Conception et implémentation d'un assistant virtuel de mise en scène

Mémoire de stage de DEA

Présenté par **Murat GOKSEDEF**

Sous la responsabilité d'Isis Truck et d'Alain Bonardi

DEA IA&OC
Universités Paris 8 & Paris 13

MSH Paris Nord, UMS 2553
4, rue de la Croix Faron
Plaine Saint-Denis
93210 Saint-Denis

Remerciements

Ce travail de stage n'aurait pas pu se faire sans l'aide d'un certain nombre de personnes que je souhaite tout particulièrement remercier ici.

Mes premiers remerciements iront à mes responsables de stage, Isis Truck et Alain Bonardi, qui m'ont dirigé tout au long de stage et qui m'ont témoigné leur soutien et confiance. Ils m'ont beaucoup aidé et encouragé, et leurs conseils et remarques m'ont permis de rendre ce document plus lisible et plus cohérent.

Un grand merci à Christine Zeppenfeld et Claire Maupetit qui nous ont accordé beaucoup de leur temps pour faire toutes les prises de vue et les vidéos.

Enfin, je tiens à remercier tout les gens de la MSH Paris Nord, qui savent si bien rendre agréable le cadre de travail, et tout particulièrement : Marie-Henriette Beaunier, Martine Samama, Alexandre Bonsard et Radja Aroquiaradja.

Résumé

Le projet qui m'a été proposé consiste en la réalisation d'un assistant virtuel de metteur en scène. Il doit aider le metteur en scène au niveau du jeu de l'acteur, c'est-à-dire lui permettre de détecter les émotions et/ou intentions que celui-ci exprime. Le travail a été réalisé à partir de représentations d'un opéra virtuel (*Alma Sola*) dans lequel la gestuelle de l'acteur est analysée automatiquement grâce, notamment, à un système à base de règles floues.

Mots-clefs : Assistant Virtuel, logique floue, émotions, jeu de l'acteur.

Abstract

This project consists in the realization of a virtual assistant for directors. It will help directors in the field of the actors' performance, i.e. it shall be used to recognize emotions and/or intentions during an actor's performance. This work has been done starting from a virtual opera (*Alma Sola*) in which the actor's gesture is automatically analyzed by means of a fuzzy rule-based system notably.

Keywords : Virtual Assistant, fuzzy logic, emotions, actor's play.

Table des matières

Table des figures

Liste des tableaux

Introduction

Les productions de divertissement sont actuellement très nombreuses et leur complexité (au niveau de la mise en scène, de la réalisation, du jeu des acteurs…) peut apparaître comme étant de plus en plus grande. L'analyse et la vérification, automatisée si possible, de l'exécution d'un spectacle par un acteur est quelque chose de souvent souhaitable dans la communauté des artistes. En effet, les directeurs, musiciens, compositeurs, metteurs en scène… ont souvent besoin d'outils intelligents (assistés par ordinateur) afin d'analyser, organiser et rectifier l'exécution des acteurs, chanteurs, artistes… qu'ils dirigent.

Dans ce projet, il s'agit de proposer un assistant virtuel de metteur en scène pour exprimer les intentions (émotions) que le *performer*[1] tente de faire passer, lors de sa représentation. Bien que cette problématique fasse appel à des notions très subjectives, on peut penser que la machine pourra extraire des caractéristiques permettant de lever quelque peu cette subjectivité – au moins en partie. C'est dans ce contexte et sur cette thématique qu'I. Truck et A. Bonardi m'ont proposé de travailler pour mon stage de DEA, à la MSH-Paris Nord

Tout d'abord, on a donc choisi de partir d'un spectacle vivant, un opéra virtuel nommé *Alma Sola* comprenant deux *performeuses*, qui chantent et dansent sur de la musique contemporaine. *Alma Sola* est un opéra virtuel créé par A. Bonardi et C. Zeppenfeld [Bonardi et Rousseaux, 2004] dans lequel un personnage de Faust et son ombre sont mis en scène. On y trouve quatre « parties » que les performeuses exécutent avec une certaine émotion dans la voix et dans les gestes qui varie selon les représentations. A certains moments, le metteur en scène décide précisément des émotions à transmettre, à d'autres moments ce sont les performeuses qui le décident en fonction de contraintes plus larges.

L'assistant virtuel sur lequel je dois travailler va donc servir de point de repère, c'est-à-dire qu'il aura pour rôle d'exprimer de façon objective les intentions/émotions jouées par la performeuse. En particulier, il doit générer un ensemble de représentations graphiques des intensités des émotions pour chaque partie du spectacle. Ainsi, il aide le metteur en scène à détecter les émotions de la performeuse en prenant une décision objective, en tous cas, fondée sur des calculs considérés comme justes. Comme tout système informatique de ce type, l'assistant présente comme intérêt de donner une réponse objective et systématique, analysant la scène de manière infatigable et donnant des résultats identiques pour des entrées identiques.

Pour ce projet, on a donc besoin d'analyser le mouvement et la voix des personnages. Les choix se portent plus naturellement sur des logiciels existants, permettant ce type de traitement. Pour ma part, j'ai surtout traité le problème de l'analyse du mouvement pour lequel j'ai défini des caractéristiques pertinentes que l'ordinateur doit prendre en compte pour évaluer la quantité, qualité, vitesse, etc. du mouvement. Il faut ensuite établir un système calculant des agrégations de ces caractéristiques pour discriminer les différentes émotions qu'on se donne à traiter. Ces différentes agrégations discriminantes permettent ensuite de prendre une décision quant à l'intention/émotion exprimée. Bien sûr, ce type de décision ne peut pas être défini de façon ultra-précise, puisque plusieurs valeurs d'agrégations regroupées peuvent mener à une même décision

[1] On utilise l'anglicisme *performer* pour désigner, de façon plus pratique, tout artiste réalisant une performance. Il peut s'agir d'un chanteur, d'un acteur, d'un danseur… ou d'un chanteur/danseur, etc.

(en tous cas, à peu près). Devant ce type de problème, on peut penser à l'utilisation de méthodes pour résoudre les problèmes d'imprécision, comme des méthodes issues de la logique floue, en particulier, des systèmes à base de règles floues. Nous savons que ce genre de système a été maintes fois utilisé avec succès dans diverses situations. Nous verrons, dans ce mémoire, comment toutes ces solutions énoncées ont été appliquées.

Plus particulièrement, ce document est organisé comme suit : le premier chapitre précise l'état de l'art dans le domaine des assistants dans les spectacles et dans le domaine des systèmes détectant les émotions.

Ensuite, le chapitre 2 étudie la faisabilité de l'assistant virtuel et pose les jalons du travail à réaliser. Nous détaillons les modules du système à développer et expliquons les étapes permettant de trouver les émotions dans le jeu de l'acteur.

Le chapitre 3, quant à lui, est consacré au système à bande règles floues qui est le cœur du raisonnement approximatif utilisé dans assistant. En particulier, quelques notions de logique floue sont rappelées et la conception et la mise en place des règles floues sont explicitées.

Enfin, l'implémentation de l'assistant est détaillée dans le chapitre 4 dans lequel quelques détails de programmation sont indiqués. En particulier, certains algorithmes pour le partitionnement flou des données sont avancées et discutés.

Chapitre 1 Etat de l'art

1.1. Les spectacles et les assistants virtuels

Les hommes assistent à des spectacles depuis des siècles. Il existe différents types de spectacles comme le théâtre, la danse, l'opéra et etc. Les spectateurs regardent et interprètent le spectacle selon leur personnalité et leurs connaissances. Chacun des spectateurs interprète le spectacle différemment. Il existe toutefois des types de spectacles hautement codifiés. Cela signifie que l'acteur joue le spectacle avec des gestes et expressions prédéfinis, chaque geste et chaque expression représentant une situation ou une émotion particulière. On peut citer par exemple le Kabuki (théâtre japonais qui est fondé au 17ème siècle par Okuni) et la Comedia dell'arte (forme de comédie populaire apparue en Italie dans les années 1550).

Dans la plupart des spectacles, il n'existe pas de gestes prédéfinis autres que ceux véhiculés par notre civilisation (par exemple dire oui ou non d'un mouvement de tête). Le performeur joue le spectacle selon les directives du metteur en scène pour exprimer quelque chose. On comprend qu'il peut exister un écart, subjectif, entre la prescription du metteur en scène et ce qu'exécute le performeur. L'acteur, le chanteur ou le danseur a donc besoin d'un retour précis sur sa performance.

D'une certaine manière, la recherche au sujet des assistants des performeur existe depuis des siècles. On pense immédiatement à l'utilisation des miroirs dans la danse. Le premier traité de danse fait publié en 1589 par Thoinot Arbeau[2], précisément à un moment où les miroirs ont commencé à se diffuser en Europe. La danse a donné pendant longtemps l'exemple parmi les arts de la performance. De nombreuse de systèmes de notation (Feuillet, Laban[3], Benesh, etc.) ont été développés au cours des siècles. Tous ces systèmes s'appuient sur une notation des causes du mouvement (les gestes précis à effectuer) et pas sur l'observation de son résultat. Cette notation des causes doit permettre par une analyse d'accéder aux intentions de l'auteur.

Une autre approche consiste en l'utilisation d'un autre ensemble d'intentions, celles exprimées par les interprètes telles que nous les éprouvons, sans référence directe à celles de l'auteur. Heureusement, ces deux systèmes d'intention ont une intersection non vide (nous sentons parfois que ce n'est pas le cas ou au contraire nous sentons parfois une adéquation parfaite entre le travail et ses interprètes).

Dans le domaine de la musique, par exemple, on peut "mesurer" une sorte de différence entre la prescription (typiquement les partitions) et la réalisation (le jeu des musiciens). Dans son approche des « partitions virtuelles »[4], le compositeur français Philippe Manoury a considéré [Manoury & Battier, 1987] dans ses morceaux pour instrument solo et électronique live le calcul de cette différence (*Jupiter* pour flûte solo et électronique live, 1987, *En Echo* pour voix et électronique live, 1991).

2 http://www.graner.net/nicolas/arbeau/
3 http://dancenotation.org/DNB/
4 http://mediatheque.ircam.fr/

La première étape dans cette approche consiste en la mesure de divers aspects de la performance, en utilisant des capteurs au sens large [Zeppenfeld, 2004]. Les divers capteurs[5] aujourd'hui disponibles (nous avons seulement choisi les plus « confortables » pour l'interprète) sont : l'appareil-photo, le microphone sans fil, les dispositifs d'ultrasons, les détecteurs disposés sur les tapis, la boussole numérique, etc.

Au début des années 2000, Antonio Camurri et son équipe à l'université de Gênes ont développé la première plateforme robuste pour l'analyse des gestes et des émotions du performeur. Celle-ci s'appelle EyesWeb [Camurri *et al.,* 1999]. Elle n'est pas fondée sur des capteurs implantés sur le corps du performeur (avec les batteries lourdes et transmission par radio), mais sur la capture visuelle avec une caméra en plan statique. Elle est fondée sur un langage graphique qui met en application beaucoup de descripteurs des gestes : quantité de mouvement, stabilité, etc. EyesWeb est devenu une norme mondiale pour l'analyse des gestes dans la performance musicale, chorégraphique et théâtrale.

Approximativement dix ans auparavant, dans les années 90, l'Ircam à Paris (et la compagnie Cycling 74) a développé une plateforme d'analyse du son en temps réel et de synthèse pour aboutir au logiciel Max (nom en hommage à l'un des fondateurs de l'informatique musicale, Max Matthews). Il s'agit d'un langage graphique pour le traitement du signal au sens large en temps réel. A été ensuite ajouté MSP (méthodes pour le traitement du son en temps réel), le logiciel s'appelle maintenant Max/MSP. Ce logiciel est une norme mondiale dans l'analyse du son en temps réel. Il existe des objets tels que l'`analyzer~`[6], développé par Tristan Jehan, très utiles particulièrement pour l'analyse de la voix (détection, par exemple, du "bruit" se dédoublant des harmoniques).

Une des grandes tendances de l'état de l'art actuellement consiste à inférer des informations significatives à partir de descripteurs calculés par EyesWeb et Max/MSP.

1.2. Travaux sur les émotions

Reconnaître les émotions des être humains est un domaine qui intéresse de nombreux chercheurs en informatique. Un système informatique qui interagit avec des êtres humains peut avoir besoin de connaître l'état émotif de son interlocuteur : cela lui permet de s'adapter à ses émotions. Un tel système peut aider les êtres humains en sachant éviter que leurs émotions aient des répercussions défavorables pour eux, d'ailleurs, il peut se servir de leurs émotions pour qu'ils réussissent mieux les tâches qu'ils souhaitent accomplir. Par exemple, il évitera de mettre en colère son élève ou calmera sa colère, il saura l'encourager quand il est démoralisé.
Il existe deux valeurs différentes qui sont liées à toute émotion humaine.

- La première est la *valeur instantanée* d'une émotion pour un individu, par exemple il est en colère.
- La deuxième est la valeur d'une *prédisposition permanente* d'un individu pour une émotion, par exemple il est colérique.

5 cf. le site de l'entreprise Interface-Z, à la pointe dans ce domaine :
http://www.interface-z.fr
6 http://web.media.mit.edu/~tristan/maxmsp.html

Dans notre projet, nous nous intéressons au premier type d'émotion. Pour découvrir le premier type d'émotion, il est possible d'utiliser l'observation des émotions d'un individu. On est capable de découvrir qu'un individu ressent des émotions en l'observant. S'il devient tout rouge, s'il utilise un vocabulaire grossier, s'il parle fort, on juge qu'il est en colère. Il est possible de créer un système qui prend certaines informations à propos de l'utilisateur et qui peut découvrir son état émotif. La difficulté est que ce système soit capable d'interpréter ces informations. Le Prototype Sensing System du MIT [Affective Computing 1997] est un exemple de ces systèmes. Il prend des informations sur l'utilisateur comme la pression sanguine, la température du corps et il essaie de trouver l'état émotif de l'utilisateur.

D'autres informations typiques sont aussi habituellement utilisées comme le langage, la modulation de la voix, les gestes, l'expression du visage…

Concernant l'expression du visage ; il existe deux type de travaux dans ce domaine. Le premier type est de découvrir l'état émotif d'un individu en observant son visage. C'est un domaine populaire de la physiologie. En 1978 Ekman et Friesen [Ekman et Friesen, 1978] ont créé un système qui s'appelle FACS (Facial Action Coding System). En utilisant ce système, on peut reconnaître les émotions d'un individu. Le deuxième type est de créer des têtes parlantes qui montrent les émotions pour améliorer la communication homme-machine [The Duy Bui, 2004]. Ces travaux utilisent FACS pour générer une tête parlante réaliste.

L'expression émotive dans la gestuelle a également été sujette à la recherche. [Camurri *et al*., 2003] ont analysé et ont modélisé l'expression émotive dans la danse. [Boone et Cunningham, 2001] ont étudié les modèles du mouvement des enfants quand ils écoutent de la musique avec différentes expressions émotives. Dahl et Friberg ont étudié des modèles de mouvement d'un musicien jouant un morceau avec différentes expressions émotives [Dahl et Friberg, 2004]. Ces études ont toutes suggéré des sélections particulières de mouvement liées à l'expression émotive, semblable à la façon dont nous décodons l'expression musicale. Nous suivons la suggestion que l'expression musicale est intimement couplée à l'expression des gestes et le mouvement biologique en général [Friberg et Sundberg, 1999], [Juslin *et al.,* 2002] Par conséquent, nous essayons d'appliquer une approche semblable d'analyse aux deux domaines.

Dans notre projet, on se propose de détecter l'état émotif d'une performeuse en observant ses gestes et le ton de sa voix lors d'une représentation du spectacle.

1.3. Approches similaires

Un projet relativement similaire au nôtre a été mené par des chercheurs italiens de l'université de Gênes. Dans ce projet, Antonio Camurri et son équipe ont essayé de reconnaître des émotions dans la performance musicale et dans la danse [Camurri *et al.,* 2004]. Dans ce but, ils ont proposé une analyse multimodale. Leur système est composé de quatre couches différentes. Dans la première couche, ils ont analysé les signaux physiques en utilisant les techniques de soustraction de fond, détection de mouvement, etc. Dans la deuxième couche, ils ont défini les descripteurs de la gestuelle et du son (comme *loudness, brillance, quantité de mouvement, indice de contraction).* Dans la troisième couche, ils ont défini la segmentation des gestes. Quant à la

dernière couche, ils ont utilisé des réseaux de neurones, arbres de décision et réseau Bayesiens pour reconnaître les émotions.

De plus, cette équipe a réalisé un projet appelé *MEGA PROJET* (pour Multisensory Expressive Gesture Applications (MEGA)) en collaboration avec d'autres chercheurs qui travaillent sur ce sujet [Megaproject, 2001]. Le projet consiste en la recherche d'un modèle pour reconnaître les émotions dans la performance musicale et le mouvement du corps. Il y a quatre grandes parties dans ce projet qui sont : l'analyse des gestes expressifs, la synthèse des gestes expressifs, les stratégies de mapping et l'intégration des données. L'équipe de MEGA PROJET a créé un système qui s'appelle « The MEGA system Environment ». Il comporte deux composants. Le premier composant est le logiciel EyesWeb. Le deuxième composant correspond aux analyses et synthèses des expressions des gestes. Il est utilisé dans différentes performances artistiques ("Medea" (A. Guarnieri), "La Fenice" Theatre, Venice, October 2002 (DEI)., Multimedia concert "Shells", Prato, Italy, July 6, 2003 (DIST).)

Un autre projet est mené par des chercheurs suédois du Royal Institute of Technology de Stockholm. Friberg et son équipe ont proposé notamment un algorithme temps réel pour analyser l'expression émotionnelle dans la performance musicale et le mouvement du corps [Friberg, 2004]. Dans le cadre d'un jeu « Ghost in the Cave », le joueur doit exprimer différentes émotions avec son corps ou sa voix, et ce sont ces émotions qui sont les valeurs d'entrées du logiciel. Ils utilisent le logiciel EyesWeb pour la récupération des mouvements du corps, et, pour le son, ils analysent les paramètres suivants : *sound level, instant tempo, articulation, attack rate* et *high-frequency* content. Dans leur système, ils utilisent la logique floue pour reconnaître les émotions du joueur. Leur système comporte trois couches.

- La première couche est l'analyse des descripteurs (comme *sound level, instant tempo*, quantité de mouvement, la vitesse de gestes…).
- La deuxième couche est la calibration de ces descripteurs. Ils agrègent les descripteurs dans cette couche.
- La dernière couche permet de découvrir l'état émotif du joueur. C'est ici qu'ils utilisent la logique floue.

