Développement d'outils d'évaluation des équivalences modales des primitives émotionnelles

Clarisse Bayol

Développement d'outils d'évaluation des équivalences modales des primitives émotionnelles

BAYOL
Clarisse

Sous la direction de Véronique Aubergé et Romain Magnani

Laboratoire Informatique de Grenoble

UFR LLASIC
Département I3L

Mémoire de master 2 professionnel – 20 crédits – Sciences du langage
Spécialité Industries De la Langue
Parcours Traitement Automatique de la Langue Ecrite et de la Parole

Année universitaire 2015-2016

Ce travail a été partiellement financé par l'Equipex Amiqual4Home ANR-11-EQPX-0002

Développement d'outils d'évaluation des équivalences modales des primitives émotionnelles

BAYOL
Clarisse

Sous la direction de Véronique Aubergé et Romain Magnani

Laboratoire informatique de Grenoble

UFR LLASIC
Département I3L

Mémoire de master 2 professionnel - 20 crédits – Mention Sciences du langage
Spécialité Industries De la Langue
Parcours Traitement de la Langue Ecrite et de la Parole

Année universitaire 2015-2016

Ce travail a été partiellement financé par l'Equipex Amiqual4Home ANR-11-EQPX-0002

Remerciements

Je tiens tout d'abord à remercier Véronique Aubergé pour m'avoir offert la possibilité de faire un stage au sein de ce laboratoire de recherche et pour son encadrement et ses conseils avisés.

Merci à Romain Magnani pour m'avoir encadrée et accompagnée sur ce projet quotidiennement, pour son investissement et sa disponibilité.

Merci également à toute l'équipe de Domus Yuko, Liliya, Fred, Natacha, Nicolas et Maxence pour tous leurs conseils et leur accueil.

Merci à mes collègues de master et à mes proches.

UNIVERSITÉ Grenoble Alpes

Déclaration anti-plagiat
Document **à scanner** après signature
et **à intégrer** au mémoire électronique

DÉCLARATION

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

NOM : BAYOL

PRENOM : Clarisse

DATE : 01/08/2016 SIGNATURE : [redacted]

Sommaire

Introduction

1. Contexte de travail

Ce stage a été effectué au LIG (Laboratoire Informatique de Grenoble) qui rassemble environ 500 chercheurs, enseignants-chercheurs, personnels en support à la recherche et doctorants. Ce laboratoire comporte 5 axes de recherche et 24 équipes de travail différentes.

Mon travail s'inscrit dans l'axe « Traitement de Données et Connaissances à Grande Echelle » au sein de l'équipe GETALP (Groupe d'Etude en Traduction Automatique/Traitement Automatisé des Langues et de la Parole) qui a été créé en 2007, lors de la création du LIG lui-même. Composée de chercheurs en traitement de l'écrit et de la parole, cette équipe pluridisciplinaire se concentre sur différents aspects méthodologiques, théoriques et pratiques du Traitement Automatique du Langage.

Ce stage a été mené avec l'INRIA (Institut National de Recherche en Informatique et en Automatique). Cet organisme public de recherche, dédié aux sciences et technologies du numérique abrite un équipement d'excellence (Equipex) dans le domaine de l'habitat intelligent. Cette plateforme d'expérimentation comprend notamment un appartement intelligent, Amiqual4Home, géré en partenariat avec d'autres plateformes d'expérimentation et de prototypages comme le FabMSTIC ou DOMUS. Dédiée à l'innovation et à l'expérimentation, cette plateforme a été créée en 2015 pour mener différents types de recherches et d'expérimentations autour de la robotique et de l'habitat intelligent.

Le travail a plus particulièrement été effectué dans le bâtiment du CTL (Centre Technologique du Logiciel) qui comprend le Living Lab Domus du LIG. Sur le même principe qu'Amiqual4Home, cette plateforme permet de mettre en œuvre des protocoles d'expérimentation dans un environnement écologique. En effet, elle comprend un appartement domotique qui s'apparente à un appartement réel. Différentes activités sont menées au sein de cet habitat intelligent, telles que la constitution de corpus d'observation, le recueil de comportements ou la capture de signaux multiples. Les différents capteurs installés permettent ainsi d'avoir des données nombreuses et variées (caméras, micros, capteurs de température, de pression, de consommation d'eau et d'électricité…). Le recueil de données au sein d'un environnement écologique est un élément clé pour les recherches

menées au sein des équipes de travail. La constitution d'un corpus nécessite de prendre en compte différentes contraintes pour que le corpus soit représentatif de la réalité.

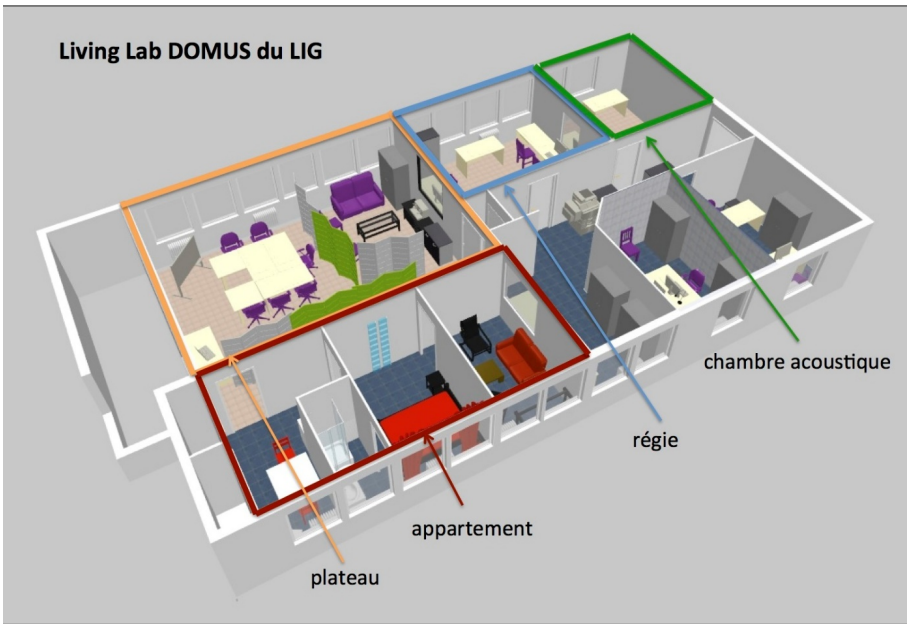

Figure 1 : Living Lab DOMUS LIG

L'utilisation de cette plateforme a évolué au fil du temps. A l'origine, elle était uniquement utilisée comme un laboratoire d'usage qui servait à tester des activités humaines en relation avec des technologies. Le but était alors de voir comment les personnes utilisaient les technologies. Aujourd'hui, l'appartement domotique est utilisé pour tester des hypothèses de recherche scientifique. La validation ou l'invalidation de ces hypothèses nécessite des technologies. Ceci permet également de modéliser les hypothèses posées.

Dans le cadre de l'utilisation de différentes technologies, il y a également un besoin de partenaires industriels qui pourront fournir ces technologies ou bien les industrialiser, une fois mises au point.

La théorie des trois « P » (Public-Privacy-People), utilisée au sein de l'équipe GETALP, implique les partenaires industriels au même titre que les chercheurs et la société. Il s'agit de mettre en place des boucles rapides et agiles qui permettront de nombreuses interactions entre les trois acteurs impliqués dans la validation d'hypothèses.

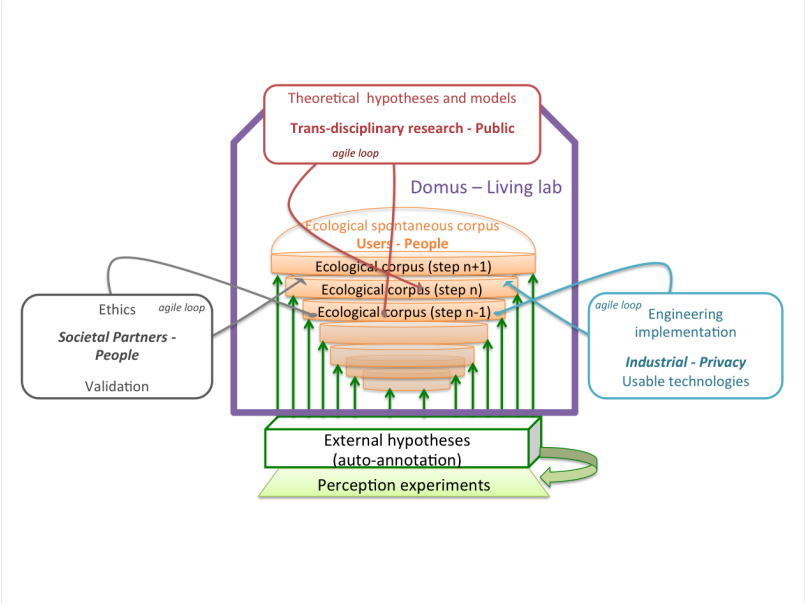

Figure 2 : La théorie des trois « P »

En complément, un protocole d'auto-annotation est utilisé pour obtenir une validation externe. Cette méthode, utilisée dans le corpus EEE par Aubergé et al. (2006), consiste à faire annoter aux sujets eux-mêmes leurs données sans qu'ils le conscientisent. Cette méthode permet d'obtenir les données endogènes au sujet en l'influençant le moins possible, comme par exemple, les informations les plus pertinentes quant au ressenti d'un sujet vis-à-vis d'un robot.

2. *Contexte du cahier des charges :*

Mon travail s'inscrit dans une étude à long terme qui tente de déterminer :

- les dimensions des inductions émotionnelles inhérentes aux entités perçues comme objets par l'humain ;
- les analogies émotionnelles entre les modalités sensori-motrices de perception des objets ;
- l'influence de ces inductions inhérentes aux objets dans l'environnement des objets connectés (Internet of Things) ;
- enfin, c'est le but final des recherches à long terme que mon travail initie, l'intégration de ces inductions émotionnelles d'objets lorsque ceux-ci sont perçus comme animés, c'est-à-dire qu'ils ne sont plus une partie augmentant le corps ou

l'environnement de l'humain, mais une entité perçue comme sujet, augmentant l'espace social de l'humain : l'objet connecté devient robot social.

Tous les travaux réalisés dans l'équipe sont déjà sur la robotique et l'interaction sociale. C'est ainsi d'ailleurs que nous avons pu disposer d'éléments acoustiques déjà bien identifiés dans leurs fonctions interactionnelles (objets acoustiques récoltés par (Audibert, 2008), mesurés perceptivement par (Vanpé, 2011) puis (De Biasi, 2011), utilisés comme déclencheur de glu socio-relationnelles par Sasa).

Pour revenir aux primitives émotionnelles de perception des entités-objets, nous avons conçu, avec mon encadrant Roman Magnani et avec Véronique Aubergé, un protocole d'association entre trois paramètres visuels primitifs combinés (taille, forme, couleur) et les primitives sonores déjà évaluées dans les expériences précitées.

Nous allons ainsi faire un court état de l'art sur l'animacité et les inductions émotionnelles.

3. Etat de l'art

Le caractère de vie associé à une chose pour sa manière de se déplacer est appelé mouvement animé ou animacité. En effet, l'empathie avec une chose est possible dès lors qu'on associe à cette chose de l'animacité. C'est le cas par exemple, lorsque l'on voit quelque chose « sauter » dans la nature. On est alors amené à estimer que ce déplacement était un saut. A l'inverse, le mouvement d'une chose portée par le vent ne sera pas perçu comme animé car on sera capable de définir que cette chose est déplacée par un élément extérieur et qu'elle n'est pas en train de se déplacer par elle-même. Dès lors, l'empathie avec cette chose n'est donc pas possible (Pratt, Radulescu, Guo, & Abrams, 2010). Ces différentes perceptions de mouvement animé ou non sont encore très peu étudiées et peu connues (Pantelis & Feldman, 2012).

Une expérience menée par Heider & Simmel (1944) consistait à afficher sur un écran trois formes abstraites différentes (un carré, un rond et un triangle) en mouvement qui semblaient interagir. Chaque sujet participant à l'expérience avait le même film. Cette expérience a permis de montrer que, bien que les formes étaient abstraites, les sujets ayant participés attribuaient une certaine forme d'animacité à ces dernières. De plus, les sujets y voient même des situations d'évitement, de poursuite, d'attaque ou de protection. Ils

estiment par exemple que « le triangle attaque le rond et le carré qui s'enfuient ». Cependant, un biais était présent dans la méthodologie utilisée pour la réalisation de cette expérience. En effet, c'est l'intentionnalité de la personne ayant mis en mouvement les différentes formes qui a été perçue par les sujets. Lors de la réalisation de l'animation, cette personne voulait sans doute que les formes soient perçues de cette façon.

Plus récemment, une autre étude a été réalisée (Tremoulet & Feldman, 2000) qui avait pour but de réduire le biais énoncé précédemment en réduisant le nombre de stimuli. Seuls un point et un tiret de couleur blanche sont testés lors de cette expérience avec seulement un ou deux types différents de mouvement. Ces deux formes sont en mouvement sur fond gris pour que la couleur blanche des formes soit visible. Lors de cette expérience, les chercheurs ont fait le choix de ne pas utiliser de couleur pour ne pas avoir à prendre en compte l'influence de la couleur (saturation, teinte, luminosité) sur la perception des sujets. Les deux formes utilisées sont présentées aux sujets comme des « microparticules ». Ils doivent alors juger si elles sont vivantes ou non. Cette méthodologie utilisée pour la réalisation de cette expérience nécessite de la part des sujets, qu'ils se mettent dans la peau d'experts en microbiologie. Cela peut être un biais, ou tout simplement un besoin de contextualisation nécessaire pour le bon déroulement de l'expérience.

L'expérience de Kholer (1929) a permis de mettre en avant l'effet Kiki Bouba, plusieurs fois testé et confirmé avec différents stimuli. Cette expérience a permis de montrer une association récurrente d'une forme à un mot, de la part des sujets (95% des sujets font la même association, (Ramachandran & Hubbard, 2001)). « Kiki » est le plus souvent associé à la forme pointue, et « Bouba » à la forme ronde. Ainsi, Sagisaka (2014) a mis en évidence par transfert de modalité que « Bouba » est perçu comme rond, blanc et « gentil ». Au contraire, « Kiki » est pointu, rouge et « agressif ». Des études complémentaires (Milan et al., 2013) ont montré que les sujets attribuent des personnalités à ces formes. De plus, le mouvement, la dynamique et la forme peuvent avoir une grande influence sur l'association de ces formes à l'un de ces deux mots. Cela pourrait donc également impacter le degré d'empathie.

