Conception d'un moteur de fusion multimodale

Objectifs

Le but de ce bureau d'étude est de **spécifier, concevoir et implémenter un moteur de fusion multimodale** pour interagir avec une palette de dessin ne disposant d'aucun bouton. Pour créer et déplacer des formes sur la palette vous utiliserez les modalités suivantes :

1. La reconnaissance de ***parole*** grâce au moteur de reconnaissance de parole (avec par exemple l'usage de l'agent ivy sra5)
2. La reconnaissance de ***geste*** grâce à la palette de reconnaissance de geste 2D (votre agent ivy $1Recognizer ou ICAR – cf. plus bas -)
3. Le ***pointage*** (souris) sur la palette de dessin

L'objectif est de **développer un moteur de fusion des différentes modalités** permettant d'approcher le célèbre « ***put that there*** » (cf. lien ci-dessous), une des premières techniques d'interaction multimodale proposée par le MIT il y a maintenant une quarantaine d'années.
Démonstration : http://www.youtube.com/watch?v=RyBEUyEtxQo

Architecture

Les outils devront communiquer selon l'architecture suivante ou équivalente (cf. Figure 1) :

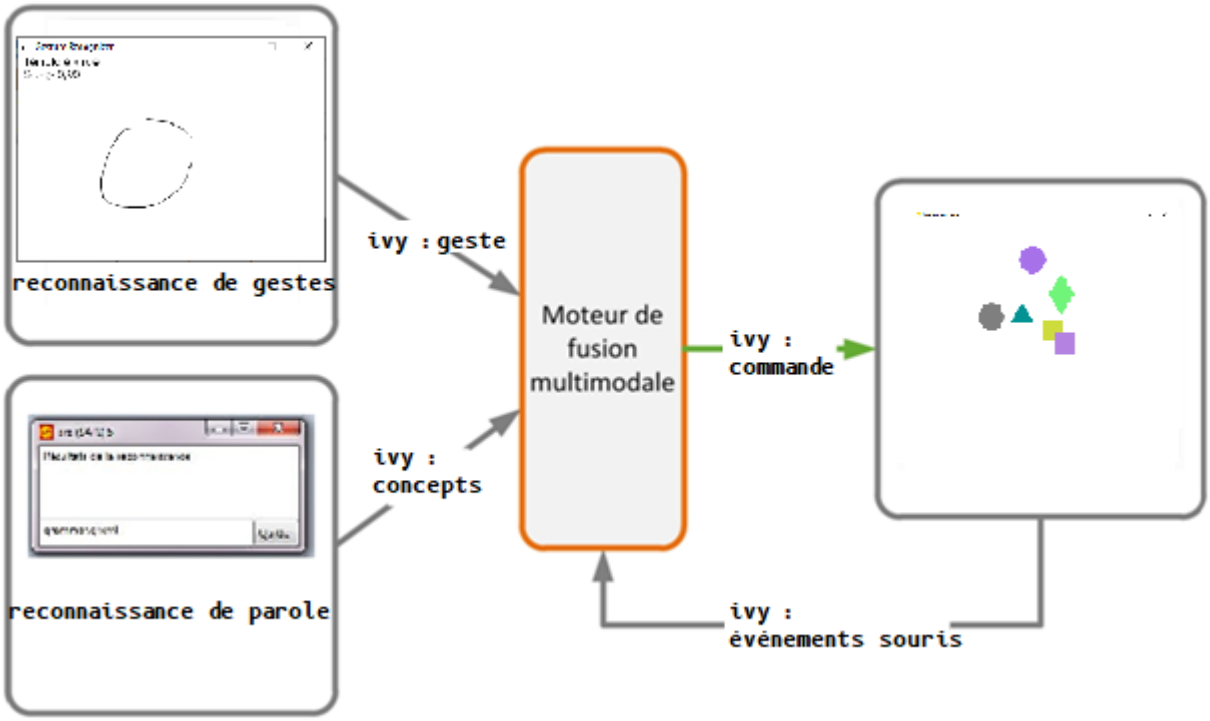

Figure 1 – architecture logicielle possible

Cahier des charges :

A partir de ces trois types d'interaction, il vous est demandé de réaliser un **moteur de fusion multimodale** qui vous permet d'effectuer les actions suivantes sur la palette de dessin.

