PERCEVAL: une station automatisée de tests de PERCeption et d'EVALuation auditive et visuelle

Alain Ghio, Carine André, Bernard Teston, Christian Cavé

PERCEVAL□: UNE STATION AUTOMATISÉE DE TESTS DE PERCEPTION ET D'EVALUATION AUDITIVE ET VISUELLE

Alain Ghio, Carine André, Bernard Teston, Christian Cavé

Résumé

Les procédures liées à la réalisation d'un test de perception sont souvent délicates à mettre en œuvre et prennent un temps considérable. Il est donc intéressant d'automatiser autant que possible toutes les opérations nécessitées par ce type d'investigation et de procéder à un recueil informatisé des réponses. Le dispositif que nous présentons répond à ces objectifs. Pour permettre une grande flexibilité, le déroulement de l'expérience est contrôlé par un fichier pilote. Le logiciel se comporte alors comme un interpréteur de commandes qui sont lues dans le fichier sélectionné. Ce type de fonctionnement offre une grande souplesse et permet de réaliser une grande variété de tests tels que décision lexicale, identification lexicale, monitoring de phonème, gating, catégorisation phonétique, mesure d'intelligibilité et évaluation de la qualité vocale. Un soin particulier a été porté à la précision temporelle du déroulement du programme qui reste un point crucial dans le domaine des tests de perception informatisés.

Mots-clés : perception, test informatisé, visuel, auditif.

Abstract

Since perception tests are highly time-consuming, there is a need to automate as many operations as possible, such as stimulus generation, procedure control, perception testing and data analysis. The computer-driven system we are presenting here meets these objectives. To enhance flexibility, the tests are controlled by scripts. The system's core software resembles that of a lexical-syntactic analyzer, which reads and interprets script files sent to it. The execution sequence (trial) is modified in accordance with the commands and data received. This type of operation is highly flexible and supports a wide variety of tests such as auditory-lexical decision making, phoneme monitoring, gating, phonetic categorization, word identification, intelligibility assessment, voice quality, etc. To ensure good performance, we were careful about timing accuracy, which is the greatest problem in computerized perception tests.

Keywords : perception, computerized test, visual, auditory.

GHIO, Alain ; ANDRÉ, Carine ; TESTON, Bernard ; CAVÉ, Christian (2003), PERCEVAL : une station automatisée de tests de Perception et d'Évaluation auditive et visuelle, *Travaux Interdisciplinaires du Laboratoire Parole et Langage,* vol. 22, p. 115-133.

1. Introduction

Il est bien connu que les tests de perception auditive et visuelle impliquent généralement des protocoles expérimentaux longs et coûteux en temps, non seulement du fait du nombre de sujets nécessaires compte tenu de la variabilité individuelle, mais aussi parce que seul un grand nombre de réponses permet de valider statistiquement les résultats. L'utilisation de l'outil informatique dans ce domaine apparaît donc essentielle. D'autre part, en plus du gain de temps, certains tests ne peuvent s'envisager que sous forme informatique comme ceux nécessitant de l'interactivité ou de l'adaptation au sujet.

Le Laboratoire Parole et Langage s'est engagé dans l'automatisation des tests de perception depuis de nombreuses années. La première version du dispositif PERCEVAL (station automatisée de tests de Perception et d'Evaluation auditive) a été développée en 1992 pour répondre à un contrat nécessitant de recueillir rapidement près d'un million de réponses pour évaluer l'intelligibilité de la parole produite par différents vocoders (Cavé *et al.,* 1993). L'enregistrement d'une telle masse de données rendait nécessaire l'utilisation d'un dispositif semi-automatique.

Une première version du dispositif permettait de tester 8 sujets simultanément (Cavé *et al.,* 1998). Chaque auditeur disposait d'un casque audio phonique, d'un écran et d'un boîtier de réponses. L'ensemble était piloté par un micro-ordinateur fournissant les stimuli audiovisuels et recueillant la réponse de chaque sujet. L'opérateur pouvait vérifier le bon déroulement de l'expérience en consultant un écran de contrôle à sa disposition. Ce système a été employé de façon intensive notamment pour la mesure de l'intelligibilité de la parole en milieu difficile (Cavé *et al.,* 1996), pour la perception des durées vocaliques (Cavé *et al.,* 1997) et pour l'étude des confusions auditives chez les adultes implantés cochléaires (Cavé *et al.,* 1999).

Il est facile d'imaginer qu'un tel dispositif à 8 postes de tests (sans compter l'installation de pilotage) nécessite une place conséquente, reste peu transportable et encore moins diffusable ou duplicable. Or, les besoins au sein du laboratoire ont fait apparaître la nécessité d'un système de test plus léger, individuel et portable, notamment pour rendre possible les expérimentations sur le terrain (écoles, instituts médicaux, personnes âgées à domicile...). D'autre part, le dispositif à huit postes simultanés ne permet pas, bien évidemment, de tests interactifs où, par exemple, un renforcement positif ou négatif doit apparaître en fin de réponse. À la suite de ce constat a été développée la version monoposte de PERCEVAL que nous présentons ici.

2. Les systèmes informatisés de tests de perception

Il existe de nombreux logiciels de tests de perception : PsyScope (Cohen *et al.,* 1993), EXPE (Pallier *et al.,* 1997), DMDX (Fallier *et al.,* 2003), Inquisit (www.millisecond.com)... Généralement, ces applications acceptent une grande variété de tests issus de la psychologie expérimentale. Certains sont faciles d'utilisation, spécialement ceux qui sont conçus autour d'une interface graphique de programmation. Toutefois, ces outils sont, pour la plupart, destinés à un type d'expériences dérivées de la psychologie expérimentale et peuvent être mal adaptés à d'autres champs d'investigation comme, par exemple, la phonétique ou plus généralement la communication parlée. De plus, certains de ces programmes ont été développés pour fonctionner avec des systèmes d'exploitation obsolètes ou pour du matériel très spécialisé (clavier de réponse ou carte audio spécifiques). Or, de telles spécifications techniques deviennent très souvent des sources de problèmes lourdes de conséquence, en particulier pour la maintenance à long terme.

