

Travaux Pratiques Interaction Vocale (Ph. Truillet) Septembre 2023

1. Une application dirigée à la voix

Nous souhaitons concevoir et réaliser une application **non-visuelle** (en **entrée** et en **sortie** incluant **parole** et éventuellement **son** -musique, messages enregistrés, etc.) permettant à un utilisateur **d'ajouter, retirer, manipuler des aliments et boissons affichés sur un écran afin de composer le contenu d'une assiette « gourmande »**. (ex : café, thé, jus d'orange, crème brûlée, profiteroles, ...). Il y a aura exactement **deux** desserts et **une** boisson dans le plat composé.

La disposition physique des desserts fait partie du problème !

Vous coderez **votre application dans le langage que vous désirez** 😊
(l'usage de *Processing.org* peut être une bonne alternative).

Il devra être possible d'effectuer toutes les actions demandées de manière purement vocale en entrée et en sortie.



Afin de réaliser notre application multimédia, **nous nous servirons prioritairement du middleware (bus logiciel) ivy**

[<https://github.com/truillet/ivy/blob/master/README.md>], support au futur Bureau d'Etudes sur la multimodalité.

Nota : Si vous êtes sous Linux ou MacOS, il vous faudra trouver des solutions alternatives pour la reconnaissance et la synthèse vocale (par exemple, utiliser **MaryTTS**, **ElevenIO** pour la synthèse vocale ou **STT** ou encore **SpeechRecognition**, librairie python pour la reconnaissance vocale – cf. liens plus bas).

2. Travail attendu de cette séance (2 h)

Après avoir pris en main les agents de reconnaissance et de synthèse vocale fonctionnant avec le bus logiciel ivy, l'objet de cette séance est :

1. de **définir la grammaire de reconnaissance** (commandes vocales ou langage « *pseudo-naturel* ») qui sera utilisée par votre application, gérer les résultats de la sortie sémantique (i.e. les concepts associés aux paroles prononcées) ainsi que le taux de confiance.
2. de **définir les retours (feedbacks) vocaux à synthétiser et sonores** utilisés par votre application.
3. de développer une application d'affichage des plats à l'écran (en java, Processing, python ... ou un autre langage).
4. et enfin **développer le contrôleur de dialogue à l'aide d'une machine à états** (qui peut être soit séparée, soit incluse dans l'application d'affichage de la forme). Le contrôleur s'appuiera sur un échange de messages ivy avec au moins les modules de reconnaissance et de synthèse vocale.



A la fin de la séance, vous aurez produit **un prototype haute-fidélité testable** du système demandé.

Nota : pour ce faire, vous pourrez utiliser quelques agents ivy déjà codés (présentés en annexe)

3. Liens de téléchargements

- **ppilot5** (Text-to-Speech), **sra5** (Automatic Speech Recognition), ... agents d'interaction vocale
<https://github.com/truillet/upssitech/tree/master/SRI/3A/IHM>
- **librairies ivy** :
<https://github.com/truillet/ivy/blob/master/README.md>
- Si vous le désirez, vous pouvez aussi utiliser :
 - **MaryTTS** (<https://github.com/marytts/marytts>), serveur Text-to-Speech écrit en Java
 - **ElevenIO** (<https://elevenlabs.io>), TTS accessible via une API REST (<https://api.elevenlabs.io/docs>)
- **STT** : Speech Recognition for Java/Processing basé sur Google Chrome et websockets :
<http://florianschulz.info/stt>
Nota : Vous pouvez utiliser la page <https://www.irit.fr/~Philippe.Truillet/stt.html> pour lancer le serveur de reconnaissance.
- **SpeechRecognition**, librairie en Python : <https://pythonprogramminglanguage.com/speech-recognition/>

N'hésitez pas à demander à l'enseignant si tel ou tel agent existe : c'est peut-être déjà le cas ! Et puis, vous pouvez **CODER** vos propres agents selon **VOS** désirs ! 😊