Créer une forme

Le moteur de fusion devra permettre de réaliser cette action de différentes manières :

Créer	*rectangle* | *cercle* | *triangle*	ici	
Créer	*rectangle* | *cercle* | *triangle*	*rouge* | *vert* | *bleu*	
Créer	*rectangle* | *cercle* | *triangle*	de cette couleur	
Créer	*rectangle* | *cercle* | *triangle*	ici	de cette couleur
Créer	*rectangle* | *cercle* | *triangle*	*rouge* | *vert* | *bleu*	Ici
Etc.			

Contraintes additionnelles :

1. Aucune interaction proposée ne devra être **monomodale** !
2. L'action de créer un objet devra être réalisée au moyen de la reconnaissance de gestes (d'abord la parole « *créer* », suivi par le geste pour choisir la forme).
3. La couleur et la désignation de la position devront être optionnelles, et l'ordre des deux doit être flexible
4. Si une couleur est spécifiée, ceci doit se faire via la parole
5. Dans le cas d'une désignation (« de cette couleur », « ici »), celle-ci se réalise à la voix et doit être complétée par un pointage/clic sur la palette de dessin

Déplacer une forme

Cette action permet de déplacer un objet créé auparavant. L'utilisateur devra pouvoir spécifier l'action de déplacement via soit un geste, soit la parole. La désignation se fera comme pour la création d'un objet.

Déplacer ce	*rectangle* | *cercle* | *triangle*	ici	
Déplacer ce	*rectangle* | *cercle* | *triangle*	*rouge* | *vert* | *bleu*	Ici
etc.			

N.B. : Dans le deuxième cas, l'ajout de la couleur permet ici de désambiguïser un cas où il y aurait 2 rectangles à l'endroit de la désignation

Autres actions

Il serait possible de définir d'autres actions (supprimer, modifier la couleur). Ceci n'est pas demandé dans ce bureau d'étude, mais vous pouvez aller plus loin si vous avez le temps.

Couleur et formes

On peut se limiter à des formes d'exemple (rectangle, ellipse, triangle, …). Il est également suffisant de choisir trois couleurs comme exemple.

Les outils

Langage de programmation recommandé

Coder en Processing ou Java via un projet IntelliJ IDEA, Eclipse, … (ou Python si vous le préférez vraiment 😊).

Communication inter-outils : Ivy

Les outils (Palette / $1Recognizer ou icar / sra5 et ppilot5) mis à disposition sont des agents Ivy.

Les agents ivy communiquent via des messages textuels sur le réseau. Ils s'abonnent aux types de messages qui les intéressent. Ces types sont définis par expression régulière. La réception d'un message provoque le déclenchement d'un appel de callback/listener.

Visualiser les messages sur Ivy

1. Un visionneur graphique Ivy

Le visionneur vous permet de gérer les agents ainsi que les messages qui transitent sur le bus ivy de manière graphique. Particulièrement pratique lors du développement d'une application utilisant le bus Ivy.

Il est téléchargeable à cette adresse :
https://github.com/truillet/upssitech/blob/master/SRI/3A/IHM/TP/Outils/visionneur_1_2.zip

2. Probe sur console

En alternative, pour utiliser la ligne de commande (cf. documentation ivy vu en enseignement d'interaction distribuée)

Reconnaissance vocale :