Pour développer la nouvelle version du dispositif PERCEVAL, nous avons considéré les contraintes suivantes :

1. possibilité de stimulation sonore et visuelle simultanée.
2. enregistrement de la réponse du sujet par boîtier à boutons ou par le clavier de l'ordinateur.
3. mesure du temps de réponse avec une précision de l'ordre de la milliseconde.
4. grande flexibilité dans la conception des expériences.
5. logiciel ergonomique, facile à utiliser.
6. compatibilité avec la station de test à 8 postes.
7. fonctionnement sur PC sous environnement Windows 98/2000/XP.

Nous aborderons chacun de ces points.

3. Considérations matérielles

3.1. Configuration minimale

Le matériel nécessaire au fonctionnement du dispositif PERCEVAL est un micro-ordinateur de type PC (Windows 98/2000/XP) équipé d'une carte vidéo SVGA et d'une carte son multimédia. Le poste peut être un portable.

3.2. Boîtier de réponse

Un boîtier à boutons peut être ajouté pour enregistrer les réponses des sujets (figure 1, d). Ces boîtiers peuvent être facilement construits à partir de joysticks ou gamepads USB

(figure 1, a) desquels sont extraits le circuit imprimé avec ses composants et câbles (figure 1, b). Le circuit est ensuite inséré dans un boîtier plastique auquel sont fixés des boutons poussoirs de type micro-rupteur à bascule caractérisés par une faible course, un contact très franc et un temps de commutation rapide, homogène, bien adapté à la mesure des temps de réaction (figure 1, c). Ces poussoirs sont reliés au circuit imprimé à la place des boutons du joystick. Le boîtier ainsi obtenu (figure 1, d) se connecte ensuite sur un port USB du micro-ordinateur. La plupart du temps, il est immédiatement reconnu par le système d'exploitation.

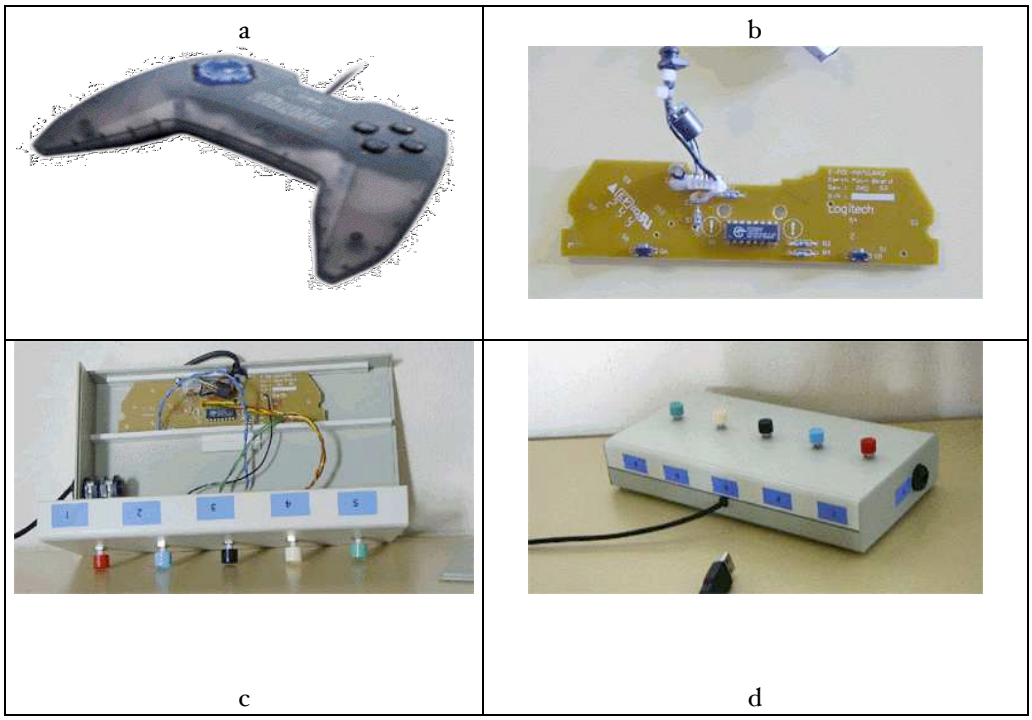


Figure 1
Fabrication d'un boîtier de réponses à partir d'un gamepad USB

3.3. Stimulation sonore

Le contrôle des conditions d'écoute lors d'un test de perception auditive est évidemment crucial. Il est possible d'utiliser directement la sortie de la carte audio du micro-ordinateur et de brancher un casque ou des haut-parleurs «multimédia». Toutefois, il est

préférable de passer par un amplificateur HI-FI auquel sont connectés soit une paire d'enceintes adaptées à l'écoute en champ libre, soit un casque audio phonique de qualité. Pour notre usage, nous avons choisi des casques électrodynamiques Beyer BT100 pour leur bonne sensibilité, leur bonne réponse en fréquence, leur robustesse et leur confort.

4. Problématique de la précision temporelle dans les dispositifs informatisés de test de perception

Selon les objectifs scientifiques, la précision temporelle du déroulement du test peut s'avérer cruciale comme, par exemple, dans les expériences de «□priming□» où un délai entre deux vues doit être contrôlé très précisément. La mesure du temps de réponse peut être aussi un facteur important dans l'exploitation des résultats. Cette mesure peut renseigner sur la difficulté de la tâche de perception et s'avère parfois le seul résultat exploitable dans le cas où le paradigme testé concerne des effets saturés. À titre d'exemple, pour tester la qualité de différentes synthèses vocales, le protocole expérimental peut consister à faire écouter des phrases synthétiques assertives (ex□: «□Paris est la capitale de la France□») et de demander au sujet de répondre par oui ou par non. La réponse en soit n'a que peu d'intérêt. Par contre, le temps de réponse peut renseigner sur la charge cognitive du sujet qui augmente dans le cas de synthèse de mauvaise qualité.

4.1. Pourquoi la précision temporelle est-elle difficile à contrôler dans les systèmes informatisés de tests de perception□?

