PURE DATA

Published : 2017-06-21
License : GPLv2+

INTRODUCTION

1. INTRODUCTION
2. À PROPOS DE CE MANUEL
3. EXEMPLES D'UTILISATION
4. PROGRAMMATION GRAPHIQUE

1. INTRODUCTION

Pure Data (souvent abrégé *Pd*) est un logiciel de création multimédia interactive couramment utilisé dans les domaines artistique, scientifique et pédagogique. Sa popularité réside notamment dans sa facilité d'utilisation. Plutôt qu'un langage de programmation textuel, Pure Data propose un environnement de **programmation graphique** dans laquelle l'utilisateur est invité à manipuler des icônes représentant des fonctionnalités et à les brancher ensemble.

UN LOGICIEL DE CRÉATION MULTIMÉDIA INTERACTIVE

Pure Data est utilisé dans tous **les champs de la création** (musique, arts visuels, danse, théâtre, robotique, etc.) pour des performances visuelles et sonores ou la création d'installations interactives, participatives et génératives. Il possède des capacités particulières dans les domaines de la musique acoustique et de la musique audio-numérique : il sert à modéliser des instruments électroniques comme les synthétiseurs. Il permet la gestion d'échantillonneurs (*samplers*) et d'effets, la composition musicale, ou encore, la création de séquenceur *MIDI*, etc. Il peut également gérer des applications vidéo et 3D.

Avec Pure Data, il est possible également d'interfacer un programme avec le monde physique en utilisant des capteurs (caméras, détecteurs de présence, etc.) pour commander des robots, interagir avec des sites internet, ou encore effectuer une visualisation de données. En art appliqué, il est utilisé en design d'interaction, en architecture et pour créer des jeux.

Par ailleurs, on trouve des utilisateurs de Pure Data dans d'autres disciplines techniques et scientifiques comme **l'électronique, les sciences physiques et les mathématiques**. Par exemple, il peut être un excellent outil de mesures acoustiques.

Enfin, c'est **un outil pédagogique** pour les analyses acoustiques et audio-numériques, la synthèse sonore, la 3D, les opérations mathématiques, et de multiples autres domaines. Grâce à son système de programmation intuitif, Pure Data favorise l'apprentissage et l'expérimentation. De nombreux enseignants et formateurs pédagogues l'utilisent pour leurs cours, formations et ateliers ; certains conçoivent à l'aide de ce logiciel des dispositifs numériques originaux d'enseignement.

L'accès à tous ces domaines constitue une formidable richesse. Le croisement des disciplines et des compétences crée une effervescence créative et stimulante. Cependant, certains domaines techniques, comme la manipulation du son, font appel à des connaissances très spécialisées qui dépassent la simple utilisation d'un logiciel. Cela peut créer des obstacles et des frustrations qu'il ne faut pas nier. Ces difficultés peuvent être dépassées en utilisant les ressources d'aide disponibles comme les forums et les listes de diffusion.

UNE EXPÉRIENCE DE PROGRAMMATION VISUELLE ET INTUITIVE

Si Pure Data emprunte à la programmation classique certaines notions comme les noms de fonctions (objets), son attrait réside dans son environnement graphique. La programmation graphique permet d'organiser des applications avec des icônes sans faire appel à des lignes de texte, contrairement aux environnements de programmation traditionnels que sont par exemple le langage C ou le Pascal.

Programmer avec Pure Data est une expérience qui s'apparente à manipuler des choses tangibles et à les brancher ensemble. L'unité de base est une boîte rectangulaire, et l'écriture du programme (le patch) consiste à relier plusieurs boîtes par des ficelles (ou cordes). Cela forme des graphes, ou diagrammes, qui mettent en abyme leur propre fonctionnement.

Le programme fonctionne en temps réel, permettant ainsi à l'utilisateur de modifier son code et de voir aussitôt les changements s'appliquer.

À ce jour, l'espace de travail (l'interface) du logiciel Pure Data est disponible uniquement en anglais. Malgré cela, son utilisation reste simple et il est très probable que les versions à venir proposeront une traduction en plusieurs langues, dont le français. Les nombreux atouts de ce logiciel libre le rendent suffisamment incontournable pour commencer à apprendre à l'utiliser sans plus attendre.

UN LOGICIEL LIBRE ET GRATUIT

Fonctionnant sur les plateformes Linux, Mac et Windows, Pure Data est un logiciel libre et gratuit. La licence « *Standard Improved BSD* » qui protège ce logiciel permet en effet un travail collaboratif, l'accessibilité des sources et une distribution gratuite auprès des utilisateurs. Le noyau de Pure Data (Pd Vanilla) est toujours maintenu par son créateur, Miller Puckette. Par la suite, de nombreux développeurs se sont joints au projet : ils ont ainsi permis d'ajouter des « librairies » qui étendent les capacités du logiciel. Celles-ci sont fournies avec *Pure Data Extended,* version du logiciel qui offre plusieurs améliorations par rapport à la version *Vanilla*. La communauté se compose de **développeurs**, d'**utilisateurs** et de personnes « **ressources** ».

Les développeurs de librairie créent de nouveaux objets en langage C ou C++, ou aident à l'amélioration générale du programme. Les utilisateurs s'y consacrent à des fins artistiques, scientifiques ou pédagogiques. Ils peuvent aussi contribuer à la communauté en partageant leurs expériences auprès des développeurs ou en éditant des documents, des tutoriels, etc. Les personnes « **ressources** » organisent des événements de transmission des connaissances (conventions), des rencontres, des festivals artistiques et participent ainsi à la vitalité des échanges dans la communauté.

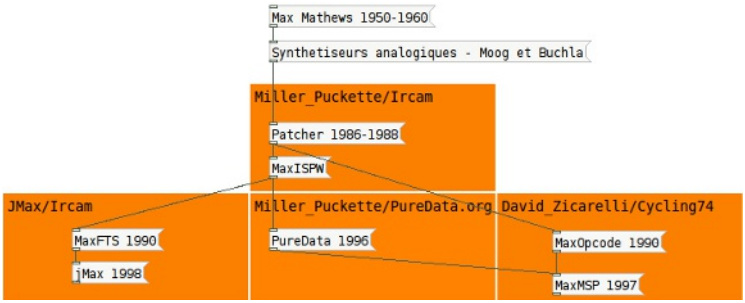
PETIT HISTORIQUE

Pure Data fait partie de la famille des langages de programmation par patchs comme JMax, VVVV, Ingen, etc. Cette famille de langages constituée de boîtes et de ficelles tire son origine de la conception modulaire expérimentée dans les premiers programmes musicaux de Max Mathews au cours des années 1950, programmes qui ont par la suite inspiré les premiers synthétiseurs analogiques.

Miller Puckette est le créateur de Pure Data. En 1988, au sein de l'Ircam, une institution française dédiée à la recherche et à la création musicale contemporaine, il développe l'éditeur Patcher (http://crca.ucsd.edu/~msp/Publications/icmc88.pdf). Ce logiciel fut revendu à la société Opcode pour créer bien plus tard Max/MSP. Pour sa part, Miller Puckette reprit la conception de Patcher pour en faire un logiciel libre à des fins musicales : Pure Data.

Les conditions de la vente du logiciel Patcher à une entreprise commerciale, alors que la phase de recherche initiale était intégralement financée par des fonds publics, font encore aujourd'hui débat. Pure Data est pour sa part publié sous la licence libre BSD. Tout en garantissant de toujours pouvoir utiliser, partager et modifier librement Pure Data, cette licenc autorise la privatisation du code source de ce programme : c'est ainsi que les créateurs de Max/MSP ont pu légalement copier le code source de Pure Data pour créer la partie qui s'occupe du traitement de signal audio de leur logiciel propriétaire.

Pour les plus historiens d'entre vous, voici un diagramme sous la forme d'un patch Pure Data retraçant ces évolutions, emprunts et influences :



La version actuelle de base de Pure Data s'appelle **pd-vanilla**. La version **pd-extented 0.42-5** regroupe Pd-vanilla et de nombreuses librairies étendant ainsi les fonctions du logiciel. Les exemples donnés dans ce manuel font référence à ces 2 versions.

Vous consultez l'édition révisée et augmentée du 20 juin 2014.