Aubergé et al. (2014) ont récemment doté un robot non androïd, ayant pour rôle domotique le contrôle d'un appartement intelligent, de primitives sonores non lexicales (bruits de bouche) extraits de corpus naturels comme feedback. Ceci a permis de montrer qu'un usager en situation d'isolement pouvait s'attacher dans une glu socio-relationnelle dont les dimensions (positives altruistes) sont contrôlables.

L'attachement est une notion large à laquelle nous sommes confrontés tous les jours. Evolutif et progressif, il peut être lié à la complexité de l'intitulé naïf « compagnon, ami ». Cet attachement est en constante évolution lorsque les individus sont en contact (physique ou non), par le biais des rôles sociaux. L'animal « de compagnie » (Lestel, 2009) a eu différents rôles fonctionnels avant d'obtenir ce dernier statut. En occident par exemple, le chien était chien de berger avant de devenir un chien social. Ou encore, en inde, la vache était outil agricole avant de devenir sacrée.

En effet, un robot peut être perçu comme ayant une apparence et des mouvements intentionnels lorsqu'il possède un rôle de technologie de services. C'est pourquoi, même si ces services sont créés ou hérités de l'humain, ce robot pourra se voir attribuer un rôle de compagnon. Ceci serait peut-être l'élément distinctif entre une machine considérée comme un objet technologique (augmentation de l'humain) et une machine considérée comme un sujet/robot (réparation et augmentation de l'espace social).

Nous pouvons ainsi dire qu'un robot ne naîtrait pas compagnon mais le deviendrait inévitablement par la perception que les individus en ont.

La vallée de l'étrange met en évidence des pics générateur d'inquiétude liés au rejet d'un robot qui aurait une apparence ou un rôle trop proche de celui de l'humain. Il est très difficile de mesurer l'impact de ces artefacts. Cependant plus le robot ressemble à quelque chose de connu, plus il est « dérangeant ».

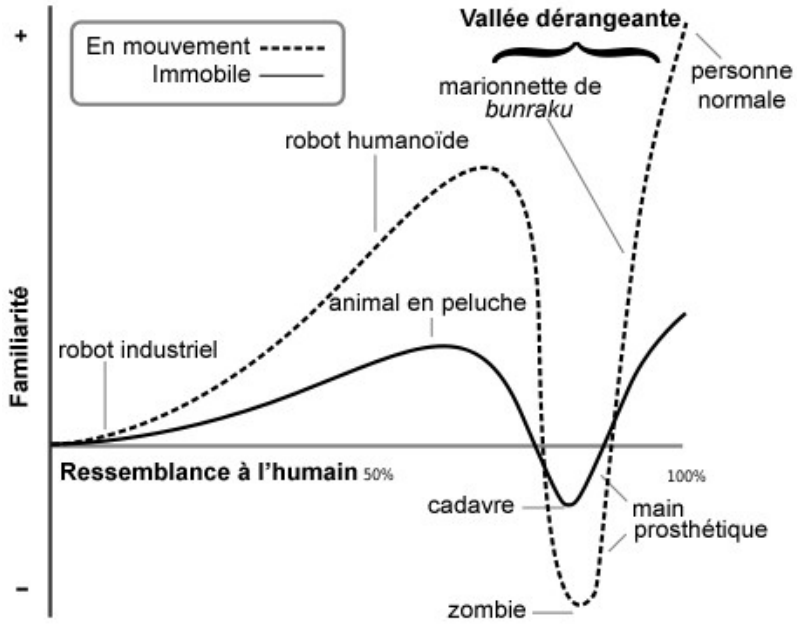

Figure 3 : La vallée de l'étrange

(http://www.paranormal-encyclopedie.com/wiki/Articles/Vall%E9e_d%E9rangeante)

Au-delà d'une apparence très semblable mais qui n'est plus dérangeante, la gestualité et les comportements du robot ont également un rôle crucial dans le contenu communicationnel.

L'apparence d'un robot joue également un rôle crucial dans la présence ou non d'illusion empathique. Une expérience de Rosenthal Von der Pütten et al. (2012) mettait en scène le robot jouet Pleo que l'on traitait de manière brutale ou affectueuse. Les chercheurs ont alors observé grâce à des capteurs que les sujets réagissaient à cette mise en scène comme si le robot était un humain.

Une autre expérience de Suzuki et al. (2015) montrait des résultats semblables lorsque l'on demandait aux sujets de couper les « doigts » d'un robot androïd non réaliste.

Enfin, une vidéo récente a suscité la polémique sur internet. En effet, cette vidéo montrait le crash test réussi, du robot androïd non communicant Atlas de Boston Dynamics. Ce robot étant capable de se déplacer, de maintenir une position et de se relever en cas de chute, la vidéo montre des expérimentateurs pousser le robot pour qu'il tombe et lui enlever des objets des mains pour voir à quel point le robot est capable de se relever ou de récupérer des objets qu'on lui a enlevé. Les expérimentateurs ont été perçus comme des individus brutaux.

C'est pourquoi les robots actuels, ayant pour but d'être des robots compagnons, tendent à s'éloigner le plus possible des pics de la vallée de l'étrange.

On obtient alors des robots, qui de par leur apparence et sans mouvement ou interaction seraient considérés comme des objets. Ainsi, quels sont les facteurs d'apparence et de dynamique qui font que l'on a plutôt une perception d'un objet (augmentation de soi ou de l'environnement) ou au contraire plutôt la perception d'un « autre » ?

4. Cahier des charges :

Le but de ce stage était de développer une application sous androïd qui permettra de tester les primitives visuelles et sonores lors d'un scénario écologique. Pour cela il fallait que cette application contienne une page avec une forme en mouvement ainsi qu'une autre page permettant de lancer des animations et des sons sur la première page (qui puisse ainsi être pilotée par le magicien). De plus, pour que l'application soit la plus complète possible, il était nécessaire d'avoir la possibilité de créer des formes, couleurs, tailles,

mouvements… Le but était donc de permettre le réglage d'un maximum de paramètres pour que l'application permette le plus de choses possibles. Ces nouvelles formes créées devaient pouvoir être enregistrées et appliquées à la page qui contenait la forme en mouvement.

La deuxième partie consistait à réaliser un test de perception avec des primitives sonores et visuelles. En effet, il était nécessaire de faire un site qui permettrait de pouvoir jouer un son et de pouvoir choisir une forme parmi les 12 sélectionnées pour le test. Le point important pour la réalisation de ce test était la facilité d'utilisation pour l'utilisateur. Il fallait que ce soit le plus ergonomique possible (le moins de clics possibles pour l'utilisateur, pas besoin de scroller…).

Nous allons donc développer ces deux parties dans ce mémoire.

Partie 1

-

Réalisation du projet

Chapitre 1. Structure NodeJS

L'enjeu principal dans la réalisation de cette interface était de pouvoir piloter une forme à distance. C'est-à-dire qu'il était nécessaire de pouvoir envoyer des instructions depuis une page qui seront exécutées sur une autre. Pour cela, une architecture NodeJs a été utilisée. Celle-ci permet de créer un serveur en JavaScript depuis lequel on pourra envoyer ou recevoir des informations des différentes pages distribuées. Cependant, l'utilisation de NodeJs implique de n'utiliser que du JavaScript. Parfois, il aurait été plus simple et rapide de coupler le JavaScript avec du PHP, mais cela n'était pas possible, le serveur ne prenant pas en compte ce langage.

Le JavaScript a été utilisé pour la réalisation de cette interface car il a notamment permis de pouvoir utiliser une bibliothèque CreateJs dans laquelle nombre de fonctions pour le dessin ou pour l'animation de formes et d'images sont déjà implémentées.

CreateJs comprend différentes bibliothèques comme TweenJs ou encore CreateJs. Ces deux dernières sont les plus utilisées dans la réalisation de ce projet. CreateJs permet de faciliter le dessin, la création de formes avec différentes tailles, couleurs ou aspects. La bibliothèque TweenJs est utilisée pour la mise en mouvement des formes et des images.

Dans un premier temps, il s'est agi de mettre en place le serveur NodeJs qui permet de distribuer les différentes pages en fonction de l'adresse entrée dans la barre d'adresse du navigateur et d'interagir avec les différentes pages distribuées. Ce fichier se trouve dans le dossier « exemple » et se nomme app.js.

Pour accéder à une page de l'application, il faut taper dans la barre d'adresse :

http://adresseIpDuServeur:8080/leNomDeLaPage

(ex http://130.000.00.000:8080/magicien.html)

L'adresse IP du serveur est affichée dans la console lorsque le serveur est lancé dans la commande NodeJs à l'aide de la commande : *node nomDuFichierServeur*. Il écoute toujours sur le port 8080. Le nom de la page pourra être :

- Magicien (la page qui sert à piloter la forme à distance)
- Domus (la page qui sera affichée à l'utilisateur dans Domus avec la forme pilotée)
- Creation (la page qui permet de créer de nouvelles formes)

1. Modules

Pour le fonctionnement du serveur, différents modules sont utilisés. L'installation se fait par le biais de NPM (Node Package Manager) qui regroupe tous les modules disponibles pour NodeJs et en gère les dépendances (si un module installé nécessite l'installation d'un autre, il sera installé automatiquement). Tous les noms des modules et leur documentation se trouvent à l'adresse suivante : http://npmjs.com. De plus, l'avantage de NPM est que si nous avons besoin d'un module qui n'existe pas, nous pouvons très simplement en créer un.

Les différents modules installés se trouvent dans le dossier « exemple » dans le dossier « node_modules ». Pour l'installation d'un nouveau module, la commande suivante est utilisée dans la commande NPM :

Npm install nomDuModule

Le module « express » est un micro-framework très utilisé en NodeJs. En effet, il permet d'éviter de coder à bas niveau. Ici, « Express » va notamment être utilisé pour gérer les différentes URLS entrées dans la barre d'adresse du navigateur par l'utilisateur. Pour pouvoir utiliser ce module dans le serveur, nous avons inclus le module, lors de la déclaration des constantes :

Const express = require('express') ;

Puis nous avons créé un objet « app » en appelant la fonction express() :

Const app = express() ;

C'est cet objet qui sera appelé lors de l'utilisation d'Express :

```
//Envoi de la page magicien.html
app.get('/magicien', function (req, res) {
    res.sendFile(__dirname + '/magicien.html');
});
```

Ici nous récupérons la route entrée par l'utilisateur, puis une fonction de callback est appelée pour envoyer la page demandée.

D'autres modules sont utilisés par le serveur, ils seront détaillés lorsque leur utilisation sera évoquée.

Après la définition des différentes constantes permettant d'inclure les modules, se trouve la déclaration des dossiers statiques. Celle-ci permet de servir des dossiers statiques

dans les différentes pages comme par exemple les feuilles CSS ou tous les fichiers nécessaires à l'utilisation de CreateJs.

```
//Dossiers statiques
app.use('/css', express.static ('../_assets/css'));
```

2. WebSocket

L'intérêt du serveur NodeJs dans le développement de cette application est la possibilité de communication entre le serveur et les différentes pages. En effet, par le biais d'une connexion à un socket, il est possible d'envoyer ou de recevoir des informations depuis le serveur. Les sockets permettent d'établir une communication entre le serveur et le client. En effet, le principe est le suivant : un client clique sur un bouton qui doit déclencher une action chez un autre client. Le socket envoie alors le message au serveur qui, lui, transmettra le message au client concerné. Il est également possible de transmettre des messages à tous les clients connectés ou de déclencher une action sur une page client à l'initiative du serveur (sans forcément avoir reçu au préalable un message à exécuter de la part de l'un des clients).

Pour l'utilisation des sockets, le module « socket.io » a été utilisé :

```
Const     io = require('socket.io')(server) ;
```

La connexion au socket s'effectue de la façon suivante :

```
//Connection au socket
io.on('connection', function(socket){
……code….
}
```

Pour recevoir un message sur le serveur :

```
socket.on('message', function (data) {
    …. Les actions à effectuer quand on reçoit un message….
}
```

Pour envoyer un message depuis le serveur :

```
//Transmet le nom de la fonction à lancer à la page
mouvementForme.js
socket.broadcast.emit('message', data);
```

Dans les deux exemples précédents, on écoute sur le socket et on reçoit ou envoie le message nommé « message ». Dans le cas d'un message reçu, une fonction de callback est appelée dans laquelle seront définies les actions à effectuer si un tel message est reçu. La variable « data » contient dans les deux cas le contenu du message, souvent une chaine de caractères.

Côté client, la connexion au socket se fera de la façon suivante :

```
var socket=io.connect('http://localhost:8080');
```

L'application devant être accessible depuis plusieurs machines différentes (la page de Domus sur la tablette et la page du magicien dans la régie) pour le besoin de l'expérience, la connexion du socket à localhost n'est pas une solution envisageable car les messages ne seraient reçus que sur la machine faisant tourner le serveur. C'est pourquoi il faut que le socket du client puisse se connecter à l'adresse IP du serveur. Cette étape s'est avérée être problématique. En effet, récupérer l'adresse IP du serveur depuis le côté client s'est révélé être compliqué. La solution la plus simple aurait été de la récupérer en PHP. Or un serveur http en NodeJs implique une utilisation exclusive du JavaScript. Cette problématique est pour l'instant toujours en suspens car aucune solution fiable n'a encore été trouvée pour récupérer automatiquement l'adresse IP du serveur côté client. Pour l'instant il est donc indispensable pour le bon fonctionnement de l'application que l'adresse IP envoyée par le serveur dans la console lorsque le serveur est lancé, soit rentrée manuellement dans les différentes pages nécessitant une connexion au socket à la place de « localhost ».