Le contrôle et la mesure du temps dans les systèmes informatisés de tests de perception ont été abordés depuis longtemps mais restent toujours des sujets d'actualité (Cohen *et al.,* 1993□; Cavé *et al.,* 1995□; Pallier *et al.,* 1997□; Myors, 1999□; Mac Innes *et al.,* 2001□; Forster *et al.,* 2003□).

Les logiciels qui fonctionnent sur un système d'exploitation mono tâche (ex□: DOS) ont l'avantage de permettre une maîtrise parfaite du déroulement temporel des instructions. En fait, dans un tel environnement, le programme s'exécute séquentiellement sans «□interruption□longue□» et la majeure partie de la puissance de l'ordinateur est employée par le programme. Un tel fonctionnement garantit, a priori, une très bonne précision pour les tests de perception informatisés où le temps est un facteur important.

Par opposition, dans des systèmes d'exploitation multi tâches (ex□: Microsoft Windows, Linux), l'exécution d'un programme peut être interrompue pendant un laps de temps variable parce que l'ordinateur partage la puissance de son processeur entre toutes les applications fonctionnant simultanément. Dans le cadre de tests de perception

informatisés, ce phénomène peut s'avérer catastrophique si, par exemple, le programme de test est inactif au moment où une image à l'écran doit être affichée où si le sujet répond à ce moment-là. Cela peut entraîner des imprécisions temporelles.

Il faut enfin préciser que ce qui importe, c'est le système d'exploitation utilisé et non l'environnement prévu dans lequel a été développé le programme. Ainsi, une application DOS (mono tâche) s'exécutant sous Windows (multi tâches) est soumise à ce partage de temps d'exécution et se comporte comme tout autre programme «□Windows□», perdant ainsi sa «□séquentialité□» parfaite. Autrement dit, pour espérer un déroulement temporel sans «□rupture□» d'exécution, il est nécessaire de faire fonctionner le PC sous DOS exclusivement (et non dans une console DOS sous Windows 98, encore moins dans une invite de commande sous Windows 2000/XP où le DOS n'a littéralement plus d'existence).

4.2. Pourquoi est-il nécessaire de développer des applications fonctionnant dans des environnements multi tâches□?

La plupart des applications développées en DOS (mono tâche) nécessitent un matériel prévu à l'avance et dédié à cette utilisation (Pallier *et al.,* 1997). En effet, les instructions relatives à l'image et au son sont écrites pour accéder directement à une carte vidéo ou audio spécifique. Or, la très grande majorité des cartes graphiques et des cartes audio actuelles sont livrées sans aucune possibilité de fonctionnement sous DOS. De plus, aucun PC récent n'est livré pour fonctionner dans cet environnement qui inévitablement va disparaître. Il apparaît donc clairement que pour assurer son avenir, toute nouvelle application doit être prévue sur la base d'un système d'exploitation moderne et donc multi tâches.

4.3. Pourquoi certaines solutions ne sont-elles que partiellement adaptées au problème de la précision temporelle□?

Une solution permettant de garder un contrôle précis du déroulement temporel d'un programme peut consister à donner une priorité maximale à la tâche de test, ce qui signifie que l'ordinateur va exécuter de façon privilégiée le process en question. Mais malgré cette précaution, il n'existe aucune garantie que le programme ne sera pas interrompu dans une phase critique par d'autres tâches de haute priorité (ex□: accès disque, accès réseau...).

L'utilisation de «□timers□» peut être aussi une solution. Un «□timer□» est un moyen d'exécuter une fonction de façon régulière. Par exemple, il peut être utilisé pour détecter si le sujet a répondu (en consultant systématiquement l'état du clavier de réponse).

Cependant, les «□timers□» standard disponibles sous Microsoft Windows ont une résolution d'environ 50 ms, ce qui est loin de nos spécifications prévues (1 ms). Enfin, les timers dits multimedia offrent une résolution de 1 ms mais cette précision n'est pas garantie.

4.4. Et le problème de la latence□?

Parallèlement à l'aspect «□interruptible□» des programmes, point précédemment traité, vient s'ajouter le problème de la latence de certaines instructions. On appelle latence le délai qui peut apparaître entre le moment où une commande est lancée (ex□: afficher une image, produire un son) et le moment où cette commande est réellement effectuée. Par exemple, sur un Pentium III 1.2 GHz, la commande *PlaySound("stimulus.wav")* qui permet de faire écouter un fichier audio possède une latence d'environ 100 ms. Cela signifie que si un chronomètre est lancé immédiatement après l'instruction *PlaySound* pour enregistrer le temps de réponse du sujet, la mesure de ce temps sera entachée d'une erreur car le stimulus audio aura commencé 100 ms après ce qui était prévu. Une solution pour pallier cet inconvénient consisterait à mesurer certains temps de latence critiques et à en tenir compte a posteriori dans l'analyse des résultats. Malheureusement, la réalité montre que ces délais peuvent varier grandement en fonction d'un ensemble de facteurs, ce qui rend la solution inadaptée.

4.5. Comment obtenir une précision temporelle acceptable□?

Tout d'abord, le micro ordinateur destiné à servir de dispositif de test de perception doit être dédié à cette application et uniquement à cette application durant le test (Forster *et al.,* 2003). L'opérateur doit arrêter toutes les tâches susceptibles d'utiliser du temps processeur ou de ralentir la machine (ex□: anti-virus, réseau, CD-Rom..)

Pour obtenir une bonne précision temporelle dans PERCEVAL, nous avons opté pour la technologie DirectX. DirectX fournit un ensemble d'outils de programmation qui contiennent des fonctions agissant comme un pont entre le matériel et le logiciel, permettant une communication directe très rapide sans passer par une succession d'échanges entre différents niveaux qui ralentissent considérablement la vitesse d'exécution. Cette technologie est utilisée de façon intensive par les concepteurs de jeux vidéo très gourmands en ressources audio, graphiques et joysticks. La conception de dispositif informatisé de test de perception répondant un peu aux mêmes contraintes, nous nous sommes donc orientés dans cette direction.