2. À PROPOS DE CE MANUEL

Les valeurs du libre ont inspiré la rédaction et la diffusion de ce manuel d'initiation à Pure Data, l'objectif étant à la fois :

- d'offrir à un public professionnel ou amateur francophone les bases d'utilisation de Pure Data ;
- de valoriser la communauté des développeurs et experts francophones de Pure Data impliqués dans la rédaction et la mise à jour de ce manuel en français ;
- de fédérer plus largement la communauté francophone de Pure Data autour d'un projet commun de documentation (tant au niveau des co-auteurs que des commentateurs), sachant que l'ouvrage dans sa forme accessible en ligne (wiki) peut être amélioré et comporter de nouveaux chapitres, notamment à l'occasion de la parution d'une nouvelle version de Pure Data.

Ce manuel est une production originale en français ; plusieurs co-auteurs francophones de différents pays ont participé à sa rédaction. L'apprentissage de Pure Data est proposé ici pas à pas et nous vous invitons à suivre l'ordre de succession des chapitres, particulièrement les premiers, qui posent les bases de l'utilisation du logiciel. Cet ouvrage est né d'une collaboration : il évolue au fur et à mesure des contributions. Pour consulter la dernière version actualisée, nous vous invitons à visiter régulièrement le volet francophone de Floss Manuals sur le site http://fr.flossmanuals.net/puredata/.

Le cœur et la structure de l'ouvrage de plus de 150 pages ont été réalisés en 5 jours dans le cadre d'un BookSprint qui s'est tenu à Paris du 7 au 11 février 2011, grâce au partenariat et au soutien de l'Organisation internationale de la Francophonie (http://www.francophonie.org), de Mains d'Œuvres (http://www.mainsdoeuvres.org) et d'Okno (http://www.okno.be).

Expérimentée et popularisée par la Floss Manuals Foundation dans le cadre de ses activités de création de manuels multilingues sur les logiciels et pratiques libres, la méthodologie du Booksprint permet de rédiger en un temps très court des livres de qualité. Un groupe de cinq experts francophones de Pure Data originaires d'Amérique du Nord, d'Europe et du continent africain se sont retrouvés dans un même lieu pour rédiger ce manuel. L'usage de la plateforme de co-rédaction en ligne a permis également à de nombreuses personnes intéressées de s'associer à distance à l'expérience.

Co-rédacteurs présents lors du booksprint :

- Farah Khelil (Tunisie)
- Pascale Gustin (France)
- Alexandre Quessy (Canada-Québec)
- Benjamin Cadon (France)
- Olivier Meunier (Belgique)

Facilitateur :

- Elisa de Castro Guerra

Ce BookSprint a été également étudié par la sociologue Anne Goldenberg, spécialisée dans le wiki et les facilitations afin de nourrir une réflexion en cours au sujet de la méthodologie « BookSprint ».

Co-rédacteurs en ligne et contributions externes (sans ordre particulier) :

- Agnès Le Foulgoc
- Corinne Laurent Dell'Accio
- Bernard Delcourt
- Christian Ambaud
- Jérôme Abel
- Olivier Heinry
- Patrick Fontana
- Roland Kossigan Assilevi
- Tad Bisaha
- Vincent Rioux
- Édouard Sufrin
- Stéphanie Castonguay
- Alexandre Castonguay

À l'occasion, les co-auteurs francophones se sont appuyés sur le manuel en langue anglaise disponible sur http://en.flossmanuals.net, où sont crédités les rédacteurs et illustrateurs concernés.

N'hésitez pas à votre tour à améliorer ce manuel en nous faisant part de vos commentaires dans la liste de diffusion francophone de Flossmanuals, ou, si vous avez des talents de rédacteur et une bonne connaissance de Pure Data, à vous inscrire en tant que contributeur pour proposer la création de nouveaux chapitres. Vous trouverez en fin d'ouvrage la liste complète des personnes ayant participé jusqu'à ce jour à la co-rédaction du manuel.

UN MANUEL LIBRE DISPONIBLE SOUS PLUSIEURS FORMATS ET SUPPORTS

Ce manuel est disponible depuis le site de Flossmanuals sous plusieurs formes : livre imprimé, pages web, pdf et ePub, ce dernier format permettant de le consulter facilement sur des appareils portatifs.

Vous consultez l'édition révisée et augmentée du 30 janvier 2013.

3. EXEMPLES D'UTILISATION

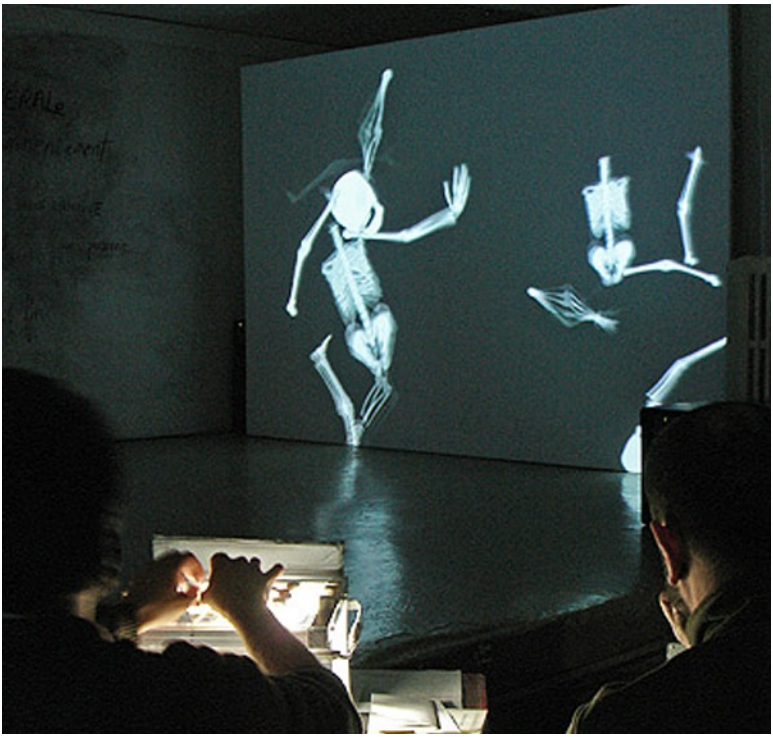
Logiciel de création multimédia en temps réel, **Pure Data est couramment utilisé dans les domaines artistiques** (musique, arts visuels, danse, théâtre, arts appliqués, etc.), **scientifiques** (électronique, robotique, sciences physiques, mathématiques) et **pédagogiques**. La diversité de ces champs d'application constitue une formidable richesse, le croisement des disciplines et des compétences favorisant une effervescence créative.

PRODUCTIONS ARTISTIQUES AVEC PURE DATA

Dans les domaines des arts et de l'expérimentation, la richesse des fonctionnalités de Pure Data stimule l'imagination des créateurs et favorise la pluridisciplinarité. Utilisé notamment dans la musique, les arts visuels et la danse, il est au cœur de performances visuelles et sonores interactives. Pure Data voyage à travers pays et cultures, par exemple au Canada, en Belgique, au Sénégal, en Tunisie, au Maroc, en Roumanie, etc.

Voici quelques exemples de productions artistiques et musicales réalisées avec Pure Data.

STUDIO DIOPTRIQUE



Utilisé dans le cadre de vidéo-performances, le dispositif Studio dioptrique déconstruit les images qui passent à travers lui : morcelées, déstructurées, elles prennent un nouveau sens. Gérées par l'ordinateur, les déformations de l'image répondent en temps réel aux mouvements des objets que l'utilisateur manipule. Studio dioptrique est développé avec des logiciels libres, notamment Pure Data.

Réalisé par Patrick Fontana et Pierre-Yves Fave (France) : http://fofana.free.fr/luca/videoperformancefr.htm

SYMPTÔMES

Symptômes *lors du Plastic Hacker Space Festival, octobre 2010.*

Des fragments de mots, visibles à l'écran, deviennent des espaces concrets et temporels. *Symptômes* explore les espaces de similitude ou de dissension entre texte et image. Utilisant le champ électromagnétique pour la capture de l'environnement sonore interne à l'ordinateur, un ou plusieurs enregistrements émergent de cet écosystème fréquentiel. À l'aide du logiciel Pure Data, de légères modulations des fréquences rendent le discours complètement inaudible ou changent la qualité du son.

