Projet DYCI2,
WP3 Dynamiques d'interaction improvisée,
SP3.3 Mémoire, connaissance, et contrôle des agents créatifs

Rapport de livrable :

L3.3.2 Mémoire, connaissance, et contrôle des agents créatifs II

Livrable	Date	Contributeurs	Rédacteurs	Contenu
L3.3.2	Juillet 2018	J. Nika (Univ. la Rochelle / Ircam-STMS), D. Schwarz (Ircam-STMS)	J. Nika (Univ. la Rochelle / Ircam-STMS)	Maquette Logicielle Agent DYCI2 0.2, Rapport scientifique

Résumé

Ce document présente la seconde technologie d'agents créatifs DYCI2 embarquant une mémoire musicale et pouvant être guidés selon un continuum de paradigmes de contrôles allant de la planification à la réactivité. Elle intègre les résultats majeurs du WP3 et est matérialisée par la seconde version d'architecture intégrative « Agent DYCI2 0.2 ». Cette maquette logicielle consiste en une librairie d'agents et de modèles pilotés par des requêtes dynamiques ou « scénarios à court terme », ainsi que par l'écoute réactive d'un flux audio. Cette architecture est une évolution du livrable L3.3.1 et sera la base sur laquelle sera construite la maquette finale du WP4.

Répertoire de développement logiciel collaboratif

https://forge.ircam.fr/p/DYCI2_library/

Adresse du livrable logiciel

DYCI2_WP3.3_L.3.3.2.zip
sur
https://forge.ircam.fr/p/Dyci2/

Ce document présente la seconde technologie d'agents créatifs embarquant une mémoire musicale, et pouvant être guidés selon un continuum de paradigmes de contrôles allant de la planification à la réactivité. Elle intègre les résultats majeurs du WP3 et est matérialisée par la seconde version d'architecture intégrative « Agent DYCI2 0.2 ». Cette maquette logicielle consiste en une librairie d'agents et de modèles pilotés par des requêtes dynamiques ou « scénarios à court terme », ainsi que par l'écoute réactive d'un flux audio. Cette architecture est une évolution du livrable L3.3.1 et sera la base sur laquelle sera construite la maquette finale du WP4.
La finalité est double : proposer une librairie de processus génératifs pouvant être interfacés avec d'autres environnements (par exemple via le protocole OSC) ainsi qu'une librairie d'agents musicaux (pouvant générer des séquences audio ou midi, de manière autonome ou guidée, en faisant appel à la librairie de processus génératifs). La « librairie DYCI2 » est donc constituée de deux librairies : une librairie Python implémentant les processus génératifs (modèles de mémoire et traitement des requêtes) et architectures réactives dans le domaine symbolique, et une librairie d'objets Max pour la performance proposant des interfaces pour le choix de mémoires musicales et de paramètres pour leur apprentissage, l'envoi de requêtes pour guider la génération, et des modules de restitution pour jouer les séquences générées dynamiquement.

Continuum des stratégies de génération entre planification et écoute réactive

Cette section résume les paradigmes de génération par navigation dans une mémoire musicale annotée (cf L.3.1.1 et L3.3.1) qu'un agent DYCI2 peut adopter successivement au sein d'une même performance : de la génération libre à la génération structurée par un scénario temporel en passant par la réaction aux évènements extérieurs analysés par un module d'écoute.

Génération libre
Une fois le modèle de mémoire construit, une requête « free » entraîne la génération d'une séquence musicale inédite suivant la logique temporelle interne du matériau appris, sans autre contrainte de structure ou d'écoute réactive. Ce mode de génération est contrôlé par des paramètres bas-niveaux modifiables en temps réel et modifiant les comportements de l'agent en influant par exemple sur la fidélité à la séquence d'origine ou la probabilité de voir des répétitions survenir dans la séquence générée. Ces paramètres sont également accessibles pour les types de guidages présentés ci-dessous.

Guidage par scénario long-terme
La génération peut être guidée par un « scénario » déterminé par l'utilisateur. Cette séquence symbolique de référence est définie sur le même alphabet que les annotations de la mémoire musicale utilisée (labels d'accords, modes de jeu, etc.) et guide l'improvisation (séquence d'accords, séquence de modes de jeu, etc.). Dans ce contexte, « improviser » signifie parcourir la mémoire pour collecter des séquences contiguës ou non et les concaténer pour créer une phrase musicale satisfaisant la séquence d'étiquettes imposée par le scenario. Il s'agit donc de trouver des segments de la mémoire correspondant aux portions successives du scenario à suivre et de les enchainer de manière créative. Pour atteindre cet objectif, le modèle proposé associe à chaque instant de la génération l'*anticipation* en assurant la continuité avec le futur du scénario, et la *cohérence avec la logique musicale de la mémoire* en assurant la continuité avec le passé de la mémoire (voir ImproteK, L3.1.1, pour plus de détails).