Le kit de développement DirectX est composé de différents éléments. DirectSound fournit des fonctions à très faible latence pour la gestion de la carte audio (écoute, mixage, enregistrement...). À titre de comparaison, sur un Pentium III 1.2 GHz, la fonction DirectSound□::*Play* qui permet de produire un son possède une latence d'environ 0,5 ms bien inférieure aux 100 ms de la commande standard *PlaySound("stimulus.wav")* (voir ci-avant le § sur la latence).

DirectDraw permet d'accéder directement à la mémoire vidéo, au blitter (puce électronique conçue pour accélérer le traitement des images) et au «□flipping surface support□» (méthode consistant à afficher une image tout en travaillant sur une autre en mémoire, ce qui permet d'éviter tout scintillement car on n'affiche les modifications qu'une fois terminées, en intervertissant un écran vidéo logique et l'écran physique). Du fait de leurs caractéristiques, ces deux composants rendent possible une réduction drastique des latences pour la production sonore et les fonctions d'affichage.

Le troisième composant DirectX utilisé est DirectInput, qui autorise le programme à accéder très rapidement aux différents matériels tels que clavier, souris ou joysticks. La lecture des informations provenant de ce type d'instrument peut s'effectuer de deux façons□: par scrutation ou par stockage des données. La scrutation («□polling□») consiste à décoder en permanence l'état du matériel. Un événement, comme par exemple l'appui sur un bouton, sera repéré par tout changement entre deux états successifs. Par opposition, le stockage des données («□buffered data□») est une méthode où un ensemble d'événements est enregistré et stocké à bas niveau jusqu'à ce que l'application vienne retrouver ces données. Si la première méthode permet une très grande précision temporelle (quelques microsecondes entre deux scrutations successives), elle reste sujette aux phénomènes d'interruption qui peuvent interrompre la scrutation pendant une durée imprévisible, ce qui peut la rendre inefficace. Par opposition, la méthode de stockage des données est immunisée contre ces phénomènes d'interruption car la lecture et le stockage des évènements s'effectue à très bas niveau (quasiment au niveau matériel) indépendamment du processeur. Ces mesures sont donc parfaitement fiables. En revanche, la précision temporelle à laquelle sont repérés les évènements reste inconnue et sans doute dépendante du matériel utilisé. Pour résumer, la première méthode fournit des informations très précises mais potentiellement fausses en cas d'interruption. La seconde fournit des informations parfaitement fiables mais potentiellement pas assez précises.

Enfin, il faut garder à l'esprit que la préparation des stimuli est aussi un facteur très important si l'expérimentateur recherche une bonne précision temporelle. En effet, dans

le cas des stimuli sonores, il serait regrettable que la mesure du temps de réaction soit entachée d'une erreur due à une période d'amorce variable selon les stimuli (figure 2). L'utilisation d'outils permettant de couper le signal sonore précisément au début du stimulus apparaît plus que conseillée.

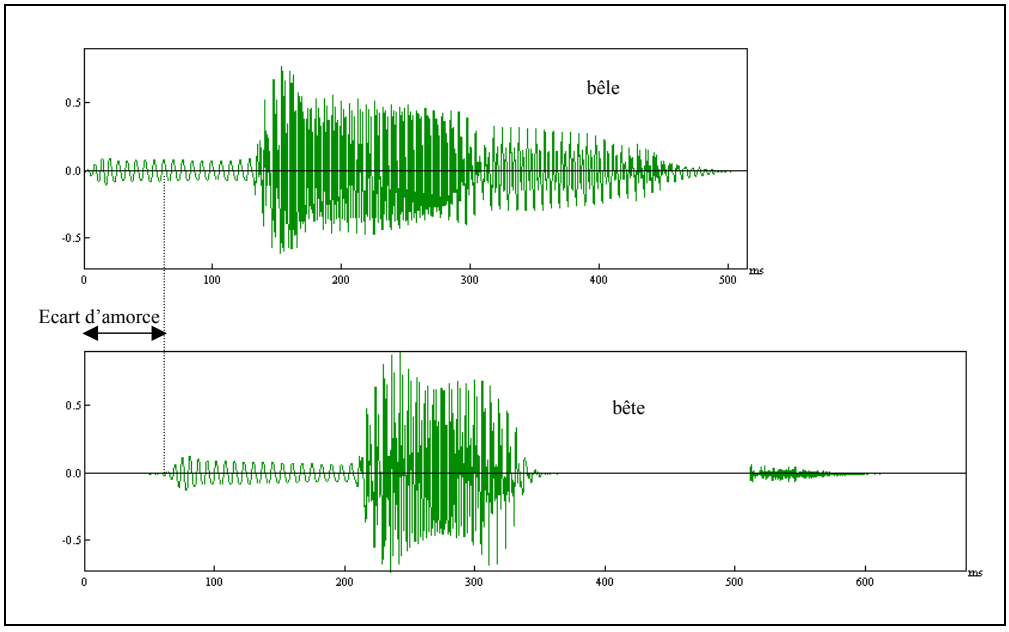


Figure 2
Mauvaise préparation de stimuli sonores ayant des temps d'amorce variables

4.6. Comment orchestrer les différentes opérations à effectuer lors d'une phase de test ?

Lors d'une phase de test, il existe généralement trois types d'opérations à effectuer : production sonore, affichage à l'écran et saisie de réponse(s). Si, d'après ce que nous avons vu précédemment, chacune de ces opérations peut être optimisée de façon à remplir parfaitement sa fonction dans les délais escomptés, la synchronisation et l'orchestration parfaite de ces processus en fonction d'une conception particulière de l'expérience restent essentielle.

Afin d'éviter une programmation séquentielle mal adaptée à ce type de comportement plutôt parallèle, nous avons opté pour une programmation multithread. Dans ce type de programmation, un même processus peut contenir plusieurs chemins d'exécutions distincts, les threads (unités d'exécution). Dans notre cas, nous avons un thread pour

l'audio, un pour l'affichage et un pour la saisie de réponse. Tous les threads tournent indépendamment l'un de l'autre. Windows partage le temps processeur entre eux en fonction de leur priorité respective. Un thread principal très peu gourmand en ressources se charge alors uniquement de fixer les priorités des threads spécialisés en fonction du déroulement prévu de l'expérience. Si, par exemple, la stimulation commence uniquement par la production d'un son puis quelques dizaines de ms après un premier affichage est effectué suivi très vite d'un second affichage, la réponse étant alors attendue, le chef d'orchestre va fonctionner de la façon suivante□:

au début, très haute priorité du thread audio (les deux autres étant en sommeil),

puis mise en sommeil de l'audio (le son continuant à être produit mais sans consommation de ressource car une fois déclenché, le travail s'effectue à très bas niveau sur la carte son) et activation très haute du thread d'affichage,

enfin, activation maximale pour la scrutation de la réponse.

