

# Interaction vocale

## 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 affichés sur un écran afin de composer le contenu d'une assiette « gourmande » de dessert(s)**. (ex : café, thé, sucre, crème brûlée, profiteroles, ... )

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** pour la synthèse vocale ou **STT** ou encore **SpeechRecognition**, librairie python pour la reconnaissance vocale – cf. liens plus bas).

## 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)

---

## Liens de téléchargements

- **ppilot5** (Text-to-Speech), **sra5** (Automatic Speech Recognition), ... agents d'interaction vocale :

<https://github.com/truillet/ups/blob/master/m2ihm/README.md>

- **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 Test-to-Speech écrit en Java

- **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 ou 10 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 parsing<sup>1</sup> **on** ou **off**)
- g      fichier de grammaire utilisé (grammaire de type grXML  
        – cf. <http://www.w3.org/TR/speech-grammar>)

<sup>1</sup> 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)

### 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 (<https://www.w3.org/TR/speech-synthesis11>) et renvoie **ppilot5 Answer=Finished** quand le buffer est vide. Les balises *<peak>* et *</peak>* 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**