Le code permettant de récupérer et d'afficher l'adresse IP du serveur dans la console lorsqu'il est lancé est le suivant :

```
//Récupérer adresse ipserveur
http.get({'host': 'api.ipify.org', 'port': 80, 'path': '/'}, function(resp) {
    resp.on('data', function(ip) {
            console.log("My public IP address is: " + ip);

    });
});
```

Figure 4 : Code pour la récupération de l'adresse IP du serveur

Plusieurs solutions ont été envisagées. Tout d'abord l'utilisation de PHP, mais impossible car le serveur NodeJs ne le prend pas en compte. Par la suite, une API « ipify » a été utilisée un temps, mais il semblerait que cette API ait été désactivée en JavaScript. Elle reste cependant utilisée dans le serveur en NodeJs. Aucune solution de repli n'a pour l'instant été trouvée.

Les points à améliorer :

- La récupération de l'adresse IP pourrait être automatique et placée dans une variable qui permettrait de ne pas devoir modifier l'adresse de connexion au socket manuellement pour l'utilisateur de l'application.

Chapitre 2. Architecture de l'application

L'application se compose donc des fichiers suivants :

- Domus.html : la page qui apparaîtra dans Domus, sur laquelle apparaît la forme en mouvement

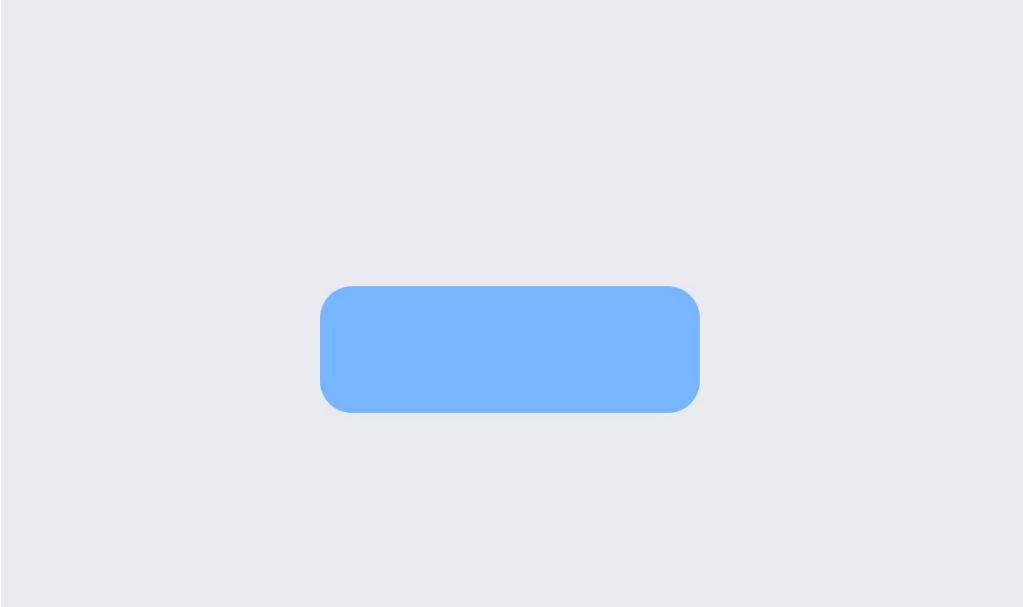

Figure 5 : Page de Domus

- Palette.html : la page qui comprend la création de forme et qui permet d'enregistrer de nouveaux profils. Elle comprend une palette de couleurs permettant d'appliquer une couleur au choix à la forme, différents boutons permettant de choisir différentes formes et un curseur permettant de régler la taille de la forme. Cette page affiche également les différents profils créés et permet de les supprimer, les modifier ou les appliquer à Domus en temps réel.

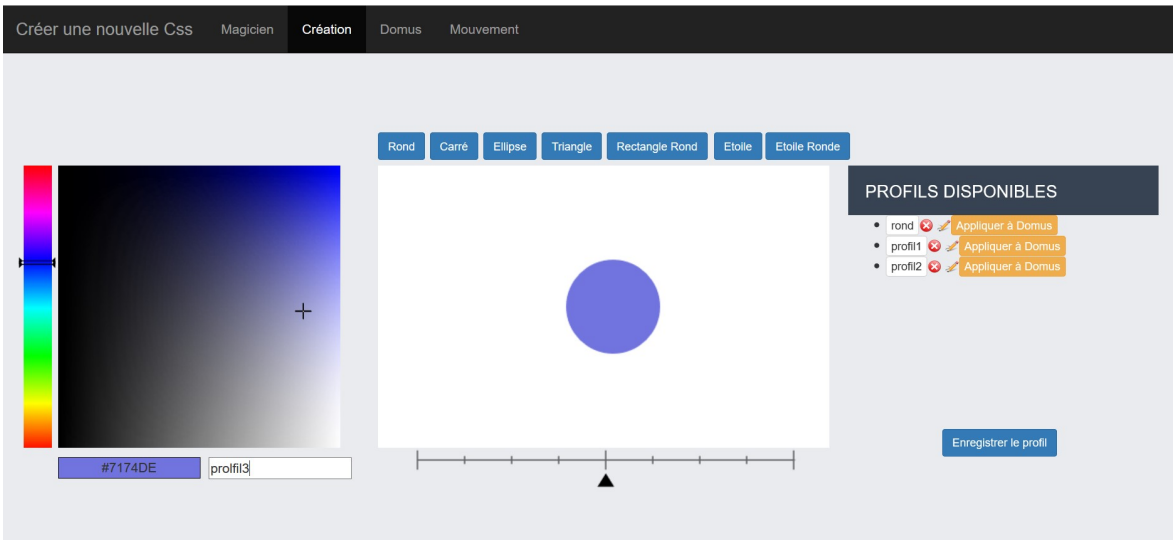

Figure 6 : Page de création

- Magicien.html : la page qui sera utilisée par le magicien. Elle comprend une palette de couleurs permettant de faire varier la couleur de la forme affichée dans domus.html avec une transition douce (en dégradé) et les différents boutons permettant de piloter la forme à distance (le mouvement et le son en simultané, le mouvement seul ou le son seul)

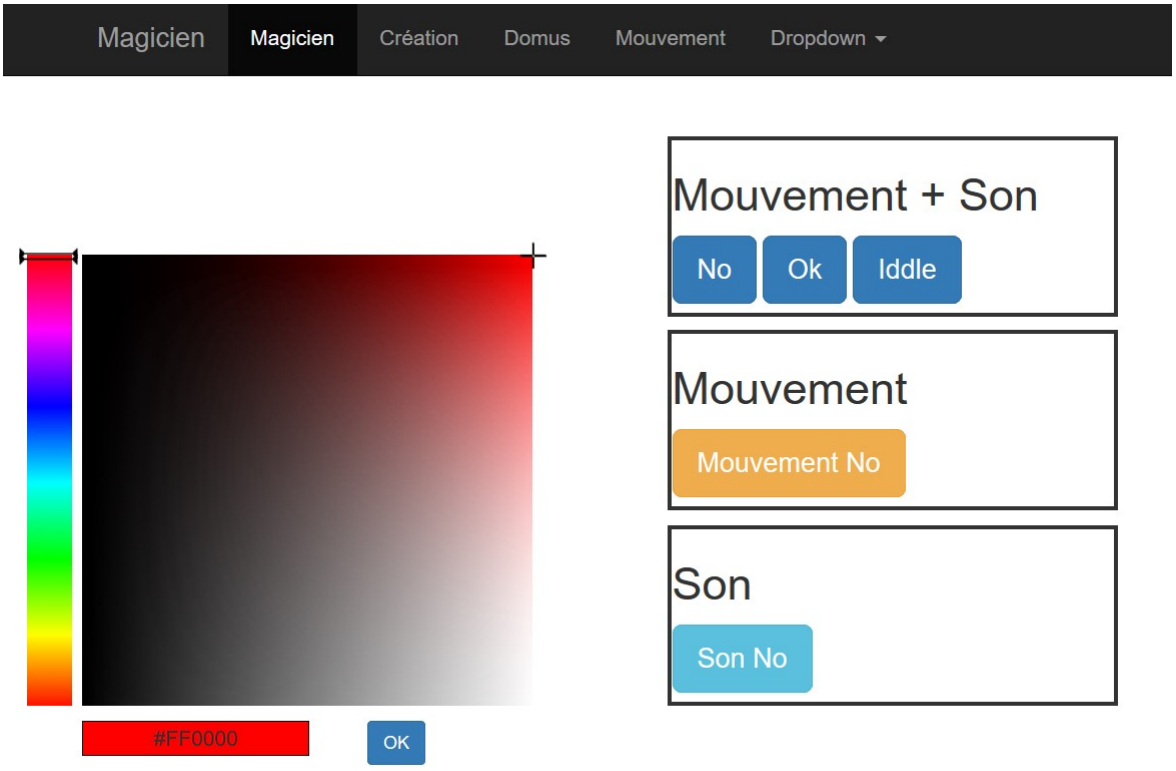

Figure 7 : Page du magicien

- App.js : le serveur http décrit précédemment.

Comme expliqué précédemment, l'architecture NodeJs permet de nombreuses interactions entre les différentes pages par le biais du serveur.

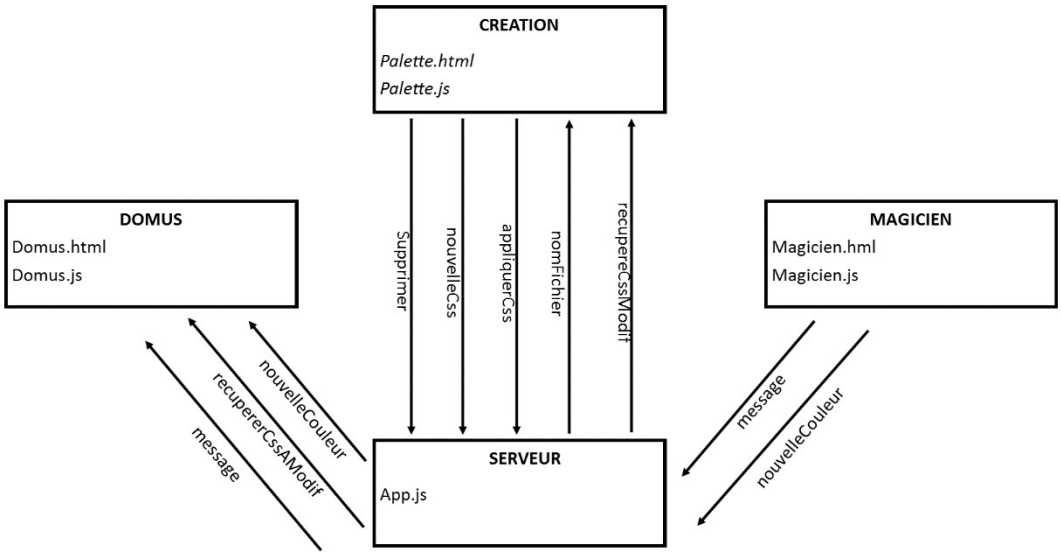

Figure 8 : Schéma des interactions

L'interaction « message » transmet le mouvement demandé par le magicien à la page de Domus pour l'exécuter par la suite.

L'interaction « nouvelleCouleur » informe la page de Domus que le magicien a demandé une nouvelle couleur à appliquer à la forme.

L'interaction « RecupererCssAModif » envoie les différentes informations (nom de la CSS, couleur de la forme, forme choisie, taille) depuis la page palette.html à la page de Domus lorsque l'utilisateur choisit d'appliquer un nouveau profil.

L'interaction « appliquerCss » envoie le nom de la CSS à afficher dans la page de création lorsque l'utilisateur veut modifier un profil.

L'interaction « nomFichier » comprend les noms des profils disponibles récupérés depuis le serveur et envoyés à la page de création.

L'interaction « supprimer » envoie le nom du profil à supprimer lorsque l'utilisateur supprime un profil. L'opération de suppression est alors effectuée du côté serveur.

L'interaction « nouvelleCss » comprend les différentes informations (nom, couleur, forme, taille) lorsque l'utilisateur crée un nouveau profil. La création du profil s'opère alors côté serveur : un fichier .txt est alors créé qui comprend toutes les informations. La fonction fsAccess() permet alors de créer un fichier à l'endroit voulu. Par la suite, la fonction fs.writeFileSync() permettra d'écrire les différents éléments dans le fichier sous le format suivant :

```
fs.writeFileSync(chemin, "enCours:"+enCours+"; couleur:"+couleur+";
wRoundRect:"+wRoundRect+"; hRoundRect:"+hRoundRect+";
xRoundRect:"+xRoundRect+"; yRoundRect:"+yRoundRect+";
radius:"+radius+"; pique:"+pique+"; taillePique:"+taillePique+";",
"UTF-8");
```

Figure 9 : Code pour écrire les propriétés CSS dans un fichier sur le serveur

Lorsque l'utilisateur voudra modifier ou appliquer cette CSS les différentes informations seront récupérées dans le fichier créé et placées dans des variables à l'aide d'expressions régulières pour être envoyées par la suite. Cette opération est effectuée sur la page app.js (le serveur) car les fichiers créés se trouvent sur ce même serveur. La solution la plus simple était donc d'effectuer cette action depuis le côté serveur plutôt que depuis le côté client.

```
//Lire le contenu du fichier css choisi
    fs.readFile(fichier, function(err,data){
            if(err){
                    return console.error(err);
            }
            //Met le contenu du fichier dans la variable contenu
            contenu = data.toString();

            //On récupère la couleur
             if(contenu.match(/couleur:([^;]*);/)){
                    couleur=RegExp.$1;
            }
            if(contenu.match(/pique:([^;]*);/)){
                    pique=RegExp.$1;
            }if(contenu.match(/taillePique:([^;]*);/)){
                    taillePique=RegExp.$1;
            }
```

Figure 10 : Code pour la récupération des propriétés CSS dans un fichier

Chaque forme ayant des paramètres différents, chaque traitement se fait en fonction de la forme choisie qui est mémorisé dans le paramètre « enCours ». Par exemple la fonction rectangle « roundRect() » prend pour paramètres de création « couleur », « wRoundRect », « hRoundRect », « xRoundRect », « yRoundRect », « radius », et deux autres paramètres de radius qui sont à 0. En revanche, la fonction carré «carre() » prend pour paramètres « couleur », « tailleCarre », « xCarre », « yCarre » et trois autres paramètres de radius qui sont toujours à 0.