Une des différences fondamentales avec notre projet est que nous n'avons pas de feedback immédiat. En effet, dans le jeu des Suédois, le joueur se meut et s'exprime *jusqu'à ce que* le logiciel réagisse en fonction de ce que le joueur souhaite. Et si la reconnaissance du mouvement et/ou du son ne s'est pas faite correctement, le joueur perd cette « manche » et le jeu continue. En revanche, dans notre travail, ce genre de feedback ne pourra se produire. Il s'agit plutôt d'une analyse et d'une synthèse fine et pertinente du jeu d'un acteur pour une scène donnée. Par ailleurs, l'étude du son que nous souhaitons faire s'effectuera sur la voix chantée, et non sur la voix parlée : en effet, les émotions sont souvent bien plus difficiles à détecter lorsque la personne chante que quand elle parle.

Enfin, nous souhaitons détecter davantage que les trois états émotionnels: *colère*, *endormi* et *hilare*. Il s'agit en fait d'un projet plus vaste, plus générique, qui a pour vocation d'analyser les performances d'un acteur sur scène de façon la plus fine possible, en utilisant des paramètres plus fins que ceux habituellement employés.

Chapitre 2 Assistant Virtuel de Metteur en Scène : Conception

2.1. Présentation du travail

Dans la tâche qui m'est confiée, je dois implémenter un système qui calcule automatiquement les émotions qu'un performer laisse paraître lors d'une représentation d'un spectacle. Mon travail se limite à l'étude des mouvements et ne comprend donc pas l'étude de la voix.

Le spectacle que l'on va utiliser pour faire les tests et qui sera donc pris comme exemple est un opéra virtuel appelé *Alma Sola*. Ce spectacle, écrit par A. Bonardi et C. Zeppenfeld, est assez longuement décrit en annexe (cf. annexe A). Je préciserai ici seulement qu'il s'agit d'un opéra dit moderne, mettant en scène un personnage de Faust féminin et son ombre, composé de quatre parties (*univers*), dans lesquels les actrices (deux femmes) peuvent aller et venir à leur gré. Plus concrètement, les univers sont matérialisés sur scène par des tapis munis de capteurs qui détectent la présence ou l'absence de la performeuse. Selon les mouvements et l'endroit où se trouvent les performeuses, l'accompagnement musical joué (par un ordinateur et deux musiciens) diffère.

Il s'agit d'un opéra en *forme ouverte*, c'est-à-dire que les représentations varient d'un soir à l'autre, que les performeuses choisissent quelles parties du spectacle elles vont jouer (sachant qu'elles doivent, au minimum, jouer la première, la dernière et au moins une autre partie) en fonction de leur humeur, de leur état (fatigué, joyeux, un peu malade…) du public, etc. En revanche, chaque partie peut être jouée de différentes façons (avec différentes intentions/émotions) et le metteur en scène a, bien entendu, donné des instructions pour cela. L'assistant virtuel de mise en scène doit donc donner un regard extérieur au metteur en scène afin de lui dire quelles émotions il a ressenti dans le jeu, lors de la représentation de telle partie.

Plus concrètement, j'ai imaginé un système qui met en correspondance les mouvements d'une performeuse avec un vecteur contenant les caractéristiques des émotions jouées. Pour décider quelle émotion plutôt que telle autre doit être sélectionnée par le système, j'ai utilisé un mode de raisonnement approximatif fondé sur les systèmes à base de règles floues. Ce point est discuté et détaillé dans la section 3.4.

Par ailleurs, afin d'avoir des bandes vidéo sur lesquelles travailler, nous avons fait venir le metteur en scène (Christine Zeppenfeld) et une performeuse (Claire Maupetit). Pour des raisons évidentes de commodité, nous avons enregistré *en studio* un seul personnage, en l'occurrence, le principal, c'est-à-dire Faust. La performeuse a donc joué et chanté les différents *univers* de l'opéra : une caméra la filmait, elle recevait l'accompagnement musical grâce à un casque et avait un micro cravate H.F. (hautes fréquences) pour l'exploitation de sa voix.

Figure 1 : Extrait de la journée d'enregistrement

Le système, (cf. Figure 2), comporte cinq modules qui sont décrits ci-dessous.

❶ *L'entrée du système.* Il s'agit d'un fichier vidéo correspondant à l'*univers* qui a été joué par la performeuse. En utilisant un logiciel de montage vidéo, on sépare le mouvement et le son de la performeuse en deux fichiers distincts pour trouver les descripteurs (caractéristiques discriminantes) correspondants.

❷ *L'établissement des descripteurs.* Les descripteurs sont des données discriminantes issues des fichiers vidéo et son (gestuelle et chant de la performeuse) permettant leur analyse. Exemples de descripteurs trouvés grâce au logiciel EyesWeb : la quantité de mouvement de la performeuse, la stabilité de la performeuse, etc. (cf. section 2.3).

❸ *Le vecteur des agrégateurs (VdA*).* Il constitue l'entrée du système à base de règles floues (en anglais : fuzzy rule-based system *(FRBS)*). C'est un vecteur qui a été obtenu par agrégation de certains descripteurs. Chaque valeur de ce vecteur est représentée par un nombre. On note :
VdA = $(a_1,a_2,\ldots,a_n)$ où a_i représente l'agrégation des descripteurs de rang *i*.
Les agrégateurs utilisés sont deux moyennes différentes et écart type. Enfin on a obtenu neuf variables dans le vecteur des agrégateurs.

❹ *Le vecteur des intensités des émotions (VIE).* Il représente l'intensité des émotions de la performeuse et constitue la sortie du système à base de règles floues. Les émotions ne seront pas détectées de façon binaire, mais de façon plus graduelle, comme « un peu de « émotion 1 », beaucoup de « émotion 2 », etc. On note :
VIE = $(e_1,e_2,\ldots,e_n)$ où $0<e_i<1$ représente l'intensité des émotions.

* Les mots ou sigles précédés d'un astérisque sont définis dans le lexique, à la fin du document.

❺ *Le système à base de règles floues (FRBS).* Il utilise des sous-ensembles flous et des règles floues, qui seront détaillés dans le Chapitre 3, pour la mise en correspondance avec les différentes émotions détectées.

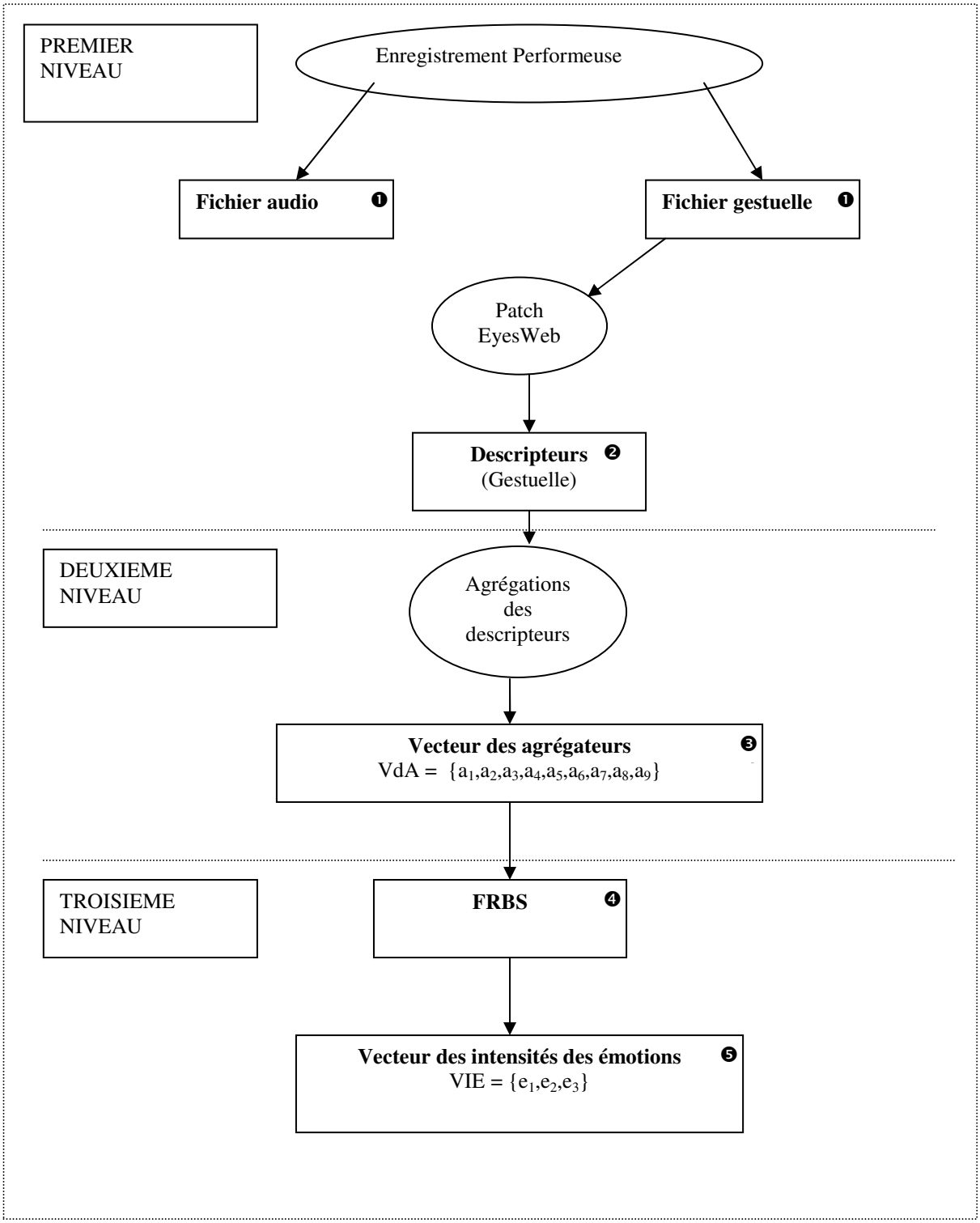

Figure 2 : Les trois niveaux et les composants du système

Le système qu'on utilise comporte trois niveaux.

- Dans le premier niveau, on analyse les vidéos et sons à l'aide des logiciels EyesWeb et Max/MSP.

  Pour analyser la vidéo, notre système utilise des patches d'EyesWeb qui nous permettent de trouver certains descripteurs. On a été intéressé aux paramètres décrivant les dispositifs généraux des mouvements plutôt qu'aux différents mouvements de chaque membre du corps. On a utilisé certains descripteurs qui ont été identifié et certains algorithmes qui ont été développées pour l'extraction automatique à partir d'entrées visuelles [Camurri *et al.*, 2004].

- Dans le deuxième niveau, on a défini des agrégateurs qui nous donnent le vecteur d'entrée de FRBS. Il s'agit de trouver des agrégateurs pour comprendre le comportement général de la performeuse. De plus, il faudra classifier les résultats des agrégateurs pour les utiliser dans le système à base de règles floues. Comme les travaux antérieurs qui utilisent la logique floue (LF), notre système offre cinq types de classes différents pour les valeurs dans le vecteur d'entrée. Il est communément admis, notamment dans la communauté de la commande floue, d'utiliser cinq classes pour caractériser les entrées de FRBS. On a modélisé ces classes en utilisant les sous-ensembles flous (cf. section 3.4.1.1).

- Le dernier niveau est la mise en correspondance (*mapping*) entre les valeurs données par les agrégateurs et les intensités des émotions. On utilise la logique floue dans cette étape. On a défini quelques règles floues pour quelques émotions à l'aide des metteurs en scène, des travaux anciens, notre observation et la performeuse. Le résultat de notre système est les intensités des émotions de chaque partie du spectacle. On a utilisé différents sous-ensembles flous pour chaque partie du spectacle.

Expliquons maintenant chaque composant plus en détail.

2.2. Gestuelle et voix

Ce composant appartient au premier niveau. Dans ce composant, il s'agit de séparer nos enregistrements en deux fichiers différents respectivement attachés à la gestuelle et à la voix.

Jusqu'à maintenant, les univers enregistrés sont le Prologue et l'univers de l'Amour. Pour qu'on puisse traiter ces enregistrements (en utilisant EyesWeb), on doit respecter quelques règles :

- La première règle à respecter est la stabilité de la caméra. Si la caméra bouge, on ne peut pas réussir à traiter ces enregistrements.
- Deuxièmement, le fond et le sol doivent être blancs et la performeuse doit s'habiller en noir (ou l'inverse) parce que EyesWeb utilise la différence de couleur pour détecter la silhouette de la performeuse.

En pratique, pour chaque univers, Claire Maupetit a chanté et joué plusieurs fois avec différentes émotions qui etaient : *Croit à l'amour, Endormi, Colère, Hilare, Peu expansif,* etc.

Après avoir enregistré ces vidéos, il fallait les analyser. Les vidéos enregistrées étaient sous le format *avi* au taux de 15 images (*frames*) par seconde.

Pour analyser et traiter ces vidéos, notre système propose de séparer le traitement des enregistrements d'une part selon la gestuelle et d'autre part selon la voix. On peut donc trouver les émotions pour chaque modalité d'expression. Notre système analyse le geste et la voix différemment pour trouver des descripteurs distincts, mais pour trouver les intensités des émotions à la fin, il prend en compte les solutions de chaque côté.

Il existe des relations communes entre l'évolution de la gestuelle et celle de la voix. Le Tableau 1 représente les résultats de travaux antérieurs [Dahl and Friberg, 2004], [Juslin, 2001]. En regardant le tableau, on constate qu'il y a des convergences entre la gestuelle et la voix dans la performance musicale. On essaie aussi de trouver des similarités entre la gestuelle et la voix dans le spectacle *Alma Sola* pour notre travail.

Emotion	**Descripteur de la gestuelle**	**Descripteur de la voix**
Colère	Large (*ample*) Fast (*rapide*) Uneven (*inégal, accidenté…*)	Loud (*fort*) Fast (*vite*) Staccato
Triste	Small (*de faible amplitude*) Slow (*lent*) Even (*régulier*)	Soft (*douce*) Slow (*lent*)
Hilare	Large (*ample*) Rather fast (*assez vite*)	Loud (*fort*) Fast (*vite*)

Tableau 1 : Une caractérisation des différentes expressions en termes de descripteurs pour les gestes et la performance musicale. (Données issues de [Dahl and Friberg, 2004] et [Juslin, 2001]).

Pour séparer les enregistrements, on utilise le logiciel Adobe Premiere qui nous donne deux fichiers qui sont respectivement le fichier de gestuelle et le fichier de voix.

2.3. Les descripteurs

Le deuxième composant du système est l'ensemble des descripteurs. Ce composant appartient au premier niveau du système comme le traitement de la gestuelle et de la voix**.** L'objectif de trouver les descripteurs de la gestuelle correspondant à notre projet. (cf. Tableau 1).

Les travaux antérieurs comme le travail de Friberg et les travaux de Camurri nous donnent une piste pour définir les descripteurs de la gestuelle. Le metteur en scène de l'opéra *Alma Sola* Christine Zeppenfeld nous a également donné quelques idées pour les descripteurs dont elle a besoin pendant le spectacle. Elle nous a communiqué les indices qu'elle utilise : par exemple, la rapidité de mouvement, l'ouverture des bras, le rapport au sol, etc.

Notre système utilise le logiciel EyesWeb pour traiter des fichiers de gestuelle et trouver les descripteurs correspondants. EyesWeb est un logiciel répandu pour le traitement et l'analyse temps réel des vidéos. On a choisi ce logiciel comme dans la plupart des travaux antérieurs. Par ailleurs, EyesWeb est un logiciel puissant, relativement facile à utiliser et gratuit. De plus, les exemples donnés par les concepteurs nous ont aidé à définir nos propres descripteurs.

On a créé un patch avec EyesWeb. L'entrée du patch EyesWeb est le fichier *avi* (le fichier de la gestuelle obtenu avec Adobe Premiere) qui contient les mouvements de la performeuse. On a implémenté quelques traitements pour définir nos descripteurs de gestuelle (cf. Tableau 2) :

- Le premier traitement est la soustraction de fond pour suivre la silhouette de la performeuse. Ce dernier nous sert à détecter les mouvements de la performeuse.
- Le deuxième traitement consiste à trouver la quantité de mouvement de la performeuse. Le composant *SMI* du module *MotionAnalysis* calcule la quantité de mouvement à partir de la silhouette de la performeuse.
- Le troisième traitement est le calcul de la stabilité de la performeuse. Les composants *Centroid Calc* et *Stability index* du module *MotionAnalysis* peuvent déterminer la stabilité de la performeuse à partir de sa silhouette.
- Dans le quatrième traitement de notre patch EyesWeb, l'indice de contraction est calculé par *Contraction Index*. Cette variable EyesWeb sert aussi à trouver les coordonnées de la matrice de contraction.
- De plus, le patch EyesWeb détermine aussi les parties où il y a des pauses dans la vidéo et les parties où il y a des mouvements. Ces parties ont été trouvées à l'aide des opérateurs mathématiques dans le module *Math* d'EyesWeb.
- Enfin, le composant *Barycenter* a été utilisé pour trouver les coordonnées du barycentre de la performeuse.

Les descripteurs de gestuelle ont été calculés par la procédure ci-dessus. On a trouvé huit descripteurs de gestuelle. Il s'agit de :

- quantité de mouvement,
- durée de pause,
- durée de mouvement,
- surface de contraction,
- index de contraction,
- stabilité
- coordonnées de matrice de contraction,
- barycentre.

Le Tableau 2 ci-après les résume.

Nom descripteur	***Description « mathématique »***	***Intervalle de valeurs***	***Description intuitive***
Durée de mouvement	Ce descripteur nous donne la durée du dernier mouvement.	[0,+∞[en ms	
Durée de pause	Ce descripteur nous donne la durée de la dernière pause.	[0,+∞[en ms	
Surface de contraction	On utilise ce descripteur pour obtenir la surface (c-à-d., le nombre de pixels) de la silhouette de l'objet analysé. C'est un entier.	[0, nombre de pixels de la vidéo[	
Indice de contraction	L'indice de contraction est calculé comme rapport entre la surface (c-à-d., le nombre de pixels) de la silhouette de l'objet et la surface de l'enveloppe de l'objet. L'index de contraction peut être considéré comme une mesure de combien de posture du corps est « *ouverte*». La sortie est entre 0 et 1	[0,1]	**Grand** : *la posture de l'acteur est ouverte* **Moyen** : …*est normale* **Bas** : … *est close*
Coordonnées de matrice de contraction	L'enveloppe (*bounding box*) calculée de la silhouette de l'objet analysé. C'est une matrice 2x2 ayant des coordonnées (x, y) du point gauche supérieur (x), et du point droit inférieur (y). Permet de savoir où se situe la silhouette de l'objet analysé.	x,y ∈ [0, nombre de pixels de la vidéo[	Situation du performer dans la scène
Stabilité	Ce descripteur est utilisé pour constater la stabilité de la silhouette de l'objet. La sortie est l'index de stabilité qui est égale au rapport entre la hauteur du centre de gravité du corps humain et la longueur du segment reliant les projections en x des pieds (ou des deux plus bas points). La sortie est entre 0 et 2.34	[0,2.34]	**Bas** : *l'acteur est proche du sol et ses jambes se sont ouvertes* **Grand :** *l'acteur est au debout et ses jambes se sont fermées*
Quantité de mouvement	Une mesure de la quantité de mouvement qui est calculée en utilisant les images de mouvement de silhouette. La sortie est un entier qui est égal au nombre de changements de pixels de la silhouette de l'objet analysé. (ramené entre 0 et 1 - égale à 0 s'il n'y a pas de mouvement - égale à 1 si toutes les parties de la silhouette sont déplacées)	[0,1] **0** : pas de mouvement **1** : toutes les parties de la silhouette sont déplacées	Vitesse de déplacement et masse déplacée (à scinder en 2 (par moyenne) On pourrait calculer la vitesse du barycentre. Donc on s'intéresse à un point et pas à un ens. de pts)
Barycentre	C'est un point qui nous donne le centre de gravité de la silhouette	x,y ∈ [0, nombre de pixels de la vidéo[	Le centre de gravité

Tableau 2 : Définitions des descripteurs.

