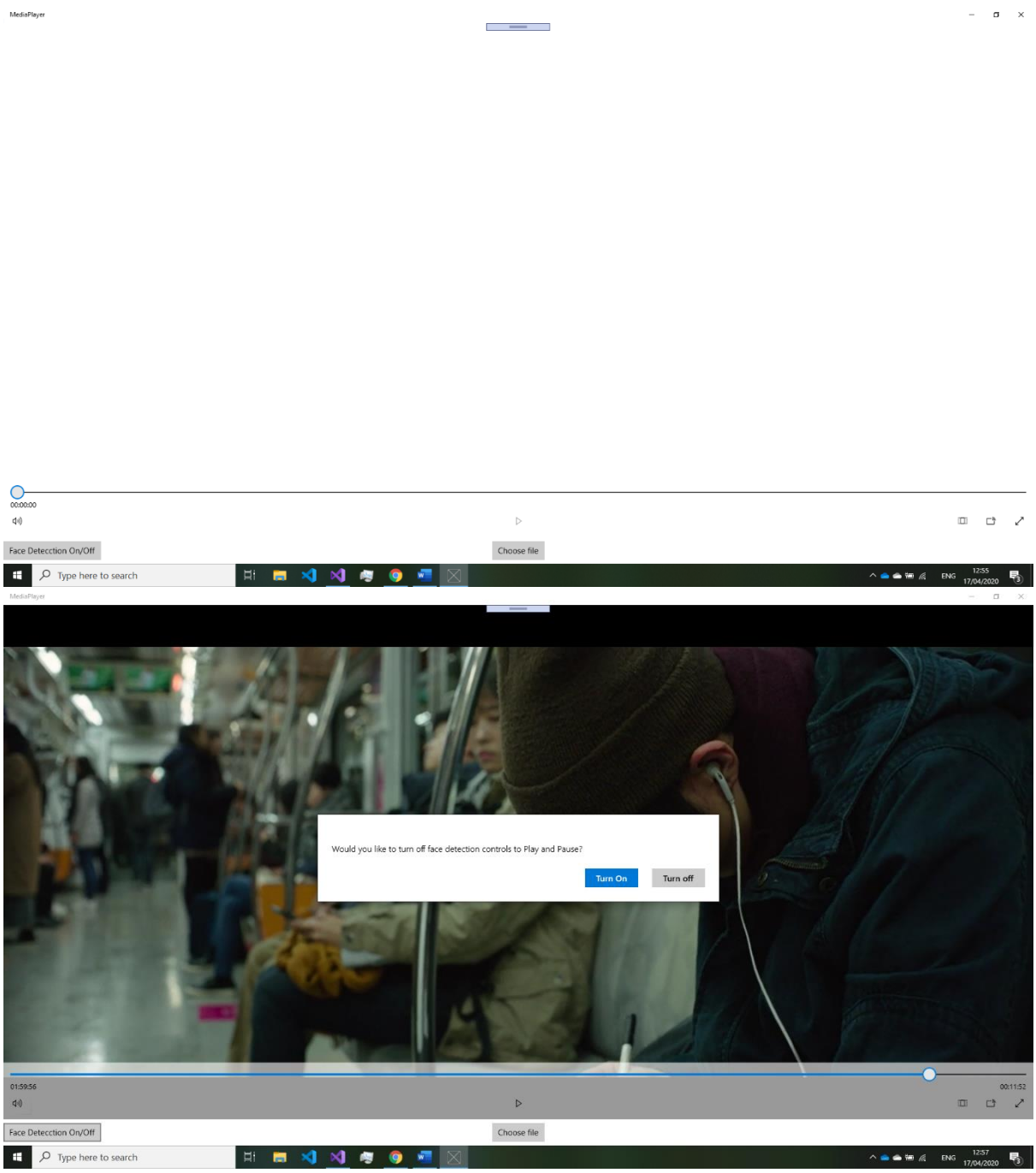
Documentation for a Media Player implementing voice detection and face detection controls for better user experience and the reasoning used for the technologies chosen