Une telle répartition des fonctions permet un bon contrôle des tâches et évite un passage aléatoire de priorité de tâches.

Enfin, pour conclure sur l'optimisation temporelle, notons qu'il est préférable de programmer de façon anticipée un certain nombre de tâches grandes consommatrices de ressources avant la stimulation proprement dite. Par exemple, lors d'une étape préliminaire, le programme effectue la lecture du fichier audio ou d'une image et stocke les données en mémoire□; puis, dans un second temps, au moment de la stimulation, il lance la production sonore et effectue le rafraîchissement de l'écran par simple flip très rapide.

4.7. Que conclure quant à la précision temporelle dans les dispositifs informatisés de test de perception□?

Le débat reste grandement ouvert comme le prouve la liste des publications opposant les défenseurs inconditionnels des systèmes mono tâche (Myors, 1999□; Mac Innes *et al.,* 2001) et les adeptes du multi tâches (Forster *et al.,* 2003). Il est évident qu'un système mono tâche reste mieux adapté à la réalisation de dispositifs informatisés de test de perception dans lequel l'aspect temporel est essentiel. Malheureusement, ce type de système est en voie de disparition et les expérimentateurs restent à la merci de vieux modèles de matériel, impossibles à remplacer en cas de panne grave.

L'évaluation de la fiabilité d'un dispositif informatisé de test de perception sous Windows (multi tâche) reste ardue. Il est évident qu'un logiciel conçu sans précaution, ni optimisation et fonctionnant sur un PC fortement sollicité par différentes tâches, est

incapable de répondre à des spécifications précises au niveau temporel. En ce qui concerne l'évaluation de la fiabilité, il apparaît nécessaire de mettre en place toute une chaîne d'instrumentation pour valider chacun des aspects du test. Par exemple, pour mesurer la précision de mesure du temps de réaction, il est possible de remplacer l'appui sur un bouton par un signal périodique parfaitement calibré sortant d'un générateur basse tension très précis. À l'instar de Plant *et al.,* (2002), nous nous orientons dans cette direction.

5. Une architecture logicielle modulaire

La partie logicielle du dispositif PERCEVAL inclut différentes applications : le module de test proprement dit, un assistant à la conception d'expérience, un gestionnaire de sujets et résultats, des outils d'aide à la création de stimuli, un module de formatage des résultats (figure 3).

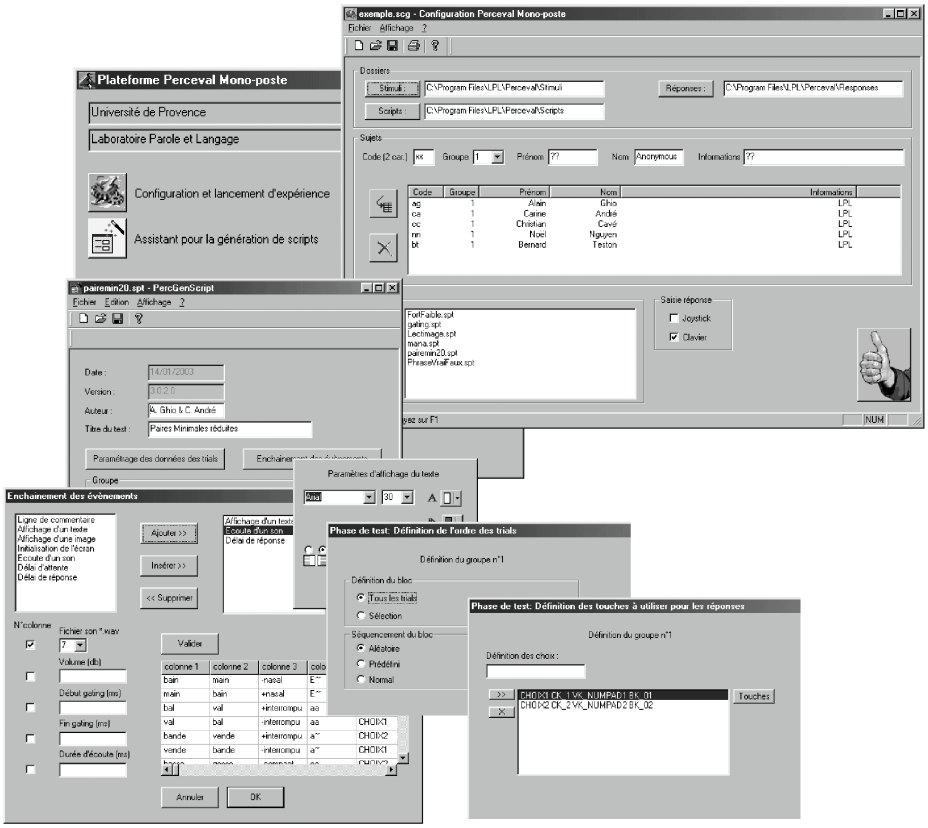


Figure 3
Différents composants logiciels de PERCEVAL
(aide à la conception d'expérience, gestionnaire de sujets...)

Le déroulement d'un test est décrit par un fichier pilote («□script□»), une technique déjà utilisée dans les recommandations du projet européen SAM (Speech Assessment Methodology) pour le système S.O.A.P. (Howard-Jones, 1991). Dans ce type de méthodes, le cœur du logiciel d'expérimentation se comporte comme un analyseur lexico-syntaxique qui lit et interprète le fichier «□script□» sélectionné (Cavé *et al.,* 1995). Il adapte alors le déroulement du test en fonction des commandes et autres informations lues dans le «□script□». Ce type de fonctionnement permet une très grande souplesse dans la conception de test.