Les interactions de suppression, modification ou ajout de profil sont gérées au niveau du serveur. En effet la suppression s'opère avec la fonction NodeJs « unlink() ». En ce qui concerne la fonction de modification, comme évoqué précédemment il s'agit de récupérer les différentes informations contenues dans le fichier .txt ayant pour nom le nom du profil choisi par l'utilisateur puis de les transmettre à la page palette.html pour pouvoir afficher le profil souhaité puis pouvoir le modifier.

1. La page utilisée par le magicien

La page magicien.html contient une barre de navigation située en haut de page qui permet d'accéder aux différentes pages de l'application : création et Domus. Lorsque l'utilisateur clique sur l'un des onglet, la fonction « page() » est appelée. Cette fonction prend un paramètre « demande » qui contient le nom de la page demandée. A l'aide de « document.location.href », nous pouvons rediriger l'utilisateur vers la page demandée. Cependant, le problème de l'adresse IP se pose ici aussi puisque l'adresse entrée devra contenir l'adresse IP.

Sur cette page nous trouvons une partie comprenant une palette de couleur. Celle-ci a été faite à l'aide du site openclassrooms.com sur laquelle il y avait un cours pour apprendre à créer une palette de couleur en HTML et JavaScript. La barre de choix et le dégradé de couleur sont deux images qui se trouvent dans le dossier « images », tout comme les images des deux curseurs correspondants. Grâce à la feuille CSS magicien.css, la palette a été positionnée à l'endroit voulu. Les différents traitements se font ensuite en fonction des coordonnées x et y de la souris de l'utilisateur que nous récupérons à l'aide de la fonction « position(axe,event) ».

C'est ensuite dans la fonction « calcul(event) » que se passent les différents traitements permettant d'afficher les couleurs correspondantes calculées en RGB. Grâce à la fonction « clique(objet) » appelée dès que l'utilisateur clique sur un élément, nous pouvons savoir s'il clique sur la barre de couleur (clic) ou sur le carré de dégradé (clic2). Par la suite, si l'utilisateur a cliqué sur la barre de couleur, différents traitements dépendants des coordonnées x et y de la souris sont opérés pour afficher le carré de dégradé avec la couleur correspondante à celle choisie dans la barre de dégradé. Cette dernière étant découpée en 6 parties égales d'où les 6 boucles if en fonction des coordonnées y (car la barre est positionnée à la verticale).

```
//1/6
if((position('y',event)-170)<=50){
        r=255;
        g=0;
        b=Math.round((position('y',event)-170)*255/50);

// 2/6 (100px)
}else if((position('y',event)-170)<=100){
        r=Math.round(255-((position('y',event)-220)*255/50));
        g=0;
        b=255;

// 3/6 (150px)
}else if....
```

Figure 11 : Code pour le traitement de la couleur par 1/6ème

La barre faisant 300 pixels de hauteur au total, un traitement différent est appliqué tous les 50 pixels.

A chaque 6ème seul un des 3 paramètres RGB est modifié en fonction des coordonnées y de la souris. Dans le premier 6ème c'est le bleu qui est modifié et le rouge et le vert gardent la même valeur peu importe où l'on clique dans ce 6ème. Une fois ce calcul effectué, on applique la couleur RGB trouvée au carré de dégradé.

Pour afficher la couleur choisie dans le carré de dégradé dans le petit rectangle d'aperçu, la fonction « afficher() » est appelée. Tout comme pour la fonction précédente, la couleur choisie est calculée en fonction des coordonnées x et y du clic de la souris. Le pourcentage de noir et de blanc est calculé puis appliqué successivement au rouge, au vert et au bleu pour obtenir la couleur voulue.

La fonction « hexadecimal() » est utilisée pour convertir la couleur RGB en hexadécimal. Cette étape est réalisée pour l'affichage du code de la couleur en hexadécimal plutôt qu'en RGB. L'hexadécimal étant plus largement utilisé.

Lorsque l'utilisateur a choisi sa couleur, il clique ensuite sur le bouton « ok » et la fonction « valider() » est alors appelée. Cette fonction est simplement chargée d'émettre un message « nouvelleCouleur » au serveur qui donnera le code couleur (en RGB) choisi par

le magicien pour que le message soit transmis à la page de Domus qui sera chargée de faire la transition de couleur .

Sur la partie droite de la page du magicien se trouvent les différents boutons qu'il pourra actionner. Ils sont classifiés en trois grandes catégories :

- Sons + Mouvements
- Sons seuls
- Mouvements seuls

Ces différentes catégories ont été créées dans le but de pouvoir envisager et permettre le plus possible d'interactions entre le magicien et la page qui se trouvera dans Domus. Lorsqu'un des boutons est cliqué, la fonction « evenement(data) » est appelée. Cette fonction est chargée d'envoyer un message « message » au serveur qui contient le nom de l'évènement cliqué. Le serveur transmettra ensuite le message par le socket à la page domus.html.

La structure NodeJs utilisée pour le développement de cette application permet ainsi la connexion de plusieurs clients en simultané. Cette fonction permet notamment au magicien de se connecter sur la page de Domus sur une machine en parallèle de celle placée dans Domus pour voir en temps réel ce qui se passe sur la page placée dans Domus, élément indispensable pour le magicien dans le but d'avoir un retour sur ce qui apparait dans Domus.

2. La page de création

Tout comme dans la page magicien.html, cette page contient la palette de couleur avec ses fonctions associées. Pour le bon fonctionnement de cette page et pour permettre les actions voulues, les différentes fonctions associées ont été modifiées. En effet, la fonction « clique(objet) » ne contient plus seulement les deux variables « clic » (pour la barre) et « clic2 » (pour le carré de dégradé) mais également la variable « clic3 » qui est modifiée lorsque l'utilisateur modifie la taille de la forme. La fonction « position(axe,event) » reste inchangée.

La fonction « calcul(event) » qui est chargée des différents traitement, a elle été adaptée aux besoins de l'application. Nous avons désormais trois boucles if majeures : une pour « clic », une autre pour « clic2 » et une dernière pour « clic3 ». Les deux premières boucles restent inchangées.

Tout comme les deux premières boucles, la boucle pour « clic3 » opère différents traitements en fonction de l'endroit où clique l'utilisateur. Lorsqu'une forme est choisie, le curseur de taille se place au milieu de la barre de taille (639 pixels). L'utilisateur déplace le curseur vers la gauche (x<639pixels) s'il veut rapetisser la forme et vers la droite (x>639pixels) s'il veut l'agrandir. Encore une fois, en fonction de la forme que l'utilisateur souhaite agrandir ou rapetisser, le traitement est différent. Il est donc nécessaire de prendre en compte la variable « enCours », d'une part parce que les paramètres de création de la forme sont différents comme expliqué précédemment, et d'autre part car le calcul pour rapetisser ou agrandir la forme est différent. En effet, les formes n'ayant pas forcément la même valeur de taille de départ, les formes seront rapetissées plus ou moins rapidement.

```
//Si la forme en cours est un rond
    if (enCours=="rond"){
            var petit=(milieu-x)/4;
            petit=d-petit;
            //~ console.log("petit si ok = "+petit);
                        //Alors d est diminué de petit
            d=petit;
            //La taille des piques
            taille=calculTaille();
                        //on relance la fonction avec la nouvelle valeur de d
            rond(couleur,d,pique1_old,taille);
            //On stocke la taille appliquée à la forme
            dTaille=d;
            tailleChoix=taille;
            //on remet d à sa va leur initiale
            d=50;
            taille=0;
    //Si la forme en cours est carré
    }else if (enCours=="carre"){
        ...
    }
```

Figure 12 : Code qui prend en compte la différence de paramètres

Lorsque l'utilisateur clique sur la barre de taille, la nouvelle taille de la forme est calculée en fonction des coordonnées x du clic de l'utilisateur. Ceci permet ainsi d'avoir un changement de taille progressif, pas saccadé. L'une des premières versions de cette page contenait un traitement de taille saccadé puisque la barre avait été séparée en quatre parties de chaque côté du milieu. En fonction de la partie dans laquelle l'utilisateur cliquait, une nouvelle taille était attribuée. Il s'est donc avéré plus logique de calculer la nouvelle taille en fonction des coordonnées x de la souris sur la barre de taille pour avoir un changement de taille plus progressif par la suite.

Par exemple, lorsque l'utilisateur souhaite rapetisser un rond, la taille est calculée de la façon suivante. Dans un premier temps, on soustrait la coordonnée x du clic de l'utilisateur au milieu (639). Puis on divise ce résultat par quatre. Enfin on soustrait le résultat obtenu au diamètre initial du rond. On obtient alors le nouveau diamètre. La taille de la forme est ensuite actualisée en rappelant la fonction de création du rond « rond(couleur,d,pique1,taille) ». Puis on remet le diamètre « d » et la taille « taille » à 0.

Différents boutons permettent de créer différentes formes :

- Rond
- Carré
- Ellipse
- Triangle
- Rectangle rond
- Etoile
- Etoile ronde

Ces formes ont été choisies pour leur pertinence dans le cadre de cette expérience. Lorsque l'utilisateur clique sur l'un des boutons, la fonction associée est appelée. Admettons que l'utilisateur clique sur le bouton « Rond », la fonction « rond(couleur,d,pique1,taille) » est alors appelée.

```
function rond(couleur,d,pique1,taille){

  //Si on change de forme, alors on remet le curseur de taille au milieu de la barre graduée
  if(enCours != "rond"){
        document.getElementById("curseur").style.left="632px";
  }
  enCours="rond";
  dChoix=d;
  //Efface d'abord le contenu du canvas
  context.clearRect(170,450,canvas.width, canvas.height) ;

  stage = new createjs.Stage("testCanvas");

  //Création de l'élément ball
  var circle = new createjs.Shape();
  circle.graphics.beginFill(couleur).drawCircle(250, 150, d);
  //Ajoute ball dans le document
  stage.addChild(circle);
  tailleChoix=taille;
  //Ajoute l'élément dans le document
  createjs.Ticker.addEventListener("tick", stage);
}
```

Figure 13 : Code pour la fonction rond(couleur,d,pique1,taille)

Lorsque la fonction est appelée, la variable « enCours » prend pour valeur « rond ». Ensuite, on efface le contenu du canvas pour pouvoir afficher la nouvelle forme. Puis on crée l'élément à l'aide de CreateJs qui la rajoute sur le canvas. On utilise ensuite la ligne de code suivante pour mettre l'élément en mouvement : « *createjs.Ticker.addEventListener("tick", stage); ».*

Comme évoqué précédemment, pour le dessin de la forme, un canvas est utilisé.

```
        <!-- LA ZONE DE DESSIN -->
    <div id="pere" class="skin_canvas">
            <canvas id="testCanvas" width="960" height="350"></canvas>
    </div>
```

Figure 14 : Canvas

C'est à cet endroit que les formes seront dessinées et mises en mouvement si nécessaire. Le même principe de canvas est adopté dans la page domus.html. A chaque dessin d'une forme, il s'agira de récupérer le canvas, d'en effacer le contenu puis d'y redessiner la nouvelle forme. La fonction « getCanvas() » est appelée au chargement de la page. Elle consiste à récupérer le canvas dans une variable.

La partie gauche de cette page est consacrée aux profils. Elle affiche les profils disponibles sous forme de liste. Chaque profil peut être modifié, supprimé ou appliqué à Domus. Les différentes actions ont été expliquées précédemment. Chaque bouton est une image qui se trouve dans le dossier « images ». Cette partie est réalisée à l'aide d'une fonction dans le serveur qui permet de lister les fichiers contenus dans le dossier « css » dans lequel sont rajoutés les nouveaux profils à chaque création.

Le projet ayant évolué au fil du temps, une première version consistait à pouvoir rendre la forme « agressive » en y ajoutant des piques. Il y avait alors une autre barre s'apparentant à la barre de taille qui permettait de régler l'agressivité de la forme. Après discussion avec mes encadrants, cette fonctionnalité s'est avérée inutile et a donc été supprimée. Cependant, le code correspondant a été gardé en commentaire. Il est donc toujours possible de rajouter cette fonctionnalité si un jour elle s'avère nécessaire.

3. La page affichée dans Domus

Seule la page de Domus ne possède pas la barre de navigation, l'objectif de cette page étant que la personne se trouvant dans Domus ne puisse pas accéder aux outils mis à disposition du magicien. Il est cependant possible depuis cette page de retourner aux autres pages en modifiant l'url dans la barre de navigation.

Cette page comprend uniquement l'affichage de la forme en mouvement sur fond gris. La couleur du fond a été choisie de façon à ce qu'une forme blanche puisse être clairement

visible. La couleur blanche étant l'une des couleurs de forme les plus importantes dans le cadre de cette expérimentation contrairement à la couleur grise. Une forme grise ne pourra donc pas y être distinguée mais si cela est nécessaire, il est toujours possible de changer la couleur du fond dans la feuille de style associée.

La fonction « getCanvas() » est appelée au chargement de la page. Cette fonction est chargée, comme dans la page palette.html, de récupérer le canvas dans une variable mais également de créer et d'afficher la forme par défaut en mouvement. La forme par défaut est donc un rectangle rond (avec les coins arrondis) de couleur bleue. En effet, le choix de ces paramètres a été fait par mon encadrant Romain Magnani, qui suite à son état de l'art sur l'animisme a établi que la forme la plus neutre serait celle-ci.

En ce qui concerne les mouvements de la forme, ils ont été créés sur la base d'animations fournies par mon encadrant. Le mouvement de repos de la forme (mouvement « iddle ») est le grossissement et le rapetissement de la forme à un rythme qui ne fasse pas penser à un battement de cœur. Il a donc fallu créer un mouvement qui ne soit pas rapide et qui effectue une pause avant de commencer à grandir ou à rapetisser.