1. Simulation

Durant le développement, la reconnaissance peut être simulée. Afin de simuler la reconnaissance des mots, on peut utiliser un panneau swing avec plusieurs `JButtons`. Chaque bouton représentera une commande vocale à reconnaitre (« *cet objet* », « *ici* », « *met ça* », …). Un clic sur le bouton enverra le message Ivy correspondant. Afin de pouvoir interchanger facilement ce panneau avec l'application `sra5`, les boutons devront envoyer des messages Ivy de la forme suivante :

```
sra5 Parsed='resReco' Confidence='proba' NP='id' Num_A='id'
```

Exemple pour la reconnaissance de l'expression « cet objet » on enverra :

```
sra5 Parsed='cetobjet' Confidence=0.8 NP=1 Num_A=0
```

[**Note** : `Confidence` précise le taux de confiance du ou des concepts reconnus ; `NP` et `Num_A` précisent le nombre d'items reconnus depuis le lancement et l'alternative de reconnaissance. Vous pouvez ces deux options à 0 dans tous les cas]

[**Note 2** : Il est possible de coder cette interface en python, Processing.org ou via … le langage de votre choix]

2. Utilisation de la reconnaissance vocale

Par la suite, nous utiliserons la reconnaissance vocale avec le module `sra5` (voir séance de TP 1). Il vous faudra donc modifier la grammaire de reconnaissance selon les besoins pour le projet.

Reconnaissance gestuelle : $1Recognizer ou algorithme de Rubine

OneDollarIvy (qui implémente l'algorithme $1Recgonizer) est composé d'une application Processing.org (cf. Figure 2)

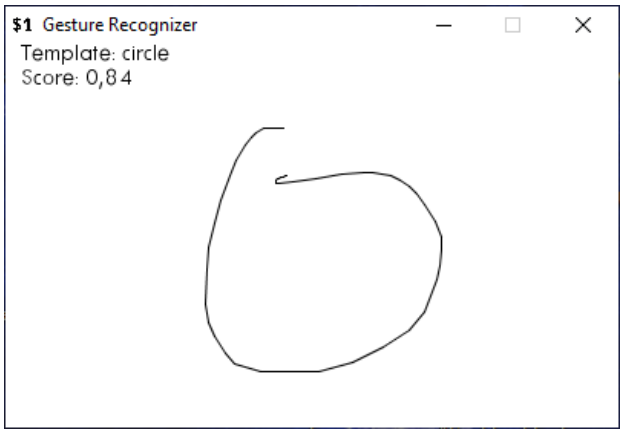

Figure 2 – reconnaissance de gestes par OneDollarIvy

OneDollarIvy permet de créer (**L**earn), d'importer (**I**mport), d'exporter (**E**xport), lister (display **T**emplates) et reconnaître (gesture **R**ecognition) des gestes appris.

ICAR (qui implémente l'algorithme de Rubine) est téléchargeable ici :
https://github.com/truillet/upssitech/blob/master/SRI/3A/IHM/TP/Outils/icar.1.2.zip

La documentation (sommaire) est disponible ici :
https://github.com/truillet/upssitech/blob/master/SRI/3A/IHM/TP/Outils/icar.pdf

ICAR est composé de 2 sous-applications : une pour l'apprentissage de gestes et l'autre pour la reconnaissance de gestes.