Les figures Figure 3 à Figure 6 sont des captures d'écran issues du patch EyesWeb que j'ai créé. On peut voir notamment l'enveloppe, la quantité de mouvement et la stabilité de la silhouette, ainsi que les graphiques générés.

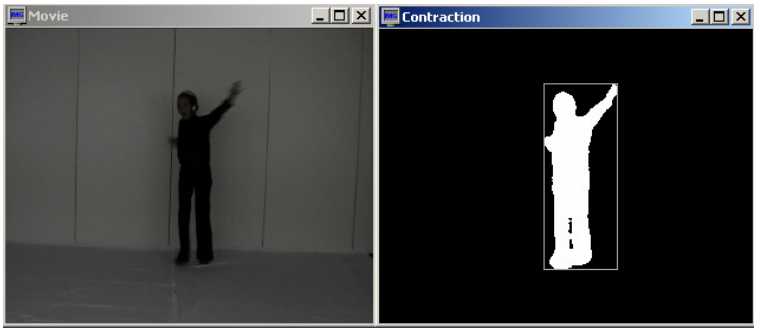

Figure 3 : A gauche, une image de la vidéo ; à droite, l'enveloppe de la silhouette correspondante.

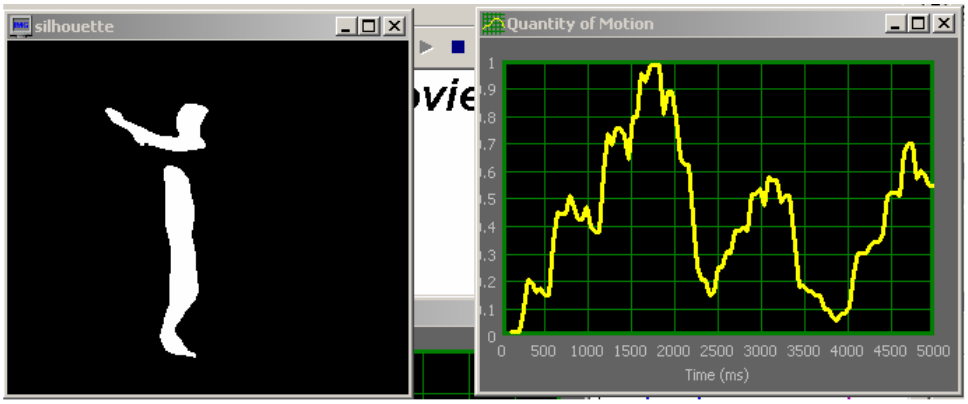

Figure 4 : A gauche, le déplacement de la silhouette ; à droite, la quantité de mouvement correspondante.

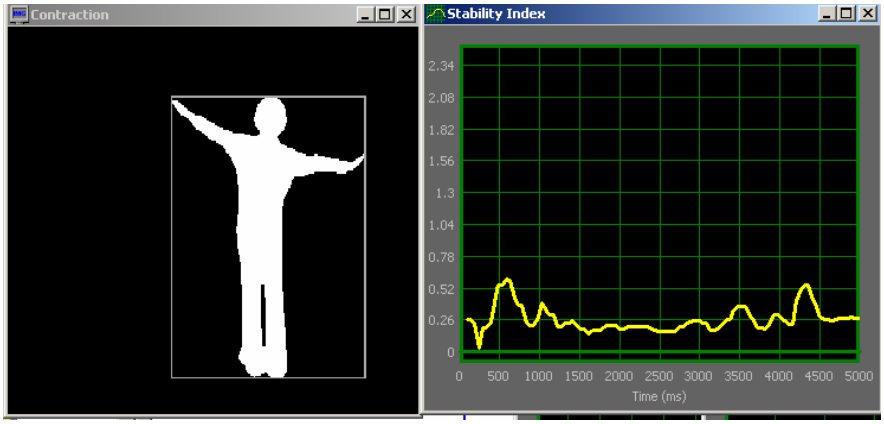

Figure 5 : A gauche, l'enveloppe de la silhouette; à droite, la stabilité correspondante.

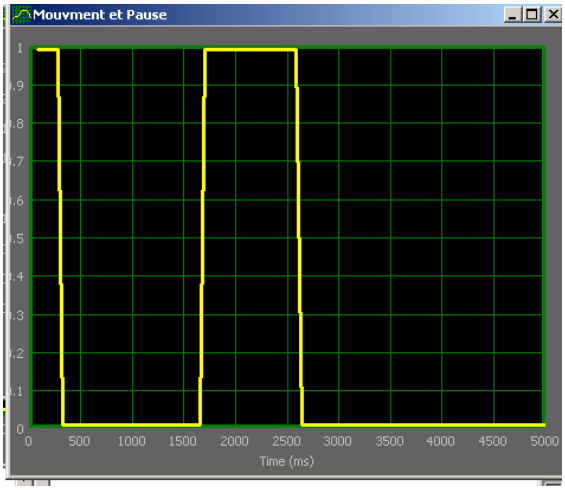

Figure 6 : Durée de mouvement et pause pour une période donnée.

Les sorties du patch EyesWeb sont des fichiers textes. Il y a un fichier texte par descripteur. Ces fichiers contiennent 15 données par seconde, qui correspondent aux valeurs du descripteur à cet instant (par exemple : pour le descripteur de quantité de mouvement, on a un fichier qui stocke les valeurs de la quantité de mouvement). Pour créer ces fichiers, j'ai utilisé le module *Output* d'EyesWeb.

On vient de présenter le premier niveau du système. Grâce à ce niveau, il est possible de définir les agrégateurs (composant suivant).

Le deuxième niveau de notre système alimente l'entrée du FRBS. Il contient le composant correspondant au vecteur des agrégateurs.

Passons maintenant au deuxième niveau.

2.4. Le vecteur des agrégateurs

2.4.1 Définition

Le vecteur des agrégateurs est donc le composant qui appartient au deuxième niveau. Le vecteur des agrégateurs est l'entrée du FRBS. Les vecteurs ont neuf composantes, qui ont été déduites à partir des descripteurs. Nous présenterons ces composantes plus tard.

Une fois qu'on a obtenu des fichiers qui contiennent les données des descripteurs, on a besoin d'information à partir de ces données. C'est la raison pour laquelle on utilise des agrégateurs. Un agrégateur permet de synthétiser plusieurs données en une seule information. Dans notre cas, les agrégateurs nous permettent de définir le vecteur d'entrée du FRBS.

En général, dans les spectacles de danse et d'opéra, il y a deux types de gestuelle :

- Le premier type correspond aux moments présence de mouvements

- Le deuxième type correspond aux moments où il n'y a pas de mouvements (Pause dans l'expression visible, mais la performeuse doit continuer à jouer intérieurement)

La performeuse peut, bien sûr, montrer ses émotions avec ou sans mouvement. Donc on doit séparer la gestuelle en deux parties. Pour cela, on utilise les descripteurs de durée de mouvement et de durée de pause.

On doit alors faire trois analyses pour la gestuelle.

- La première analyse concerne les phases avec mouvements,
- La deuxième analyse concerne les phases avec pauses
- La dernière analyse concerne les deux ensembles.

Après avoir segmenté le spectacle en deux parties (mouvement/pause), il suffit de trouver des agrégateurs pour les descripteurs. On a utilisé des formules statistiques pour agréger les données des descripteurs. En statistique, il y a plusieurs formules pour produire un résultat à partir des données. Par exemple, la moyenne. La valeur moyenne générale pour chaque descripteur est une information indispensable. On calcule d'ailleurs les valeurs partielles de chaque partie (mouvement/pause). Avec ces données, on calcule les variances et les écarts-types pour chaque partie. Ces informations sont très importantes parce qu'elles nous donnent le comportement général de la gestuelle.

Le vecteur des agrégateurs contient neuf composantes.

VdA={a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9}, où :
a_1: Taux
a_2: Moyenne générale de quantité de mouvement
a_3: Moyenne générale de stabilité
a_4: Moyenne générale de indice de contraction
a_5: Moyenne partielle de quantité de mouvement
a_6: Moyenne partielle de stabilité
a_7: Moyenne partielle de indice de contraction
a_8: Ecart type de quantité de mouvement
a_9: Ecart type de stabilité

Taux: C'est le rapport de la durée de pause sur la durée de totale. Cette variable prend toujours ses valeurs entre 0 et 1.

Moyennes Générales : Elles nous donnent le comportement général de la performeuse.

Moyennes Partielles : Elles nous donnent le comportement de la partie pause et de la partie mouvement.

Ecarts Types : Elles nous donnent les distributions des données

Après avoir défini le vecteur des agrégateurs, on a calculé ses valeurs pour nos enregistrements. Pour pouvoir établir des règles floues (cf. section 3.4.1.2) qui seront utilisées dans le FRBS, il faut classifier les résultats des agrégateurs. En général, les règles floues s'écrivent sous la forme

« si la performeuse bouge beaucoup, alors elle est heureuse », par exemple. On a donc besoin d'une classification pour chaque variable du vecteur des agrégateurs.

L'étape de classification était difficile parce que les valeurs sont spécifiques à chaque univers; heureusement, les émotions ont une certaine généralité. Par exemple, j'ai constaté que quand la performeuse exprime une émotion liée au bonheur, elle bouge plus que pour les autres émotions quel que soit l'univers du spectacle. Mais en revanche, si elle se déplace pendant le prologue, elle bouge généralement plus que pendant l'univers de l'amour. Donc la classification des variables dépend de la partie du spectacle que l'on traite. C'est pour cela que l'on fait des classifications différentes pour chaque partie.
Ci-dessous, le Tableau 3 indique les résultats des agrégateurs pour l'univers de l'amour, tandis que la Figure 7 montre graphiquement les résultats.

Agrégateurs	Croit à l'amour	Endormi	Furieux	Hilare	Peu expansif
Taux	0,601	0,848	0,6642	0,514545	0,841
Moyenne générale de QdM	0,19	0,1582	0,265042	0,302257	0,1852
Moyenne générale de Stabilité	0,19	1,597	0,314922	0,215988	0,23702
Moyenne générale d'Indice de Contraction	0,4816	0,609	0,569448	0,517243	0,635798
Moyenne partielle de QdM	0,07911	0,0226	0,0915993	0,14875	0,0302762
Moyenne partielle de Stabilité	0,213867	1,56968	0,333692	0,235331	0,248647
Moyenne partielle d'Indice de Contraction	0,483434	0,606	0,560919	0,462505	0,626581
Ecart-type QoM	0,144138	0,07958	0,202213	0,23105	0,0912
Ecart-type Stabilité	0,0879	0,646255	0,155892	0,095	0,0738346

Tableau 3 : Résultats des agrégateurs pour l'univers de l'amour.

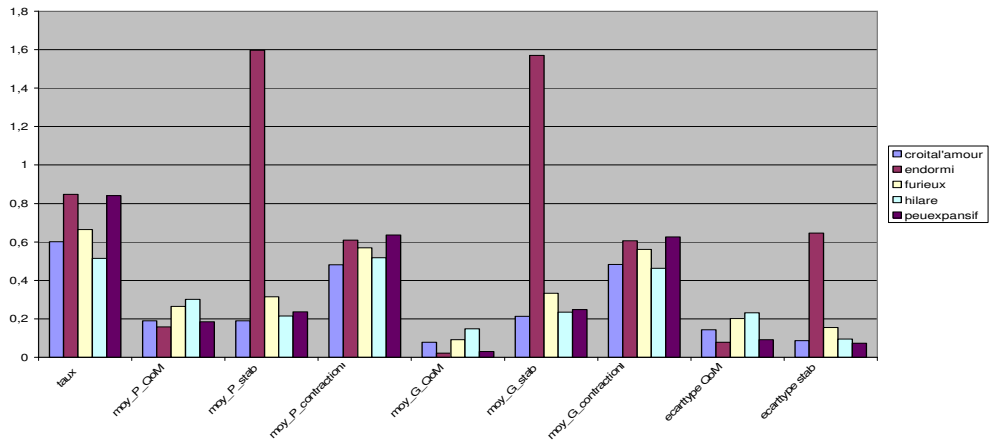

Figure 7 : Résultats des agrégateurs pour l'univers de l'amour sous forme de graphe

2.4.2 Construction des classes et classification

Pour l'étape de construction des classes, l'idée est de répartir les différentes valeurs prises par chacun des agrégateurs. Une distribution uniforme des valeurs sur l'axe des abscisses semblait cohérente.

C'est pourquoi le premier algorithme que j'ai proposé utilisait les mêmes intervalles pour tous les univers du spectacle. L'intervalle des valeurs prises par le vecteur des agrégateurs était alors divisé en cinq classes qui étaient : Très Petit, Petit, Moyen, Grand, Très Grand. Toutes les classes étaient définies sur une même longueur d'intervalle. Par exemple, pour la variable *taux* comprise entre 0 et 1, on obtenait cinq intervalles égaux : 0-0.2 → très petit, 0.2-0.4 → petit, 0.4-0.6 → moyen, 0.6-0.8 → grand, 0.8-1 → très grand. Pour mémoire, l'annexe B présente ce premier algorithme de construction de classes.

Mais j'ai constaté qu'avec cet algorithme, la plupart des données du vecteur appartenait à la même classe. Il fallait donc trouver une autre solution.
L'autre algorithme que j'ai trouvé distribue mieux les données dans les différentes classes. Il distribue aussi les données en cinq intervalles comme l'algorithme précédent: Très petit, Petit, Moyen, Grand, Très Grand. Cette fois, la valeur minimum de nos données est prise comme borne inférieure pour la classe Petit. Ce qui signifie que les valeurs (calculées ultérieurement, lors d'un autre enregistrement) qui seraient plus petites que cette valeur minimum seront catégorisées comme des valeurs de la classe Très Petit. De même, la valeur maximum de nos données est prise comme borne supérieure pour la classe Grand. Ce qui signifie que les valeurs ultérieures qui seraient plus grandes que cette valeur maximum seront catégorisées comme des valeurs de la classe Très Grand. Ensuite, j'ai divisé l'intervalle entre la valeur maximum et la valeur minimum en trois sous-intervalles, puisqu'il reste trois classes à créer : Petit, Moyen et Grand. Pour mémoire, l'annexe B présente ce deuxième algorithme de construction de classes.

Cette façon de catégoriser est meilleure que le premier algorithme de classification mais on constate à nouveau que certaines données qui devraient discriminées sont regroupées.
Pour éviter ce problème, j'ai proposé une catégorisation qui dépend aussi de la distribution des données. Cette méthode permet de construire les classes Très Petit et Très Grand de la même façon que le deuxième algorithme présenté. La différence réside dans l'utilisation de la valeur moyenne. Cette fois, je divise l'intervalle entre la valeur minimum et la valeur maximum en trois parties inégales, c'est-à-dire que je procède à une distribution non uniforme des données, comme le font notamment Martinez *et al.* avec des systèmes de notation utilisant des hiérarchies linguistiques sur des échelles non uniformément distribuées [Martinez *et al*., 2005].
Tout d'abord, on divise l'intervalle situé entre la valeur minimum et la valeur moyenne en trois parties. Il y a donc trois intervalles égaux entre la valeur minimum et la valeur moyenne. La classe Petit reprend les deux premiers intervalles de cette partie, on fait la même chose pour l'intervalle entre la valeur moyenne et la valeur maximum. La classe Grand correspond aux deux derniers fragments de cet intervalle. Quant à la valeur moyenne, une donnée est classifiée comme une valeur moyenne, si elle se trouve entre les intervalles Petit et Grand. Avec cet algorithme, j'ai constaté que les données obtenues sont correctement distribuées (cf. Figure 8). L'annexe B présente ce dernier algorithme retenu.

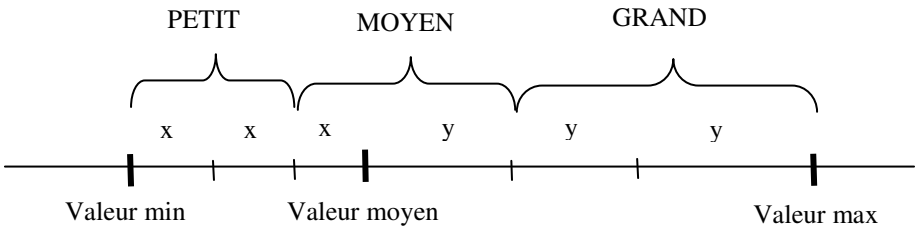

Figure 8 : Exemple de catégorisation des données

A titre d'exemple, le Tableau 4 présente l'ensemble des bornes inférieures de chaque intervalle pour chaque agrégation pour l'univers de l'amour.

Agrégateurs	Très Petit	Petit	Moyen	Grand	Très Grand
Taux	0	0,514545	0,63401433	0,745166	0,848
Moyenne générale de QdM	0	0,1582	0,1994932	0,2475122	0,302257
Moyenne générale de Stabilité	0	0,19	0,40399067	0,8729906	1,597
Moyenne générale d'Indice de Contraction	0	0,4816	0,53561187	0,5870112	0,635798
Moyenne partielle de QdM	0	0,0226	0,05717807	0,0992280	0,14875
Moyenne partielle de Stabilité	0	0,213867	0,41811793	0,8700556	1,56968
Moyenne partielle d'Indice de Contraction	0	0,462505	0,51942687	0,5741188	0,626581
Ecart-type QoM	0	0,07958	0,12628413	0,1767741	0,23105
Ecart-type Stabilité	0	0,0738346	0,16579575	0,3566025	0,646255

Tableau 4 : Bornes inférieures pour chaque agrégateur dans le cas de l'univers de l'amour.