Guidage par scénarios dynamiques à court terme
L'implémentation des stratégies de gestion de requêtes concurrentes et de l'approche de la réaction en tant que réécriture d'anticipations préalablement générées (exposées dans L3.1.1) a permis d'introduire dans L3.3.1 la notion de scénarios dynamiques à court terme. Au cours de la performance, un opérateur-musicien peut ainsi envoyer des requêtes correspondant à des séquences de labels spécifiant ce que l'agent doit générer et jouer sur le champ, tout en

maintenant la cohérence avec ce qui vient d'être joué ainsi qu'avec les anticipations préalablement générées dans le cas où une révision des anticipations est nécessaire. Ce paradigme de guidage introduit un mode de jeu que l'on pourrait qualifier de « meta DJing » ou « DJing d'intentions » : en effet, un opérateur-musicien peut improviser en contrôlant un agent à l'échelle de la narration musicale (par exemple « à partir du temps prochain : générer et jouer une séquence correspondant à la suite d'accords Dm7 G7 CMaj7», « maintenant : générer et jouer une séquence partant de 'grave rugueux' pour arriver 'aigu brillant' »). Comme nous le verrons dans la suite de ce document, ces scénarios dynamiques à court terme peuvent également provenir d'un module d'écoute analysant le jeu d'un musicien en temps réel.

Génération libre structurée
La modularité de la librairie permet de chaîner deux (ou plusieurs) agents de manière à ce que le type de contenus retournés par le premier soit le type de labels utilisé pour construire les requêtes pilotant le second. Ce type d'utilisation permet d'ajouter une dimension verticale à l'improvisation avec, par exemple, une chaîne dans lequel un agent se spécialise dans l'harmonisation et le second dans l'arrangement. Il permet également d'introduire un intermédiaire en les stratégies de génération « libre » et « structurée ». Un premier agent peut ainsi, par exemple, naviguer de manière « libre » dans sa mémoire et retourner non pas les contenus musicaux des états par lesquels il passe, mais les labels correspondants. La séquence de labels ainsi construite sera donc inédite mais cohérente avec la structure de sa mémoire et pourra servir de scénario pour un autre agent générant du contenu musical.

Les trois modes de guidages suivants constituent un apport de ce nouveau livrable et sont plus amplement détaillés dans la dernière section de ce document.

Guidage par scénario de descripteurs audio
Les stratégies de guidage « par scénario » ou « par scénarios dynamiques à court terme » peuvent également être utilisées pour naviguer dans une mémoire constituée par l'analyse automatique d'un fichier ou d'un flux audio. Au cours de la performance, l'utilisateur peut ainsi envoyer des requêtes spécifiant des séquences de classes que l'agent doit générer et jouer. Les classes constituant l'alphabet de labels sont obtenues par une analyse du fichier « mémoire » selon une sélection de descripteurs effectuée par l'utilisateur, puis par un clustering discrétisant l'espace de descripteurs en un nombre de classes également choisi par l'utilisateur.

Guidage par écoute réactive
La navigation dans une mémoire ainsi constituée par analyse automatique peut également être pilotée par un module d'écoute/analyse temps-réel. Un flux audio capté en temps réel – par exemple un musicien co-improvisant avec l'agent – est analysé selon la même sélection de descripteurs que la mémoire pour créer automatiquement des requêtes de génération.

Guidage hybride écoute réactive / scénarios
La fusion de cette dernière stratégie avec le paradigme de scénarios à court terme a également permis d'introduire de nouvelles stratégies de générations pilotées par l'écoute entre lesquelles un agent peut alterner au cours d'une même performance : le label déterminé par l'écoute peut être utilisé pour créer un scénario court terme dans lequel il est répété, dont il est le point de départ, ou encore dont il est la destination. Ces stratégies introduisent un contrôle hybride entre le musicien produisant le stimulus et l'opérateur-musicien « composant » en temps réel les modalités de la réactivité dans une dynamique de compagnonnage humain / humain-machine.

Vue d'ensemble de la librairie

Nous donnons ici une vision d'ensemble des différentes composantes de la librairie : les classes de la librairie Python illustrées par les interfaces correspondantes dans la librairie Max, et les modules de rendu audio également disponibles dans la librairie Max.

Agent

L'atome de base de la librairie est l'« agent DYCI2 » embarquant un modèle de mémoire appris à partir d'une séquence symbolique : une séquence de couples (date, label) éditée au préalable et pouvant par exemple correspondre à l'annotation d'un fichier, l'analyse automatique d'un flux ou un fichier audio, etc. L'agent reçoit des requêtes de génération, des séquences de *labels*, et suit la stratégie de navigation adaptée (voir plus haut) pour retourner des séquences de *contenus* correspondants dans la mémoire (par exemple des dates à lire dans un fichier audio envoyées à un module de rendu). L'agent possède son temps interne qui peut être indépendant du temps des autres agents, maître du temps d'un autre agent, asservi à celui d'un agent maître, asservi au temps d'un module de rendu.

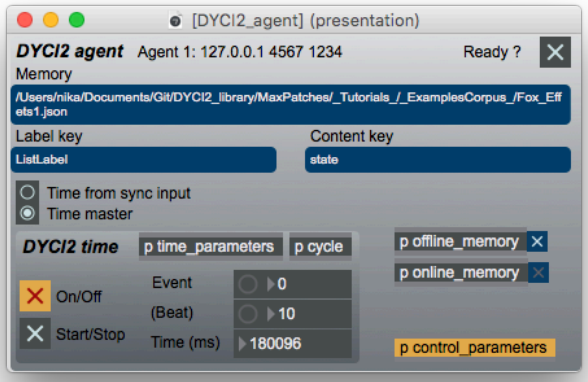

Figure 1. Agent DYCI2 (interface Max).