1. **Apprentissage de gestes :**

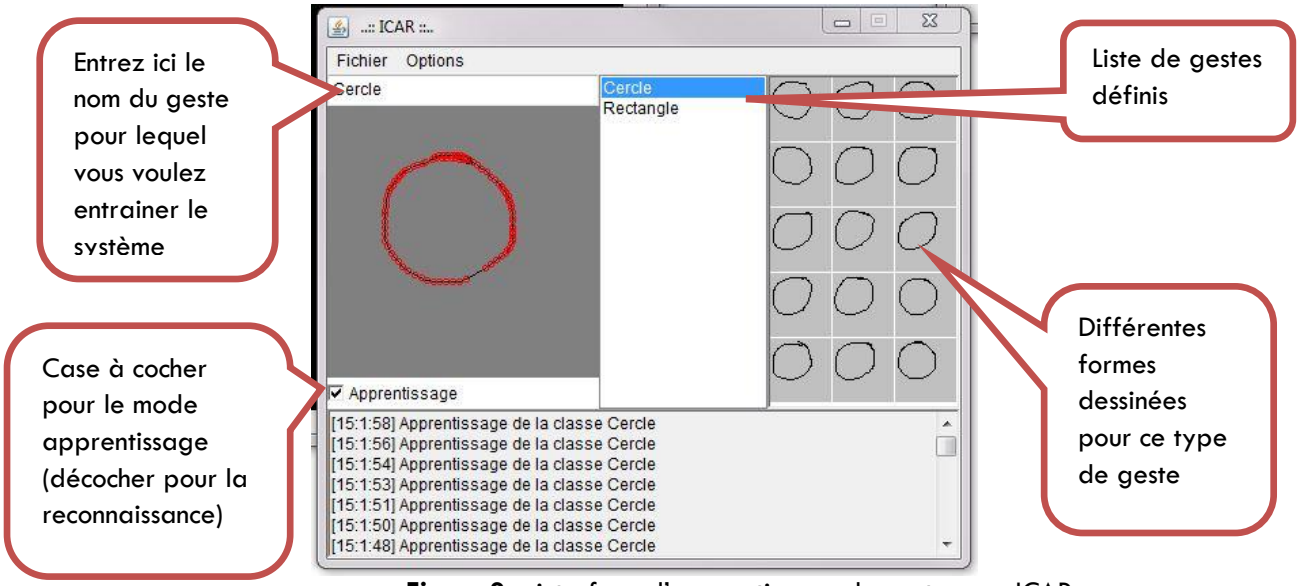

Figure 3 – interface d'apprentissage de gestes par ICAR

L'éditeur de dictionnaire « *IcarAdmin* » permet d'enregistrer un dictionnaire de gestes (cf. Figure 3).

[**Note** : Il convient de définir un seul dictionnaire qui contiendra plusieurs gestes différents.]

Une fois le dictionnaire créé et enregistré il vous suffit de modifier le **.bat** « *Icarlvy* » pour qu'il utilise en paramètre votre nouveau dictionnaire.

2. Reconnaissance de gestes :

L'outil de reconnaissance « Icarlvy » reconnait les gestes selon le dictionnaire fourni en entrée. Lorsque « *iCarlvy* » reconnait un geste, il envoie un message Ivy de la forme suivante :

```
ICAR 'nomGesteReconnu'
```

Palette de dessin

Celle-ci répond à des messages Ivy, permettant de créer, déplacer, colorier, supprimer, etc. des éléments. Afin de communiquer avec elle, votre application doit émettre des messages ivy.

Nous fournissons une version de Palette développée en Processing.org permettant de gérer différentes formes (triangles, cercles, rectangles et losanges) :
https://github.com/truillet/upssitech/blob/master/SRI/3A/IHM/TP/Code/Palette.zip

Vous pouvez bien évidemment recoder une palette selon vos envies.

Déroulement des séances

Pour réaliser ce bureau d'études, vous avez **trois** séances de deux heures. Voici une organisation possible des séances :

Séance 1 :

1. Spécifier les **grammaires** de gestes et de parole
2. Faire l'apprentissage des gestes ($1Recognizer ou ICAR)
3. Décrire vos **commandes multimodales** sous la forme de chronogrammes (cf. figure 4 ci-dessous). L'idée de ces chronogrammes est d'explorer les différentes possibilités d'ordre de commandes sur le bus ivy afin de mieux prévoir la flexibilité dans le système.