Deux autres mouvements m'ont été demandé de réaliser : le mouvement « no » et le mouvement « ok ». Le mouvement « no » est un déplacement de la forme de droite à gauche et le mouvement « ok » est le déplacement de la forme vers la droite, vers le haut, vers la gauche puis un retour à sa position initial. Ces mouvements ont été réalisés à l'aide de TweenJs qui permet facilement de récupérer un objet dans le canvas et de changer ses coordonnées (tout en gérant le fait que la forme placée aux coordonnées précédentes soit effacée). De plus, cela permet également de gérer plutôt simplement la vitesse de déplacement de la forme.

```
if(data=='no'){
    //On déclenche le son associé au mouvement
    var son = new Audio("../son/no.mp3");
    son.currentTime = 0;
    son.play();
    //override : arrêter le mouvement iddle dès le déclenchement de l'évènement
    var tween = createjs.Tween.get(ball, {loop: false, ignoreGlobalPause: true, override: true})
    //.to({coordonnées de destination, largeur de la forme, hauteur de la forme},vitesse de déplacement,rebondit)
    //Part à droite tout en retrouvant la taille initiale
    .to({x: 0, y:0, x: 50, scaleX: 1, scaleY: 1}, 1500, createjs.Ease.bounce)
    .to({x: 0, y: 0}, 1500, createjs.Ease.bounce)
    .to({x: 50, y: 0}, 1500, createjs.Ease.bounce)
    .to({x: 0, y: 0}, 1500, createjs.Ease.bounce)
    //Une fois le mouvement terminé (5800ms), on relance le mouvement iddle
    setTimeout(iddle,5800);
}
```

Figure 15 : Code pour l'animation « no » de la forme

Contrairement au mouvement « iddle », lors du déclenchement des deux autres mouvements, le mouvement ne s'effectue qu'une seule fois. Le mouvement iddle, au contraire, est un mouvement qui s'effectue en boucle et qui se déclenche dès qu'aucun autre mouvement n'est en cours.

La transition des mouvements était un point important dans la réalisation de cette application. En effet, les transitions ne devaient être brutales ou saccadées, et les mouvements devaient absolument être déclenchés dès que le magicien le demandait. La solution qui a été trouvée a ainsi été de lancer un mouvement quand il était demandé et si le mouvement iddle n'est pas terminé, la forme revient à sa taille initiale tout en commençant le mouvement. Pour l'instant le cas où l'utilisateur demande le mouvement no alors que le mouvement ok a déjà été commencé mais n'est pas terminé est mal géré par l'application. La transition n'est pas claire et la forme fait un mouvement qui n'est pas harmonieux. Le mouvement « no » a été associé à un son qui a été fourni par mon

encadrant. Ce son sera toujours modifiable en changeant le nom du son déclenché dans la fonction du mouvement et en ajoutant le fichier du nouveau son dans le dossier « son ».

Pour que la forme soit toujours centrée, peu importe la taille de l'écran, on utilise les propriétés screen.width et screen.height qui permettront de connaître la taille (en pixels) de l'écran de l'utilisateur et ainsi calculer les coordonnées de départ de la forme en fonction.

```
//Déclaration des variables
var resolutionEcranX=screen.width;
var resolutionEcranY=screen.height;

//Calcul de la position de départ au milieu de l'écran
var departX=(resolutionEcranX/2)-150;
var departY=(resolutionEcranY/2)-200;
```

Figure 16 : Code pour les coordonnées de départ de la forme

Enfin, la transition de couleur devait se faire de façon progressive. C'est pourquoi une fonction permettant de changer la couleur en passant par toutes les couleurs intermédiaires a été intégrée dans le code. Trois fonctions différentes sont associées : l'une permettant de récupérer la couleur choisie par le magicien, et faisant les calculs permettant de passer par des stades de couleurs différents, une autre permettant de savoir si la couleur choisie par le magicien a été atteinte, et une dernière qui applique la couleur. Ces différentes fonctions ont été adaptées aux besoins de l'application à partir de fonctions trouvées sur un forum.

Pour l'affichage des formes, au même titre que dans la page de création, les fonctions implémentées dans CreateJs ont été utilisées comme par exemple la fonction « drawRoundRect() ».

Point à améliorer :

- La transition entre le mouvement ok et le mouvement no alors que le mouvement ok a été démarré mais pas terminer n'est pas harmonieuse. La transition n'est pas claire.
- La transition entre deux profils sur la page de domus.html n'est pas progressive. Il pourrait être intéressant de faire un fondu pour le changement de couleur et de forme.

Partie 2

-

Test de perception

Chapitre 1. Choix des primitives utilisées

Ce test de perception a pour but de récupérer des données et en déduire des tendances au sujet de l'association entre une forme et un son. Nous avons pour cela sélectionné des primitives sonores et visuelles qui permettront de réaliser au mieux le test de perception qui s'appuie fortement sur l'effet Kiki Bouba énoncé précédemment.

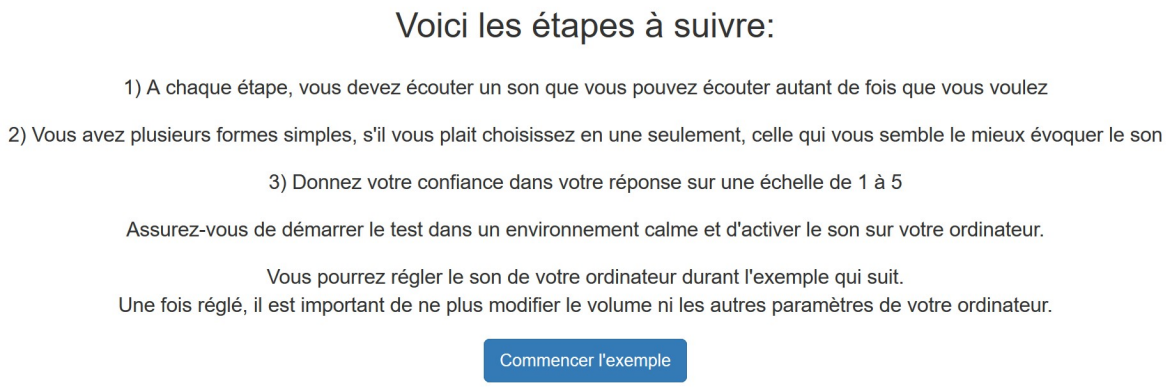

Figure 17 : Consignes

1. Primitives sonores

Ces sons non lexicaux de « pure prosodie », récoltés dans des corpus d'interaction naturels (Audibert, 2008) et très largement sélectionnés, évalués, mesurés et calibrés depuis ont été choisis (si on se réfère par exemple au modèle bidimensionnel de Russel) pour leur variation égale entre leur valence positive/plaisant vs. négative/déplaisant, dont l'hypothèse est que l'analogie avec les formes soit dans arrondi vs. pointu et/ou blanc vs. rouge, et de leur activation passive vs. active, dont l'hypothèse est que l'analogie avec les formes soit dans petit vs. grand. Quant aux analogies entre les morphologies acoustiques et sonores, nous avons choisi des sons selon :

- leur variation de hauteur de la fréquence fondamentale ;
- leur variation de l'amplitude dont l'hypothèse (rappelons que la perception de la variation de la fréquence fondamentale et de l'amplitude sont des perceptions intégrées et croisées, c'est à dire que si l'une augmente on perçoit une augmentation de l'autre)
- leur variation de qualité de voix (dont l'hypothèse est que l'analogie avec les formes soit dans arrondi vs. pointu car nous n'avons pas introduit de vision de

texture qui ferait appel à l'analogie du toucher, c'est-à-dire que nous avons contrôlé ce biais inévitable (d 'inférence du toucher par la perception visuelle) avec des objets tous perçus selon le même aspect (lisse)) ;

Nous donc avons créé des sons « whouaou », « ah », « euh », « Hum » en fonction des paramètres énoncés précédemment.

Ces sons ont été normalisés pour essayer de limiter l'artefact de l'intensité (volume de la voix). En effet, celle-ci était biaisée à cause de la distance du micro qui variait lors de l'enregistrement. Nous avons ainsi gardé les sons originaux et les sons avec l'amplitude normalisée pour tester l'impact de la modification de l'intensité.

La F0 (fréquence fondamentale) a été modifiée. Mais il était important de suivre l'artefact : pour la voix d'homme, la F0 a été baissée, et pour la voix de femme, elle a été augmentée. Le signal naturel a lui aussi été gardé, c'est la F0 modale. La modification de F0 a donc été opérée sur le son naturel et sur le son normalisé.

Nous avons donc un total de 64 sons : quatre sons différents : « wouhaou », « ah », « euh », « Hum2 ». Pour chacun de ces sons nous avons une version masculine et une version féminine (V et R) en prosodie positive et en prosodie négative (POS et NEG). Il y a donc pour chaque son :

- La version originale
- La version originale normalisée
- La version originale non normalisée avec modification de F0
- La version normalisée sans modification de F0

4 (sons) x 2 (genre) x 2 (prosodie) x 2 (F0) x 2 (amplitude) = 64 sons différents.

Nous avons par la suite créé aléatoirement quatre modèles différents d'ordre d'écoute des sons pour ne pas avoir de biais dû à l'ordre d'écoute.

2. Primitives visuelles

Pour cela, nous avons donc sélectionné trois formes différentes (rond, rond piquant, piquant), deux couleurs (rouge et blanc : une couleur réputée comme plutôt associée au négatif et à la dominance, et une autre plus neutre qui est réputée pour être plutôt neutre voire positive) et deux tailles (petit et grand : le petit sera plutôt considéré comme pas dominant à l'inverse du grand). En créant toutes les combinaisons possibles, nous obtenons ainsi un panel de douze formes qui seront toutes proposées à l'utilisateur à

l'écoute d'un son. L'utilisateur devra alors associer une forme (parmi les douze proposées) au son (en fonction de ce que le son lui évoquera). Il devra ensuite donner un indice de confiance, grâce à l'échelle de Likert qui apparait lorsqu'une forme est sélectionnée.

Le choix des formes a nécessité une certaine réflexion car elles ne devaient représenter rien de connu dans l'environnement naturel de l'humain. C'est pourquoi les formes ont été dessinées avec paint.net à l'image de ce que l'on voulait. Il fallait qu'elles soient déstructurées mais sans trop l'être et pour la forme piquante elle devait être agressive mais pas trop non plus.

Dans le but de ne pas stresser les sujets, nous avons permis une écoute illimitée du son, de même pour le choix de forme. Le choix de l'utilisateur sera définitivement validé lorsqu'il aura choisi son indice de confiance et appuyé sur valider.

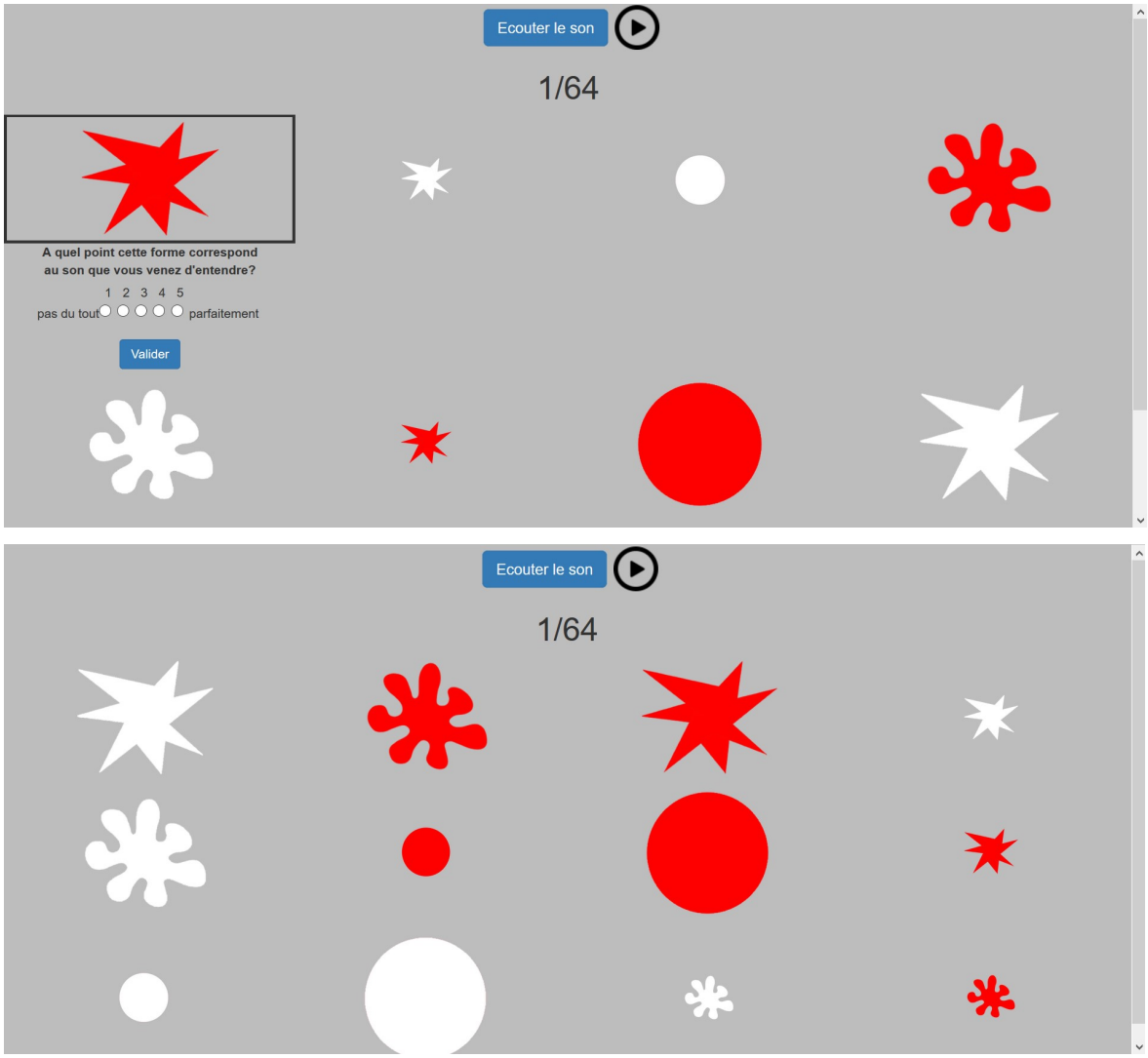

Figure 18 : Association forme/son

Chapitre 2. Réalisation du test

1. Données récupérées et stockage

Dans un test de perception, il est important de récupérer le plus d'informations pertinentes possibles quant au profil du sujet. Ici, nous avons donc pu récupérer les données suivantes.