Requêtes

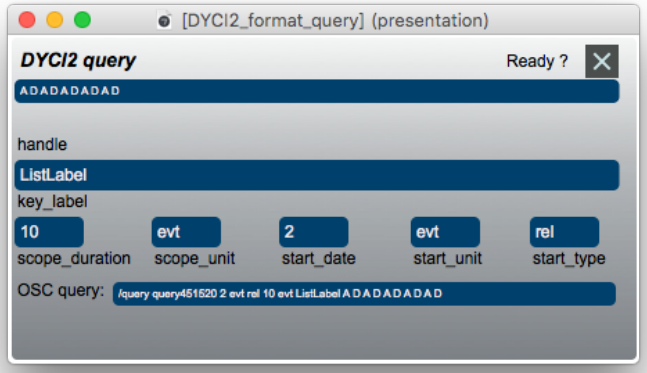

Figure 2. Requête de génération (interface Max).

L'élément principal d'une requête de génération est une « poignée » (*handle*) : la spécification que la génération doit respecter (une valeur de descripteur audio, un symbole représentant un label d'accord ou un mode de jeu, un identifiant de classe dans un espace de descripteurs discrétisé, une séquence de valeurs ou de symboles, etc.). La requête porte également des informations de périmètre temporel : en effet celle-ci peut s'exprimer, selon les stratégies de guidage, en date absolue (pour le guidage par un scénario fixe) ou en date relative (pour le guidage par un scénario dynamique ou par écoute réactive). Cette approche permet d'avoir un format de requête unique pour toutes les stratégies : l'objet créant une requête peut être activé aussi bien par un ordre manuel provenant d'un opérateur-musicien (demandant par exemple un scénario court terme) ou par un module d'analyse/écoute réactive (demandant par exemple un évènement correspondant à la même classe de centroïde spectrale que le dernier évènement joué par un musicien co-improvisant avec l'agent).

Rendu audio et synchronisation

La librairie Max comporte deux types de modules de rendu audio permettant de jouer les séquences générées dynamiquement par un agent. Le premier module, mettant en œuvre un séquencement dynamique réalisé à l'aide d'une partition dynamique écrite dans le langage Antescofo (cf L3.1.1) couplé à des technologies de time stretching, offre des possibilités de synchronisations. Il permet de synchroniser la restitution audio sur une source de temps extérieure (par exemple un métronome, une liste de cues, ou un signal extérieur). Il permet également d'établir des relations entre le temps de plusieurs voix jouant les improvisations générées par plusieurs agents : par exemple en synchronisant la restitution de deux voix d'accompagnement sur une voix maîtresse de solo, elle-même synchronisée tantôt sur une source extérieure, tantôt sur son temps interne.

Figure 3. Module de rendu audio synchronisé.

Rendu audio, analyse automatique, et module d'écoute

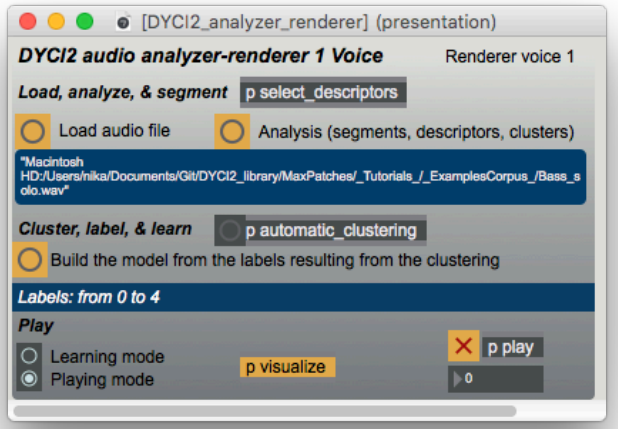

Figure 5. Module d'analyse/rendu audio.

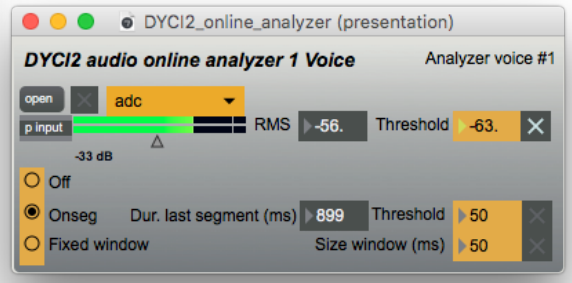

Figure 4. Module d'analyse/écoute réactive.

Le deuxième module de rendu, enrichi de capacités d'analyse, est détaillé dans la dernière section de ce document. Basé sur les technologies MuBu et CataRT, celui-ci permet de naviguer dans un fichier ou un flux audio analysé et segmenté selon des descripteurs audios choisis. Il peut être couplé avec un module d'analyse/écoute dans le cadre d'un guidage de la génération par l'écoute réactive d'un flux audio en temps réel.