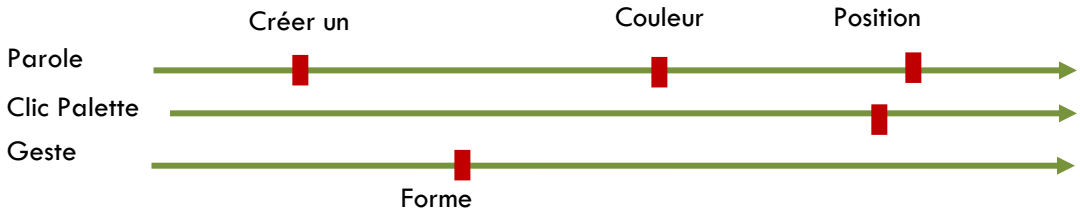

Figure 4 – chronogrammes des commandes multimodales

Séance 2 :

1. Décrire **l'automate du contrôleur de dialogue** (machines à états, voir cours M. Serrano).
   Cet automate est crucial pour le fonctionnement de votre moteur de fusion et nécessite de la réflexion !

   [**Rappel** : il faut commencer par décrire les actions et événements possibles, ensuite l'automate et ensuite la matrice états/événements]
2. Vous allez devoir utiliser une **structure** de données pour réaliser la fusion des informations. Elle sera remplie au fur et à mesure que les messages arrivent. Celle-ci sera implémentée dans une classe à part. Quelles données et quelles méthodes doit-elle contenir ? Quand devra avoir lieu la réinitialisation ? Spécifier et coder cette structure.

Séance 3 :

1. Coder le moteur de fusion. Vous veillerez également à appliquer les techniques de conception et d'implémentation systématique d'un contrôleur de dialogue (déduction du code par rapport à une machine à états).
2. Terminer de coder le moteur de fusion.
3. Tester et valider le fonctionnement de votre travail.
4. Ecrire le rapport.

Travail à rendre

Vous enverrez par mél un lien vers un repository git ou vers une archive zip contenant :

1. Votre conception de la fusion des modalités devra être argumentée dans un **rapport de quelques pages**. Vous y décrirez
   a. les aspects temporels de la fusion multimodale (chronogrammes)
   b. la conception logicielle de votre système dans sa globalité (diagramme de classe)
   c. la machine à états
   d. Illustrez l'utilisation de votre application avec un ou plusieurs exemples (impressions d'écrans de l'utilisation).
2. Le code source
3. Un exécutable ou un point d'entrée de votre projet (exemple : un fichier **.bat** ou **.sh** lançant l'exécution de tous les outils nécessaires au fonctionnement de l'application). En alternative, décrivez le mode d'utilisation dans le rapport.]
4. En option, vous pourrez fournir une vidéo avec un exemple d'exécution de l'application.

Délai pour le rendu : dimanche 27 novembre 2022, 23h55 UTC+1

Le travail sera envoyé à Philippe.Truillet@univ-tlse3.fr
(Si vous avez des fichiers trop lourds à envoyer, vous pouvez utiliser un service cloud ou de transfert comme https://www.wetransfer.com)

Chaque jour de retard se verra infligé 0,25 pt de pénalité.

Liens

- Richard Bolt, "Put that there": Voice and Gesture at the Graphics Interface, Proceedings of SIGRAPH'80, **https://dl.acm.org/citation.cfm?id=807503**
- Denis Lalanne, Laurence Nigay, Philippe Palanque, Peter Robinson, Jean Vanderdonckt, Jean-François Ladry, "Fusion Engines for Multimodal Input: A Survey", Proceedings of ICMI-MLMI'09, **http://iihm.imag.fr/publs/2009/FinalSurvey.pdf**
- Sharon Oviatt, "Ten Myths of Mutimodal Interaction", Communication of the ACM, november 1999, **https://pdfs.semanticscholar.org/440a/4e4e842968c58a45ac1e920abfda1c4803bc.pdf**
- Dean Rubine, "Specifying gestures by example", Proceedings of SIGRAPH'81, **https://dl.acm.org/citation.cfm?id=122753**
- Marcos Serrano, « Interaction multimodale en entrée : Conception et Prototypage », Thèse en Informatique de l'Université de Grenoble, 2010, **https://tel.archives-ouvertes.fr/tel-01017242/document**