Réalisé en 2010 par Pascale Gustin (France) :
http://www.pascsaq.org/weblog/archives/2010/11/11/sympt%C3%B4mes/index.html

GHOST ELECTRIC GEARS

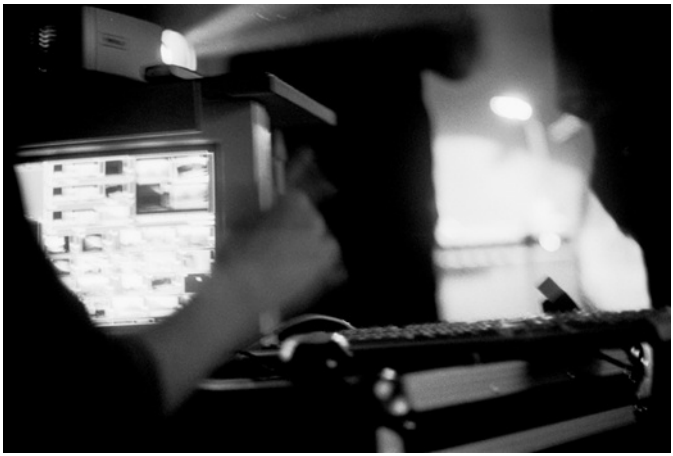
Ghost electric Gears *au Festival Garmerz 06*

Le principe de l'installation Ghost electric Gears est d'implémenter dans le monde de l'art l'esthétique des vitrines des magasins de matériel informatique, de produits multimédia et d'appareils électroménagers. L'idée consiste à utiliser ces vitrines comme instruments de musique assistés par ordinateur. Le dispositif est composé d'une carte Arduino et d'un séquenceur MIDI contrôlés par le logiciel Pure Data : la carte d'interface envoie la lecture d'un fichier audio (MIDI). Ainsi, on peut animer ces objets qui « trainent dans nos placards » et créer une ambiance faite de lumières et de sons.

Réalisé en 2010 par Fenshu (France) :

- Vidéo de l'installation au Festival Gamerz 06 : http://www.dailymotion.com/video/xfz1fu_festival-gamerz-06-fenshu-version-longue_videogames
- Patch Pd.zip
- http://reso-nance.org/?p=272&lang=fr

L.T.D.M.S.



L.T.D.M.S. est une formation belge de post-rock noisy créée début 2000 sur les cendres du groupe punk-rock *Les Trucks*. Le groupe apprécie de récréer sur scène un lieu qui lui soit propre en installant téléviseurs, projecteurs ou lumières tamisées. Le VJ (Bernard Delcourt) utilise Pure Data avec l'extension vidéo PDP pour des projections multi-écrans et pour transformer le son de certains instruments en direct.

L.T.D.M.S. (Belgique) : http://www.ltdms.be

LE POULPE

Les composants matériels et logiciels du Poulpe : capteurs de son, table de mixage, ordinateur, carte son, patch Pure Data, haut-parleurs.

Le POULPE s'installe dans des lieux de vie quotidienne et citadine. Il forme le corps virtuel de la ville qui exprime par le son ses mouvements invisibles et l'ensemble des flux qui la traversent et la constituent. À l'aide de capteurs de sons installés en différents endroits, le POULPE déploie ses tentacules pour lier des contextes toujours différents et qui échangent leurs flux sonores via internet : ces flux continus s'immiscentdans l'environnement sonore d'un milieu et le modifient.

Selon sa propre logique, un automate virtuel utilisant Pure Data filtre en direct les sons captés, les transforme, les mixe et les redistribue.

Réalisé en partenariat avec la Cellule d'Intervention d'APO33, l'ECM de la Région Centre Bandits-mages, Labomedia et l'atelier nUM de l'École supérieure des beaux-arts de Tours (France) :
http://www.apo33.org/poulpe/doku.php

Patch Pd du Poulpe orléanais :
http://yamatierea.org/papatchs/lapoule33/lapoule33.zip

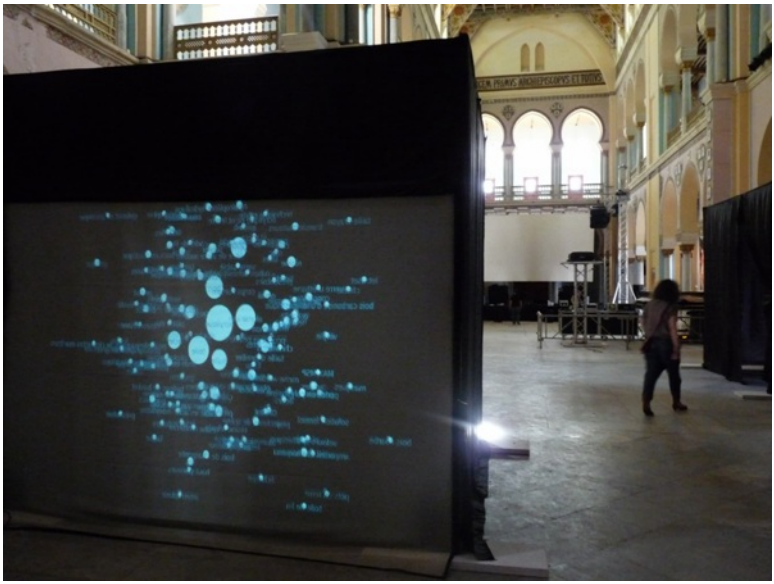
LA CHORALE A ROULETTES

Téléphones à cadran bricolés, ordinateur, logiciel Pure Data.

La *Chorale à roulettes* est une installation sonore qui met en scène des téléphones à cadran, symboles de la communication moderne. Chaque appareil a conservé sa sonnerie originale. Additionnées et composées, celles-ci génèrent un répertoire étendu où des timbres joyeux et entraînants peuvent répondre à des sons d'une infinie tristesse. La *Chorale à roulettes* propose une gamme de timbres allant de la sonnerie mélodieuse au clic à peine audible d'un ronron métallique assourdi.

Réalisé en 2007 par Darsha Hewitt et Alexandre Quessy (Canada-Québec) : http://alexandre.quessy.net/?q=rotarianchoir

TECHNIQUE MIXTE



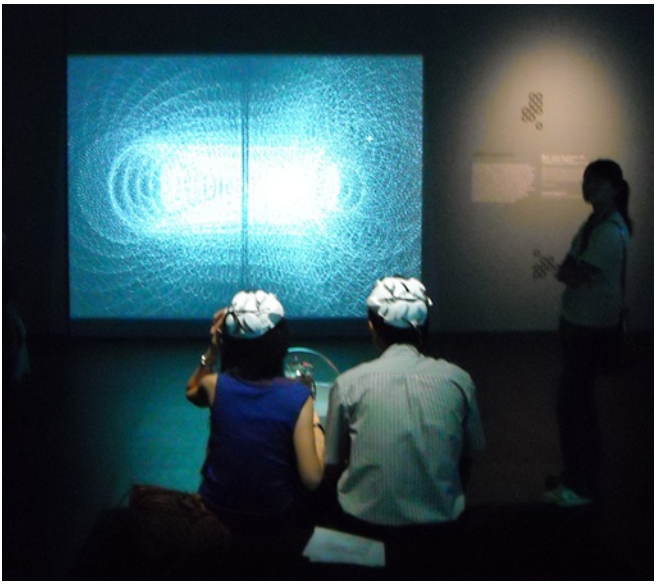
L'œuvre participative Technique mixte *au Festival E-Fest 2010 à Tunis : écran de rétro-projection, vidéoprojecteur, documents papier format A4, ordinateur, logiciel Pure Data, carte son, capteurs pyroélectriques 60° et enceintes.*

Technique mixte est un dispositif qui propose d'exploiter une base de données virtuelle, actualisée et déclinée sous différents formats (vidéo, papier et audio). Il introduit le processus d'exploration de données (*Data Mining*) dans le domaine de l'art. À partir d'une fouille de données, de techniques et de matériaux utilisés pour décrire les œuvres d'art réunies sous forme de liste au sein d'un document poétique, un programme lit le texte et génère en temps réel une représentation dynamique et diagrammatique.