Un script est un fichier texte composé de différentes sections□(figure 4)□:

des informations générales (ex□: auteurs, date, titre du test...)□;

des données sous forme de tableau (ex□: des mots, des durées, des caractéristiques phonétiques, des noms de fichiers audio ou images...)□;

des évènements (produire un son, afficher une image...)□;

des réglages (ordre des stimuli, touches de réponses, police de caractère, couleur d'écran, format du fichier de réponses...).

Les scripts peuvent être directement écrits à la main à condition d'utiliser une syntaxe correcte. En complément, un utilisateur novice peut suivre les instructions données par une application assistant la conception, ceci dans un environnement facile à utiliser à base de menu déroulant (figure 3).

Pour chaque sujet soumis au test, un fichier de réponses contenant les informations nécessaires à l'exploitation des résultats est généré. Ces fichiers sont écrits sous forme de texte tabulé et peuvent être directement lus dans un tableur (ex : Excel) ou dans des applications de traitement statistique (figure 5).

6. Conception de l'expérience et résultats

Nous présentons quelques exemples d'expériences adaptées aux sciences du langage.

6.1. Exemple de script pour le test des paires minimales (fig. 4)

Le test de paires minimales est construit sur une opposition de traits distinctifs avec réponse à choix forcé (Peckels *et al.,* 1990). Le déroulement est simple et consiste à faire écouter un mot, à afficher deux mots entre lesquels le sujet doit choisir et à enregistrer la réponse. L'exploitation des réponses est facilitée en incluant le trait phonétique testé ainsi que le contexte vocalique dans le fichier de réponses.

Pour comprendre la structure d'un script, nous utiliserons l'exemple de la figure 4 ci-après.

```
[INFORMATION]
AUTHOR=A. Ghio & C. André
DATE=14/01/2003
TITLE=Paires Minimales réduites
VERSION=3.0.2.0

[TRIAL_DATA]
TRIAL1=  <1) main 2)bain>     <bain.wav>    <Choix2>  <-nasal>          <E~>
TRIAL2=  <1)bain 2)main>      <main.wav>    <Choix2>  <+nasal>          <E~>
TRIAL3=  <1)bal 2)val>        <bal.wav>     <Choix1>  <+interrompu>     <aa>
TRIAL4=  <1)val 2)bal>        <val.wav>     <Choix1>  <-interrompu>     <aa>
…        #1                   #2            #3        #4                #5

[TRIAL_EVENTS]
X10=BEGIN
X20=DISPLAY_TEXT<#1>
X30=PLAY_SOUND<#2>
X40=GET_INPUT<DELAY 2000>
X50=END

[SETTINGS_GROUP1]
INSTRUCTION_FORMAT=<Pairemin.txt>     *1
TRAINING_ORDER=<1 3 4 6>              *2
TRIAL_ORDER=<RANDOM>                  *3
TEXT_FORMAT=<FONT Arial><SIZE 30><BKCOLOR 0x0000FF>…< POSITION HCenter|VCenter>    *4
INPUT=<Choix1 CK_1 VK_NUMPAD1 BK_01><Choix2 CK_2 VK_NUMPAD2 BK_02> *5
CORRECT=<#3>                          *6
PAUSE=500                             *7
RESPONSE_FORMAT=<$SUBJECT><$TRIAL><#1><#2><#3><$RESPONSE><$ERROR><#4><#5><$RTIM
E> *8
```

Figure 4
Exemple de script

Comme nous l'avons décrit précédemment, la section ***[INFORMATION]*** apporte des informations générales. La section ***[TRIAL_DATA]*** contient un ensemble de données.

La section ***[TRIAL_EVENTS]*** décrit le déroulement du test. Dans l'exemple de la figure 4, la commande de la ligne X20 permet d'afficher à l'écran le contenu de la 1ère colonne (#1) des *TRIAL_DATA*. Par exemple, pour le trial n°1 sera affiché le texte « 1) main 2) bain ».

La commande de la ligne X30 permet de faire écouter le fichier audio dont le nom est fourni dans la 2ème colonne (#2) des *TRIAL_DATA*. Dans notre exemple, le fichier « *bain.wav* »sera produit. La commande de la ligne X40 déclenche l'attente de la réponse du sujet. Dans notre exemple, un délai de réponse de 2000 ms sera accordé au sujet. Si ce temps est dépassé, le résultat sera considéré comme une non réponse.

La section ***[SETTINGS_GROUP1]*** définit la configuration du test. Plusieurs configurations sont possibles dans un même script, chacune correspondant à un groupe de sujets. Au début du

test, des instructions au sujet sont affichées à l'écran. Ces instructions sont contenues dans un fichier texte dont le nom est, par exemple, « Pairemin.txt » (cf. *1).
Une phase d'entraînement peut être exécutée. Les trials utilisées sont définis dans *2.
Durant la phase de test proprement dite, l'ordre des trials peut être fixé à l'avance ou proposé dans un ordre aléatoire (cf. *3).
Le format d'affichage (police de caractère, taille, couleur, etc.) est paramétré en *4.
Les touches de réponses sont listées en *5: touches du clavier standard (CK_...), du pavé numérique (VK_...) ou du boîtier à boutons (BK_...).
Selon la nature du test, il est possible de coder directement la réponse comme correcte ou incorrecte (cf. *6). Dans notre exemple, cette information est contenue dans la 3ème colonne (#3) des *TRIAL_DATA*.
La pause (cf. *7) définit le temps d'inactivité entre deux trials.
Les résultats seront écrits dans un fichier réponse dans un format décrit en *8. La figure 5 fournit un exemple obtenu avec le script ci-dessus.