Avec ces intervalles, on obtient une classification distribuée comme recherchée. Ainsi, le Tableau 5 montre le résultat des classifications, toujours dans le cas de l'univers de l'amour.

Agrégateurs	Croit à l'amour	Endormi	Colère	Hilare	Peu expansif
Taux	Petit	Grand	Moyen	Petit	Grand
Moyenne générale de QdM	Moyen	Petit	Grand	Grand	Petit
Moyenne générale de Stabilité	Petit	Grand	Petit	Petit	Petit
Moyenne générale d'Indice de Contraction	Petit	Grand	Moyen	Petit	Grand
Moyenne partielle de QdM	Moyen	Petit	Moyen	Grand	Petit
Moyenne partielle de Stabilité	Petit	Grand	Petit	Petit	Petit
Moyenne partielle d'Indice de Contraction	Petit	Grand	Moyen	Petit	Grand
Ecart-type QoM	Moyen	Petit	Grand	Grand	Petit
Ecart-type Stabilité	Petit	Grand	Petit	Petit	Petit

Tableau 5 : Classification pour l'univers de l'Amour.

2.5. Le vecteur des intensités des émotions

Avant d'expliquer le FRBS (quatrième composant de notre système) on présente le vecteur des intensités des émotions. Ce dernier est le cinquième composant de notre système. Il appartient au troisième niveau comme le FRBS. Le vecteur des intensités des émotions est la sortie du FRBS. Il contient trois composants qui représentent les intensités de l'émotion *hilare*, *endormi* et *colère*.

VIE = $\{e_1,e_2,e_3\}$

où
e_1 = intensité de l'émotion « *hilare* »
e_2 = intensité de l'émotion « *endormi* »
e_3 = intensité de l'émotion « *colère* »

Chaque donnée du vecteur VIE prend ses valeurs entre 0 et 1. Autrement dit, si l'intensité de l'émotion est Petit, alors la valeur de e_i est faible. Sinon la valeur de e_i est grande.

Chapitre 3 Le système à base de règles floues

Le système à base de règles floues (FRBS) est le quatrième composant de notre système. Il appartient au troisième niveau du système (cf. Figure 2). Ce composant prend le vecteur des agrégateurs (VdA) en entrée. La sortie du FRBS est le vecteur des intensités des émotions.

Pour mieux expliquer ce composant, je propose de définir et expliquer sommairement ce qu'est le concept de logique floue.

3.1. Présentation sommaire de la logique floue

Les êtres humains doivent souvent traiter une information qui n'est pas sous une forme précise ou numérique. Inspiré de cette observation, Zadeh a développé une théorie, dite *théorie des ensembles flous*, qui utilise des concepts qui n'ont pas de frontières pointues bien définies [Zadeh, 1965]. Contrairement à la théorie des ensembles classiques dans laquelle un objet peut seulement être un « élément à part entière » ou bien « non élément d'un ensemble », un objet dans la théorie des ensembles flous peut posséder une appartenance *partielle* à un ensemble.

Le degré d'appartenance mesure dans quelle proportion un objet appartient à un ensemble flou. Cette valeur est un nombre réel entre 0 et 1. Un ensemble flou est caractérisé par une fonction d'appartenance qui indique la valeur d'appartenance de l'objet. Un sous-ensemble *ordinaire* (en anglais : *crisp set*) est un cas très spécial d'un ensemble flou dans le sens que maintenant les valeurs d'appartenance sont limitées à deux éléments qui sont 0 et 1, valeurs évidemment incluses dans l'intervalle [0 ; 1].

Une proposition floue sous la forme « si X est A » est partiellement satisfaite si l'objet X a une appartenance partielle à l'ensemble flou A. D'après ceci, la logique floue a été développée pour traiter les règles « si-alors » où la condition du « si » est une combinaison booléenne de propositions floues. Quand la « condition si » est partiellement satisfaite, la conclusion d'une règle floue est calculée selon le degré avec lequel la condition est satisfaite.

3.2. Quelques définitions

Dans cette partie, on explique quelques notions fondamentales de la logique floue.

Dans un ensemble de référence E, d'après [Zadeh, 1965], un sous-ensemble flou de ce référentiel E est caractérisé par une fonction d'appartenance μ de E dans l'intervalle des nombres réels [0,1] (degré d'appartenance qui est l'extension de la fonction caractéristique d'un sous-ensemble classique). En fait, un sous-ensemble flou est formellement défini par l'application μ, mais pour se ramener au langage des mathématiques classiques, nous parlerons d'un ensemble flou A, et noterons μ_A sa fonction d'appartenance.

Pour un sous-ensemble flou A d'un référentiel E, on donne les définitions suivantes, visibles en Figure 9.

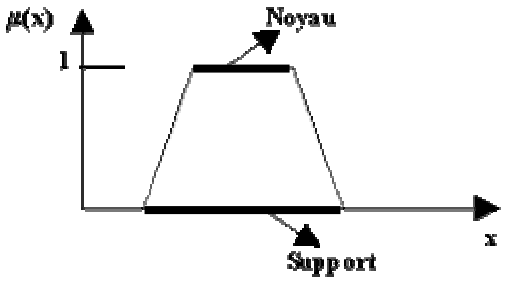

Figure 9 : Noyau et support

Noyau : Les éléments « vraiment » dans A.

$$Noyau(A) = \{x / \mu_A(x) = 1\}$$

Support : Les éléments appartenant au moins un peu

$$Support(A) = \{x / \mu_A(x) \neq 0\}$$

Hauteur : C'est la borne supérieure de la fonction d'appartenance

$$Hauteur(A) = \sup_{(x \in E)} \mu_A(x)$$

D'autres définitions mettent en jeu plusieurs ensembles de référence.

3.2.1. Relation floue

Soient X et Y deux ensembles de référence. Une relation floue R entre X et Y est définie par:

$$R = \{((x, y), \mu_R(x, y)) \mid (x, y) \in X \times Y\}$$
$$\mu_R : X \times Y \rightarrow [0,1]$$

3.2.2. t-norme et t-conorme

t-norme :

$T : [0,1] \times [0,1] \rightarrow [0,1]$
$x, y \rightarrow z = x \; T \; y$

avec les propriétés suivantes:
- commutativité : $x \; T \; y = y \; T \; x$
- associativité : $(x \; T \; y) \; T \; z = x \; T \; (y \; T \; z)$
- non-décroissance par rapport aux arguments : si $x \leq y$, $w \leq z$, alors $x \; T \; w \leq y \; T \; z$
- avec 0 comme élément absorbant et 1 comme élément neutre :
$0 \; T \; x = 0$; $1 \; T \; x = x$

Ces propriétés sont importantes et doivent être vérifiées afin que le raisonnement soit valide, puisque les t-normes sont utilisées dans la phase de *raisonnement approximatif* (cf. section 3.2.3).

Ex: "Min" est une t-norme.
L'opérateur *et* peut être défini à l'aide d'une norme triangulaire.

- ET (intersection) : $\mu_{A\cap B}(x,y) = \mu_{ET}(x,y) = \min(\mu_A, \mu_B)$

t-conorme:

$\perp : [0,1]\times[0,1] \rightarrow [0,1]$

avec les propriétés suivantes:
- commutativité $x \perp y = y \perp x$
- associativité $(x \perp y) \perp z = x \perp (y \perp z)$
- non-décroissance par rapport aux arguments si $x \leq y$, $w \leq z$, alors $x \perp w \leq y \perp z$
- avec 1 comme élément absorbant et 0 comme élément neutre :
$1 \perp x = 0$; $0 \perp x = x$

On peut définir une t-conorme à partir d'une t-norme :
$x \perp y = 1 - (1 - x)\ T\ (1 - y)$ (loi de De Morgan dans la théorie des ensembles)

Ex : « Max » est une t-conorme.
L'opérateur *ou* peut être défini par la donnée d'une t-conorme :

- OU (union) : $\mu_{A\cup B}(x,y) = \mu_{OU}(x,y) = \max(\mu_A, \mu_B)$

Les t-normes et t-conormes les plus fréquemment utilisées sont données dans le tableau suivant.

t-norme	**t-conorme**	**Négation**	**Nom**
min(x,y)	max(x,y)	1 – x	Zadeh
x.y	x + y – xy	1 – x	Probabiliste
max(x + y - 1, 0)	min(x + y, 1)	1 – x	Lukasiewicz
x si y = 1 y si x = 1 0 sinon	x si y = 0 y si x = 0 1 sinon	1 – x	Drastique

Tableau 6 : Quelques exemples de t-normes et t-conormes.

3.2.3. Modus Ponens Généralisé (MPG)

Le problème de représentation en logique floue consiste à passer d'une règle floue, qui est un objet linguistique à une relation floue, qui est un objet mathématique.

Zadeh a étendu la règle du *modus ponens* de la logique classique au contexte flou :
- *modus ponens :* { *x est A*, *si x est A alors y est B* } implique *y est B*

- *modus ponens généralisé :* on connaît les deux prémisses
(1) *si x est A'*
(2) *si x est A alors y est B*
On peut déduire la conséquence *y est B'*
A, B, A' et B' sont des sous-ensembles flous.
B' est caractérisé par la fonction d'appartenance suivante:

$$\forall y, \mu_{B'}(y) = \sup_{x \in X} \min(\mu_{A'}(x), \mu_R(x, y))$$

où R est la relation floue ou règle d'inférence représentant la règle (2) et *min* la t-norme associée à l'opérateur *et*.

Raisonner de la sorte constitue la base de ce que l'on nomme habituellement le *raisonnement approximatif*.

3.2.4. Commande Floue

La commande floue est utilisée dans les FRBS. Notre système utilise certaines notions de commande floue pour calculer le vecteur des intensités des émotions.

Un problème typique en commande floue est souvent caractérisé par trois composantes importantes. La première est l'ensemble de règles floues à utiliser. La deuxième composante est l'entrée numérique du système. La dernière est la sortie numérique.

Pour résoudre ce problème, on distingue trois étapes :

- La quantification floue des entrées/sorties du système
  - Fuzzification
- L'établissement des règles liant les sorties aux entrées
  - Humain/experts
- La combinaison des règles pour génération des sorties
  - Défuzzification et MPG

3.3. FRBS

Un système à base de règles floues (FRBS) se compose d'un ensemble de règles floues pour déterminer la relation entre les entrées et les sorties. Chaque règle du système est de type :

$$\begin{aligned} &Si \quad (x_1 \ est \ I_{1,k}) \ et \ (x_2 \ est \ I_{2,k}) \ et \ \mathrm{K} \ et \ (x_n \ est \ I_{n,k}) \\ &Alors \quad y \ est \ O_k \qquad\qquad (k = 1..m) \end{aligned}$$

Où m est le nombre de règles floues du système ; x_1, x_2..., x_n représentent les entrées avec

$I_{1,k}$, $I_{2,k}$.., $I_{n,k}$ leurs sous-ensembles flous associés de fonction d'appartenance $\mu_{I_{1,k}}$, $\mu_{I_{2,k}}$, …, $\mu_{I_{n,k}}$; y est la sortie et O_k son ensemble flou associé de fonction d'appartenance μ_{O_k}

Quand les valeurs données en entrée au FRBS sont des valeurs numériques (ce qui est le cas dans ce projet), celles-ci sont converties en un degré d'appartenance aux sous-ensembles flous associés utilisés par le système. Si une valeur numérique en entrée appartient partiellement à plusieurs sous-ensembles flous, alors elle peut déclencher plusieurs règles floues. Le degré selon lequel la règle k est activée est calculé de la façon suivante :

$$\alpha_k = \mu_{I_{1,k}}(x_1) \wedge \mu_{I_{2,k}}(x_2) \wedge \mathrm{K} \wedge \mu_{I_{n,k}}(x_n)$$

Où $\mu_{I_{i,k}}(x_i)$ est la fonction d'appartenance de x_i au sous-ensemble flou $I_{i,k}$. Le degré d'appartenance à O_k de la sortie y est calculé par:

$$\mu_{O_k}^{\inf} = \alpha_k \wedge \mu_{O_k}(y)$$

Le degré d'appartenance de la sortie y est calculé par :

$$\mu_{O}^{total}(y) = \mu_{O_1}^{\inf}(y) \vee \mu_{O_2}^{\inf}(y) \vee \mathrm{K} \vee \mu_{O_m}^{\inf}(y)$$

Ensuite, la fonction d'appartenance associée à la sortie est *défuzzifiée* dans le but d'obtenir une sortie numérique. En fait, il s'agit de transformer une information *qualitative* (ex. : y est B' et C') en une information *quantitative* (ex. : y vaut telle valeur numérique).

Il existe plusieurs méthodes pour la défuzzification, mais pas de procédure permettant de conclure sur le choix de la meilleure méthode. Les méthodes les plus couramment utilisées sont la méthode du centre de gravité ou celle de la moyenne des abscisses de maximum.

Méthode du centre de gravité:

$$y_{final} = \frac{\int \mu_O^{total}(y) \times y \times dy}{\int \mu_O^{total}(y) \times dy}$$

Un FRBS fournit une manière rapide, simple et efficace pour décrire un système dont les règles floues sont écrites en langage naturel comme, par exemple, "si la vitesse du véhicule est faible alors la consommation d'essence est basse". L'utilisation des sous-ensembles flous est appropriée pour traiter des concepts naturellement vagues tels que *petit*, *grand*, etc. De plus, les règles floues nous permettent d'incorporer des descriptions *qualitatives* comme *petit*, *grand*… et de les mettre en relation avec d'autres descriptions *qualitatives*. Ainsi, dans notre projet, la modélisation de relations entre les agrégations des descripteurs et les émotions peut se faire avec des règles floues, du type de celles que l'on vient de voir ci-dessus.

3.4. Le Système

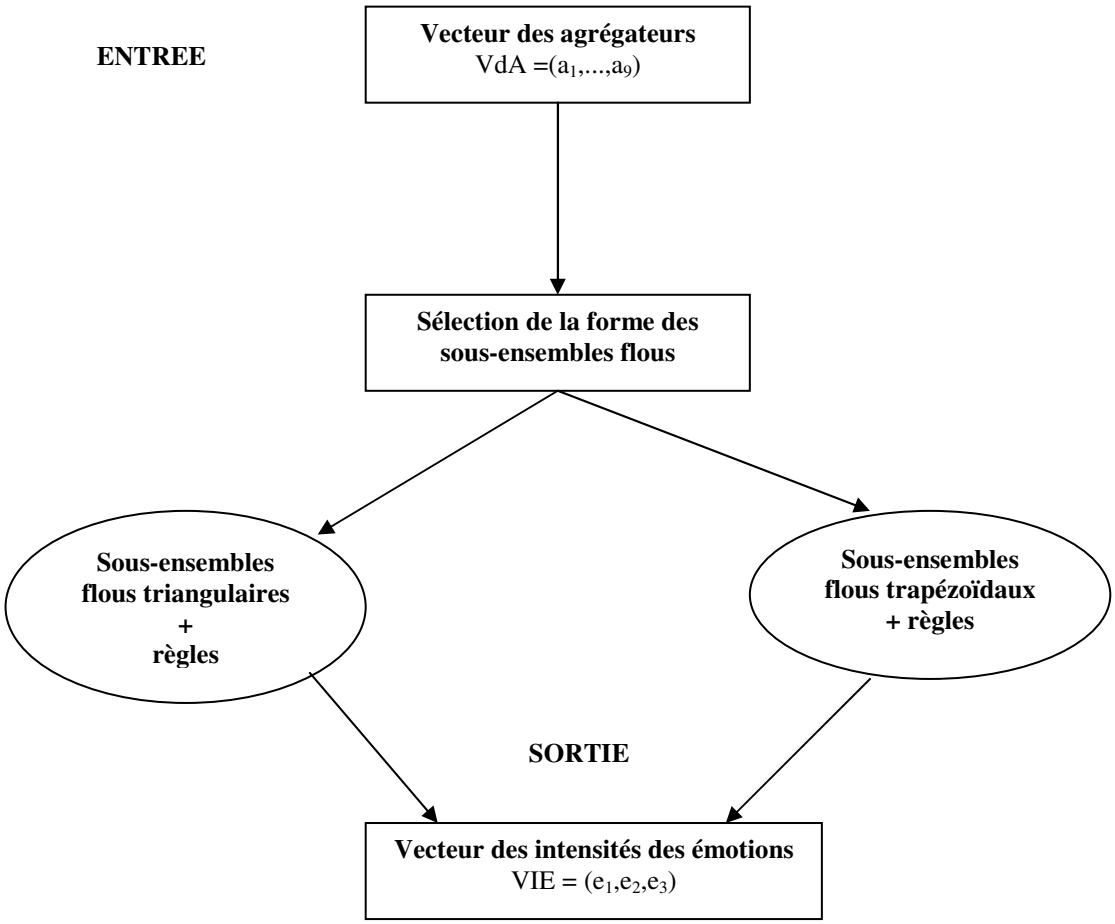

Figure 10 : Le FRBS de notre système

Notre système est fondé sur le FRBS. On prend pour entrée le vecteur des agrégateurs (VdA) et pour sortie le vecteur des intensités des émotions (VIE). Afin de déterminer au mieux les valeurs prises par le VIE, on utilise les notions de logique et commande floue évoquées ci-dessus.