Pour les données propres au profil du sujet :

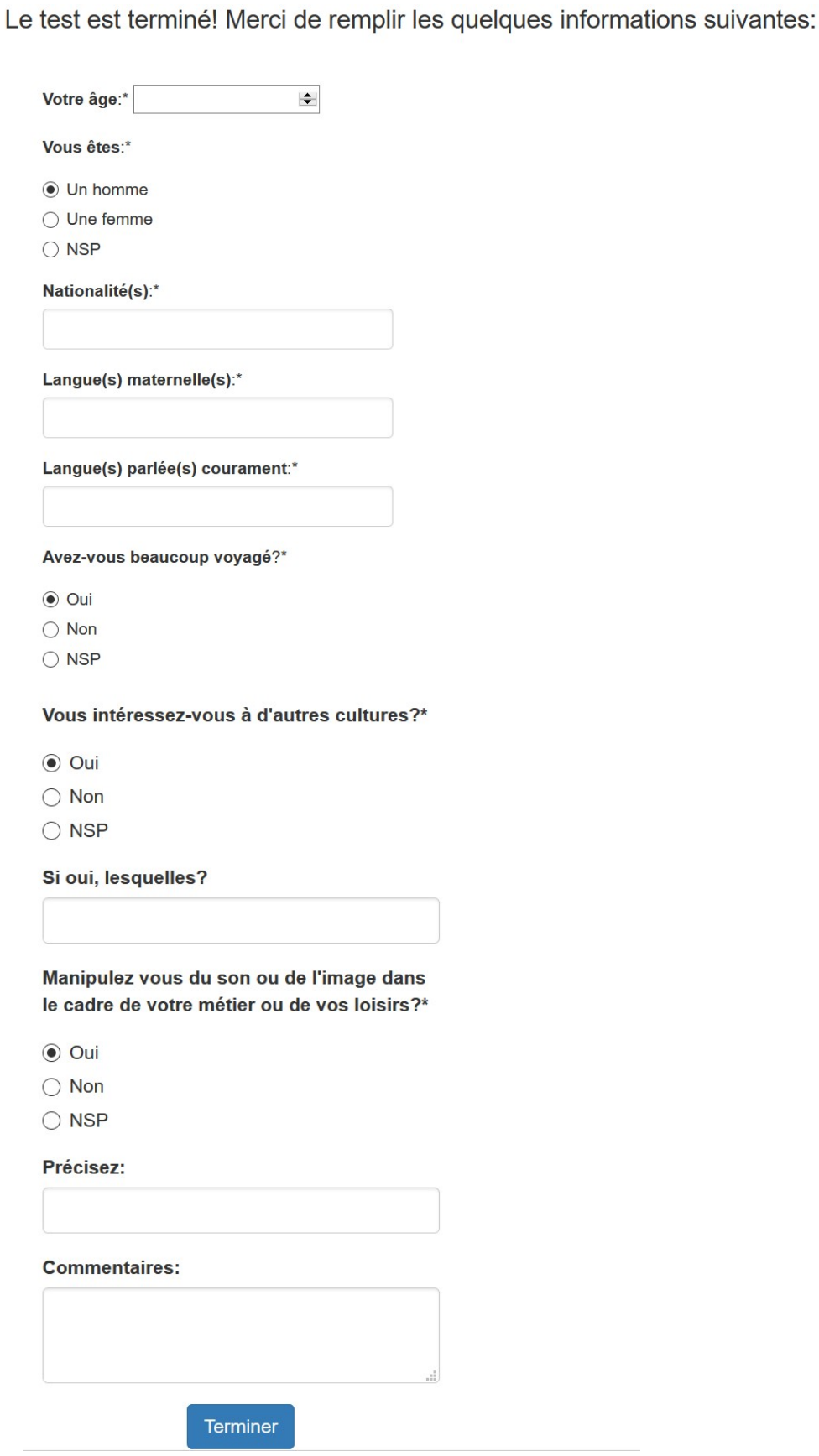

Figure 19 : Questionnaire de fin de test

Pour les données de réponse :

- Le temps que le sujet a mis pour répondre définitivement (le temps écoulé entre la première écoute du son et la validation de l'indice de confiance)
- La forme choisie définitivement
- Le son associé
- Le nombre d'écoutes du son entre la première écoute et le choix définitif de la forme
- L'indice de certitude (échelle de Likert)
- Le nombre de fois où le sujet a choisi une autre forme avant de choisir la forme définitive
- Les formes sur lesquelles le sujet a hésité (stockées dans un tableau de tableaux : un tableau global qui comprend un tableau par son qui comprend lui-même à chaque indice (en fonction du numéro de clic, la forme qui a été cliquée)
- Le timeStamp de la première écoute : avec la fonction « substr(microtime(true)*1000, 0,13); » car PHP a la fonction microtime(true) qui permet de récupérer le timestamp depuis l'époque unix en micro secondes, mais il n'existe pas de fonction pour récupérer cela directement en millisecondes. C'est pourquoi nous avons dû utiliser cette fonction dérivée. En JavaScript, la fonction utilisée permet directement de récupérer le timestamp en millisecondes.
- Le timeStamp du choix définitif de la forme
- Le timeStamp de chaque clic sur une forme (si le sujet a hésité) : y compris la dernière forme cliquée car le sujet clique sur la forme puis choisi sa certitude. C'est pourquoi il est intéressant de garder les 2 temps qui sont différents.
- Le timeStamp de chaque clic sur écouter (avec le clic de début global)

Ces différents paramètres nous permettent ainsi de voir si l'utilisateur a beaucoup hésité, en fonction de s'il a eu besoin d'écouter plusieurs fois le son, s'il a choisi d'autres formes avant de choisir la définitive, de recouper les hésitations de différents sujets pour savoir si un grand nombre de sujet a hésité entre deux mêmes formes…

Le questionnaire que le sujet devra remplir à la fin du test nous permet d'avoir les informations nécessaires pour connaître son profil. En effet, la nationalité, les langues parlées couramment, et la/les langue(s) maternelle(s) et l'intérêt particulier pour d'autres cultures, et les voyages sont des questions qui ont été posées pour récupérer au mieux les influences culturelles que l'on pourrait retrouver dans les réponses données par le sujet.

D'un point de vue technique, le test de perception a été réalisé en PHP, HTML5 et JavaScript. Les données récupérées pour chaque sujet sont stockées dans un fichier CSV global sur le serveur (ouvert à chaque connexion avec le curseur à la fin, pour conserver les données récupérées précédemment). Par précaution, à chaque nouveau sujet, un fichier ayant pour nom son identifiant, est créé. Cela permet de ne pas perdre la totalité des données si le fichier CSV global vient à se perdre par exemple.

A chaque connexion, un identifiant unique est attribué avec la fonction uniqid() de PHP qui permet d'attribuer un identifiant unique préfixé basé sur la date et l'heure de connexion de l'utilisateur. Cet identifiant unique sera placé dans la première colonne du fichier CSV et nous permet de garder l'anonymat des sujets qui ont participés au test tout en les distinguant les uns des autres. Le fichier CSV est rempli à la fin de chaque test (lorsque l'utilisateur clique sur « terminer » après avoir rempli le questionnaire), ce qui implique de garder en mémoire les différentes données récupérées tout au long du test.

Pour réaliser cela, j'ai utilisé des variables de session contenant des tableaux. Lorsque l'on remplit le CSV, les différents tableaux sont parcourus à l'aide de l'indice des sons (il y a autant de données différentes pour un même paramètre que de sons).

L'envoi des données récupérées en JavaScript est réalisé avec la méthode GET, sauf pour le questionnaire de fin de test (la méthode POST a été utilisée pour plus de facilités) et pour le clic de l'utilisateur pour l'écoute d'un son. Pour écouter le son, un formulaire a été créé. Lorsque l'utilisateur clique sur « Ecouter le son » ou sur le bouton « play », la fonction « sauvegardeDonnees() » est appelée. Cette fonction permet de garder en mémoire, lors du chargement de la page, l'ancien nombre de clic sur des formes (clicForme) puisqu'il est envoyé en tant que paramètre GET. Cette valeur sera réinitialisée, seulement lorsque l'utilisateur aura validé une forme et passera donc au son suivant à l'aide de la fonction « sonSuivant() ».

Pour l'envoi des données et le lancement du son, un bouton a été créé sous forme d'image. L'appel d'une fonction JavaScript ne fonctionnant pas sur un bouton de type « submit » dans la configuration de la page, il a donc fallu créer une image à la place du bouton. Le formulaire a été gardé pour une question de mise en page, qui permet simplement d'avoir le bouton « écouter le son » et le bouton « play » côte à côte sans créer de feuille CSS.

Pour la sortie en CSV lors de la phase de test, nous avons remarqué un problème d'affichage des caractères spéciaux, mais uniquement sous Excel car Excel ne détecte pas l'encodage du fichier mais utilise sa valeur par défaut qui varie en fonction de la langue de

l'utilisateur. Pour remédier à cela il y a 2 solutions possibles : soit on force Excel à utiliser de l'UTF-8 pour ouvrir le fichier (il faut alors insérer un caractère spécial appelé B UTF8 au début du fichier lors de sa génération), soit on ouvre Excel puis dans l'onglet « Données » on clique sur « Fichier Texte » et on importe notre fichier en prenant soin de bien choisir « 65001 : Unicode (UTF-8) ».

Un autre problème a été rencontré dans le fichier CSV lorsqu'il est ouvert avec Excel : lorsque l'utilisateur a cliqué uniquement sur deux formes (une forme hésitation et une forme choisie définitivement) le TimeStamp du deuxième choix est tronqué à l'affichage. Lors de l'ouverture dans un éditeur de texte tel que SublimeText, la donnée apparait bien correctement et non tronquée.

Les différents traitements tels que l'initialisation des variables, le choix des formes à afficher, sont faits sur la page consigne.php qui est affichée juste avant que l'utilisateur fasse l'exemple.

De plus, il a fallu contourner le fait que le JavaScript ne prenne pas en compte des variables qui fonctionnent sur le même type que les variables de session en PHP. C'est pourquoi nous avons dû créer des fonctions qui seront appelées par le PHP à chaque changement de situation auquel on devra réinitialiser les variables de nombre de clics sur les formes, et la variable qui contient les formes respectives sur lesquelles l'utilisateur a cliqué. Pour garder en mémoire ces variables, le choix a été fait de garder le contenu de ces variables par le biais de la barre d'adresse. En envoyant les valeurs par GET, on peut ensuite les récupérer pour pouvoir les modifier. Cependant, JavaScript n'a pas de fonction pré implémentée permettant de récupérer des paramètres d'une URL. Il a donc été nécessaire de parser l'URL pour récupérer le paramètre voulu. Une fonction « $_GET(param) » permet d'effectuer ces différentes actions, et tout comme PHP de renvoyer un tableau dans lequel se trouvent les différentes valeurs envoyées en GET.

En ce qui concerne les différentes formes cliquées entre la première écoute et le choix définitif de l'utilisateur, le choix a été fait de concaténer la taille (petit ou grand) à la fin du nom de l'image. Lorsque l'utilisateur a fini le test, nous obtenons donc une chaine de caractères contenue dans « chaineFormeClic » du type :

piqueBlanc.pnggrand,piqueRondRouge.pnggrand,piqueRouge.pnggrand

Le .png n'apportant aucune information, juste avant l'écriture des résultats dans le CSV dans la page questionnaire.php, la chaine est splittée en fonction du « .png ». Nous obtenons alors un tableau contenant tous les éléments de la chaine splittée. Il faut alors

parcourir le tableau pour concaténer tous les éléments du tableau et obtenir la même chaîne sans le « .png ». Nous obtenons alors des données propres et exploitables en l'état, qu'il suffira de splitter en fonction du caractère de séparation (« , »,« $ », « @ ») lors de l'analyse automatique des données.

De plus, pour pouvoir récupérer des données propres dans le champ de commentaire rempli par le sujet, nous avons décidé d'interdire le retour charriot grâce au code suivant placé dans la balise <form> :

onkeypress="return event.keyCode != 13;"

2. Choix de mise en page

Pour réaliser la mise en page de cette interface, il y avait deux possibilités. Soit on réalisait un tableau HTML à l'aide des balises <tr> et <td>, soit on utilisait des . La deuxième solution a été choisie dans un premier temps car elle permettait l'utilisation de Bootstrap pour la mise en page. Mais après avoir essayé de modifier cette mise en page, il s'est avéré qu'il serait bien plus simple d'utiliser le simple tableau HTML composé de balises <tr> et <td> pour différentes raisons. Il était tout d'abord plus simple de faire en sorte que l'image affichée dans une cellule du tableau soit alignée verticalement sur le bas de cette même cellule pour que lorsque l'échelle de Likert s'affiche, on puisse l'avoir collée à l'image concernée et non pas à l'image du dessous. En effet, même si avec les , les les plus petites ne prenaient pas la taille des plus grandes sur la même ligne, celles qui étaient placées en-dessous s'affichaient alignées à celles qui étaient les plus grandes au-dessus. Dans un tableau HTML avec <tr>, il a suffi d'utiliser la propriété CSS « vertical-align » avec la valeur « bottom » pour pouvoir aligner l'image verticalement sur le bas de la cellule. Enfin, il était plus évident de rendre le tableau responsive avec un simple tableau HTML.

Pour l'affichage des formes sur l'écran de l'utilisateur, nous avons fait le choix d'avoir quatre modèles différents d'affichage des formes mélangées. Pour ceci, il a fallu faire un tableau multidimensionnel contenant les 4 différents tableaux qui comprennent les différentes formes dans l'ordre d'affichage. Pour opérer un affichage aléatoire à chaque nouvel utilisateur, nous avons fait toutes les associations possibles entre modèle de forme et modèle de son.

```
//Les différentes associations de modèles
    $associationModele[0]=array("0","0");
    $associationModele[1]=array("0","1");
    $associationModele[2]=array("0","2");
    $associationModele[3]=array("0","3");
```

Figure 20 : Code pour l'association des modèles de forme et de son

L'indice de $associationModele est inscrit dans le fichier CSV. Enfin, pour déterminer quel modèle sera attribué à un sujet, nous avons créé un fichier ids.txt dans lequel nous inscrivons l'identifiant de chaque sujet qui se connecte (un sur chaque ligne). Lorsqu'un sujet se connecte, le nombre de ligne de ids.txt est alors compté. A l'aide d'un modulo, nous pouvons alors savoir quel modèle attribuer au sujet en fonction de son rang de connexion.