Utilisation, documentation, tutoriels

La **librairie Python DYCI2**, dont la première version a été introduite dans le livrable L3.3.1, implémente les modèles temporels, stratégies de navigations guidées, et architectures d'agents génératifs. Sa documentation est disponible à l'adresse suivante : http://repmus.ircam.fr/downloads/docs/DYCI2_library/. Elle a été conçue de manière modulaire afin de pouvoir être prise en main par des développeurs souhaitant utiliser les outils

qu'elle implémente pour créer leurs propres agents improvisateurs, étendre simplement le panel de stratégies de génération, et intégrer ces outils dans un environnement externes, par exemple en utilisant le protocole OSC (ou l'encapsulation en langage C décrite dans le livrable 4.2). Dans cette optique, un fichier tutoriel est associé à chacune de ses classes.

La **librairie Max DYCI2** propose une interface pour chacune des classes de la librairie Python ainsi que différents modules de rendu audio. Elle communique en arrière-plan avec la librairie Python via le protocole OSC et permet de construire aisément son propre dispositif de performance en associant un ou plusieurs agents, modules de rendu, sources de requêtes, et stratégies de génération. Des tutoriels associés à la librairie Max ont été réalisés en collaboration avec le saxophoniste Rémi Fox et la compositrice Marta Gentilucci. Ces patchs d'exemples ont été adaptés de patchs de concerts réalisés pour des créations associées au projet DYCI2. Ces exemples sont accompagnés de matériaux musicaux originaux créés par des artistes partenaires du projet.

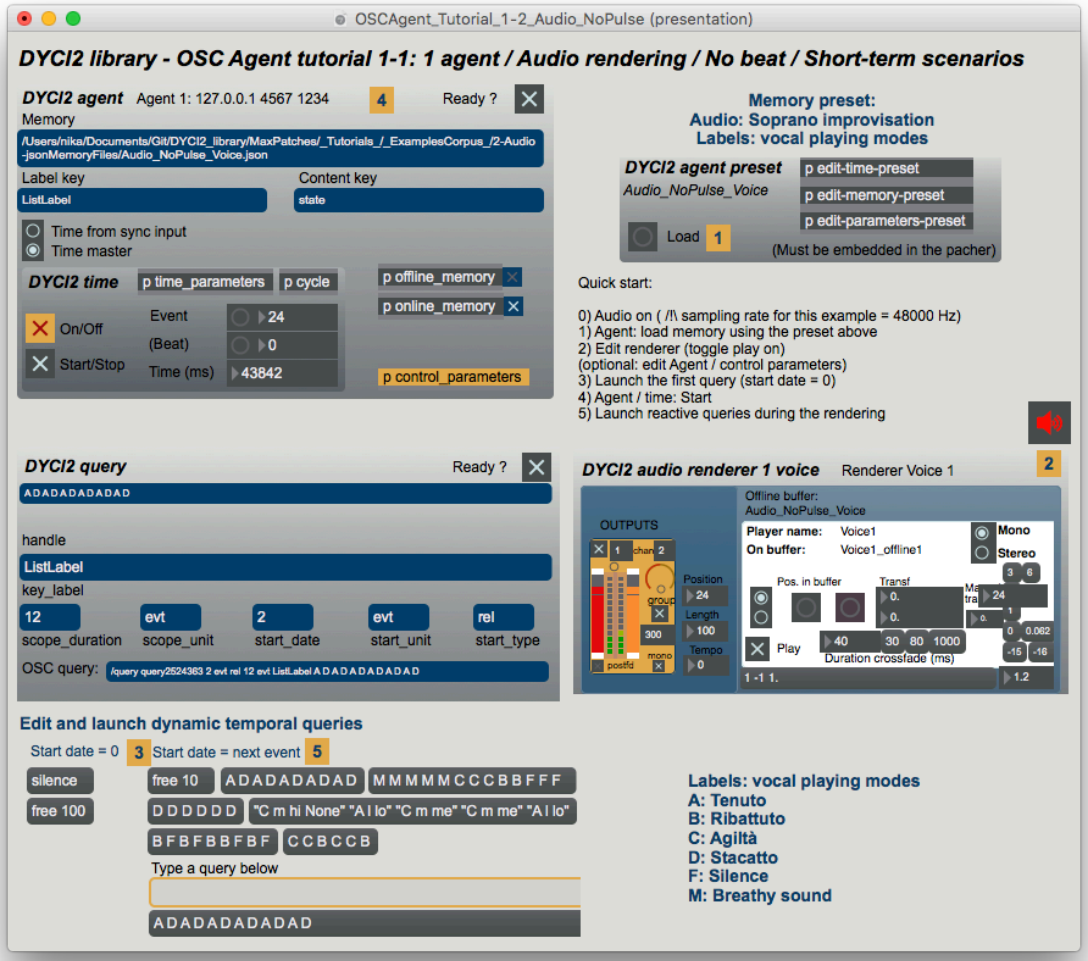

Figure 6. Exemple de patch tutoriel utilisant les objets de la librairie : 1 agent embarquant un modèle appris sur une mémoire audio annotée, des exemples de requêtes pouvant lui être envoyées pour guider la génération, 1 module de restitution audio permettant de jouer les séquences dynamiquement générées (exemple issu de l'utilisation de la librairie DYCI2 par la compositrice Marta Gentillucci lors de sa résidence à l'Ircam, cf. livrable 4.1.4).