La première étape est l'étape de *fuzzification*. Pour cette étape, il existe plusieurs méthodes, dont la plus simple (celle que nous avons utilisée) est appelée méthode du singleton. Cette méthode est utilisée quand la mesure de la variable x_0 (l'entrée de la commande flou) est certaine. Ainsi si la mesure x_0 est exacte, le sous-ensemble flou X_0 doit être représenté par un fait précis. Par conséquent, on utilise comme opérateur de fuzzification la transformation dite du singleton. La fonction d'appartenance du sous-ensemble flou X_0 est alors définie par:

$$\mu_{X_0} : E \to E, \mu_{X_0}(x) = 1 \quad si\ x = x_0; \mu_{X_0} = 0\ si\ x \neq x_0$$

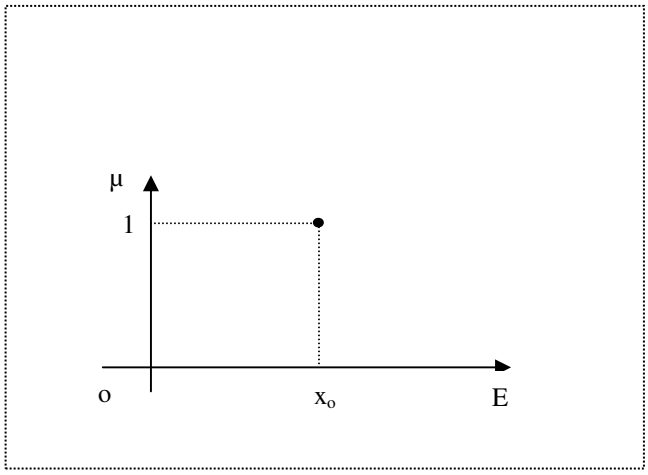

Figure 11 : Méthode du singleton

Par contre, si la mesure de la variable est entachée d'erreur, le sous-ensemble flou X_0 doit être représenté par un fait imprécis. Dans notre projet, pour la fuzzification, on utilise la formule suivante (cf. Figure 12) :

$$\mu_{x_0}(x) = \max\{0;1 - \frac{|x - x_0|}{x_0 \times p}\} \text{ où:}$$

p est un paramètre (fixé à 0,3)

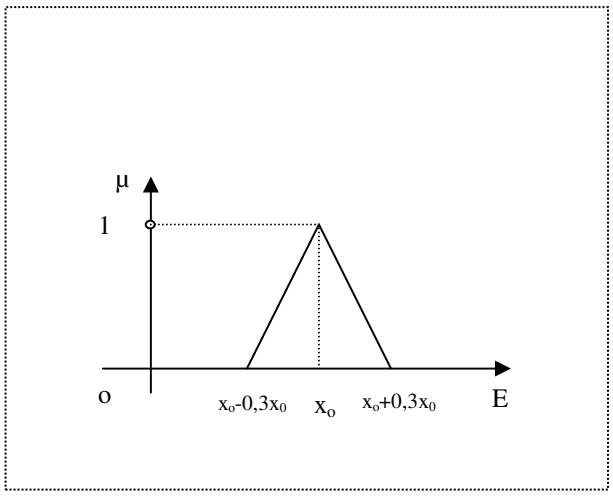

Figure 12 : Méthode du SEF triangulaire

En outre, l'utilisateur peut choisir la méthode de fuzzification, puisque les deux méthodes sont implémentées.

La deuxième étape concerne les familles des sous-ensembles flous manipulés. En particulier, notre système utilise deux collections de sous-ensembles flous qui sont employés pour reconnaître l'intensité des émotions. Des sous-ensembles trapézoïdaux constituent la première collection et des sous-ensembles flous triangulaires la deuxième. Ainsi, l'utilisateur peut choisir l'une des deux collections dans notre système, selon son choix. Celle-ci sera donc utilisée dans

l'étape de fuzzification et dans les règles pour obtenir une sortie. A la fin, la sortie est défuzzifiée pour obtenir une sortie numérique.

Les règles floues "si-alors" sont fondées sur un assez long travail de concertation avec le metteur en scène Christine Zeppenfeld. J'ai aussi utilisé certains travaux antérieurs comme celui de Friberg et celui de Camurri. Par ailleurs, j'ai observé longuement les mouvements de la performeuse, et cette analyse a aussi joué un rôle dans la constitution des règles.

Ainsi le FRBS se divise en trois parties : la sélection de la forme des sous-ensembles flous, les sous-ensembles flous triangulaires et les règles, les sous-ensembles flous trapézoïdaux et les règles.

3.4.1. Le FRBS créé

Les deux types de sous-ensembles flous (triangulaires et trapézoïdaux) sont utilisés dans les règles floues. Deux sous-systèmes doivent donc convertir le vecteur d'agrégateurs en vecteur d'intensités des émotions.

Dans un spectacle comme un opéra, des émotions qualitatives comme « *hilare* » ou « *endormi* » peuvent être trouvées. On peut décrire ces émotions en utilisant le langage naturel, ce qui peut donner, par exemple : « si la performeuse est hilare, elle bouge beaucoup », ou encore : « si elle est endormie, elle est souvent proche du sol », etc. Pour pouvoir utiliser ces concepts dans notre travail, je les ai transformés en règles floues.

Expliquons maintenant les sous-ensembles flous et les règles utilisées dans notre système.

3.4.1.1. Les sous-ensembles flous

Le système utilise 9×5 (cinq sous-ensembles flous pour chaque agrégateur), soit 45 sous-ensembles flous pour prendre une décision.

Les sous-ensembles flous pour chaque composant du vecteur des agrégateurs sont :
Très Petit, Petit, Moyen, Grand, Très Grand.

Les fonctions d'appartenance (supports, noyaux…) sont déterminées par les algorithmes de fuzzification détaillées dans les sections 3.4.1.1.1 et 3.4.1.1.2, sous forme de triangles ou de trapèzes.

3.4.1.1.1. SEFS triangulaires

Un exemple de SEFs triangulaires pour les classes Très Petit (TP), Petit (P), Moyen (M), Grand (G), Très Grand (TG) est visible dans la Figure 13.

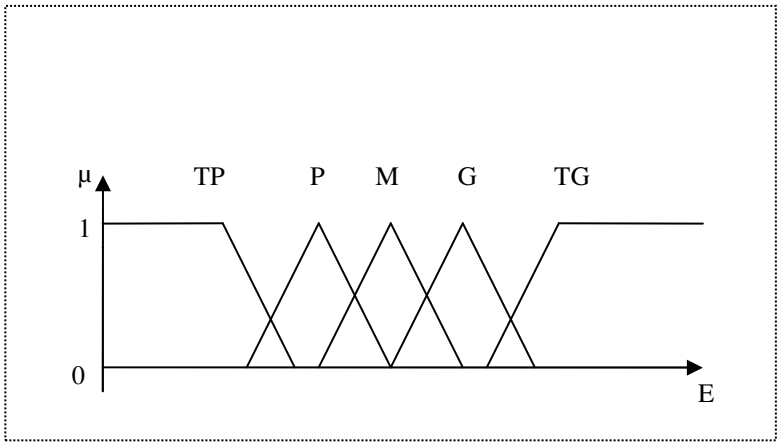

Figure 13 : Un exemple de SEFs triangulaires

Les fonctions d'appartenance et les supports de sous-ensembles flous dans ce mode (cf. la section 2.4.2 qui explique la construction de classe) sont définis ainsi:

Le sous-ensemble Très Petit:

int x ;

x = la borne inférieure de la classe moyen – la borne inférieure de la classe petit ;

borne inférieure du support = 0

borne supérieure du support = la borne inférieure de la classe petit – x/10 ;

borne inferieure du noyau = 0

borne superieure du noyau = la borne inférieure de la classe petit

:

Le sous-ensemble Petit

int x ;

x = borne inférieure de la classe moyen – la borne inférieure de la classe petit;

borne inférieure du support = la borne inférieure de la classe petit - x/10 ;

borne supérieure du support = la borne inférieure la classe moyen + x/10 ;

Noyau (singleton) : borne inférieure du support + borne supérieure du support

2

Le sous-ensemble Moyen:

int x ;

x= la borne inférieure de la classe grand – la borne inférieure de la classe moyen;

borne inférieure du support = la borne inférieure de la classe moyen- x/10 ;

borne supérieure du support = la borne inférieure de la classe grand + x/10 ;

Noyau (singleton) : borne inférieure du support + borne supérieure du support

2

Le sous-ensemble Grand

int x ;

x= la borne inférieure de la classe très grand– la borne inférieure de la classe grand;

borne inférieure du support = la borne inférieure de la classe grand - x/10 ;

borne supérieure du support = la borne inférieure de la classe très grand + x/10 ;

Noyau (singleton) : borne inférieure du support + borne supérieure du support

2

Le sous-ensemble Très grand

int x ;

X= la borne inférieure de la classe très grand – la borne inférieure de la classe grand;

borne inférieure du support = la borne inférieure de la classe très grand - x/10 ;

borne supérieure du support= la valeur maximum possible ;

borne inférieure du noyau = la borne inférieure de la classe très grand;

borne supérieure du noyau= la valeur maximum possible ;

3.4.1.1.2. SEFs trapézoïdaux

Un exemple de SEFs trapézoïdaux pour les classes Très Petit (TP), Petit (P), Moyen (M), Grand (G), Très Grand (TG) est visible dans la Figure 14.

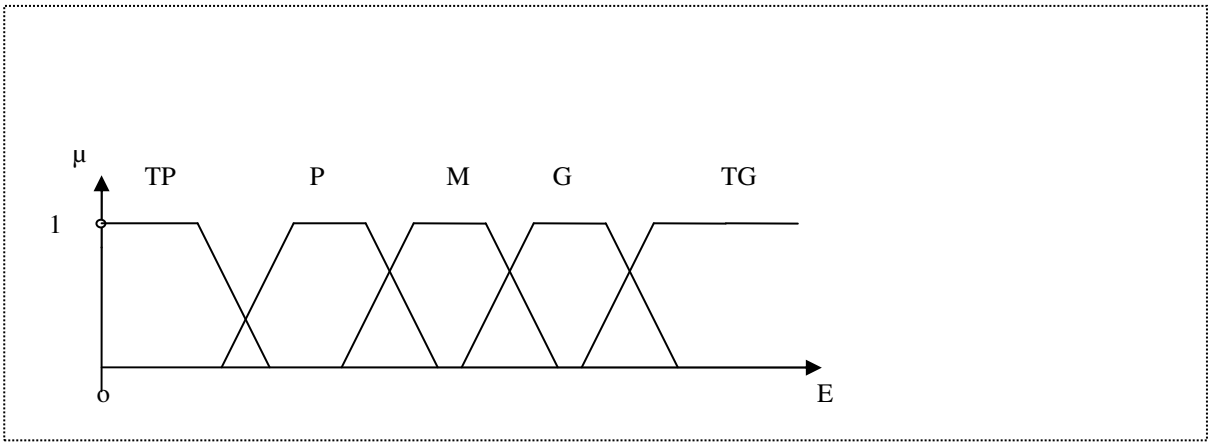

Figure 14 : Un exemple de SEFs trapézoïdaux

Les fonctions d'appartenance et les supports de sous-ensembles flous dans ce mode (cf. la section 2.4.2 qui explique la construction de classe) sont définis ainsi:

Le sous-ensemble Très Petit:
int x ;
x = la borne inférieure de la classe moyen – la borne inférieure de la classe petit ;
borne inférieure du support = 0 ;
borne supérieure du support = la borne inférieure de la classe petit –x/10 ;
borne inférieure du noyau = 0 ;
borne supérieure du noyau= borne inférieure de la classe petit;

Le sous-ensemble Petit:
int x ;
x = borne inférieure de la classe moyen – la borne inférieure de la classe petit;
borne inférieure du support = la borne inférieure de la classe petit - x/10 ;
borne supérieure du support = la borne inférieure la classe moyen + x/10 ;
borne inférieure du noyau = la borne inférieure de la classe petit;
borne supérieure du noyau= la borne inférieure de la classe moyen ;

Le sous-ensemble Moyen
int x ;
x= la borne inférieure de la classe grand – la borne inférieure de la classe moyen;
borne inférieure du support = la borne inférieure de la classe moyen- x/10 ;
borne supérieure du support = la borne inférieure de la classe grand + x/10 ;
borne inférieure du noyau = la borne inférieure de la classe moyen;
borne supérieure du noyau = la borne inférieure de la classe grand;

Le sous-ensemble Grand:
int x ;
x= la borne inférieure de la classe très grand– la borne inférieure de la classe grand;
borne inférieure du support = la borne inférieure de la classe grand - x/10 ;
borne supérieure du support = la borne inférieure de la classe très grand + x/10 ;
borne inférieure du noyau = la borne inférieure de la classe grand;
borne supérieure du noyau = la borne inférieure de classe très grand ;

Le sous-ensemble Très grand:
int x ;
X= la borne inférieure de la classe très grand – la borne inférieure de la classe grand;
borne inférieure du support = la borne inférieure de la classe très grand - x/10 ;
borne supérieure du support= la valeur maximum possible ;
borne inférieure du noyau = la borne inférieure de la classe très grand;
borne supérieure du noyau= la valeur maximum possible ;

3.4.1.2. Les Règles Floues

L'établissement des règles constitue la partie la plus importante du FRBS. En effet, il s'agit de trouver des règles floues "si-alors" qui nous permettront de reconnaître les différentes émotions avec leurs intensités.

La classification (cf. section 2.4.2) des agrégateurs nous aide à définir ces règles. Les résultats de l'étape de classification donnent les résultats espérés. Par exemple, pour l'émotion « *hilare* », on constate que la variable *taux* prend des valeurs faibles par rapport à ce qu'elle vaut dans le cas des autres émotions. Ou encore, la *moyenne générale de quantité de mouvement* est grande par rapport à ce qu'elle vaut dans le cas des autres émotions.

J'ai d'ailleurs essayé d'établir des règles pour les émotions qui n'ont pas une caractéristique similaire pour chaque univers. L'émotion « triste » n'a pas les mêmes valeurs pour chaque univers. On a donc défini des règles différentes pour chaque univers.

Dans le Tableau 7, on représente les résultats des agrégateurs pour l'émotion « *hilare* » dans le prologue.

Agrégateurs	**Hilare**
Taux	Petit
Moyenne générale de QdM	Grand
Moyenne générale de Stabilité	Petit
Moyenne générale d'Indice de Contraction	Petit
Moyenne partielle de QdM	Grand
Moyenne partielle de Stabilité	Petit
Moyenne partielle d'Indice de Contraction	Petit
Ecart-type QoM	Grand
Ecart-type Stabilité	Petit

Tableau 7 : Résultats des agrégateurs pour l'émotion « *hilare* » dans le prologue.

Dans le Tableau 8, on représente les résultats des agrégateurs pour l'émotion « *hilare* » dans l'univers de l'amour.

Agrégateurs	**Hilare**
Taux	Petit
Moyenne générale de QdM	Grand
Moyenne générale de Stabilité	Petit
Moyenne générale d'Indice de Contraction	Petit
Moyenne partielle de QdM	Grand
Moyenne partielle de Stabilité	Petit
Moyenne partielle d'Indice de Contraction	Petit
Ecart-type QoM	Grand
Ecart-type Stabilité	Petit

Tableau 8 : Résultats des agrégateurs pour l'émotion « *hilare* » dans l'univers de l'amour.

On constate que les résultats des agrégateurs sont les mêmes pour chaque univers. En conséquence, on peut définir une règle pour reconnaître l'émotion « *hilare* » :

"SI
Le taux
ET la moyenne générale de stabilité
ET la moyenne générale de l'indice de contraction
ET la moyenne partielle de stabilité
ET la moyenne partielle d'indice de contraction
ET l'écart type de stabilité *sont PETIT*

ET la moyenne générale de quantité de mouvement
ET la moyenne partielle de quantité de mouvement
ET l'écart type de quantité de mouvement *sont GRAND*

ALORS la performeuse est HILARE"

Quant à l'émotion « *endormi* », le Tableau 9, montre les résultats des agrégateurs pour l'univers de l'amour.

Agrégateurs	**Endormi**
Taux	Grand
Moyenne générale de QdM	Petit
Moyenne générale de Stabilité	Très Grand
Moyenne générale d'Indice de Contraction	Grand
Moyenne partielle de QdM	Petit
Moyenne partielle de Stabilité	Très Grand
Moyenne partielle d'Indice de Contraction	Grand
Ecart-type QoM	Petit
Ecart-type Stabilité	Très Grand

Tableau 9 : Résultats des agrégateurs pour l'univers de l'Amour.

Les résultats des agrégateurs pour l'émotion « *endormi* » au prologue sont représentés dans le Tableau 10.

Agrégateurs	**Endormi**
Taux	Grand
Moyenne générale de QdM	Petit
Moyenne générale de Stabilité	Très grand
Moyenne générale d'Indice de Contraction	Grand
Moyenne partielle de QdM	Petit
Moyenne partielle de Stabilité	Très grand
Moyenne partielle d'Indice de Contraction	Grand
Ecart-type QoM	Petit
Ecart-type Stabilité	Très grand

Tableau 10 : Résultats des agrégateurs pour l'univers de l'amour.

Pour l'émotion « *endormi* », on constate que
"SI
le taux
ET la moyenne générale de l'indice de contraction
ET la moyenne partielle de l'indice de contraction sont GRAND,

ET la moyenne générale quantité de mouvement
ET la moyenne partielle quantité de mouvement
ET l'écart type de quantité de mouvement sont PETIT,

ET la moyenne générale,
ET la moyenne partielle
ET l'écart type de stabilité sont TRES GRAND

ALORS la performeuse est ENDORMI"

Le Tableau 11 montre les résultats de l'émotion « *colère* » pour le Prologue et l'univers de l'Amour. J'ai défini deux règles pour la reconnaître. La première est pour le Prologue et la deuxième est pour l'univers de l'Amour.

Agrégateurs	**Colère –Prologue**	**Colère -Amour**
Taux	Grand	Moyen
Moyenne générale de QdM	Moyen	Grand
Moyenne générale de Stabilité	Petit	Petit
Moyenne générale d'Indice de Contraction	Grand	Moyen
Moyenne partielle de QdM	Moyen	Moyen
Moyenne partielle de Stabilité	Petit	Petit
Moyenne partielle d'Indice de Contraction	Grand	Moyen
Ecart-type QoM	Moyen	Grand
Ecart-type Stabilité	Petit	Petit

Tableau 11 : Résultats de l'émotion *Colère* pour l'univers de l'Amour et le Prologue.

La règle pour reconnaître la colore dans le Prologue est :
"SI
le taux
ET la moyenne générale de l'indice de contraction
ET la moyenne partielle de l'indice de contraction sont GRAND,

ET la moyenne générale quantité de mouvement
ET la moyenne partielle quantité de mouvement
ET l'écart type de quantité de mouvement sont MOYEN

ET la moyenne générale de stabilité
ET la moyenne partielle de stabilité
ET l'écart type de stabilité sont PETIT

ALORS la performeuse est COLERE"

La règle pour reconnaître la colore dans l'univers de l'Amour est :
"SI
le taux
ET la moyenne partielle quantité de mouvement
ET la moyenne générale de l'indice de contraction
ET la moyenne partielle de l'indice de contraction sont MOYEN,

ET la moyenne générale quantité de mouvement
ET l'écart type de quantité de mouvement sont GRAND

ET la moyenne générale,
ET la moyenne partielle
ET l'écart type de stabilité sont PETIT

ALORS la performeuse est COLERE"

Pour conclure, notre système contient, pour l'instant, quatre règles floues et 45 sous-ensembles flous qui sont utilisés dans la commande floue. Grâce à ce système flou, on peut reconnaître les intensités des émotions à partir du vecteur des agrégateurs.

Seulement quatre règles ont été établies à l'heure où j'écris ces lignes car on ne dispose pas de davantage d'émotions jouées par la performeuse sur bande vidéo.