Un processus de traduction des données, entre visible et lisible, génère des bulles qui grossissent, flottent et s'entrechoquent, évoluant ainsi dans le temps en fonction de leurs cohabitations récurrentes au sein des différentes techniques artistiques énoncées dans le texte. Par son intrusion dans ce dispositif interactif, le spectateur déclenche la relecture du texte et renouvelle ainsi tout le processus.

Réalisé en 2010 par Farah Khelil (Tunisie) :
http://farahkhelil.free.fr/index.php?/projets/techniques-mixtes

AURORA CONSURGENS



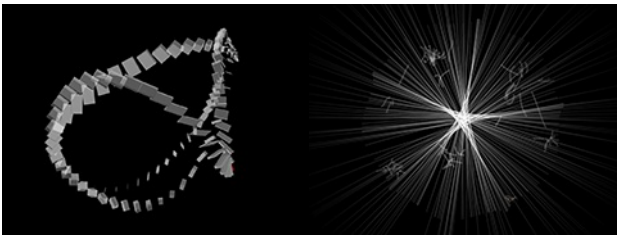
Aurora consurgens *au Symposium International sur les arts numériques (ISEA 2008, Singapour) : casques, capteurs d'activité cérébrale, écran de rétro-projection, vidéoprojecteur, ordinateur, logiciel Pure Data, carte sons et enceintes.*

Aurora Consurgens explore la question de l'intelligence collective et de l'inconscient. A l'aide d'un simple casque posé sur leur tête, deux personnes unissent l'activité de leur cerveau pour créer une œuvre visuelle et sonore évoluant en temps réel. Formes archétypales abstraites et fréquences binaurales se déploient dans l'espace, plongeant les spectateurs-acteurs dans une expérience aux frontières de la science et de la para-science.

Cette installation numérique a notamment été exposée à Singapour (ISEA 2008), Liban (Crea numerica, 2009) et Chine (FIAM 2010).

Réalisé en 2008 par Horia Cosmin Samoïla et Marie Christine Driesen (Roumanie / France) en collaboration avec le Mixed Reality Lab : http://www.ghostlab.org/installations/

CHDH

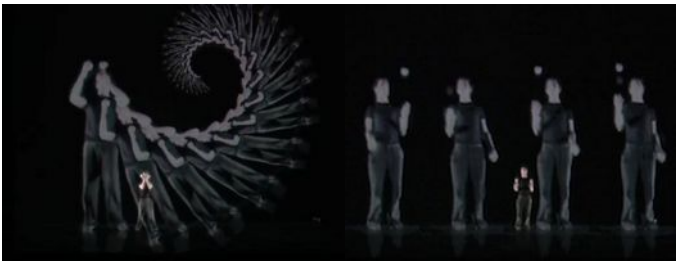


Deux ordinateurs portables, logiciel Pure Data, deux interfaces potentiomètres motorisés MIDI (BCF2000), une carte son (deux voies, sortie cinch), un splitter VGA.

La performance Chdh explore les relations entre l'image et le son, entre autres à l'aide du logiciel Pure Data couplé à la librairie GEM. Ce projet évoque un monde virtuel, constitué de créatures abstraites plus ou moins autonomes. Le mouvement de ces créatures comme des instruments audiovisuels singuliers est influencé en temps réel par deux instrumentistes qui les font vivre et réagir. Ces objets virtuels créent alors des données utilisées pour la synthèse de la vidéo et du son, générant ainsi une forte cohésion entre les médias utilisés. Le public assiste alors à une projection cinéma générée en temps réel par les deux artistes placés à côté de l'écran.

Réalisé par Cyrille Henry, Nicolas Montgermont, Damien Henry (France) : http://www.chdh.free.fr/spip.php?article2

RETIME/KRONOSCOP



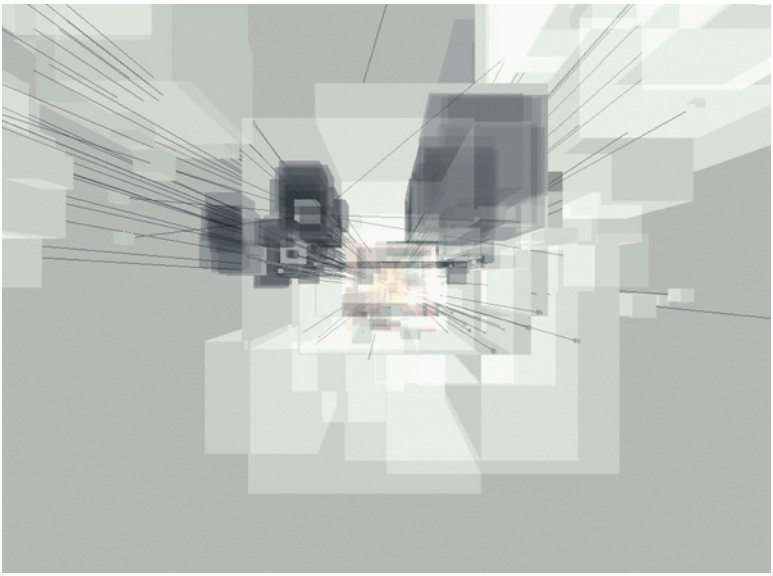
Caméras, un écran en tulle, éclairage, ordinateur, logiciel Pure Data.

Grâce à un dispositif informatique, on peut enfin jongler avec le temps : faire ressentir l'incroyable dilatation des secondes, suspendre des instants choisis, retrouver le présent qui file inexorablement ou, encore, revenir sur une erreur. Ces sensations prennent un sens particulier lorsqu'elles sont confrontées au jonglage, art de la contrainte. *ReTime* propose d'explorer ce bref instant du jonglage où naît l'erreur... quelques secondes où tout s'échappe... ce moment où, hors de contrôle, le monde patiemment bâti dans la maîtrise, s'écroule.

Entre le désarroi et le désaveu, le jongleur, dans son jeu avec la gravité, a encore perdu. Au-delà de l'expérience de la chute et sa temporalité éphémère, *reTime* joue avec le temps et l'espace comme le jongleur avec la pesanteur. Cette observation est d'autant plus perceptible lors d'un spectacle puisque le public a une autre vision de ce que vit « l'acteur », il ne perçoit pas la temporalité de celui qui agit. Aussi *reTime* propose un dispositif qui joue de cette perception du temps.

Réalisé en 2006 par la Compagnie Adrien (France) : http://www.adrienm.net/spectacles/retime/index.html

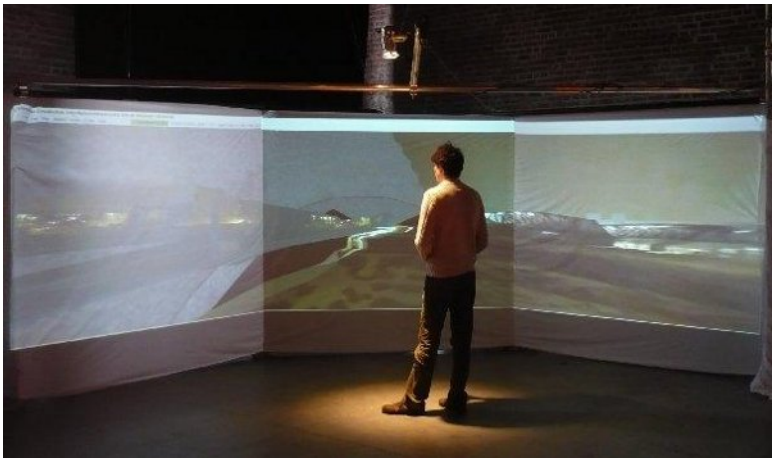
RYBN



Depuis mars 2006, le collectif artistique RYbN développe un projet de recherche autour du *Data Mining*, basé sur la récupération et la visualisation de données accessibles sur Internet. En détournant les objectifs initiaux de *Data Mining* et de technologies de surveillance numériques, les données collectées servent à créer une série d'images. Ces visualisations numériques reposent sur des principes cartographiques, et sont mises à jour en temps réel. Le projet intitulé Antidatamining (ADM) est à la fois une enquête artistique, socio-économique et géopolitique, ainsi qu'un processus archéologique en temps réel se concentrant sur les flux de données qui composent une partie de notre société contemporaine.