Il a été important d'avoir une interface la plus responsive possible : l'utilisateur doit avoir à fournir un minimum d'effort pour répondre au test. Il ne doit donc pas avoir à scroller pour voir la totalité des formes proposées par exemple. Il a donc fallu rendre l'interface responsive aussi bien en hauteur qu'en largeur. Pour obtenir une page responsive en largeur, il a suffi d'attribuer une position relative au tableau contenant les douze formes à afficher : « width : 100% » qui indique que le tableau occupe la totalité de la largeur de l'écran peu importe la taille de l'écran. Enfin, pour obtenir une page responsive en hauteur, il a fallu attribuer une taille relative aux images en fonction de leur propriété (petite ou grande). Toutes les images affichées ont donc la même taille, mais les tailles d'affichage varient dans la feuille CSS en fonction de la largeur attribuée à chacune d'elle : 50% pour les grandes, 20% pour les petites.

Pour l'affichage de l'échelle de Likert, il y a une ligne vide en dessous de chaque ligne où sont affichées les formes, dans laquelle on affiche l'échelle de certitude en fonction de l'image sur laquelle a cliqué l'utilisateur ce qui permet ainsi d'avoir l'échelle de Likert affichée en dessous de la forme cliquée. Pour les formes placées sur la dernière ligne (et celles de la ligne du milieu) il a fallu mettre en place un scroll automatique pour que l'utilisateur puisse voir l'échelle de Likert sans avoir besoin de Scroller lui-même. Notons, qu'un test de perception doit être le plus facile possible pour l'utilisateur et lui demander le moins d'effort possible. De plus, on aurait risqué que l'utilisateur ne voie pas l'échelle de Likert s'il n'avait pas vu que le scroll n'était pas tout à fait en bas.

Enfin l'un des points importants pour le bon déroulement du test de perception était qu'il soit compatible avec un maximum de navigateurs. Lors des tests sous les différents navigateurs, des problèmes de compatibilité avec Internet Explorer ont été repérés. En effet, le navigateur ne prend pas en compte le format « wav » pour la lecture des sons dans l'objet <audio>. Il a donc fallu détecter si l'utilisateur utilisait IE pour utiliser dans ce cas un lecteur avec les balises « embed ». Ce lecteur aurait pu être utilisé pour tous les types de navigateurs, or l'attribut « hidden » des balises « embed » n'est pas pris en compte par Firefox et Chrome. IE modifiant le contenu de la variable $_SESSION['http_AGENT'] contenant les informations nécessaires côté serveur pour connaître le navigateur utilisé par le client, il a fallu détecter le navigateur coté client et transmettre cette information par la donnée GET au même titre que les autres informations récupérées côté JavaScript.

3. Sécurisation du serveur PHP

Lors de la mise en ligne du test de perception, il était important de s'assurer que les données étaient bien protégées.

Nous n'utilisons pas de bases de données mais uniquement des CSV, c'est pourquoi nous n'avons pas eu besoin de nous préoccuper des injections SQL.

Les droits d'accès aux fichiers ont été modifiés '444' aucun droit d'écriture, que des droits de lecture, pour éviter que quiconque puisse déposer un script sur le serveur et supprimer le site ou faire du pishing.

Le fichier .htaccess a été rajouté dans les différents dossiers pour interdire l'accès et la liste des fichiers contenus dans la totalité de public.html

4. Analyse des résultats

L'analyse des résultats est une étape nécessaire pour pouvoir tirer des conclusions du test de perception. Le fichier CSV étant composé des réponses de chaque sujet à la suite, il a été nécessaire de trier les réponses en fonction des sons et non plus en fonction des identifiants des sujets. Pour cela, il a suffi d'effectuer un tri par ordre alphabétique de la colonne contenant les sons (colonne numéro 13). Cela nous a ainsi permis d'obtenir un fichier constitué de façon à ce que les 36 réponses pour chaque son (car 36 sujets ont passé le test à ce jour) soient regroupées les unes à la suite des autres. Ce tri a été effectué avec la fonction de tri d'Excel.

Dans un second temps, un programme Perl a été réalisé qui permet de prendre en entrée ce fichier CSV de réponses global trié en fonction des sons et d'en ressortir des données facilement exploitables pour pouvoir en faire des graphiques sous Excel. A noter que la ligne de titre des colonnes a été supprimée pour que les lignes de réponses commencent à 1. Le fichier est ainsi lu ligne par ligne et splitté en fonction du séparateur du fichier CSV « ; ». A chaque fois que l'on se situe sur la première ligne d'un nouveau son, nous écrivons dans le fichier CSV de sortie le nom du son traité. Cette information est récupérée à l'aide d'un modulo. A partir de cette ligne et jusqu'à la dernière ligne de réponse d'un même son, nous allons chercher le contenu de la colonne correspondant à la forme choisie définitivement par le sujet, qui se trouve à la colonne numéro 17. Cette donnée est alors stockée dans un tableau associatif qui comptabilisera le nombre de fois où cette forme a été choisie. Lorsque l'on arrive à la dernière ligne de réponse pour un même son, ces différentes données sont alors inscrites dans le fichier CSV de sortie. Cela nous permettra ainsi de savoir quelle est la forme la plus choisie pour un son.

Pour avoir des informations concernant l'influence des paramètres variables de la forme (forme, taille, couleur) et du son (masculin, féminin, positif, négatif, F0 modifiée, amplitude normalisée, amplitude normalisée avec F0 modifiée, son original), le nombre de réponses en fonction de ces différents paramètres est également comptabilisé par le programme Perl et les résultats inscrits dans ce même fichier CSV de sortie. Nous obtenons alors les résultats suivants :

- Le nombre de personnes qui a associé la forme piquante, ronde ou la forme intermédiaire pour les sons à prosodie positive vs les sons à prosodie négative, pour les sons avec la modification de F0 vs les sons sans modification de F0 (son original), pour les sons avec une voix masculine vs les sons avec une voix féminine, pour les sons avec une amplitude normalisée vs les sons sans amplitude normalisée (son original), et enfin pour les sons avec une prosodie positive vs les sons avec une prosodie négative.
- Le nombre de personnes qui a associé une forme de couleur rouge ou de couleur blanche pour les sons à prosodie positive vs les sons à prosodie négative, pour les sons avec la modification de F0 vs les sons sans modification de F0 (son original), pour les sons avec une voix masculine vs les sons avec une voix féminine, pour les sons avec une amplitude normalisée vs les sons sans amplitude normalisée (son original), et enfin

pour les sons avec une prosodie positive vs les sons avec une prosodie négative.

- Le nombre de personnes qui a associé une grande forme ou une petite forme pour les sons à prosodie positive vs les sons à prosodie négative, pour les sons avec la modification de F0 vs les sons sans modification de F0 (son original), pour les sons avec une voix masculine vs les sons avec une voix féminine, pour les sons avec une amplitude normalisée vs les sons sans amplitude normalisée (son original), et enfin pour les sons avec une prosodie positive vs les sons avec une prosodie négative.

Pour obtenir tous les croisements de paramètres, ces trois paramètres liés aux formes ont également été croisés avec les sons à amplitude normalisée avec la F0 modifiée, et les sons à prosodie positive vs négative avec la F0 modifiée et l'amplitude normalisée, et enfin les sons à prosodie positive vs négative avec la F0 modifiée sans normalisation de l'amplitude.

A partir de ces différentes données nous avons ainsi pu réaliser des camemberts représentatifs des résultats obtenus.

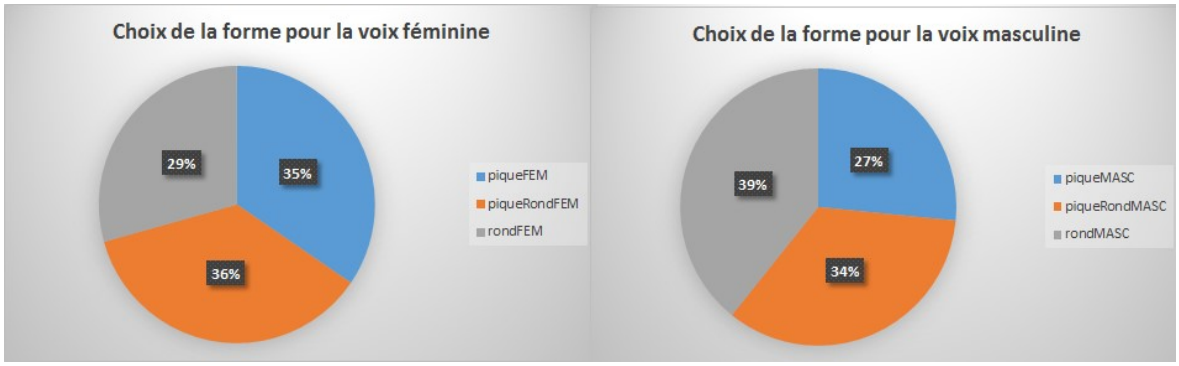

Figure 21 : Graphique du choix de la forme pour la voix masculine vs féminine

Sur ce graphique, nous pouvons observer que les sujets ont plutôt associé la voix masculine à la forme ronde. En revanche, la voix féminine a été presque autant associée à la forme piquante (35%) qu'à la forme intermédiaire (36%). La forme ronde étant plutôt perçue comme une forme pas agressive, il serait probable que la voix masculine ait été perçue comme moins agressive que la voix féminine.

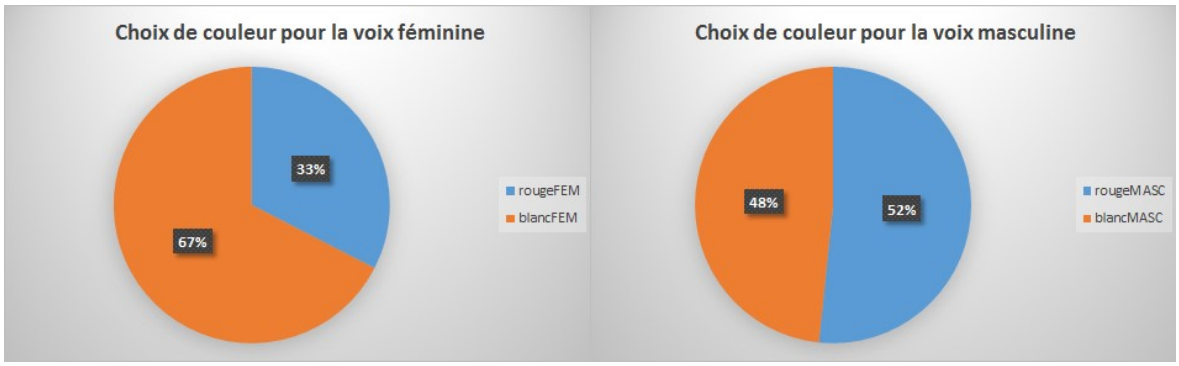

Figure 22 : Graphique du choix de couleur pour la voix féminine vs masculine

Ici, il semblerait que le choix de la couleur blanche est largement dominant pour la voix féminine. La voix masculine a plutôt été associée à la couleur rouge mais la différence n'est pas claire (il n'y a que 4% de différence entre les deux couleurs). A l'inverse du graphique précédent, la couleur blanche étant considérée comme « pas dominante », nous pourrions déduire que la voix masculine ait été perçue comme plus « dominante » que la voix féminine.

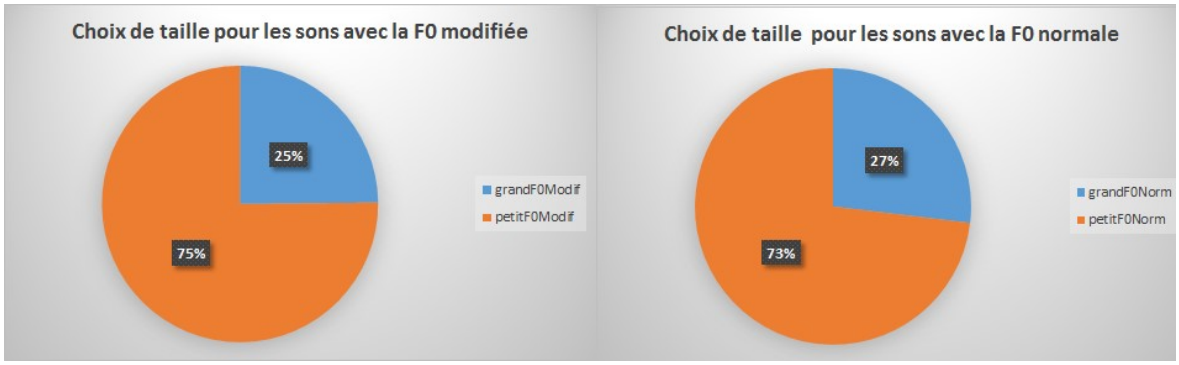

Figure 23 : Graphique de choix de taille pour la F0 modifiée vs la F0 normale

Il semblerait que la modification de la F0 ait eu très peu d'impact sur le choix de la forme. Pour tous les paramètres variables de la forme (taille, couleur, forme), le même est prédominant lorsque la F0 a été modifiée que lorsque la F0 est normale.