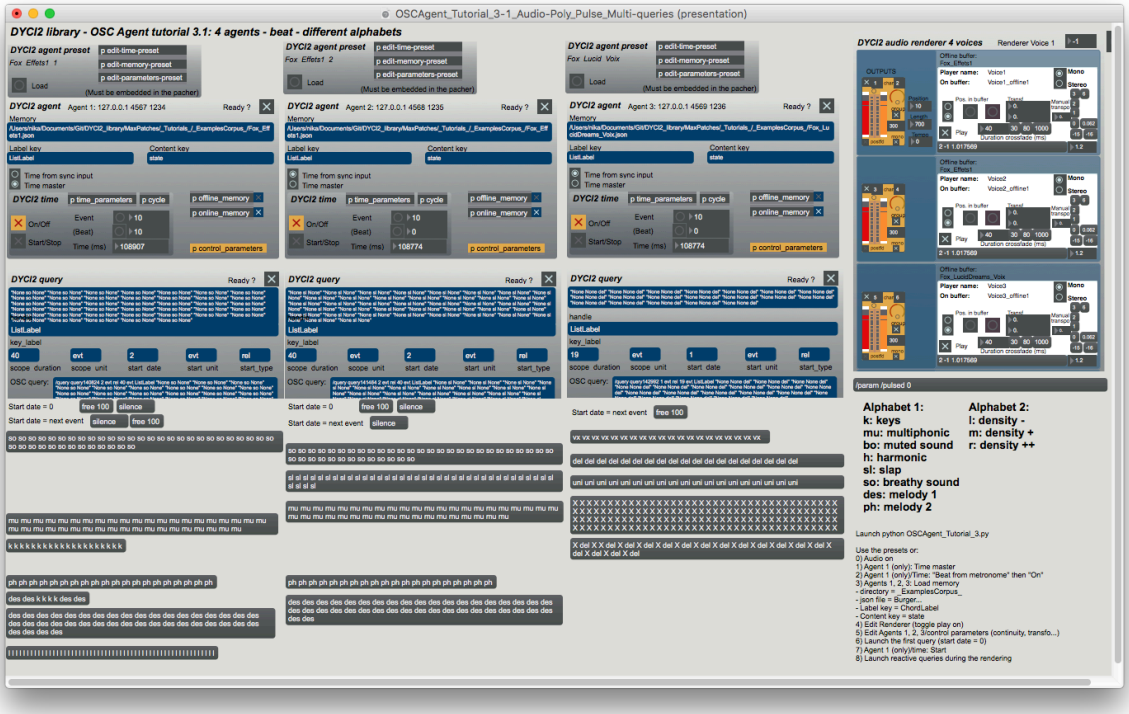

Figure 7. Exemple de patch tutoriel utilisant les objets de la librairie : 3 agents embarquant des modèles appris sur une mémoire audio annotée, des exemples de requêtes pouvant leur être envoyées pour guider la génération, 1 module de restitution audio permettant de jouer les séquences dynamiquement générées en les synchronisant sur le premier agent (exemple issu de l'utilisation de la librairie DYCI2 par le saxophoniste Rémi Fox lors d'une performance au festival « Improtech Paris-Philadelphie », cf. livrable 4.1.4).

Figure 8. Exemple de patch tutoriel utilisant les objets de la librairie : 1 agent embarquant un modèle appris sur l'analyse automatique d'un fichier audio, un module de requêtes manuelles et requêtes provenant d'une écoute réactive d'un flux audio analysé en temps réel, un module de restitution audio basé sur le moteur CataRT (exemple issu de l'utilisation de la librairie DYCI2 par le saxophoniste Rémi Fox lors d'une performance à l'Académie de Darmstadt, juillet 2018).

Guidage par l'écoute : analyse automatique, descripteurs audios, et clustering

Nouveau module d'analyse/rendu construit à partir de l'environnement MuBu
Cette section documente la principale nouveauté introduite depuis le livrable L3.3.1 : l'analyse automatique de fichier ou flux audio et les possibilités génératives associées : le guidage par scénarios de descripteurs audio, le guidage par écoute réactive, et le guidage hybride écoute réactive / scénarios. Ce travail a été mené en collaboration avec Diemo Schwarz et l'équipe ISMM de l'Ircam, et a conduit au développement du nouveau module d'analyse/rendu utilisant l'environnement MuBu (http://forumnet.ircam.fr/fr/produit/mubu/). MuBu *multi-buffer* est un conteneur de données sonores et de mouvement fournissant de la mémoire structurée pour le son et le mouvement enregistrés à travers des interfaces et opérateurs en temps réel en tant qu'objets externes complémentaires pour Max. Chaque piste d'un MuBu buffer peut représenter un flux de données échantillonné ou une séquence d'événements temporels étiquetés, comme par exemple : des échantillons audios, des descripteurs audios, des données de mouvement de captation, des marqueurs, des segments, et des événements musicaux. Les paragraphes ci-dessous détaillent les recherches et développement associés à ces nouveaux modules et modes de guidage de l'improvisation.