Réalisé depuis 2006 par RYbN (France / Belgique) :
http://www.imal.org/WorkshopADM/index.php?page=rybn_fr

THIN SYNCH'D SPACES



Installation au BRASS, 2010, sur un écran en plastique végétal. Un ordinateur, trois projecteurs vidéo, quatre enceintes acoustiques, caméra IR, spot, structure aluminium, tendeurs.

Sur un grand écran tendu en travers de la salle, l'image reflète un monde étrange fait d'assemblages d'espaces distincts et dans une temporalité déconcertante. Lorsque le spectateur s'en approche et pénètre dans cet univers, le son et l'image suivent ses mouvements. Des lieux d'un paysage urbain ont été filmés durant plusieurs jours, et par une synchronisation toujours changeante des boucles d'images, le temps perçu est alors « déréglé ».

L'espace visuel de cette installation est composé de surfaces vidéo dans un espace 3D virtuel, de prises de vue de plusieurs caméras placées en différents lieux et reliées par un réseau sans fil. Le visiteur parcourt ce lieu étrange grâce à un capteur de mouvement permettant un contrôle corporel du point de vue : il se meut physiquement, et ses déplacements se répercutent virtuellement dans l'architecture en mouvement, créant ainsi l'impression d'être devant un miroir ou une fenêtre sur un autre monde. Les surfaces 3D sont « perméables » et la gravité est absente : ceci ajoute à l'étrangeté de cet univers tout en trahissant la facture de l'illusion. L'interaction et la spatialisation du son, générées d'après des prises de son réelles retravaillées, sont développées dans Pure Data. L'image est calculée en temps réel avec le moteur de jeux de Blender.

Réalisé depuis 2008 par Ogeem / Olivier Meunier, Stephanie Laforce (Belgique) : http://ogeem.be

LA FAIM JUSTIFIE LES MOYENS

Sculpture interactive sur la thématique de la faim.

En s'inspirant des émeutes de la faim, les créateurs de cette oeuvre interactive veulent susciter une réflexion autour de la pauvreté en Afrique. L'installation en bois est constituée d'une accumulation de manches de balais, de cuillères, de fer, etc. En ouvrant une boîte de conserve dans laquelle est installé un capteur de luminosité, le visiteur enclenche le défilement d'une série d'images représentant des scènes de famine tout en faisant entendre des sons de cuillères qui s'entrechoquent.

Réalisé en novembre 2010-janvier 2011 par Charles Seck, Roland Kossigan Assilevi, Samba Tounkara - Kër Thiossane (Sénégal): http://sicap1695.phpnet.org/spip.php?article57

VEDERSI #1

Musique mixte pour une peinture, 2 surfaces vitrées, Pure Data et 2 capteurs piezzo.

Aux frontières entre l'installation sonore et visuelle, l'*action painting* et la musique mixte, Vedersi#1 met en acte la transdisciplinarité en transformant le geste pictural en un flux sonore et musical. L'artiste peint sur des surfaces verticales en verre sur lesquelles sont positionnées des capteurs audio de type piezzo. Les phénomènes sonores ainsi provoqués sont transmis et modifiés immédiatement grâce à Pure Data. Ce travail de composition éphémère tend à développer une forme musicale à partir d'un geste non instrumental exécuté et filtré par traitement du signal audio numérique en temps réel.

Réalisé par Sitan adèle K / Tad Bisaha au festival Dix'Art 2011 :
http://vedersi.blogspot.com

PROJETS PÉDAGOGIQUES AVEC PURE DATA

Grâce à son système de programmation intuitif, Pure Data favorise l'apprentissage et l'expérimentation. De nombreux artistes et pédagogues l'utilisent pour leurs cours, formations et ateliers, ou encore conçoivent à l'aide de ce logiciel des dispositifs numériques originaux d'enseignement.

LA MALINETTE

Vue de la malinette en 3d.

La Malinette est un projet réalisé avec le logiciel Pure Data dont l'objectif est de faciliter la programmation interactive. Elle regroupe un ensemble logiciel et matériel constitué de patchs et d'interfaces électroniques permettant de brancher rapidement des capteurs et actionneurs physiques (ex: Arduino, capteurs électroniques, vidéo, son, moteurs, 3d, OSC, etc.). C'est un outil de prototypage rapide d'installation interactive car elle regroupe de nombreuses abstractions souvent utiles, classé dans une fenêtre principale qui sert de barre d'outils.

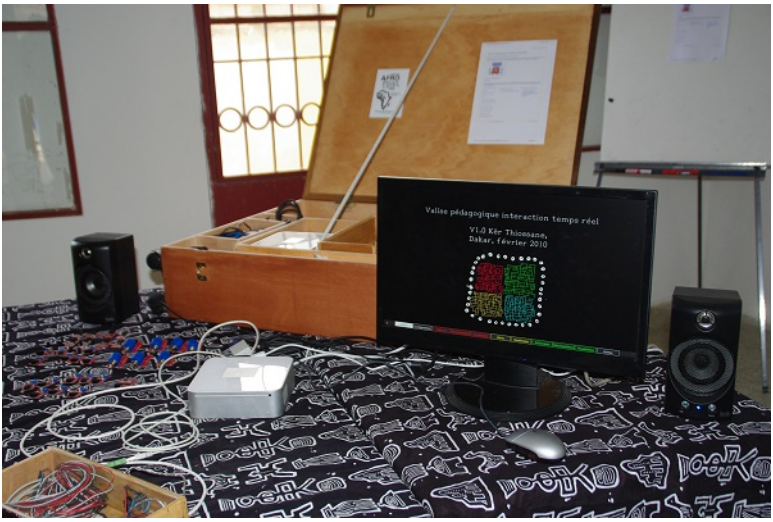
La Malinette est conçue pour :

- **des démonstrations rapides ;**
- **des initiations à Pure Data et Arduino à la création interactive ;**
- **des expérimentations ludiques dans les classes en utilisant des capteurs pour des travaux pratiques ludiques ;**
- **des créations avec la possibilité d'ajouter ses propres objets.**

Projet développé par l'association Reso-nance Numérique à Marseille.

- Site de présentation : http://reso-nance.org/malinette
- Depot Git : https://gitorious.org/malinette/malinette/trees/master
- Téléchargement : http://reso-nance.org/malinette/download/

VALISE PÉDAGOGIQUE CRÉATION INTERACTIVE



Présentation de la valise pédagogique à Kër Thiossane, Villa des arts et du multimédia.

Réalisé dans le cadre d'un projet d'essaimage de pratiques artistiques numériques en Afrique de l'Ouest et dans les Caraïbes, la Valise Pédagogique Création Interactive est un ensemble matériel, logiciel et documentaire pour l'apprentissage des technologies d'Interaction Temps Réel dans la création contemporaine, tous champs artistiques confondus (arts plastiques, danse, théâtre, musique, architecture, design, etc.).

Équipée de Pure Data, la valise peut servir aussi bien de plateforme d'apprentissage dans le cadre d'un atelier de découverte de l'interaction en art que d'outil de création pour artiste en permettant d'inventer, de simuler puis de réaliser des milliers de dispositifs interactifs différents.

Réalisé en 2010-2011 par Jean-Noël Montagné (artiste plasticien), Jérôme Abel (artiste développeur) et les électroniciens africains de ENDA Ecopole en partenariat avec Kër Thiossane (Sénégal) et le CRAS (France) :

- Kër Thiossane : http://sicap1695.phpnet.org/spip.php?article6
- Code de la valise : https://gitorious.org/valise-pedagogique

MIAM - MALETTE INTERACTIVE ARTISTIQUE MULTIMEDIA

Illustration des périphériques et dispositifs contenus dans la Malette Interactive Artistique Multimédia

Destinée à être **tant un outil pédagogique qu'un instrument/système à vocation artistique**, la **Malette Interactive Artistiques Multimédia** (MIAM) est constituée d'un ordinateur équipé des périphériques les plus couramment utilisés dans les dispositifs et instruments interactifs (capteurs divers, webcam, joystick, wiimote, carte arduino, etc.). Elle offre aux enseignants et formateurs de nombreuses ressources numériques et multimédias « prêtes à l'emploi » destinées à un large public pour un décryptage et une approche de l'histoire de l'art numérique et interactif de façon didactique et illustrée.