James Mullarkey & Liam Higgins

G00345716@gmit.ie & G00348489@gmit.ie

Purpose of the application

This application is a Media Player application, capable of Voice Recognition Commands and Facial Detection Controls. Users can interact with this media player by picking a video to play and using the various voice commands to control the app e.g. “Play”, “Fast Forward” etc. They can also use the face detection feature to play and pause the video (e.g. if face detected play else pause). When the popup to turn Off/On the Face Detection appears the app outputs the Speech of the test box using Text To Speech technology.



Working out which controls would have priority was a challenge. The decision was to use facial recognin only when the user has used the “Play” voice command i.e. plays and pauses based on

whether it can detect a face from the camera. But face detection controls are not used when the user used the “Pause” voice command because face detection would overwrite that command which made the voice command “pause” redundant.

Gestures identified as appropriate for this application

The gestures used was Voice Recognition Commands and Facial Recognition. These features were chosen because they worked well with what we wanted to achieve which is a media player that is easier use for the consumer, and it introduced a more minimal effort control system for the user to controll their media.

At first it was just going to use Voice Recognition but shouting commands at the media player grew very tired very quick. It was then decided to find another way to control the app with minimal hardware resources. Facial recognition seemed to be next best thing because it cuts out the user having to talk to the app and the app interoperates when the user wants to watch by using face detection. The face detection part of the app can prove to be irritating when you want the media to continue playing when you are not in the view of the camera, so an option to turn this feature on and off was introduced.

There was a lot of trial and error when creating this appliction due to our limited knowledge on guesture based user interface but there was ample resourses in the module and online which helped on getting out app to conclusion.

Throughout the project there was different classes added and removed from the project testing out diftterent technologies and techneques this can be seen in out git commit history and git Issues page.

Hardware used in creating the application

Webcam microphone, speaker

For this project the hardware used was a Webcam, Microphone and Speaker. The speaker was used for the TTS feature. The microphone was used for speech recognin to pick up user commands to the media player which could the be used to implement the relevant command. The webcam was used to detect a face to see of someone is watching the player and if so play and if not pause.

To setup the microphone and webcam I had to go into my laptops setting and allow Visual Studio permission to access my hardware. The options for hardware were very limited and unfortunately. There wasn’t any other option where we could incorporate a more innovated exciting technology. The original plan was to use a Microsoft Kinect to control the media player with hand gestures, but we ended up not having access to a Microsoft Kinect.

Architecture for the solution

For this app it was decided that the best framework to use to create this was UWP application because we planned on using the app on Windows laptops and it contains various libraries for Gesture detection which proved extremely useful compared to using python or Django framework

which would have taken more time to implement the same app, with the time constraints it would have ended up being clunky. There was a worry that we would not be able to achieve what we had set out to do which was create an app that was capable of voice and face detection to control a media player if we used python.

Speech To Text (STT) was also used to ask the user if they would like to turn off face detection.

Assets used:

- For Facial Recognition **`Windows.Media.FaceAnalysis;`** was used.
- For Voice Commands Recognition **`Windows.Media.SpeechRecognition;`** was used.
- For Text To Speech (TTS) Recognition **`Windows.Media.SpeechSynthesis;`** was used.

The decision was made to use the Windows.Media libraries for these features because they are the most tried and tested libraries and there is more documentation on them online. This was a big advantage when issues with the library arose to have the documentation to hand, rather than searching the internet for an answer to a lesser known library and less in-depth documentation.

Conclusions & Recommendations

The app that has been created is fit for purpose and has useful features that the average player does not have. I learned how to interact between static and non-static methods, how to access variable from different classes and how to implement features in a UWP application. If I was to start over on this project, I would use a Microsoft Kinect for hand gesture controls and create the media player from scratch using python, if I was to have no time constraints. this would make the app more robust and I could implement it how I want rather than the way windows encourage you to create UWP apps.