Chapitre 4 L'Application

Maintenant que l'analyse détaillée du problème (ainsi que les algorithmes) ont été explicités, il convient de décrire l'application que j'ai programmée pour mettre en place l'Assistant Virtuel. Compte tenu qu'aucun compilateur n'a été imposé mais que la machine à ma disposition était installée avec seulement un OS Windows, et que Visual Studio.Net y était installé, j'ai naturellement décidé de programmer sous Visual Studio.Net, et plus particulièrement sous Visual C++.Net. En effet, l'environnement Visual Studio.Net propose, entre autres, un compilateur C++ qui permet de programmer en langage objet et d'utiliser facilement des interfaces graphiques.

Comme le système décrit précédemment, notre application comporte trois couches :

- **La couche de pré-traitement :** Dans cette couche, on récupère les données dans les fichiers des descripteurs et on calcule le vecteur des agrégateurs. Cette couche est modélisée par la classe `Movie`.
- **La couche du FRBS** : Elle nous permet de créer un FRBS et de trouver le vecteur des intensités des émotions. Celle-ci est aussi modélisée par la classe `CdeFloue`.
- **La couche de visualisation :** Grâce à cette couche, on affiche les résultats obtenus par notre système.

Expliquons brièvement notre application. La première couche traite les fichiers qui ont été générés par le patch EyesWeb et calcule le vecteur des agrégateurs. Elle envoie ensuite ce vecteur à la deuxième couche. La deuxième couche prend donc ce vecteur en entrée et calcule le vecteur des intensités des émotions en utilisant le FRBS. Enfin, la dernière couche pour afficher les résultats.
Passons maintenant aux descriptions des couches.

4.1. La couche de pré-traitement :

La première étape du système est de récupérer les données dans les fichiers qui sont générés par le patch EyesWeb (cf. 2.3.). On a cinq fichiers qui stockent les valeurs des descripteurs :

- Fichier des états des *frames* (pause ou mouvement)
- Fichier de stabilité
- Fichier de quantité de mouvement
- Fichier d'indice de contraction
- Fichier de coordonnées de barycentre

La classe `Movie` sert à modéliser cette couche. Les données des fichiers vont servir indirectement à alimenter les champs de la classe `Movie`. C'est-à-dire que les méthodes de `Movie` vont permettre d'agréger ces données avant le stockage dans les différents champs. Les agrégateurs utilisés sont :

- Moyenne générale
- Moyenne partielle
- Ecart type

Pour définir le vecteur des agrégateurs, la classe `Movie` contient 13 champs qui sont détaillées dans le Tableau 12.

Nom du champ	Définition	Type
`Taux`	rapport de la durée de pause sur la durée totale	float
`Moyenne_generale_de_QdM`	valeur moyenne générale de quantité de mouvement	float
`Moyenne_partielle_de_QdM`	valeur moyenne de quantité de mouvement pour la partie pause	float
`Ecart_type_de_QdM`	valeur de l'écart type de quantité de mouvement pour tout le bloc considéré	float
`Moyenne_generale_de_Stabilite`	valeur moyenne générale de stabilité	float
`Moyenne_partielle_de_Stabilite`	valeur moyenne de stabilité pour la partie pause	float
`Ecart type_de_Stabilite`	valeur de l'écart type de stabilité pour toute la partie	float
`Moyenne_generale_de_IC`	valeur moyenne générale de l'indice de contraction	float
`Moyenne_partielle_de_IC`	valeur moyenne de l'indice de contraction pour la partie de pause	float
`Baricentre_x`	coordonnée x du barycentre de la silhouette	float
`Baricentre_y`	coordonnée y du barycentre de la silhouette	float
`Mouvement_tab`	tableau qui contient les frames où il y a présence de mouvement	int [][]
`Pause_tab`	tableau qui contient les frames où il y a absence de mouvement (pause)	int [][]

Tableau 12 : Champs de la classe `movie`.

Le but de cette couche est d'agréger les données obtenues à partir des fichiers des descripteurs. Les méthodes de la classe `Movie` nous permettent d'agréger ces données. Quatre de ces méthodes sont particulièrement importantes (cf. Tableau 13).

Nom de la méthode	Type de la valeur en retour	Type de la valeur en entrée	Définition

Calculer_Taux	double	int tab_pause[MAX][2],
		// le tableau de durées de pause
		int tab_mouvement[MAX][2]
		//le tableau de durées de mouvement
		char * fichier
		//le nom de fichier
			Cette méthode modifie les tableaux de mouvement et de pause, et renvoie le `Taux`.

Get_Moyenne_de_fichier	double	char * fichier //le nom du fichier	Cette méthode renvoie la valeur moyenne des valeurs contenues dans le fichier « fichier » (des variables) correspondants
Get_Partielle_moyenne_de_fichier	double	int tab[1000][2] //le tableau de durées de pause ou mouvement char * fichier//le nom du fichier	Cette méthode renvoie la valeur moyenne partielle du bloc dont les valeurs sont contenues dans le fichier « fichier »

Get_EcartType	double	char * fichier
		//le nom du fichier
		double moy
		//la valeur moyyenne de ce fichier
			Cette méthode renvoie la valeur d'écart type des valeurs contenues dans le fichier « fichier »

Tableau 13 : Méthodes de la classe `movie`.

Les quatre méthodes que nous venons de voir donnent les valeurs du vecteur des agrégateurs recherché.

La Figure 15 :montre le vecteur des agrégateurs qui a été calculé.

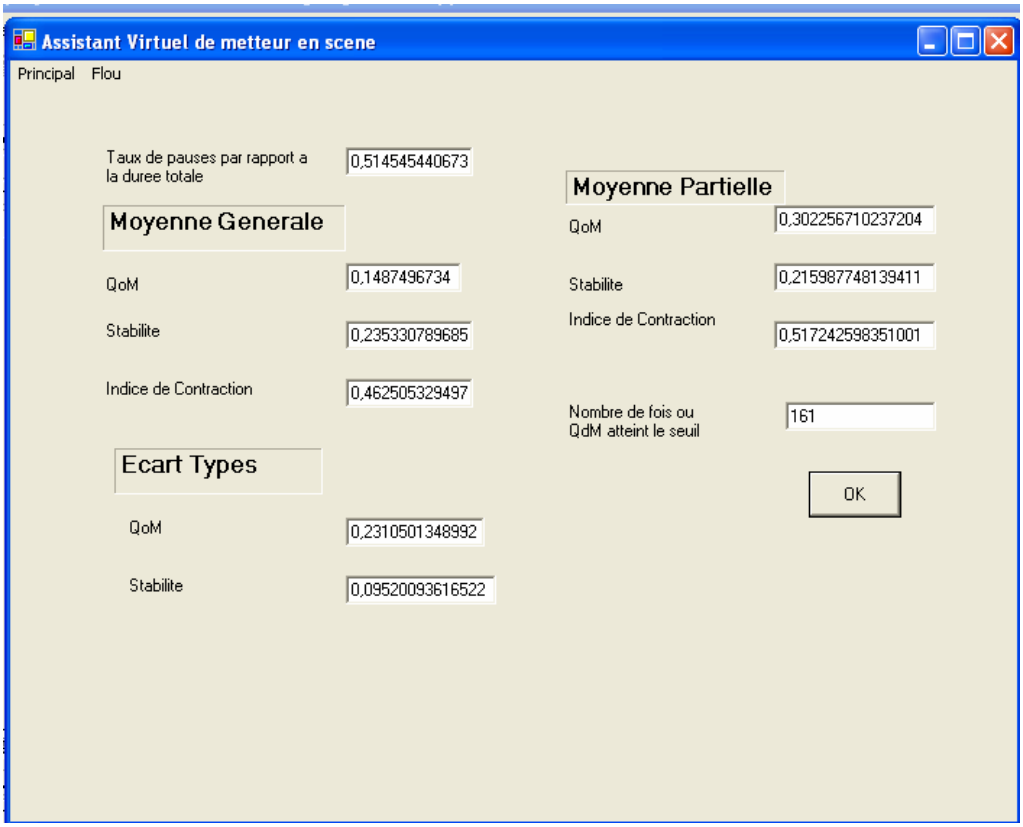

Figure 15 : Vecteur des agrégateurs pour un enregistrement

4.2. La couche du FRBS

La deuxième étape de notre travail est de créer un FRBS (cf 3.4.1.) afin de prendre une décision quant aux émotions jouées par la performeuse.

C'est la deuxième couche modélisée par la classe `CdeFloue` réalise le FRBS. En entrée, cette deuxième couche prend le VdA, calculé grâce à la première couche.

Puis, elle calcule la sortie correspondant à l'entrée (= le VdA) grâce à un mécanisme de *raisonnement approximatif* fondé sur la LF. En utilisant les méthodes et les champs de la classe `CdeFloue`, on fuzzifie l'entrée, on applique les règles pour trouver une sortie floue, puis on défuzzifie la sortie. Le résultat obtenu est un vecteur des intensités des émotions (VIE).

Pour la fuzzification, on a décidé d'utiliser deux algorithmes différents qui débouchent sur deux méthodes différentes afin de donner le choix à l'utilisateur du logiciel. Le premier algorithme correspond à la méthode du singleton qui suppose que la mesure (l'entrée) est exacte (cf. 3.4.1.) et le deuxième à la méthode du SEF triangulaire qui suppose, elle, que la mesure est entachée d'imprécision (cf. 3.4.1.).

Ensuite, pour calculer la sortie du système, on se propose d'utiliser des SEFs et des règles floues, comme on le fait habituellement dans le FRBS, ainsi qu'en commande floue. Pour modéliser les SEFs, on utilise la classe `CdeFloue` avec ses 12 champs visibles dans le Tableau 14.

Nom du Champ	Définition	Type
`Pente1`	Pente à gauche du SEF	`double`
`Pente2`	Pente à droite du SEF	`double`
`Xdep`	Coordonnée en x gauche du support du SEF	`double`
`Xfinal`	Coordonnée en x droite du support du SEF	`double`
`Xmil`	Valeur du noyau du SEF (pour SEF triangulaire)	`double`
`Yfuzz`	Degré d'appartenance associé	`double`
`Ymil`	Hauteur du SEF	`double`
`Ydep`	Début de valeur de degré d'appartenance du SEF	`double`
`Yfinal`	Fin de valeur de degré d'appartenance du SEF	`double`
`noy_dep`	Coordonnée en x gauche du noyau du SEF	`double`
`noy_fin`	Coordonnée en x droite du noyau du SEF	`double`
`Type`	Type de sous-ensemble flou 0 : triangulaire 1 : trapézoïdal	`int`

Tableau 14 : Champs de la classe `CdeFloue`.

L'annexe C détaille les différents constructeurs de la classe `CdeFloue`.

Grâce à ces derniers, on définit nos sous-ensembles flous.

Les règles floues, quant à elles, sont définies par un tableau à deux dimensions :

```
int Tab_regles[4][9] ;
```

La première dimension définit le nombre de règles. La deuxième est pour le nombre de conditions.

Les méthodes de la classe `CdeFloue` vont permettre d'appliquer ces règles pour le calcul de la sortie floue. Cette classe contient 23 méthodes détailles dans le Tableau 15.

Nom de la méthode	Type de la valeur en retour	Type de la valeur en entrée	Définition
Get_xmil	double	sans objet	Méthodes utilisées pour obtenir les valeurs des différents champs privés
Get_ymil	double	sans objet	
Get_ydep	double	sans objet	
Get_yfinal	double	sans objet	
Get_pente1	double	sans objet	
Get_pente2	double	sans objet	
Get_noy_dep	double	sans objet	
Get_noy_fin	double	sans objet	
Get_type	double	sans objet	
Get_xfinal	double	sans objet	
Get_xdep	double	sans objet	
Get_yfuzz	double	sans objet	
Get_xmil	double	sans objet	
Get_ymil	double	sans objet	
Get_ydep	double	sans objet	
Set_xfinal	void	double xf	Utilisées pour définir les valeurs des différents champs privés
Set_noy_dep	void	double sd	
Set_noy_fin	void	double sf	
Set_xmil	void	double xm	
Set_yfuzz	void	double val	
Set_pente1	void	double p1	
Set_pente2	void	double p2	
Set_xdep	void	double xd	
Calculer_pentes	void	sans objet	Utilisé pour calculer les pentes des SEFs

calculer_degre_dappartenance	double	int type
		//type de SEF
		double valeur
		// valeur d'entrée
		int fuz_type
		// type de fuzzification
			Cette fonction utilisée pour calculer le degré d'appartenance

AppliRegles	void	double*val
		//vecteur des agrégateurs
		double*resultats
		//tableau qui contient la degré d'appartenance des entrees
		double * int_emotions
		//vecteur des intensités des émotions
		CdeFloue **f1, CdeFloue **f2, CdeFloue **f3, CdeFloue **f4, CdeFloue **f5, CdeFloue **f6, CdeFloue **f7, CdeFloue **f8, CdeFloue **f9,
		// SEFs utilisés
		int fuz_type
		//type de fuzzification
			Cette fonction utilisée pour calculer le vecteur des intensités des émotions

Tableau 15 : Méthodes de la classe `CdeFloue`.

La dernière méthode évoquée permet de trouver le vecteur des intensités des émotions.
A titre d'exemple la Figure 17 montre la sortie du FRBS (i.e., le VIE) correspondant au vecteur des agrégateurs visible dans la Figure 7.

4.3. La couche de visualisation

Après avoir obtenu le vecteur des intensités des émotions, il convient de l'afficher. Etant donné qu'il n'existe pas d'objet pour afficher des graphiques sous l'environnement Visual Studio.Net, je souhaitais trouver une *API* proposant cela sous cet environnement. Après plusieurs recherches, je me suis tourné vers l'API de Gigasoft qui est gratuite et qui est très bien documentée.

Avec cette API, j'ai créé plusieurs interfaces graphiques qui sont :

- L'interface pour le vecteur des intensités des émotions
- Les interfaces pour afficher les sous-ensembles flous

4.3.1. L'interface du vecteur des intensités des émotions

Le résultat du vecteur des intensités (cf. Figure 17.) des émotions est exprimé au travers de l'interface visible dans la Figure 7.

4.3.2. Les interfaces pour afficher des sous-ensembles flous

L'application que j'ai développée contient aussi les interfaces pour afficher les sous-ensembles flous du système. A titre d'exemple, la Figure 16 montre l'interface affichant les sous-ensembles flous triangulaires des *moyennes générales* et *taux* pour l'univers de l'Amour (pour l'émotion *endormi*).

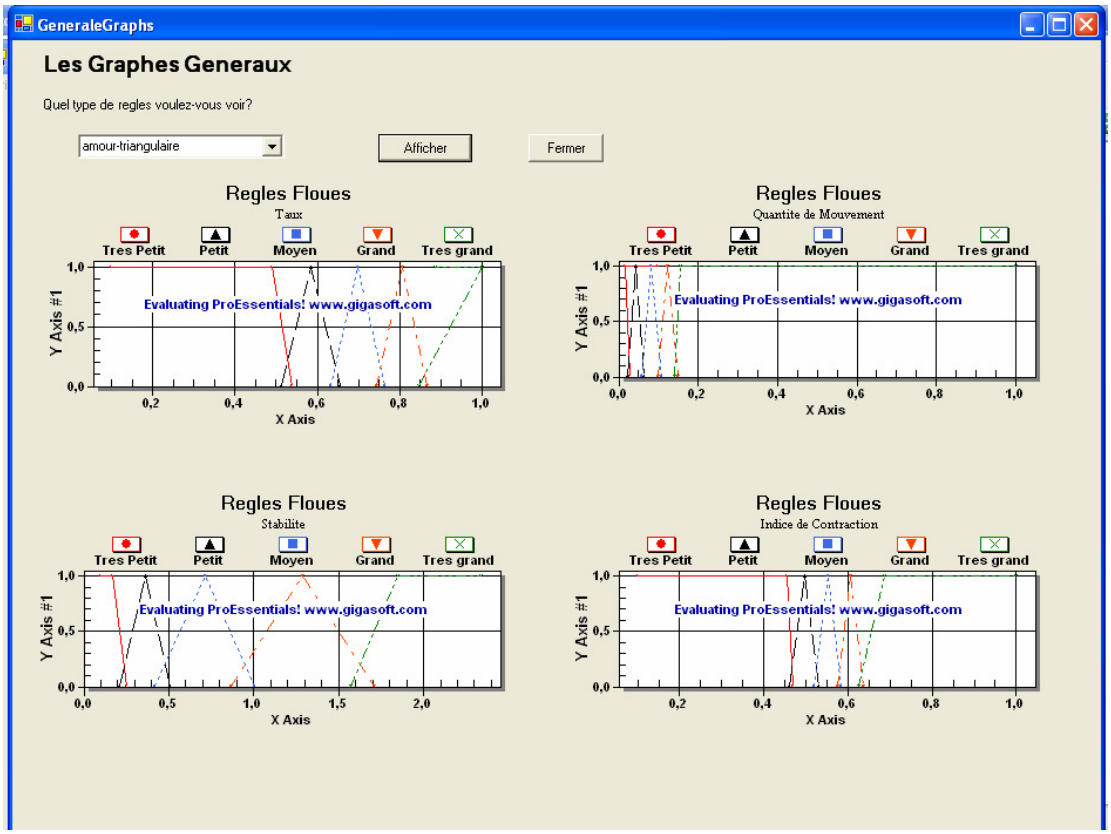

Figure 16 : Quelques exemples de sous-ensembles triangulaires

4.4. Les résultats obtenus

Les enregistrements de la performeuse d'*Alma Sola* ont donc permis de tester notre application. Comme on peut le constater dans les captures écran qui suivent, les résultats obtenus sont bons et cohérents. Les intensités des émotions (*endormi*, *colère* et *hilare*) pour le Prologue et l'univers de l'Amour exprimées à l'aide des sous-ensembles triangulaires et sous-ensembles trapézoïdaux sont montrées dans les figures 17 à 20.