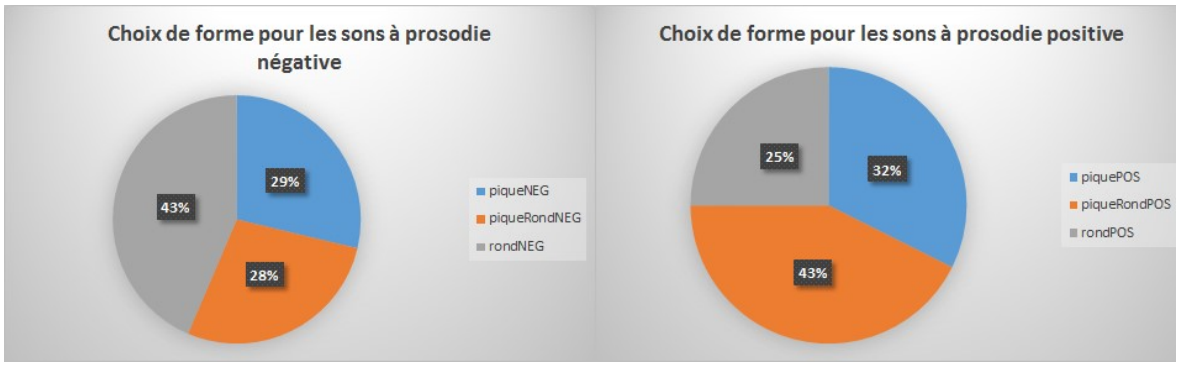

Figure 24 : Graphique de choix de forme pour la prosodie positive vs négative

En ce qui concerne la variation de prosodie (négative vs positive), il apparait que le choix de forme varie un peu plus clairement. En effet, lorsque la prosodie est négative, la forme ronde a majoritairement été choisie à 43%. En revanche, lorsque la prosodie est positive, c'est la forme intermédiaire qui a été choisi à 43%.

Figure 25 : Graphique de choix de taille pour les sons originaux vs à amplitude normalisée

Ici, nous pouvons observer que la différence de perception entre les sons originaux et les sons avec amplitude normalisée n'est pas prononcée. En effet que le son entendu soit le son original ou le son avec amplitude normalisée, la taille choisie par la plupart des sujets reste la petite taille. Cela pourrait s'expliquer par une nuance entre les sons à amplitude normalisée et les sons originaux parfois très peu audible. De plus, le fait que les sons soient mélangés lors de l'écoute et non écoutés les uns à la suite des autres, si la nuance entre les deux est faible, alors elle ne sera pas entendue car l'élément de comparaison aura été suivi par d'autres sons.

Nous observons donc quelques tendances mais qui restent plutôt faibles. Cependant cela pourrait s'expliquer par le nombre plutôt faible de réponses obtenues. Il serait donc intéressant de faire passer ce test de perception à plus de sujets, pour ensuite pouvoir faire une analyse sur plus de réponses.

Points à améliorer :

- L'attribution d'un modèle à chaque utilisateur se fait au début (lorsque l'utilisateur clique sur « commencer le test », cependant si l'utilisateur ne finit pas le test, ses données ne sont pas récupérées, et le test de ce modèle n'aura donc pas été fait. Mais le compteur des identifiants aura tout de même été incrémenté. Cela a donc demandé de vérifier de temps en temps quels modèles n'avaient pas été testés, pour pouvoir régler le problème. Cependant, nous avions également évoqué la possibilité d'incrémenter le compteur des modèles à la fin du test, mais cela posait un autre problème : si un utilisateur se connectait alors qu'une autre n'avait pas encore terminé, les deux utilisateurs testaient le même modèle. Le test étant relativement long, cela n'était pas envisageable.
- Le test de perception ne fonctionnait pas sur tablette ou téléphone. C'est un problème que nous n'avons eu le temps de poser dans le cadre de ce stage mais permettrait probablement d'avoir un nombre de réponses plus élevé.
- Il faudrait probablement un nombre de réponses plus élevé pour que le test soit plus représentatif.

Conclusion

Bilan

Dans ce mémoire, nous avons présenté le développement dans un premier temps d'une application qui permettra de tester dans un milieu écologique des primitives sonores et visuelles. Réalisée à l'aide de différents outils tels que NodeJs ou CreateJs, elle permet la création de différentes formes et couleurs qui pourront ensuite être testées en mouvement.

Dans un second temps, nous avons présenté le test de perception qui a été réalisé au cours de ce stage. Celui-ci a permis d'avoir un premier aperçu sur la perception des formes et des couleurs en fonction de primitives sonores sélectionnées pour leur valence positive ou négative. Les résultats obtenus ont permis de voir que certains paramètres de son ou de forme ont effectivement un impact. Cependant, le nombre de réponses restant peu élevé, il serait très intéressant de récupérer plus de réponses pour avoir une analyse plus poussée.

Perspectives

Ce stage s'est ainsi composé en deux parties distinctes mais complémentaires. Le test de perception ayant été réalisé après l'application, toutes les formes ayant été testées ne sont pas présentes dans l'application. Il sera probablement nécessaire, pour les besoins de l'expérience, d'implémenter ces formes dans l'application pour pouvoir les tester en mouvement notamment. Le problème étant que ces formes sont complexes, il faudrait probablement, soit les dessiner avec une fonction de dessin dans le canvas, soit pouvoir mettre en place avec CreateJs le mouvement d'une image qui comprendrait la forme et y appliquer des filtres pour pouvoir modifier la couleur de la forme par exemple.

De plus, les paramètres de création d'une forme ou d'un mouvement étant multiples dans l'application, le temps a manqué pour finaliser tous les paramètres de création que j'avais prévu initialement. Notamment au niveau de la création des mouvements, qui n'est pour l'instant pas possible car le traitement JavaScript de la page n'a pas pu être finalisé. La première étape de création de la forme dans le canvas de la page

a été finalisée. De plus, il est possible de déplacer la forme. Il faut cependant encore récupérer les coordonnées de la souris pour pouvoir mémoriser le déplacement et gérer les paramètres de vitesse de déplacement souhaité par l'utilisateur.

Enfin, comme expliqué dans le mémoire, la récupération automatique de l'adresse IP serait un élément qui pourrait considérablement faciliter l'utilisation de cette application.

Cette application pourra ainsi être utilisée pour tester différentes primitives sonores et visuelles en mouvement dans le milieu écologique et sera ainsi un outil qui pourra être utilisé pour mener des expériences dans le cadre d'une thèse.

Bibliographie

Aubergé, V., Audibert, N., Rilliard, A. (2006). Auto-Annotation : An Alternative Method to Label Expressive Corporta. *In , pp.* https://hal.archives-ouvertes.fr/hal-00364542

Aubergé, V., Sasa, Y., Bonnefond, N., Meillon, B., Robert, T., Rey-Gorrez, J., Schwartz, A., et al. (2014). The EEE Corpus: Socio-Affective "glue" Cues in Elderly-Robot Interactions in a Smart Home with the Emoz Platform. In , 27-34. Reykjavik, Iceland.

Audibert, N. (2008). Prosodie de la parole expressive : dimensionnalité d′énoncés méthodologiquement contrôlés authentiques et actés. Thèse de doctorat en ingénierie de la Cognition, d la Création et des Apprentissages, Institut polythechnique de Grenoble, Grenoble.

De Biasi, G. (2011). Étude de la perception des émotions dans les micro-événements non verbaux chez l'être humain. Mémoire de master en Traitement Automatique de la Langue Ecrite et de la Parole à l'université Stendhal, Grenoble.

Heider, Fritz, et Marianne Simmel (1944). An Experimental Study of Apparent Behavior. *The American Journal of Psychology 57 (2): 243. doi:10.2307/1416950*

Köhler, W. Gestalt Psychology, 1929. Traduction française La psychologie de la forme, *Gallimard, Paris, 1964*.

Lestel, D. (2009). Oublier la frontière homme/animal. *Le Carnet PSY 140 (9): 26. doi:10.3917/lcp.140.0026*

Milan, E., O. Iborra, de Cordoba, V. Juarez-Ramos, M.A.R. Artacho, et J.L. Rubio (2013). The Kiki-Bouba Effect A Case of Personification and Ideaesthesia. *Journal of Consciousness Studies 20 (1-2): 84- 102*.

Pantelis, Peter C., et Jacob Feldman (2012). Exploring the Mental Space of Autonomous Intentional Agents. *Attention, Perception, & Psychophysics 74 (1): 239-49. doi:10.3758/s13414-011-0215-6*

Pratt, J., P. V. Radulescu, R. M. Guo, et R. A. Abrams (2010). It's Alive!: Animate Motion Captures Visual Attention. *Psychological Science 21 (11): 1724-30. doi:10.1177/0956797610387440*

Ramachandran, Vilayanur S., et Edward M. Hubbard (2001). Synaesthesia–a window into perception, thought and language. *Journal of consciousness studies 8 (12): 3–34*.

Rosenthal-von der Pütten, Astrid, M., Nicole C. Krämer, Hoffmann, L., Sobieraj, S., et C. Eimler. S. (2012). An Experimental Study on Emotional Reactions Towards a Robot. *International Journal of Social Robotics 5 (1): 17-34. doi:10.1007/s12369-012-0173-8*

Sagisaka, Y. (2015). Sentiment analysis of color attributes derived from vowel sound impression for multimodal expression. *Asia-Pacific Signal and Information Processing Ass.*

Suzuki, Yutaka, Lisa Galli, Ayaka Ikeda, Shoji Itakura, et Michiteru Kitazaki (2015). Measuring empathy for human and robot hand pain using electroencephalography. *Scientific Reports 5 (novembre): 15924. doi:10.1038/srep15924*

Tremoulet, Patrice D, et Jacob Feldman (2000). Perception of Animacy from the Motion of a Single Object. *Perception 29 (8): 943-51. doi:10.1068/p3101*

Vanpé, A. (2011). Expressions et micro-expressions spontanées de la face et de la voix en Interaction Homme-Machine : esquisse d'un modèle du *"Feeling of Thinking"*. Thèse de doctorat en Langues, Littératures et Sciences Humaines, université de Grenoble, Grenoble.

Sitographie

Createjs. [En ligne]. [dernière consultation en août 2016]. Disponible sur : http://createjs.com/

Paint.net. [En ligne]. [dernière consultation en août 2016]. Disponible sur : http://www.getpaint.net/index.html

Nodejs. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://nodejs.org/en/

Npm. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://www.npmjs.com/

LIG plateforme Domus. [En ligne]. [dernière consultation en août 2016]. Disponible sur : http://domus.liglab.fr/

LIG Laboratoire d'Informatique de Grenoble [En ligne]. [dernière consultation en août 2016]. Disponible sur : http://www.liglab.fr/

INRIA. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://www.inria.fr/

Openclassrooms. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://openclassrooms.com/

Amiqual4home. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://amiqual4home.inria.fr/fr/home/

Stackoverflow. [En ligne]. [dernière consultation en août 2016]. Disponible sur : http://stackoverflow.com/

Wikistuce. [En ligne]. [dernière consultation en août 2016]. Disponible sur : http://www.wikistuce.info/doku.php?do=index

Creativejuiz. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://www.creativejuiz.fr/blog/javascript

Ipify. [En ligne]. [dernière consultation en août 2016]. Disponible sur : https://www.ipify.org/

Glossaire

Javascript : créé en 1995, ce langage orienté objet est un langage interprété client-side. C'est un langage basé sur les évènements qui est majoritairement utilisé dans la programmation Web mais qui commence de plus en plus à être utilisé dans des serveurs ou pour créer des applications. Souvent couplé au HTML, il interagit très facilement avec cet autre langage. C'est pourquoi JavaScript et HTML5 ont été utilisés pour la réalisation de cette application.

NodeJs : Si JavaScript est un langage client-side (exécuté par le navigateur internet), NodeJs offre la possibilité d'utiliser et d'exécuter du JavaScript côté serveur. NodeJS est aujourd'hui très utilisé pour le développement d'applications web de tous types car il est rapide et non bloquant. En effet, la rapidité de NodeJs réside dans le moteur d'exécution V8 de Google Chrome. Cet outil open source créé par Google analyse et exécute du code JavaScript très rapidement. NodeJs permet ainsi de pouvoir exécuter du JavaScript aussi bien du côté serveur que du côté client. NodeJS contenant une bibliothèque de serveur http intégrée, il permet de faire un serveur web autre que Apache par exemple (serveur web le plus utilisé). NodeJS comprend également un grand nombre de modules disponibles sur la plateforme NPM de NodeJS qui permettent de faciliter l'écriture du code sur le serveur.

Fonction de callback : C'est le principe de fonction de callback. En effet, un code non bloquant est capable d'effectuer une action et de continuer à exécuter le code même si cette action n'est pas terminée. Lorsque l'action en question sera terminée, le système y reviendra et exécutera la fonction de callbak qui dépendait de l'exécution de la première partie de l'action. Cette notion importante est très utilisée dans NodeJs.

CreateJS : CreateJs est une bibliothèque JavaScript permettant de faciliter l'écriture de code liée au Canvas de HTML5. Elle permet de créer du contenu interactif. Elle comprend différentes sous-librairies dont EaselJS qui permet de travailler avec des images par exemple ou TweenJS qui est la libraire utilisée en majorité dans ce projet car elle permet de créer des animations sur des objets JavaScript ou sur des images importées.

Table des illustrations

Table des matières

MOTS-CLÉS : application androïd, primitives, test de perception, mouvement, induction émotionnelle

RÉSUMÉ

Nous avons développé une application androïd permettant de tester les inductions émotionnelles à partir de primitives sonores et visuelles en mouvement. Il était nécessaire de permettre un « pilotage » des formes à distance pour mener des expériences au sein d'un environnement écologique dans l'appartement domotique. Le développement de cet outil a donc nécessité l'utilisation de NodeJs pour un serveur http en JavaScript et CreateJs pour l'animation des formes.

Dans un second temps, nous avons développé un test de perception en ligne de ces primitives sonores et visuelles sans mouvement. Plusieurs paramètres de formes (forme, taille, couleur) ont été retenus ainsi que différents paramètres sonores (amplitude, prosodie, genre masculin/féminin, fréquence fondamentale). Le développement de cette interface en ligne a été fait en PHP.

KEYWORDS : android app, primitives, perception test, motion, emotional inductions

ABSTRACT

We have developped an android app in order to test emotional inductions through loud and visual primitives in motion. It was important to allow a remote control to make experiences in an ecological environment in a flat with home automation. The development of this tool needed the use of NodeJs to creae an http server in JavaScript and CreateJs in order to animate the shapes.

In the second part, we have developped an online perception test with these loud and visual primitves without animation this time. Several shape parameters (shape, size, color) and loud paraeters (amplitude, male and female voice, fundamental frequency) where tested. The perception test was made with PHP.