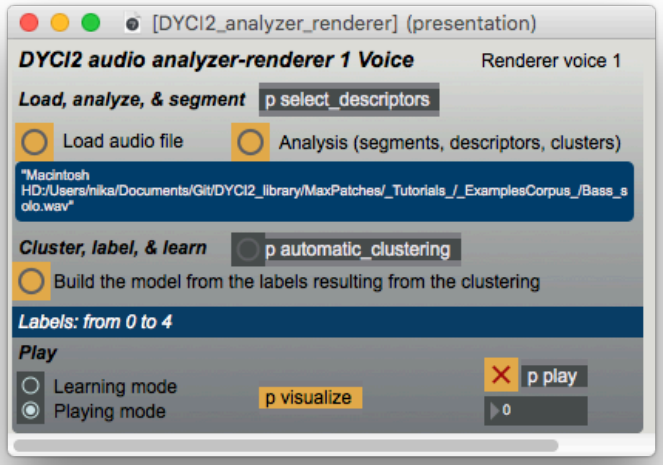

Figure 9. Module d'analyse/rendu audio.

Analyse et segmentation

Le fichier ou flux audio à partir duquel sera construit le modèle de mémoire est analysé selon un ou plusieurs descripteurs choisis par l'utilisateur en fonction de la nature du matériau et des dimensions souhaitées pour guider la génération. Ces données sont ensuite segmentées selon une fenêtre temporelle de taille fixe ou par évènements dont les bornes sont calculées à partir des variations du vecteur de descripteurs (algorithme « onseg » de l'environnement MuBu), pour obtenir enfin un unique vecteur pour chaque segment. Ces opérations sont invisibles pour l'utilisateur qui obtient, une fois le fichier chargé, une représentation visuelle de celui-ci dans une projection de l'espace des descripteurs où chaque segment audio est représenté par un point.

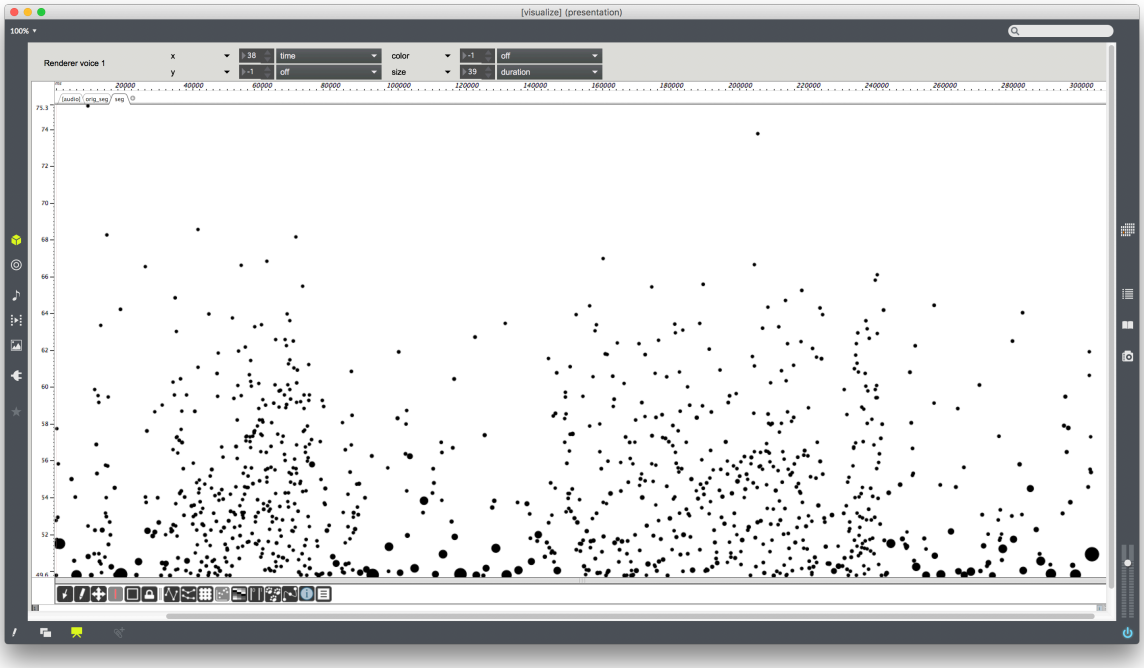

Figure 10. Représentation de la segmentation d'un fichier audio dans une projection d'un espace de descripteurs choisis (construit à partir de l'environnement MuBu et du moteur CataRT).

Clustering, labellisation, et construction du modèle de mémoire

Afin de construire un modèle de mémoire, l'espace des descripteurs est discrétisé pour obtenir un alphabet de labels fini qui servira à étiqueter les différents segments du fichier audio. Un nouvel algorithme de clustering non supervisé, développé à partir de la librairie de modèles de mélanges de gaussiennes XMM (http://ismm.ircam.fr/software/xmm-probabilistic-models-for-motion-recognition-and-mapping/), discrétise l'espace des vecteurs descripteurs en N classes (N gaussiennes dans le modèle). Chaque segment audio est ainsi étiqueté par l'indice de la classe/gaussienne à laquelle il appartient. Une fois encore, l'exécution de l'algorithme est cachée à l'utilisateur qui choisit simplement les descripteurs audios souhaités et le nombre N de classes/labels et obtient une représentation visuelle du fichier « mémoire » dans une projection de l'espace des descripteurs où chaque segment audio est représenté par un point coloré en

fonction de son label. La séquence de labels décrivant la mémoire est finalement envoyée à un Agent DYCI2 afin de construire son modèle de mémoire.