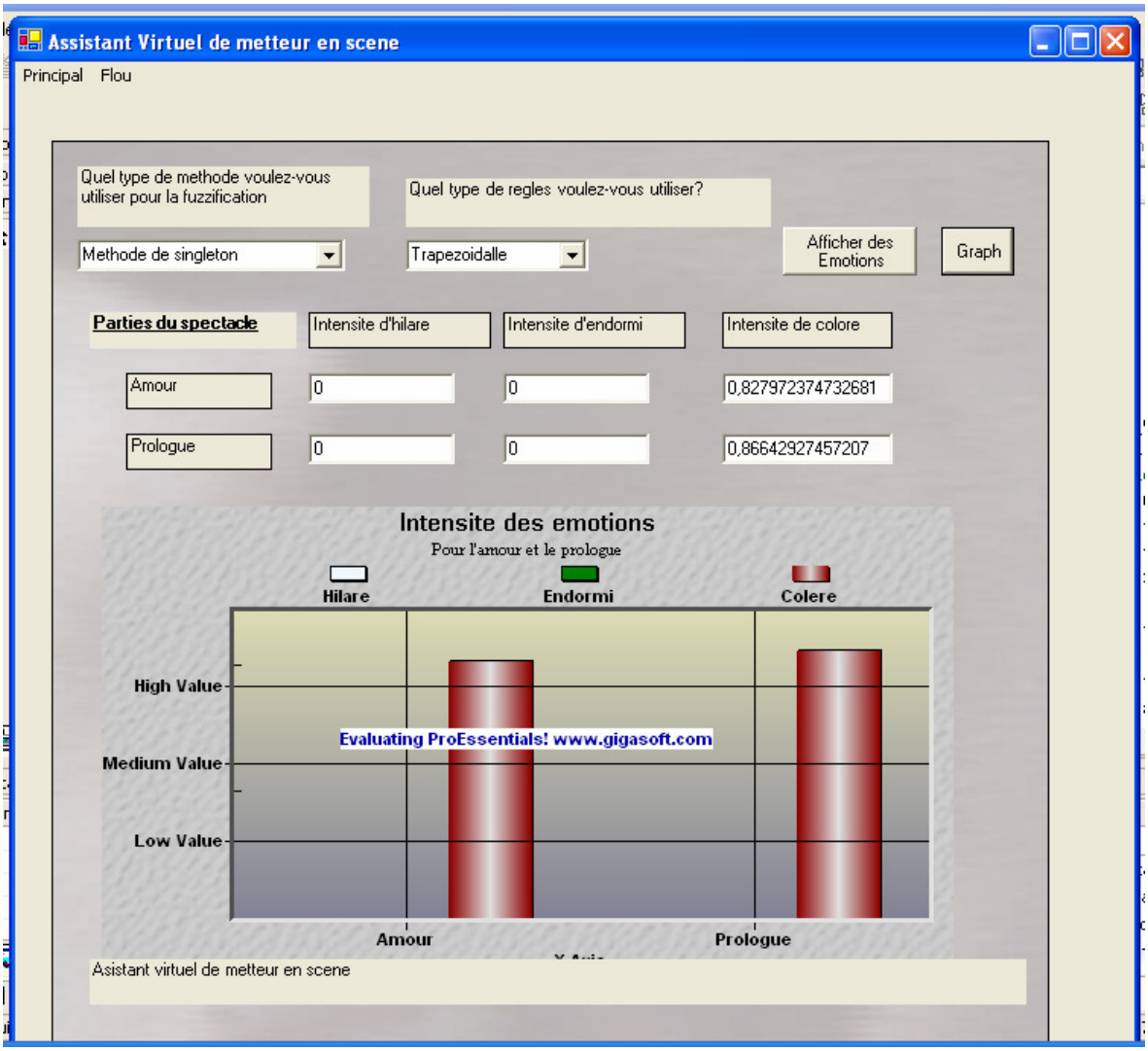

Figure 17 : Résultats de l'émotion *colère* avec la méthode du singleton et les sous-ensembles flous trapézoïdaux

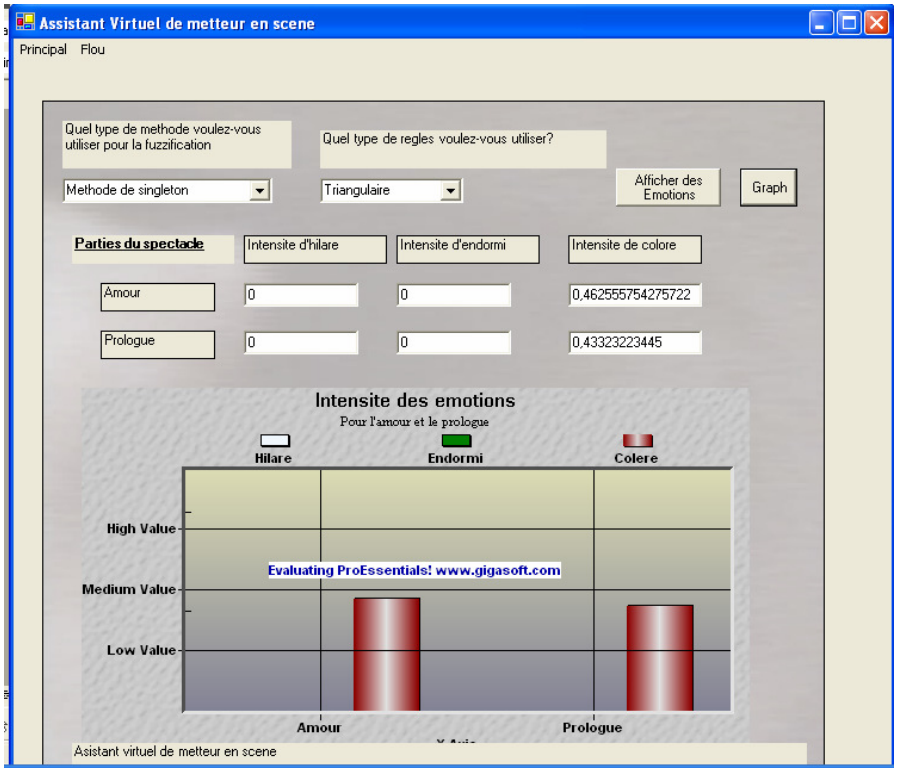

Figure 18 : Résultats de l'émotion *colère* avec la méthode du singleton et les sous-ensembles flous triangulaires

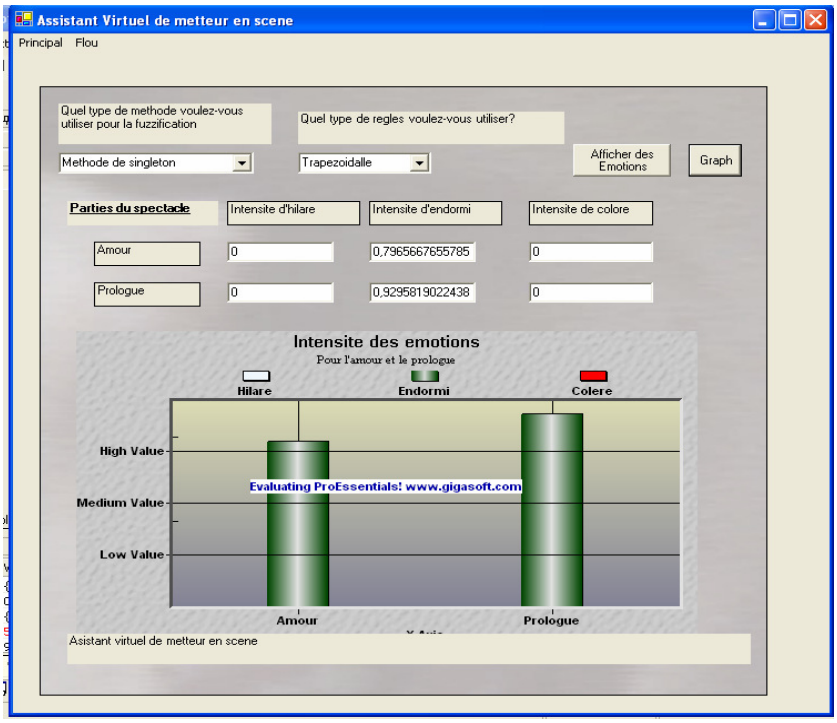

Figure 19 : Résultats de l'émotion *endormi* avec la méthode du singleton et les sous-ensembles flous trapézoïdaux

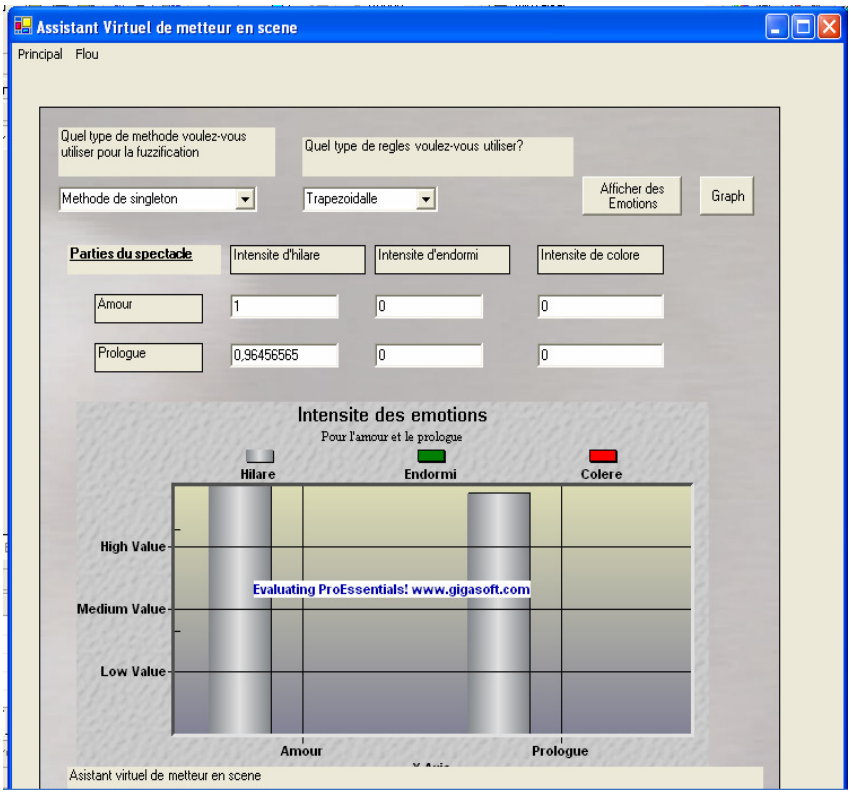

Figure 20 : Résultats de l'émotion *hilare* avec la méthode du singleton et les sous-ensembles flous trapézoïdaux

Avant de faire les tests, j'avais supposé que les sous-ensembles trapézoïdaux donnaient des résultats plus cohérents, ce qui est effectivement le cas. Par exemple, dans la figure 18, l'intensité de l'émotion *colère* est 0,46 et 0,43 en utilisant les sous-ensembles triangulaires mais si on utilise les sous-ensembles trapézoïdaux, on obtient 0,82 et 0,86 pour le même vecteur des agrégateurs. En utilisant les sous-ensembles trapézoïdaux, on obtient les intensités 0,79 et 0,92 pour l'émotion *endormi* et 1 et 0,96 pour l'émotion *hilare*. Ces résultats nous montrent qu'en utilisant nos règles et les sous-ensembles trapézoïdaux, on peut mieux détecter les émotions qu'on cherche pour le Prologue et l'univers de l'Amour.

Conclusion

Le système que j'ai imaginé et implémenté permet de donner des indications quant au jeu d'un performer, lors d'une représentation, à partir d'une analyse de la gestuelle.

Dans un premier temps, on a filmé une des performeuses de l'opéra virtuel *Alma Sola*, ce qui nous a permis d'obtenir deux fichiers de données brutes : un fichier correspondant aux images enregistrées, l'autre à la capture de la voix. Mon travail a porté principalement sur le traitement du fichier « images » (vidéo muette). Dans un deuxième temps, j'ai utilisé le logiciel EyesWeb pour créer un *patch* afin d'extraire de la vidéo des paramètres intéressants et discriminants sur les mouvements et la gestuelle de la performeuse. Cette étude a été réalisée en collaboration avec une metteur en scène (professionnelle). Elle a nécessité de nombreux essais sur divers fichiers vidéo trouvés, le plus souvent, sur Internet. Une fois ces descripteurs de mouvement et leurs agrégations associées définis, j'ai, dans un troisième temps, imaginé un système permettant de déduire, raisonner approximativement. J'ai choisi d'élaborer un système à base de règles floues dans lequel les descripteurs – ou plutôt leurs agrégations – sont partitionnées en sous-ensembles flous. Il a fallu ensuite établir des règles, i.e. des conditions que les agrégations doivent réaliser pour que telle ou telle émotion soit retenue.

Le système que j'ai programmé fonctionne pour trois émotions : *hilare, colère* et *endormi*. Les tests effectués ont permis de vérifier que notre système fonctionnait très bien pour ces trois émotions. Seules trois émotions ont été prises en compte car les enregistrements de la performeuse effectués jusqu'à présent ne comportent que ces émotions. Notre système est également flexible de sorte qu'on puisse l'utiliser dans les autres spectacles. Il suffit de changer les règles floues et sous-ensembles flous, ce qui est facilement faisable grâce à l'interface.

Ce projet a été prévu sur le long terme et quelques évolutions sont possibles pour améliorer le logiciel. On peut tout d'abord augmenter le nombre d'émotions à caractériser et surtout multiplier le nombre d'enregistrements pour une émotion donnée. Cela nous permettra d'avoir beaucoup plus de données sur lesquelles s'appuyer et par conséquent d'affiner les règles floues.

Par ailleurs, le *poids* donné, pour l'instant, à chaque descripteur est égal à un. En effet, la quantité de mouvement, la stabilité ou l'indice de contraction ont la même importance dans la caractérisation d'une émotion. On pourrait décider de pondérer davantage l'un ou l'autre de ces descripteurs.

En outre, il serait intéressant de tester notre logiciel sur d'autres spectacles qu'*Alma Sola*. Les spectacles de danse, par exemple, s'y prêteraient certainement très bien.

On peut aussi créer une interface qui demanderait au metteur en scène de définir l'émotion recherchée. Suivant les résultats de l'enregistrement, le logiciel pourrait dire ce que doit faire la chanteuse pour arriver à cette émotion (par exemple, la chanteuse doit *bouger beaucoup plus* et avoir des *mouvements saccadés* pour jouer de façon *angoissée*).
Enfin, l'idéal serait, à un méta-niveau, que le logiciel puisse apprendre à reconnaître une nouvelle émotion, donc sans les tests, l'étude, ni l'expertise de l'humain. Mais cette question est probablement loin d'être résolue.

Bibliographie

[Affective Computing 1997] Site Web du Affective Computing Research Group dirigé par Rosalind Picard, http://vismod.www.media.mit.edu/vismod/demos/affect/.

[Bonardi et Rousseaux, 2004] A. Bonardi & F. Rousseaux, New Approaches of Theatre and Opera Directly Inspired by Interactive Data-Mining, Conférence Internationale Sound & Music Computing (SMC'04), pages 1-4, Paris, 20-22 octobre 2004.

[Boone et Cunningham, 2001] R.T. Boone & J.G. Cunningham, (2001). Children's expression of emotional meaning in music through expressive body movement. Journal of Nonverbal Behavior, 25(1).

[Camurri *et al.,* 1999] A. Camurri, M. Ricchetti & R. Trocca. EyesWeb - toward gesture and affect recognition in dance/music interactive systems, in Proc. IEEE Multimedia Systems '99, Italy, 1999.

[Camurri *et al*., 2001] A.Camurri, G.De Poli, M.Leman & G.Volpe A Multi-layered Conceptual Framework for Expressive Gesture Applications, Proc. Intl MOSART Workshop, Barcelona, November 2001.

[Camurri *et al*., 2003] A. Camurri, I. Lagerlöf & G. Volpe (2003). Recognizing Emotion from Dance Movement: Comparison of Spectator Recognition and Automated Techniques. International Journal of Human-Computer Studies, 59(1-2), 213-225.

[Camurri *et al*., 2004] A. Camurri, B. Mazzarino & G. Volpe (2004). Analysis of Expressive Gesture: The EyesWeb Expressive Gesture Processing Library. In A. Camurri, G. Volpe (Eds.), Gesture-based Communication in Human-Computer Interaction, LNAI 2915, Springer Verlag.

[Camurri *et al.,* 2005] E. Lindström, A. Camurri, A. Friberg, G. Volpe & M.-L. Rinman, Affect, attitude and evaluation of multi-sensory performances. Journal of New Music Research,9 (in press).

[Dahl et Friberg, 2004] S Dahl & A. Friberg (2004). Expressiveness of musician's body movements in performances on marimba. In A. Camurri, G. Volpe (eds.) Gesture-based Communication in Human-Computer Interaction, LNAI 2915 (pp.479-486), Berlin, Springer Verlag.

[Ekman et Friesen, 1975] P. Ekman, & W. V. Friesen (1975). Unmasking the Face: A Guide To Recognizing Emotions From Facial Clues. Prentice-Hall, Englewood Cliffs, New Jersey.

[Ekman et Friesen, 1978] P. Ekman, & W. V. Friesen, (1978). Facial Action Coding System. Consulting Psychologists Press, Palo Alto, CA.

[Friberg et Sundberg, 1999], A. Friberg & J. Sundberg (1999). Does music performance allude to locomotion? A model of final ritardandi derived from measurements of stopping runners. Journal of the Acoustical Society of America, 105(3), 1469-1484

[Friberg, 2004] A. Friberg. A fuzzy analyzer of emotional expression in music performance and body motion, in Proc. of Music and Music Science, Stockholm, 2004.

[Juslin *et al*., 2002] P. N. Juslin, A. Friberg & R. Bresin, (2002). Toward a computational model of expression in performance: The GERM model. Musicae Scientiae special issue 2001-2002, 63-122.

[Manoury & Battier, 1987] P. Manoury & M. Battier. Les partitions virtuelles, Paris, documentation Ircam, 1987.

[Martínez *et al*., 2005]L. Martínez, J. Liu, Da Ruan & J.B. Yang, Fuzzy Tools to Deal with Heterogeneous Information in Engineering Evaluation Processes. Information Sciences. 2005. In Press.

[Megaproject, 2001] A. Camurri, G. De Poli, A. Friberg, L. Leman & G. Volpe, (in press). The MEGA project: analysis and synthesis of multisensory expressive gesture in performing art applications. Journal of New Music Research.

[Ng *et al*., 2004] Ng., K.C., A Camurri,& N. Bernardini, (eds.), Proc. of the Symposium on Gesture Interfaces for Multimedia Systems, AISB 2004 Convention: Motion, Emotion and Cognition, The Society for the Study of Artificial Intelligence and the Simulation of Behaviour (SSAISB), ISBN: 1 902956 37 4, 2004.

[Ng, 2003] Ng, K.C., Expressive Gesture, in Proc. of Electronic Imaging and the Visual Arts London Symposium 2003, UCL, London, UK, 2003.

[Seif El-Nasr *et al*., 2000] Seif El-Nasr, M., Yen, J. & Iorger, T. R. (2000). FLAME – Fuzzy logic adaptive mode of emotions. Autonomous Agents and Multi-Agent Systems, 3, 219-257.

[The Duy Bui, 2004] Creating emotions and facial expressions for embodied agents. Phd thesis, University of Twente

[Volpe *et al.,* 2001]A.Camurri, B.Mazzarino, R.Trocca & G.Volpe Real-Time Analysis of Expressive Cues in Human Movement, in Proc. Cast01 - Conference on artistic, cultural and scientific aspects of experimental media spaces, pp. 63-68, Bonn, Germany, September 2001.