Le projet de Malette Interactive Artistique Multimédia a reçu le soutien financier du ministère français de la Culture et de la Communication.

Réalisé en 2010-2011 par les associations Labomedia et Ping (France) en collaboration avec la Fabrique du Libre : http://lamiam.fr/

Les images des réalisations présentées dans ce chapitre sont la propriété de leurs auteurs respectifs.

4. PROGRAMMATION GRAPHIQUE

La programmation graphique permet d'organiser des applications avec des icônes reliées entre elles par des cordes sans faire appel à des lignes de texte, contrairement aux environnements de programmation traditionnels que sont par exemple le C ou le Pascal.

PROGRAMMATION TEXTUELLE

Les programmes informatiques sont le plus souvent écrits sous forme de texte. Ils sont enregistrés dans des fichiers que vous devez ensuite compiler pour rendre le programme exécutable. Il s'agit de « programmation textuelle ».

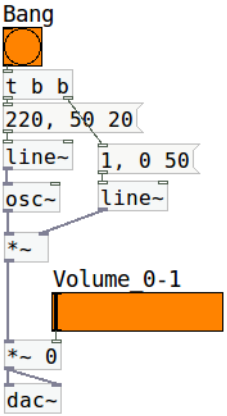
```
<CsoundSynthesizer>;
<CsOptions>
csound -W -d -o tone.wav
</CsOptions>
<CsInstruments>
sr     = 44100           ; Sample rate.
kr     = 4410            ; Control signal rate.
ksmps  = 10              ; Samples pr. control signal.
nchnls = 1               ; Number of output channels.

instr 1
a1     oscil p4, p5, 1   ; Simple oscillator.
           out a1            ; Output.
    endin
</CsInstruments>
<CsScore>
    f1 0 8192 10 1           ; Table containing a sine wave.
    i1 0 1 20000 1000        ; Play one second of one kHz tone.
    e
</CsScore>

</CsoundSynthesizer>
```

Ci-dessus, un exemple de programme texte :
http://fr.wikipedia.org/wiki/Csound

Maîtriser les concepts et les langages de ces logiciels demande un certain temps d'étude et de pratique. Pour ces raisons, aux yeux des non-programmeurs, ces méthodes d'écriture et de compilation paraissent souvent peu intuitives.

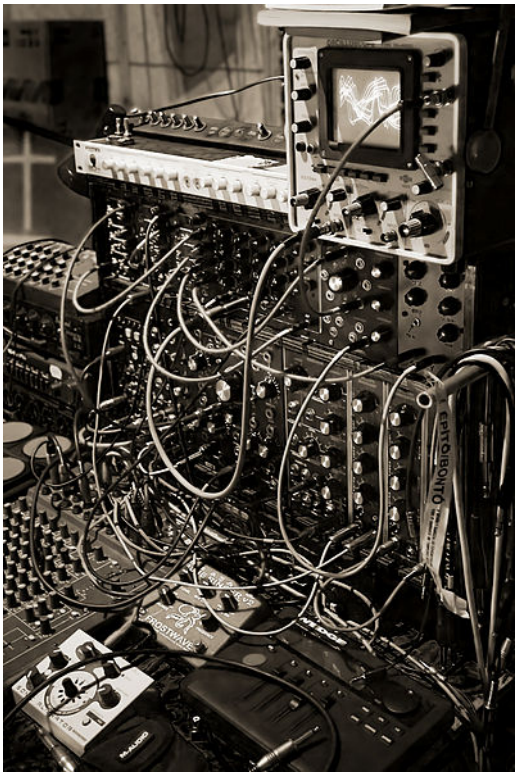
PROGRAMMATION GRAPHIQUE

Pure Data propose un environnement de programmation graphique. Au lieu d'écrire des lignes de texte, vous ajoutez et liez des éléments graphiques : les « boîtes ». Celles-ci sont la représentation graphique de fonctions qui, quant à elles, sont écrites sous forme de texte et compilées. L'action de ces boîtes est transparente pour l'utilisateur, il n'a pas à se soucier des codages induits. Manipuler ces objets à la souris et les déplacer dans l'espace de la page demeure plus accessible et ludique que d'écrire des pages de code. En outre, l'aisance d'exécution de Pure Data et la souplesse de son interface permettent à chacun de s'initier à l'univers de la programmation.



PATCH

Un programme dans Pure Data s'appelle un « patch » c'est-à-dire un ensemble de boîtes connectées par des fils. Ce terme vient des synthétiseurs modulaires. Les câbles audio y relient des modules sonores et créent un patch. Ce terme fut aussi utilisé pour désigner un programme à l'époque des premiers gros ordinateurs comme l'ENIAC.


QUINCITO
FREESTYLE

INSTALLATION

5. PD-EXTENDED VS PD-VANILLA
6. INSTALLATION SUR GNU/LINUX
7. INSTALLATION SUR MAC OS X
8. INSTALLATION SUR WINDOWS

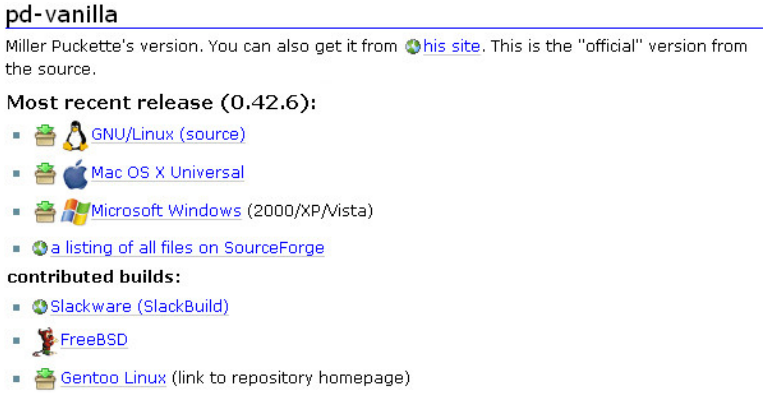
5. PD-EXTENDED VS PD-VANILLA

Pure Data se décline en deux saveurs, deux versions : Pure Data vanilla et Pure Data Extended. Vous serez amené à utiliser l'une ou l'autre en fonction de vos besoins et de votre expérience.

PD-VANILLA

La version de base de Pure Data développée par Miller Puckette s'appelle **pd-vanilla** (Pure Data vanilla). Elle permet de manipuler de l'audio et du MIDI. Vous trouverez sur le site officiel (www.puredata.info), les versions les plus récentes au téléchargement.

Comme on peut le constater sur cette copie d'écran, Pd-vanilla est disponible sur plusieurs systèmes d'exploitation (GNU/Linux, Mac OS X et Microsoft Windows) :



PD-EXTENDED

La version **Pd-extended** (Pure Data Extended) regroupe Pd-vanilla et de nombreuses librairies créées par une communauté de développeurs. Ces librairies étendent les fonctions de Pd-vanilla pour pouvoir, par exemple, traiter de la vidéo, communiquer avec des périphériques USB et FIREWIRE ou réaliser des opérations complexes en une seule fois, etc. Les versions récentes de Pure Data Extended se téléchargent également depuis le site officiel du projet.

Pure Data Extended est souvent représenté par le logo de Pure Data connecté à un bouton.

Tout comme Pd-vanilla, Pd-extended est disponible sur plusieurs systèmes d'exploitation :

Debian GNU/Linux

- stable lenny (i386)
- oldstable etch (i386)
- testing squeeze (i386)

Apple Mac OS X (10.4/Tiger or newer)

- Mac OS X Intel (Mac Pro, MacBook, all Intel Macs)
- Mac OS X PowerPC (PowerMac, PowerBook, iMac, with G4 or G5)

Ubuntu GNU/Linux

- Maverick 10.10 (i386 32-bit)
- Lucid 10.04 (i386 32-bit)
- Lucid 10.04 (amd64 64-bit)
- Karmic 9.10 (i386 32-bit)
- Karmic 9.10 (Netbook Remix lpia)
- Hardy 8.04 (i386 32-bit)
- Hardy 8.04 (Netbook Remix lpia)

Microsoft Windows 2000/XP/2003/Vista/Windows 7 (32-bit)

- Microsoft Windows (2000/XP/Vista/7)

QUELLE SAVEUR PRIVILÉGIER ?

