SpeechToCodeCSharp

Contexte : Aujourd’hui, peu ou pas de sources permettent de programmer grâce à la voix.

Programmer en parlant permet de renforcer notre esprit créatif.

Besoin :

* Améliorer sa productivité en couplant les entrées clavier à la possibilité de parler
* => Améliorer sa créativité

But : Réalisation d’un logiciel de traitement interactif à la fois vocal & clavier.

Structure :

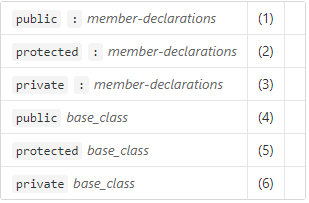
Le logiciel se décompose en deux parties distinctes :

1. L’éditeur de texte, en C# WPF
   * TileBar de l’application, avec
     + Le titre de l’application
     + Un bouton pour minimiser/maximiser l’application
     + Un bouton pour fermer l’application
   * Barre d’outils (MenuBar) pour gérer
     + La gestion du projet et de ses fichiers (New/Open/Save/SaveAs)
     + La configuration du projet
     + La configuration de l’éditeur
   * En dessous de la MenuBar, il y a la Toolbar qui gère les boutons cliquables
     + Open, new, save(+saveAs)
     + Start vocal listening / Stop vocal listening, pour la reconnaissance vocale
     + (autres boutons à venir)…
   * La zone de texte où s’affiche le code. Il est édité dans cette zone.
   * Un volet « Project Explorer » où s’affichent le projet et ses fichiers.
2. Le module de Reconnaissance Vocale des mots-clés & instructions du langage C++

Fonctionnalités :

1. Module de reconnaissance des mots-clés & instructions du langage c++

* Reconnait les keywords du c++ :
  + D
* Fonction de génération du reste des instructions suivant le mot-clé en fonction de la nature de ce dernier (ex : un typedef finira par un « ; », un if…else finira par un « } »
* Réconnaissance de mots-clés d’**instructions** pour effectuer des actions de :
  + Création de variable
  + Création de classe / struct
  + Création de fonction / méthode
  + Création de template
  + Instructions Préprocesseur + related keywords
    - #Define
    - #error
    - #include
    - #line
    - #pragma
  + Import de library/header
* Reconnaissance des **words de la librairie** **std ::  ???**
* Reconnaissance des « access-specifiers », ou specificateurs d’accès :



(1) : public (optional keyword after public could be : “”)

(4) : public inheritance

(5) : protected inheritance

(6) : private inheritance

* ETC / AUTRES FONCTIONALITés

1. La zone de texte de l’éditeur

CMU SphinxBase & PocketSpinx : Cross Platform c++

* Download page: <https://cmusphinx.github.io/wiki/download/>