[Volpe *et al.,* 2004]A. Camurri, B. Mazzarino, M. Ricchetti, R. Timmers & G. Volpe Multimodal analysis of expressive gesture in music and dance performances, in A. Camurri, G. Volpe (Eds.), "Gesture-based Communication in Human-Computer Interaction", LNAI 2915, Springer Verlag, 2004.

[Zadeh, 1965] L. A. Zadeh (1965). Fuzzy sets. Information and Control, 8: pp. 338-353.

[Zeppenfeld, 2004] C. Zeppenfeld. L'acteur face aux technologies, Master dissertation, 2004.

Lexique

API : Application Programming Interface, en français : interface de programmation

FRBS: Fuzzy Rule-Based System, en français : système à base de règles floues

LF: Logique Floue

SEF : Sous-Ensemble Flou

VdA: Vecteur des Agrégateurs

VIE: Vecteur des Intensités des Emotions

Annexe A : Le projet d'opéra/forme ouverte

Nota : ce document a été écrit par A. Bonardi.

L'opéra est un art qui joue traditionnellement de la continuité et des paroxysmes du récit dramatique : la notion de progression musicale et dramatique y possède un rôle fondamental.
Sortant volontairement de cette conception traditionnelle, *Alma Sola* est un opéra interactif reposant sur une forme ouverte.
Les machines et programmes informatiques, par leur puissance calculatoire permettent de combiner à l'infini les éléments textuels, musicaux et visuels, et de forger en permanence un nouvel opéra.
L'espace des possibles est rendu explicite, il se confond avec celui du plateau : la scénographie met en scène ces choix, ces parcours à constituer. Le personnage, Faust(e) féminin, arpente cet espace qu'elle construit dans sa forme musicale. En retour, des voix et des images lui manifestent la présence d'une altérité qu'elle cherche à maîtriser pour franchir les limites de la connaissance et du pouvoir humain et faire l'expérience de l'ultime liberté. Pour Faust(e), le savoir permet de soumettre la nature par la science, de la modifier et de la plier ainsi à ses propres fins. L'informatique ouvre au personnage un espace qu'il peut investir dans la forme ouverte. La forme ouverte devient un jeu où il se perd et se retrouve ; elle lui offre la liberté des possibles, l'illusion de saisir le monde en un schéma synoptique et celle d'abolir le temps.
Plus de 35 ans après la création de l'opéra « ouvert » *Votre Faust* d'Henri Pousseur et Michel Butor, il apparaît que les nouvelles technologies peuvent donner un nouvel élan et un nouveau sens à la forme ouverte sur scène. Il s'agit de faire de chaque représentation d'*Alma Sola* une expérience unique, un parcours singulier dans les espaces de la sensation, du désir et de la volonté. La forme ouverte introduit les notions de « zapping », de bifurcation, de dédoublement, qui trouvent un support idéal dans les plateformes logicielles interactives comme Max/MSP/Jitter ou Arkaos.

L'espace du livret

La thématique retenue est celle d'un Faust(e) féminin explorant des univers thématiques : l'Amour, le Pouvoir, le Plaisir, l'Opulence, la Connaissance. Ces univers sont divisés en blocs selon la structure suivante :

Univers	**Prologue**	**Amour**	**Pouvoir**	**Plaisir**	**Opulence**	**Connaissance**
Blocs des Univers	*Prologue1 : *entrée rythmique* *Prologue2 : *entrée mélodique* *Prologue3 : *entrée harmonique* *Pacte	*Amour1 : *amour blanc 1* *Amour2 : *amour noir 2* *Amour3 : *amour blanc 3* *Amour4 : *amour noir 4*	*Pouvoir1 : *puissance/obéissance/ assistance* *Pouvoir2 : *labeur/soumission/ anéantissement*	*Plaisir1 : *Geisha/tabac* *Plaisir2 : *âme/chair* *Plaisir3 : *joint/sexe* *Plaisir4 : *folie/vin* *Plaisir5 : *poignard/près de moi*	*Opulence1 : *euros/picasso/casino* *Opulence2 : *hôtels/voyages/cities* *Opulence3 : *valets/mets/vins* *Opulence4 : *carte gold/yachts/bijoux* *Opulence libre	*Connaissance1 : *onomatopées* *Connaissance2 : *langage* *Connaissance3 : *hyper-langage*

Selon le principe retenu de non-linéarité, ces 23 blocs peuvent être articulés dans n'importe quel ordre. L'ordinateur propose des continuations possibles à chaque bloc et des enchaînements grâce aux modules d'intelligence artificielle implémentés. Il en résulte un vaste nombre d'opéras possibles lequel s'écrit avec plus de 40 chiffres.

Une forme ouverte incarnée dans la scénographie

La forme ouverte est incarnée dans le dispositif scénographique. Chaque zone de l'espace scénique correspond à un univers d'expérimentation faustienne. En pénétrant sur une zone donnée, Faust(e) fait basculer l'opéra dans l'univers associé. Des tapis sensibles et des dalles de pression détectent la présence du personnage qu'ils indiquent aux deux ordinateurs chargés de gérer son et image et de faire basculer les lumières. Ce dispositif suit également les sorties d'univers qui peuvent se produire en plusieurs points de chaque module. La chanteuse laisse ainsi une empreinte singulière à chaque représentation.

La forme ouverte ainsi associée intimement à l'espace devient complètement lisible pour le spectateur : à chaque zone de l'espace est associé un type de texte, d'image et de lumière. Lorsque le personnage quitte une zone, l'ordinateur mémorise la situation dans laquelle il se trouvait. Lorsqu'il revient sur cette même zone après un parcours, l'ordinateur réactive la situation mémorisée, renvoyant les stimuli sonores et visuels déjà actifs au passage précédent. Tout un travail sur la mémoire sensible et sur son rappel est mené ici.

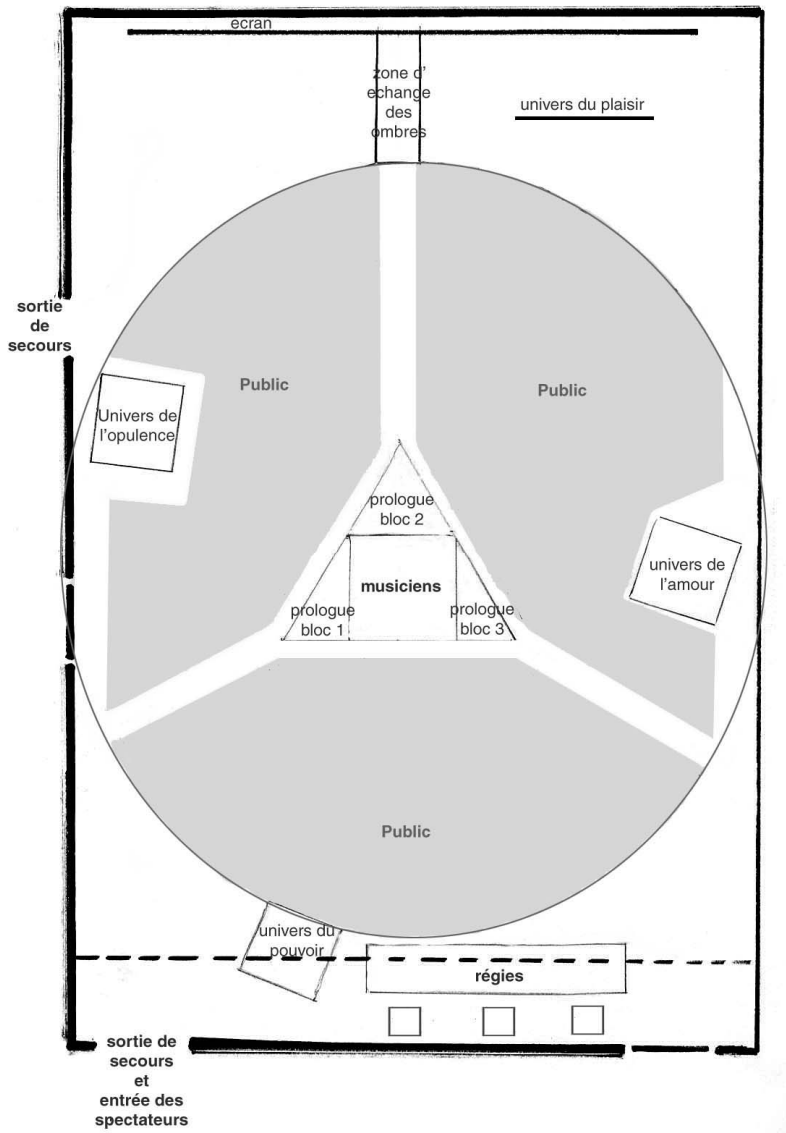

Figure A.1 :.Exemple de dispositif scénographique pour le Cube – schéma : Candice Moïse

Conçue dans une optique labyrinthique, la scénographie s'inscrit dans une spatialité en trois dimensions.

Eclaté en univers, le dispositif scénographique est modulable et peut s'adapter à des espaces de tailles variées avec différentes façons de placer le public (rapport frontal ou éclaté).

Une forme ouverte assistée par ordinateur

L'interprète ne joue pas seule à ce jeu de forme ouverte, l'ordinateur est son partenaire, il l'assiste dans ses choix de construction de l'opéra. Nous souhaitons que l'ordinateur propose à Faust(e) une continuation possible en sortie de chaque univers. La proposition de l'ordinateur se manifestera sous forme lumineuse, avec un éclairage au sol intégré à la scénographie et guidant l'interprète. Ce dernier a le choix de suivre ou non chacune des propositions de la machine, qui au passage apprend ce qu'est une forme ouverte intéressante. Mais comment « calculer » une

« bonne » forme ouverte ? Nous avons fait appel à une technique de l'Intelligence Artificielle parfaitement adaptée à l'apprentissage et à la génération des formes ouvertes : il s'agit des modèles de Markov cachés (Hidden Markov Models, HMM en abrégé) utilisés dans la modélisation des enchaînements et des séquences. Nous avons mis au point, sous la forme d'un nouvel objet Max/MSP, un module informatique implémentant ces modèles de Markov adaptés à la génération de formes ouvertes.

C'est à notre connaissance la première utilisation de tels formalismes dans un opéra.

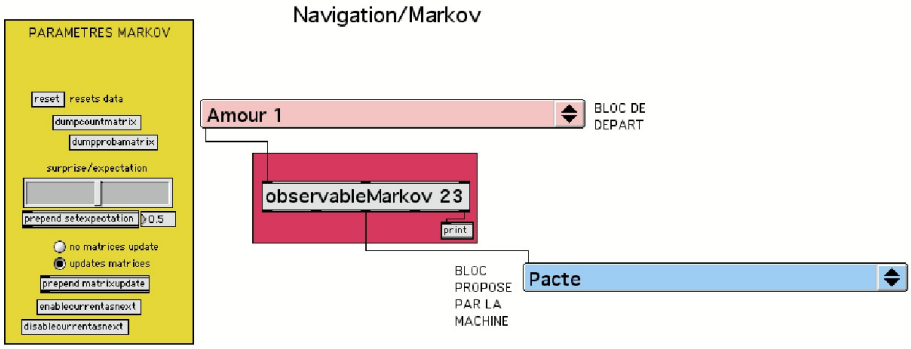

Figure A.2 : Exemple de patch Max/MSP de proposition de bloc d'opéra (en bleu) en fonction du bloc en cours (en rouge) . Conception : Alain Bonardi

Conception musicale

L'opéra fraye sa voie entre écriture traditionnelle et calcul musical sur ordinateur. Fondée sur les catégories de coupure et de commentaire, la conception musicale joue sur les registres opposés de la prescription et de l'imprévu. S'opposent la ductilité du matériau confié aux instrumentistes et la puissance combinatoire du matériau informatique. Les blocs d'un même univers sont composés et programmés en variations les uns des autres.

L'œuvre est écrite pour deux chanteuses (Faust(e) et son Ombre), guitare, cor et dispositif d'informatique musicale temps réel. La représentation commence par le *Lamento d'Arianna* de Monteverdi, conçu pour une chanteuse solo (déjà « âme seule », « alma sola »), et que nous adaptons à l'effectif d'*Alma Sola*. L'écriture de Monteverdi, qui laisse libres de nombreux paramètres de réalisation (effectif, instruments, tempi) propose un certain nombre de degrés d'ouverture s'accordant bien à l'esprit d'*Alma Sola*. Ecriture baroque et écriture contemporaine de l'opéra se croisent, se renvoient l'une à l'autre dans les deux sens.

Temps réel et générativité

L'espace de l'écran

La scène est conçue comme une métaphore de l'univers mental du personnage de Faust(e). Elle est fermée par un écran où se projettent des images filmées et d'autres traitées en temps réel, projection des rêves, des phantasmes, des aspirations de cet être en recherche d'absolu.

L'ordinateur chargé des images gère en temps réel un ensemble de vidéos prédéfinies, organisées comme pour la musique selon le principe de variations : les différents blocs d'un même univers ont une unité qui permet au spectateur de saisir instantanément le positionnement dans la forme

ouverte. Mais par ailleurs, chaque bloc a son originalité. Les transformations et traitements appliqués à ces vidéos sont modulés par des informations provenant de la musique. En effet, « l'ordinateur-image » et « l'ordinateur-son » échangent des informations en réseau. Par exemple, dans le bloc 1 du Prologue, « l'ordinateur-son » envoie à son homologue chargé de l'image deux informations : une concernant le « beat » ou pulsation, l'autre concernant l'attaque de la voix.

A ces vidéos prédéfinies mais transformées en temps réel s'ajoute un flux vidéo live provenant d'une caméra manipulée par l'Ombre de Faust(e). Il s'agit d'une deuxième chanteuse, qui incarne la partie obscure de Faust(e), le recul par rapport à l'immédiateté de sa décision, et qui, à l'instar des manipulateurs de marionnettes du Bunraku japonais, crée et façonne l'image de Faust(e). Ses prises vidéo se situent par rapport à l'image directe de Faust(e), comme son chant par rapport au personnage principal, en contrepoint ou en contre chant. Faust(e), face à une image de lui-même, est une âme seule en recherche d'absolu.

Figure A.3 : Exemples d'images (Univers de l'opulence, du plaisir, du pouvoir, du Prologue) créées par Julien Piedpremier

L'espace sonore

L'espace sonore dans lequel est immergé le spectateur est élaboré à partir des voix (2 sopranos) et des instruments (guitare et cor) et des dispositifs électroniques (notamment grâce au logiciel Max/MSP). Ces derniers sont soit des sons prédéfinis diffusés et/ou transformés en temps réel,

soit des sons obtenus à partir des voix et des instruments (en utilisant des transformations comme l'harmonizer, le frequency shifter, le delay, etc). La notion de mémoire a été plus particulièrement travaillée : à certains endroits, des échantillons de voix, de guitare ou de cor, sont prélevés, puis traités et réinjectés dans un autre univers.

L'ensemble de ces sons, acoustiques et électroniques, directs et diffusés est intégré dans un espace sonore prévu en quadriphonie, mais qui est modulable en fonction du lieu (de la stéréophonie à l'octophonie).

Annexe B : Les algorithmes de construction des classes

Première algorithme :

int intervalle ;
int x ;
intervalle = la valeur maximum possible – la valeur minimum possible ;
x=intervalle/5 ;
Pour chaque donnée du vecteur des agrégateurs
    Si la donnée est plus petite que (la valeur minimum possible +x)
        Alors elle est très petite
    Sinon si la donnée est entre (la valeur minimum possible +x) et (la valeur minimum possible +2x)
        Alors elle est petite
    Sinon si la donnée est entre (la valeur minimum possible +x) et (la valeur minimum possible +x))
        Alors elle est moyenne
    Sinon si la donnée est entre (la valeur minimum possible +x) et (la valeur minimum possible +x)
        Alors elle est grande
    Sinon elle est plus grande
Fin pour

Deuxième algorithme:

int Intervalle ;
int x ;
X= intervalle/3 ;
intervalle = valeur maximum – valeur minimum
Pour chaque donnée du vecteur des agrégateurs
    Si la donnée est plus petite que la valeur minimum
        Alors elle est très petite
    Sinon si la donnée est entre la valeur minimum et (la valeur minimum+ x)
        Alors elle est petite
    Sinon si la donnée est entre (la valeur minimum+x) et (la valeur minimum +2x)
        Alors elle est moyenne
    Sinon si la donnée est entre (la valeur minimum+2x) et (la valeur maximum)
        Alors elle est grande
    Sinon elle est plus grande
Fin pour

Troisième Algorithme :

```
int intervalle1 ;
int intervalle2 ;
int x ;
int y ;
intervalle1= la valeur moyenne – la valeur minimum ;
intervalle2 = la valeur maximum – la valeur moyenne ;
X = Intervalle1/3 ;
Y = intervalle2/3 ;
Pour chaque donnée du vecteur des agrégateurs
    Si la donnée est plus petite que la valeur minimum
        Alors elle est très petite
    Sinon si la donnée est entre la valeur minimum et (la valeur minimum+ x)
        Alors elle est petite
    Sinon si la donnée est entre (la valeur moyenne-x) et (la valeur moyenne+y)
        Alors elle est moyenne
    Sinon si la donnée est entre (la valeur moyenne+y) et (la valeur maximum)
        Alors elle est grande
    Sinon elle est plus grande
Fin pour
```

Annexe C : Constructeurs de la classe `cdeFloue`

On définit les sous ensembles flous à partir des constructeurs. Il y a deux types de constructeur dans cette classe. Le premier qui permet de définir les sous-ensembles flous triangulaires et est de la forme suivante:

```
CdeFloue (double extremite1,double extremite2, double pic,
double ydepart, double ymilieu, double yfin);
```

Où

- **Extremite1:** borne inférieure du support
- **Extremite2:** borne supérieure du support
- **Pic**: noyau (comme le sous ensemble flou est triangulaire, c'est réduit à un point)
- **Ydepart**: valeur de début de la fonction d'appartenance
- **Ymilieu**:valeur de la hauteur
- **Yfin:** valeur de fin de la fonction d'appartenance

Le deuxième constructeur définit les sous-ensembles flous trapézoïdaux:

```
CdeFloue(double extremite1,double extremite2,double noy_dep,
double noy_fin, double ydepart,double ymilieu, double yfin);
```

Où

- **Extremite1:** borne inférieure du support
- **Extremite2:** borne supérieure du support
- **noy_dep:** borne inférieure du noyau
- **noy_fin:**borne supérieure du noyau
- **Ydepart:** valeur de début de la fonction d'appartenance
- **Ymilieu:**valeur de la hauteur
- **Yfin:** valeur final de fonction d'appartenance