Il est recommandé aux utilisateurs qui débutent avec Pure Data d'installer Pd-extended. Cette version demeure la plus conviviale pour un débutant tout en offrant un maximum de possibilités (pour un même plaisir). Mais de nombreux utilisateurs plus expérimentés choisissent Pd-vanilla pour la simplicité de son noyau, qui requiert peu de dépendances. Cette option implique cependant d'installer soi-même les extensions nécessaires à ses besoins.

SYSTÈMES D'EXPLOITATION

Pd-vanilla comme Pd-extended existent pour les trois principaux systèmes d'exploitation : **GNU/Linux, Mac OS X et Microsoft Windows**. Récemment portées vers les plates-formes **iOS, Android et Maemo**, ces applications se déclinent aussi sur le Web via le projet WebPd.

Les développeurs Pure Data privilégient le système GNU/Linux mais exportent souvent leurs travaux pour les autres systèmes d'exploitation, parfois en y ajoutant des optimisations spécifiques à chaque environnement. À noter aussi que certains objets n'existent que sur une des plates-formes, tandis que d'autres reposent sur les librairies de votre système d'exploitation. Par exemple, nous pouvons citer des optimisations de la librairie graphique GEM pour Mac OS X, l'objet [pix_video] de GEM décliné en [pix_videoDS] pour Windows, la défunte librairie vidéo Framestein pour Windows, etc.

Quelle que soit votre plateforme, Pure Data préservera sa convivialité et vous constaterez de meilleures performances si votre système est bien en ordre.

6. INSTALLATION SUR GNU/LINUX

Voyons maintenant comment installer Pure Data sur un système libre.

PD-EXTENDED

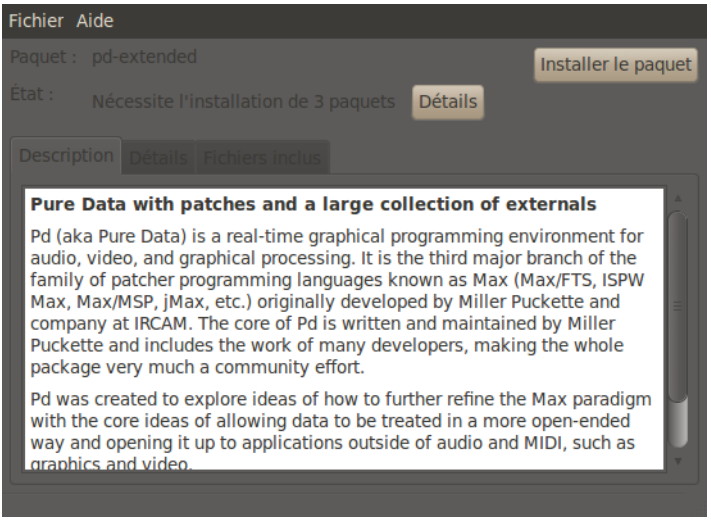
- **Nom du logiciel :** Pure Data Extended
- **Page de téléchargement :** http://puredata.info/downloads
- **Version du logiciel utilisée ici :** Pd-extended 0.42-5
- **Système d'exploitation utilisé ici :** Ubuntu 10.04 Lucid Lynx (similaire pour les autres versions)
- **Matériel minimum recommandé :** Processeur à 300 Mhz, 128 Mo RAM
- **Matériel pour un usage audio/vidéo intense :** Processeur à 2 Ghz, 1 Go RAM, carte graphique supportant bien l'OpenGL (NVIDIA, ATI, etc.)

Télécharger et installer un paquet prêt à l'emploi

Le logiciel Pure Data est disponible sous forme de « paquets » pour les systèmes d'exploitation Debian et Ubuntu : cela signifie que l'on peut l'installer en quelques clics.

Pour d'autres systèmes d'exploitation GNU/Linux ou pour des besoins plus spécifiques, il est possible de « compiler » Pure Data, c'est-à-dire de créer l'exécutable à partir des codes sources du logiciel : cette opération est plutôt réservée aux spécialistes et ne sera pas traitée dans ce chapitre.

Il existe plusieurs versions de Pd-extended en fonction du processeur de votre ordinateur (32 bits, 64 bits), et une version optimisée pour certains netbooks.



1. Quand le paquet adapté à votre processeur est téléchargé, faites un clic droit sur celui-ci et choisissez « Ouvrir avec installateur de paquets GDebi ». (avec la logithèque Ubuntu à partir de la version 11.04)
2. L'installateur de paquet s'ouvre et vous indique s'il est nécessaire de charger des librairies complémentaires : si c'est le cas, vous DEVEZ avoir une connexion internet pour terminer l'installation.
3. Cliquez sur « Installer le paquet ».
4. Tout se passe bien, l'installation se termine, vous avez désormais une nouvelle icône dans votre menu : *Applications > Programmation > Pure Data*

Méthode alternative : ajouter un dépôt pour installer Pd-extended

Plutôt qu'une installation de Pd-extended par téléchargement d'un paquet, la méthode d'installation par le biais d'un dépôt permet des mises à jour simplifiées. L'installation est légèrement plus longue, car elle nécessite de rajouter le dépôt contenant Pd-extended à son système d'exploitation. On peut procéder en ligne de commande ou via le gestionnaire de paquets.

Pour connaître l'adresse du dépôt correspondant à votre système, consultez cette page du site pure-data.info

Synaptic est un gestionnaire de paquets en mode graphique couramment utilisé. Pour y accéder, vous devez identifier votre distribution Linux :

- Menu *Applications > Système pour Xubuntu*
- Menu *Système > Administration pour Ubuntu*

Pour aller plus loin : http://doc.ubuntu-fr.org/synaptic

PD-VANILLA

Pd-vanilla est présent dans les dépôts Debian et Ubuntu pour une installation plus simple :

- soit **avec le gestionnaire de paquets, cherchez "Puredata"** (sans avoir ajouté de dépôt supplémentaire) et faire un clic droit > « Sélectionner pour installation »
- soit **en ligne de commande**

```
sudo apt-get install puredata
```

Enfantin ! Si par la suite vous avez besoin d'extensions spécifiques, vous les trouverez peut-être dans les dépôts d'Ubuntu. La plupart de ces paquets ont un nom qui commence par "pd-", sauf le paquet "gem". Pour trouver ces paquets, taper dans un terminal :

```
apt-cache search pd- | grep ^pd-
```

DISTRIBUTIONS GNU/LINUX ORIENTÉES MULTIMÉDIA

Il vous est possible d'installer une distribution plus orientée pour le multimédia (voir http://www.linux-sound.org/distro.html). C'est-à-dire une distribution qui comprend les logiciels multimédias pré-installés, un noyau Linux temps réel, etc. En général, vous pouvez en choisir une avec la version « live » et la version normale installable. La version « live » vous permet d'essayer la distribution sans l'installer. Vous pouvez ainsi tester la compatibilité de votre matériel.

Notez que ces distributions sont toujours en cours de développement (comme tout le monde !) et donc plus ou moins en retard par rapport aux distributions qui servent de base (Ubuntu, Debian, etc.).

Pour certaines distributions, vous pouvez garder votre version actuelle et n'installer que les paquets multimédias et le noyau temps réel. Par exemple, pour Ubuntu, vous pouvez garder votre version et installer les paquets : ubuntustudio-desktop, linux-rt, ubuntustudio-graphics, ubuntustudio-video, ubuntustudio-audio, ubuntustudio-audio-plugins (voir http://doc.ubuntu-fr.org/ubuntu_studio)

Quelques distributions : Puredyne, Ubuntu Studio, Apodio, Dyne:bolic, Debian Multimedia, TangoStudio, ArtistX.

De plus, le site francophone http://www.linuxmao.org propose des articles sur l'optimisation de certaines distributions et l'utilisation de nombreux logiciels audio.