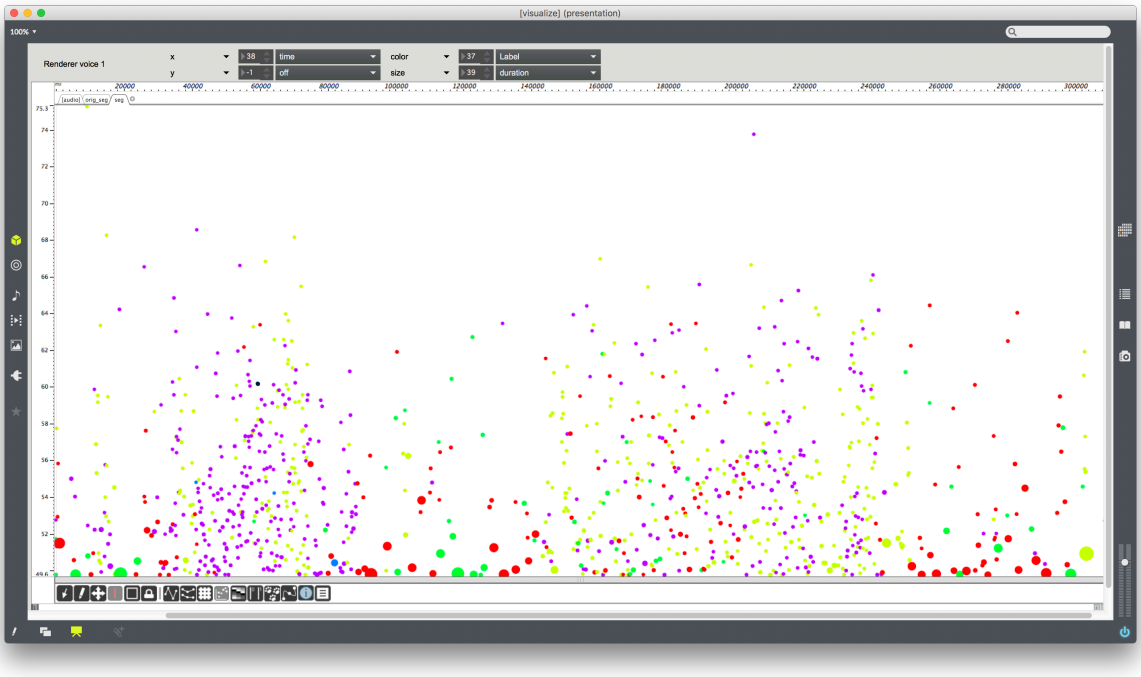

Figure 11. Représentation du clustering appliqué aux segments d'un fichier audio dans une projection d'un espace de descripteurs choisis (construit à partir de l'environnement MuBu et du moteur CataRT).

Génération : guidage par scénarios de descripteurs audios

Une mémoire constituée par le processus d'analyse décrit plus haut peut naturellement être utilisée dans le cadre de stratégies de guidage « par scénario » ou « par scénarios dynamiques à court terme ». Au cours de la performance, un opérateur-musicien peut ainsi envoyer des requêtes correspondant à des séquences de classes que l'agent doit générer et jouer sur le champ tout en maintenant la cohérence avec ce qui vient d'être joué ainsi qu'avec les anticipations préalablement générées dans le cas où une révision des anticipations est nécessaire. L'agent suit la stratégie de navigation adaptée pour retourner la séquence de segments à lire dans la mémoire du module d'analyse/rendu. Cette restitution est effectuée grâce au moteur CataRT (http://imtr.ircam.fr/imtr/CataRT) associé à la librairie MuBu, tout en affichant dans l'espace des descripteurs le segment joué à chaque instant. Notons que cette nouvelle technologie enrichit CataRT d'une prise en compte de la temporalité du fichier « mémoire » et d'une optimisation des chemins choisis dans l'espace de descripteurs grâce à celle-ci.

Ce mode de jeu permet à l'utilisateur une exploration préalable de l'espace des classes de descripteurs avant la performance, comme le montre le patch tutoriel associé à cette stratégie de guidage où le fichier « mémoire » est un solo de basse, et le nombre de classes/labels pour discrétiser l'espace de descripteurs choisi est fixé à 5. En seulement quelques requêtes, l'exploration des 5 classes permet d'observer que l'une d'entre elle correspond à des notes jouées dans un registre medium, une autre dans un registre aigu, une autre des notes jouées en « slap »,

une autre les notes jouées en harmoniques, et une autre aux applaudissements du public.

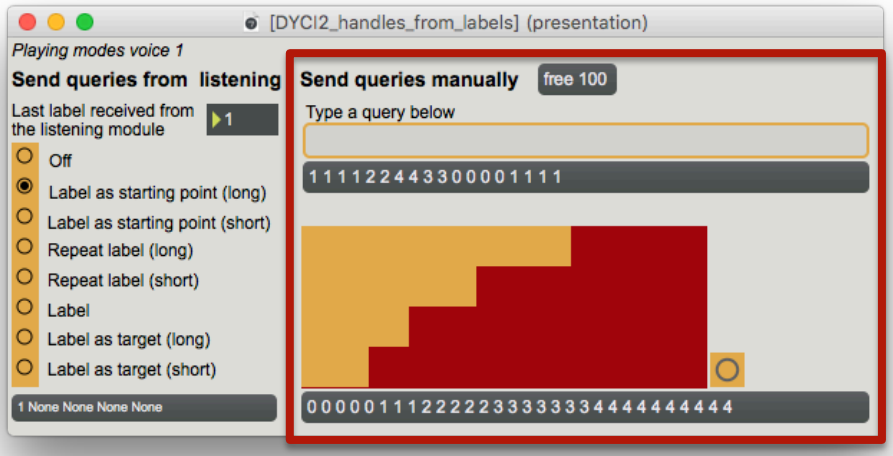

Figure 12. Requêtes de générations : scénarios de classes de descripteurs.