A	B	C	D	E	F	G	H	I	J
$S	$T	#1	#2	#3	$RESP	$ER	#4	#5	$RT
ca	4	1)val 2)bal	val.wav	Choix1	Choix2	err	-interrompu	aa	1072
ca	1	1)main 2)bain	bain.wav	Choix2	Choix2	ok	-nasal	E~	748
ca	14	1)latte 2)ratte	ratte.wav	Choix2	Choix1	err	+compact	aa	1012
ca	17	1)veste 2)ouest	veste.wav	Choix1	Choix1	ok	-vocalique	EE	830
ca	20	1)furent 2)surent	surent.wav	Choix2	Choix2	ok	+grave	yy	1066
ca	12	1)dette 2)bette	bette.wav	Choix2	Choix2	ok	-grave	EE	1218
ca	8	1)bosse 2)gosse	bosse.wav	Choix1	xxx	xxx	+compact	oo	0

Figure 5□
Exemple de fichier de réponses au test des paires minimales

A: $SUBJECT spécifie le code du sujet.
B: $TRIAL fournit l'index du trial.
C: #1 est le contenu de la 1ère colonne des *TRIAL_DATA*. Dans l'exemple, il s'agit du texte affiché à l'écran qui correspond au choix forcé.
D: #2 est le contenu de la 2ème colonne des *TRIAL_DATA*. Dans l'exemple, il s'agit du fichier audio produit.
E: #3 est le contenu de la 3ème colonne des *TRIAL_DATA*. Dans l'exemple, il s'agit de la réponse correcte à donner.
F:$RESPONSE est la réponse donnée par le sujet.

G:$ERROR spécifie si la réponse du sujet est correcte ou non (“ok” si correct, “err” si incorrect).
H: #4 est le contenu de la 4ème colonne des *TRIAL_DATA*. Dans l’exemple, il s’agit du trait distinctif testé.
I: #5 est le contenu de la 5ème colonne des *TRIAL_DATA*. Dans l’exemple, il s’agit du contexte vocalique testé.
J: $RTIME est le temps de réponse.

6.2. Exemple de script permettant de faire varier dynamiquement le volume du stimulus sonore (fig. 6)

Dans cet exemple de test, le sujet doit répondre si le son qu’il entend («□aaa.wav□») lui semble fort ou faible. Nous utilisons l’instruction *VOLUME* qui permet d’ajuster dynamiquement le niveau d’écoute. Dans l’exemple de la figure 6, le trial n°1 sera produit de façon normale (0 dB)□; le trial n° 2 sera produit avec une atténuation de 3 dB□; le n° 3 avec une atténuation de 6 dB.

```
[TRIAL_DATA]
TRIAL1=<0>
TRIAL2=<-3>
TRIAL3=<-6>
…

[TRIAL_EVENTS]
X10=BEGIN
X20=DISPLAY_TEXT <1)Fort 2)Faible>
X30=PLAY_SOUND <aaa.wav><VOLUME #1>
X40=GET_INPUT
X50=END
```

Figure 6□
Possibilités de variation dynamique du volume d’écoute

6.3. Exemple de script permettant de faire varier dynamiquement le début et la fin du stimulus sonore (fig. 7)

Dans cet exemple de test, le sujet écoute l’amorce d’un des deux mots produits (“bêle” ou “bête”). Les temps initial (*TIME_BEGIN*) et final (*TIME_END*) de la production sonore sont définis en paramètres (cf. ligne X30, figure 7). Dans cet exemple, la durée de l’écoute est définie dans la 3ème colonne (#3) des *TRIAL_DATA*. Ainsi, le trial n°1 produira les 200 ms initiales de «□bele.wav□», le n° 5 produira les 275 ms initiales de «□bette.wav□»...

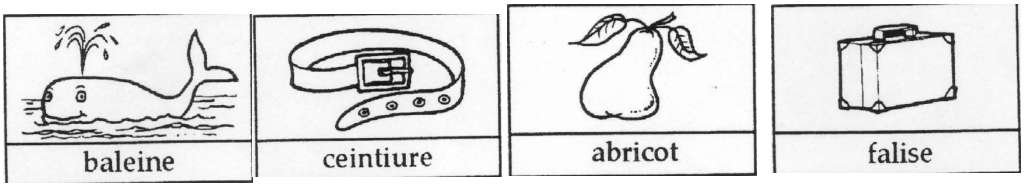
```
[TRIAL_DATA]
TRIAL1= <bêle> <bele.wav>  <200>
TRIAL2= <bêle> <bele.wav>  <250>
TRIAL3= <bêle> <bele.wav>  <275>
TRIAL4= <bête> <bette.wav> <200>
TRIAL5= <bête> <bette.wav> <250>
TRIAL5= <bête> <bette.wav> <275>

[TRIAL_EVENTS]
X10=BEGIN
X20=DISPLAY_TEXT<1) bêle 2) bête>
X30=PLAY_SOUND<#2><TIME_BEGIN 0><TIME_END #3>
X40=GET_INPUT<DELAY 2000>
X50=END
```

Figure 7
Possibilités de variation dynamique du début et de la fin du stimulus sonore

6.4. Exemple de script avec utilisation de feedback (fig. 8)

Certains tests, en particulier ceux s'adressant à des enfants, nécessitent une action après la réponse du sujet (effet de feedback). Il s'agit souvent de renforcement positif si la réponse donnée est correcte et négatif dans le cas contraire. Il peut s'agir d'image ou de son encourageant le sujet ou lui signalant son erreur.

Dans l'exemple ci-après, le sujet est soumis à une image à laquelle est associée un mot écrit et doit décider si le mot correspond à l'image. La non correspondance peut être de différente nature : erreur orthographique, incohérence sémantique…



À la suite de la réponse du sujet, un applaudissement (clap.wav) renforce une bonne réponse tandis qu'un bruit de verre cassé (glass.wav) sanctionne une mauvaise réponse.