Annexe 1 - utiliser sra5

sra5 est un agent utilisant le moteur de reconnaissance natif SAPI 5.x de Windows Vista, 7, 8.1, 10 et 11 et peut renvoyer **deux types de solutions** issues de la reconnaissance **sous deux formats différents** :

Lancement de l'agent en ligne de commandes

```
sra5 -b 127.255.255.255:2010 -p on -g grammaire.grxml
```

Par défaut, **sra5** utilise le fichier de grammaire locale **grammaire.grxml**

- b adresse IP + port
- p mode de renvoi des données (mode parsing¹ **on** ou **off**)
- g fichier de grammaire utilisé (grammaire de type grXML
– cf. <http://www.w3.org/TR/speech-grammar>)

¹ Le mode « parsing » consiste à renvoyer comme résultat les sorties sémantiques plutôt que la chaîne orthographique.

Retours (**UNIQUEMENT** sur le bus ivy)

- **sra5 Text**=chaîne_orthographique **Confidence**=taux_de_confiance (si le flag *parse* est positionné à off)
- **sra5 Parsed**=resultat **Confidence**=taux_de_confiance **NP**=xx **Num_A**=xx où NP est le numéro du résultat courant et Num_A le numéro d'alternative (si le flag *parse* est positionné à on)
- **sra5 Event**={Grammar_Loaded | Speech_Rejected} : envoi d'événements provenant du moteur de reconnaissance.

Commandes (**UNIQUEMENT** sur le bus ivy)

- **sra5 -p** {on | off} **sra5** change le mode de retour de la reconnaissance (on → mode de retour sous forme de concept ou off → mode de retour orthographique)
- **sra5 -g** **sra5** active une nouvelle grammaire (sur un chemin local à la machine)

Annexe 2 - utiliser ppilot5

ppilot5 permet d'utiliser des systèmes de synthèse vocale compatibles SAPI5.

Lancement de l'agent

```
ppilot5 -b 127.255.255.255:2010 -r Hortense -o "Microsoft Hortense"
```

Par défaut, **ppilot5** utilise le premier moteur de TTS trouvé et apparaît sur le bus ivy sous le nom « **ppilot5** »

- b adresse IP + port
- r nom sous lequel apparaîtra l'agent sous ivy (dans l'exemple précédent, « *Hortense* »)
- o nom du moteur de synthèse utilisé (ici, la TTS "*Microsoft Hortense*", TTS par défaut sous windows)

Commandes (**UNIQUEMENT** sur le bus ivy)

* Synthèse

- **ppilot5 Say**=hello **ppilot5** prononce via la TTS utilisée la chaîne de caractères "hello"
- **ppilot5 SaySSML**=<sequence_SSML> **ppilot5** prononce la séquence SSML et renvoie **ppilot5 Answer=Finished** quand le buffer est vide. Les balises <speaking> et </speaking> sont automatiquement ajoutées au flux
Exemple de séquence SSML :
ppilot5 SaySSML=Je peux parler <emphasis level="strong">très fort</emphasis>
si je veux !

* Commandes

- **ppilot5 Command=Stop** la synthèse vocale est stoppée. **ppilot5** renvoie **ppilot Answer=Stopped**
- **ppilot5 Command=Pause** la synthèse vocale est mise en pause. **ppilot5** renvoie **ppilot Answer=Paused**
- **ppilot5 Command=Resume** la synthèse vocale est relancée si elle était en pause précédemment. **ppilot5** renvoie **ppilot Answer=Resumed**
- **ppilot5 Command=Quit** l'application se ferme

* Paramètres

- **ppilot5 Param=Pitch:value** le pitch est changé par la valeur donnée. **ppilot5** renvoie **ppilot Answer=PitchValueSet:value**
- **ppilot5 Param=Speed:value** la vitesse est changée par la valeur donnée. **ppilot5** renvoie **ppilot Answer=SpeedValueSet:value**
- **ppilot5 Param=Volume:value** le volume est changé par la valeur donnée. **ppilot5** renvoie **ppilot Answer=VolumeValueSet:value**