Génération : guidage par écoute réactive, scénarios créés par l'écoute, guidage hybride
Ce mode de guidage introduit un nouveau module d'écoute/analyse temps-réel. Un flux audio capté en temps réel – par exemple un musicien co-improvisant avec l'agent – est analysé selon la même sélection de descripteurs que le fichier « mémoire » et est segmenté selon une fenêtre temporelle de taille fixe ou par évènements pour obtenir un unique vecteur pour chaque segment. Le vecteur de descripteur associé au dernier segment entendu est projeté dans la cartographie apprise sur le fichier « mémoire » afin d'obtenir la gaussienne/classe dont il est le plus proche. Notons que le choix du mode de segmentation à un impact fort sur le résultat musical : la segmentation selon une fenêtre temporelle de taille fixe assez courte permet à l'agent de réagir quasi instantanément (perceptivement) mais l'identification d'une classe pertinente est plus difficile sur une petite fenêtre. A l'inverse, si la segmentation par évènements permet une bonne identification de la classe, la requête de génération ne peut être envoyée, par définition, qu'une fois l'évènement terminé.

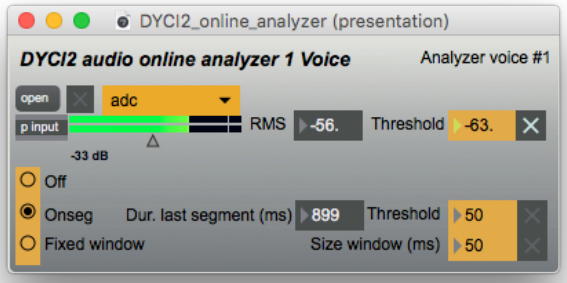

Figure 13. Module d'analyse/écoute réactive.

L'identifiant de classe attribué au dernier segment entendu constitue donc un label à partir duquel une requête de génération est construite puis envoyée à l'agent qui génère et joue instantanément un élément ou une séquence d'éléments. Comme dans Somax (L3.1.2), la requête créée à partir de ce label peut simplement être constituée du label seul (« maintenant, génère et joue un évènement correspondant au dernier évènement entendu »). La fusion de ce paradigme avec celui des scénarios à court terme a également permis d'introduire de nouvelles stratégies de générations pilotées par l'écoute entre lesquelles un agent peut alterner au cours d'une même performance : le label déterminé par l'écoute peut être utilisé pour créer un scénario court terme dans lequel il est répété, dont il est le point de départ, ou encore dont il est la destination.

Ce mode de jeu a été utilisé pour la première fois avec succès lors de la création de deux pièces à l'Académie de Darmstadt en juillet 2018 avec le saxophoniste Rémi Fox. Ce contrôle hybride entre le musicien produisant le stimulus et l'opérateur-musicien « composant » en temps réel les modalités de réaction ouvre une nouvelle direction prometteuse de guidage hybride et de compagnonnage humain / humain-machine.

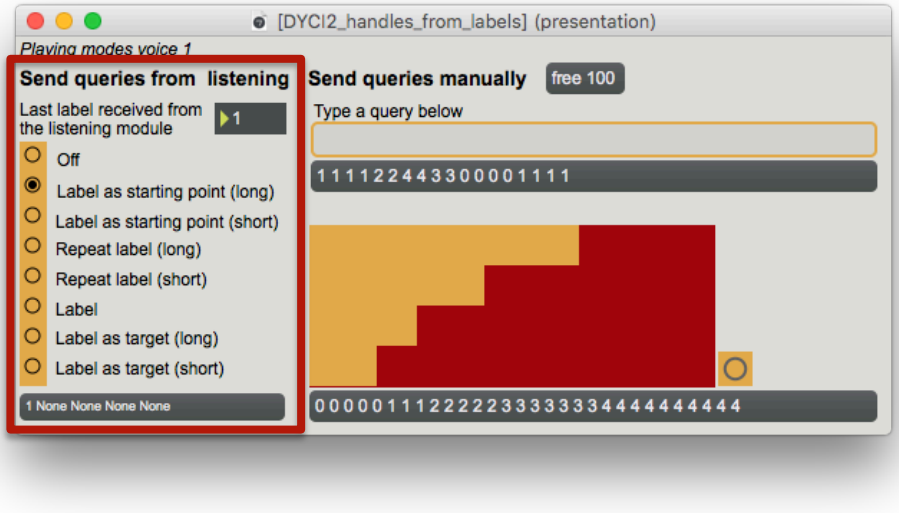

Figure 14. Requêtes de générations : scénarios issus d'une écoute réactive.