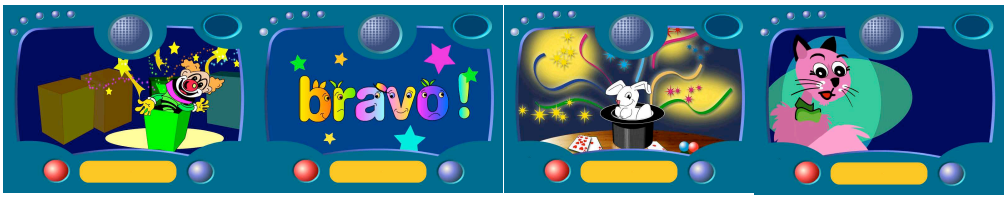
```
[TRIAL_DATA]
TRIAL1=<baleine><correct><baleine.bmp>
TRIAL2=<centuire><faute><centuire.bmp>
TRIAL3=<abricot><faute><abricot.bmp>
TRIAL4=<falise><faute><falise.bmp>

[TRIAL_EVENTS]
X10=BEGIN
X20=DISPLAY_FILEBMP<#3>
X40=GET_INPUT<DELAY 3000>
X50=END

[SETTINGS_GROUP1]
…
SOUND_FEEDBACK=<POSITIVE clap.wav><NEGATIVE glass.wav>
```

Figure 8
Possibilité de renforcement

Pour éviter la monotonie d'un renforcement unique, il est possible d'utiliser une série d'images et/ou de sons.

```
[SETTINGS_GROUP1]
…
SOUND_FEEDBACK=<POSITIVE manaPositive1.wav manaPositive2.wav
manaPositive3.wav manaPositive4.wav><NEGATIVE manaNegative1.wav
manaNegative2.wav manaNegative3.wav manaNegative4.wav>
DISPLAY_FEEDBACK=<POSITIVE manaPositive1.bmp manaPositive2.bmp
manaPositive3.bmp manaPositive4.bmp manaPositive5.bmp><NEGATIVE
manaNegative.bmp>
```

Figure 9
Renforcement multiple



manaPositive1 *manaPositive2* *manaPositive3* *manaPositive4*

7. Conclusion

Le dispositif informatisé de test de perception PERCEVAL donne la possiblité de mettre en œuvre un grand nombre de procédures expérimentales utilisables dans les sciences du langage. Il a été conçu pour automatiser le plus possible les différentes tâches impliquées dans la conception, la réalisation et l'exploitation de ce type d'expérience. L'utilisation en laboratoire a fait apparaître des besoins non satisfaits comme la possibilité d'introduire des intervalles inter-stimuli de durée aléatoire contraignant le sujet à une attention accrue ou encore de réaliser des expériences à doubles tâches («□Dual Task Paradigm□») nécessitant deux réponses de la part du sujet. Nous nous efforçons donc d'ajouter progressivement des fonctionnalités au dispositif actuel. La possibilité de stimulation par des séquences audiovisuelles est aussi étudiée. L'expérience acquise durant la phase de développement de cette version mono poste va, à présent, être exploitée pour la mise à niveau de la station multi postes, ce qui permettra de doter le laboratoire à la fois d'un outil mono poste portable et d'un dispositif permettant d'acquérir rapidement une grande quantité de résultats.

PERCEVAL est disponible sur le site web du Laboratoire Parole et Langage à l'adresse□: www.lpl.univ-aix.fr/-lpldev/perceval/

8. Bibliographie

CAVÉ, C.; MEUNIER, C. (1993) Mesure de l'intelligibilité de parole codée, Procédures expérimentales et mise en œuvre. *Rapport Thomson CSF/CNRS*.

CAVÉ, C.; DI CRISTO, P.; GHIO, A.; TESTON, B. (1995)□Une station de tests automatisée pour l'évaluation de la qualité et l'intelligibilité de la parole, *Travaux de l'Institut de Phonétique d'Aix*, 16, p. 123-139.

CAVÉ, C.; MEUNIER, C.; GHIO, A.; MELLIET, J.-L.; MARCHAL, A. (1996) Effect of speech conditions and gas mixture on the intelligibility of diver's speech as assessed under real diving conditions at 50 and 100 meters, *Proceedings of the 3rd European Conference on UnderWater Acoustics*, p. 765-770.

CAVÉ, C.; MEUNIER, C.; GHIO, A. (1997) Sarah a atterri au Sahara ou la durée vocalique comme indice pour distinguer deux voyelles identifiques consécutives, *Actes du 4ème Congrès Français d'Acoustique*, 1, p. 341-344.

CAVÉ, C.; TESTON, B.; GHIO, A.; DI CRISTO, P. (1998) A Computer-Driven System for Assessing Speech Quality and Intelligibility, *Acta Acustica*, 84, p. 157-161.

CAVÉ, C.; DE SMET, G.; TRIGLIA, J.-M. (1999) French distinctive feature confusions in adult cochlear implantees, *Proceedings of the XIVth International Congress of Phonetic Sciences*, San Francisco, p. 787-788.

COHEN, J.D.; MAC WHINNEY, B.; FLATT, M.; PROVOST, J. (1993) PsySCope: a new graphic interactive environment for designing psychology experiments", *Behavioral Research Methods, Instruments and Computers,* 25(2), p. 257-271.

FORSTER, K.; FORSTER, J. (2003) DMDX: A Windows Display Program with Millisecond Accuracy, *Behavioral Research Methods, Instruments and Computers* (forthcoming).

HOWARD-JONES, P.; SAM Partnership (1991) 'SOAP' - A Speech Output Assessment Package for Controlled Multilingual Evaluation of Synthetic Speech, *Proceedings of Eurospeech 91*, vol. 1, p. 281-283.

MAC INNES, W.J.; TAYLOR, T.L. (2001) Millisecond Timing on PCs and Macs, *Behavioral Research Methods, Instruments and Computers,* 33(2), p. 174-178.

MYORS, B. (1999) Timing accuracy of PC programs running under DOS and Windows, *Behavioral Research Methods, Instruments and Computers,* 31(2), p. 322-328.

PALLIER, C.; DUPOUX, E.; JEANIN, X. (1997) EXPE: an Expandable Programming Language for On-line Psychological Experiments, *Behavioral Research Methods, Instruments and Computers,* 29(3), p. 322-327.

PECKELS, J.-P.; ROSSI, M. (1990) Le test de diagnostic par paires minimales, adaptation au français du «□Diagnostic Rhyme Test□» de Voiers, *Revue d'Acoustique,* 27, p. 245-262.

PLANT, R.; QUNILAN, P.; HAMMOND, N.; WHITREHOUSE, T. (2002) Benchmarking precision in the real world, *Proceedings of 4th International Conference on Methods and Techniques in Behavioral Research,* 27-30 August 2002, Amsterdam, the Netherlands.