7. INSTALLATION SUR MAC OS X

Nous allons à présent voir comment installer Pure Data version **Pd-extended** sur Mac OS. Elle comprend de nombreuses extensions et fonctionne bien sous Mac OS X. Pour commencer la programmation avec Pd, cette « mouture » permet d'en apprécier rapidement les possibilités,

La page de téléchargement : http://puredata.info/downloads

PD-EXTENDED

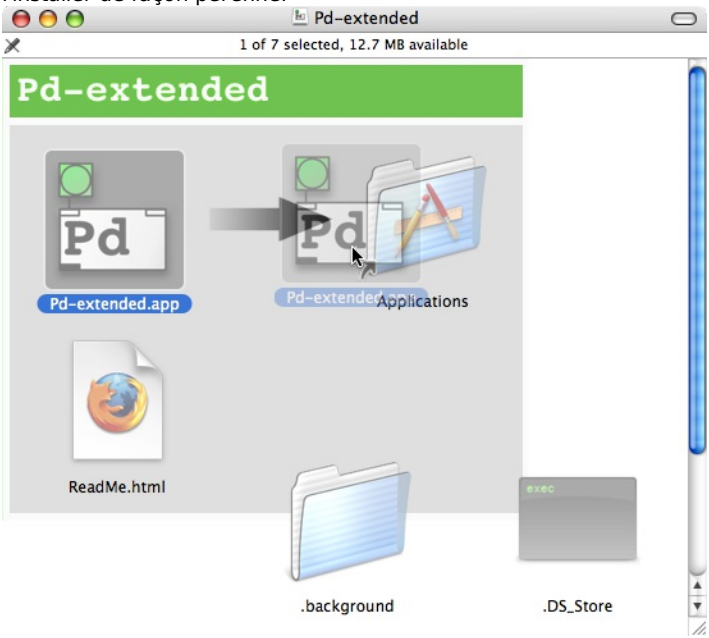
- **Nom du logiciel :** Pure Data Extended
- **Page de téléchargement :** http://sourceforge.net/projects/pure-data/files/pd-extended/0.42.5/Pd-0.42.5-extended-macosx104-i386.dmg/download
- **Version du logiciel utilisée ici :** Pd-extended 0.42-5
- **Système d'exploitation utilisé ici :** Mac OS 10.4 ou 10.5
- **Matériel minimum recommandé :** Processeur à 300 Mhz, 128 Mo RAM
- **Matériel pour un usage audio/vidéo intense :** Processeur à 2 Ghz, 1 Go RAM, carte graphique supportant bien OpenGL (NVIDIA, ATI, etc.)

Télécharger et installer un paquet prêt à l'emploi

Le logiciel Pure Data est disponible sous forme d'un paquet prêt à installer.

Il existe plusieurs versions de Pd-extended en fonction du processeur de votre ordinateur : Intel (Mac Pro, MacBook, tous Macs Intel) ou PowerPC (PowerMac, PowerBook, iMac, avec G4 ou G5)(32 bits, 64 bits).

1. Une fois le paquet adapté à votre processeur téléchargé, double-cliquez sur celui-ci, « **lire et accepter la Licence d'utilisation** » apparaît.
2. Cliquez sur **"Agree"** :l'image du disque se monte et s'ouvre automatiquement. Il suffit alors de déplacer Pd-extended.app dans le dossier "Applications" (ou dans tout autre dossier de votre choix) pour copier Pd-extended sur votre disque dur et l'installer de façon pérenne.



3. Une fois Pd-extended copié sur votre disque, vous pouvez éventuellement lire le fichier "**ReadMe**" (en anglais) pour un complément d'information sur l'installation.

Pour lancer le logiciel, cliquez désormais sur l'icône présente dans votre dossier « **Applications** ». Pendant que le logiciel se lance, vérifiez dans la fenêtre qui s'ouvre que les librairies se chargent correctement. Si un message d'erreur comme celui reproduit ci-dessous s'affiche, il vous faudra en plus installer X11.

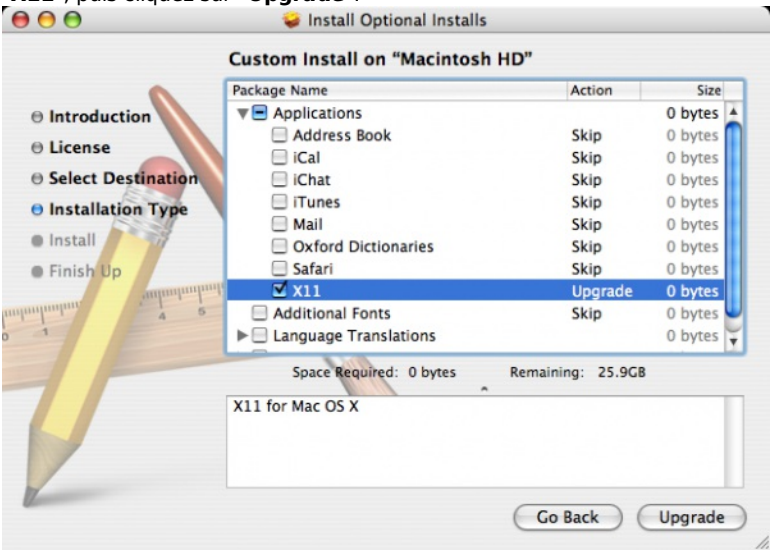
```
libdir_loader: added 'flatspace' to the global objectclass path
/Applications/Pd-extended.app/Contents/Resources/extra/pdp.pd_darwi
n: dlopen(/Applications/Pd-extended.app/Contents/Resources/extra/pd
p.pd_darwin, 10): Library not loaded: /usr/X11R6/lib/libX11.6.dylib
  Referenced from: /Applications/Pd-extended.app/Contents/Resources
/extra/pdp.pd_darwin
  Reason: image not found
pdp: can't load library
/Applications/Pd-extended.app/Contents/Resources/extra/pidip.pd_dar
win: dlopen(/Applications/Pd-extended.app/Contents/Resources/extra/
pidip.pd_darwin, 10): Library not loaded: /usr/X11R6/lib/libX11.6.d
ylib
  Referenced from: /Applications/Pd-extended.app/Contents/Resources
/extra/pidip.pd_darwin
  Reason: image not found
```

Installation complémentaire de la librairie X11 pour Mac OS X 10.3 Panther et 10.4 Tiger

Cette installation n'est plus nécessaire à partir de Mac OS 10.5 puisque la librairie X11 y est installée par défaut.

La librairie dédiée au traitement vidéo PDP nécessite l'installation de la librairie X11 contenue dans le DVD d'installation de Mac OS et ce uniquement pour Mac OS 10.3 et 10.4. Si vous n'avez plus ce DVD, vous pourrez le trouver en ligne. La Foire aux questions concernant Mac OS est ici : http://puredata.info/docs/faq/macosx

1. Insérez votre disque OS X Tiger (#1). Descendez dans la fenêtre pour localiser l'icône "**Optional Installs**" et double-cliquez sur cette icône.
2. Cliquez sur « **Continuer** » au premier écran.
3. Vous pouvez lire la licence d'utilisation, puis cliquez sur « **Continuer** ».
4. Cliquez sur **« Accepter** ».
5. Choisissez un emplacement et cliquez sur « **Continuer** ».
6. Déroulez la liste des applications, cochez la case correspondante "**X11**", puis cliquez sur "**Upgrade**".



7. Entrez votre mot de passe et cliquez sur « **OK** ».
8. Attendez jusqu'à la fin de l'installation.
9. Quand la librairie X11 est installée correctement, cliquez sur « **Fermer** ».

8. INSTALLATION SUR WINDOWS

Voyons maintenant comment installer Pure Data sur Windows. La page de téléchargement des différentes versions de Pure Data est http://puredata.info/downloads.

PD EXTENDED

- **Nom du logiciel :** Pure Data Extended
- **Page de téléchargement :** http://sourceforge.net/projects/pure-data/files/pd-extended/0.42.5/Pd-0.42.5-extended-windowsxp-i386.exe/download
- **Version du logiciel utilisée ici :** Pd-Extended 0.42.5
- **Système d'exploitation utilisé ici :** Windows XP
- **Matériel minimum recommandé :** Processeur à 300 Mhz, 128 Mo RAM
- **Matériel pour un usage audio/vidéo intense :** Processeur à 2 Ghz, 1 Go RAM, Carte graphique supportant bien l'OpenGL (Nvidia, ATI...)

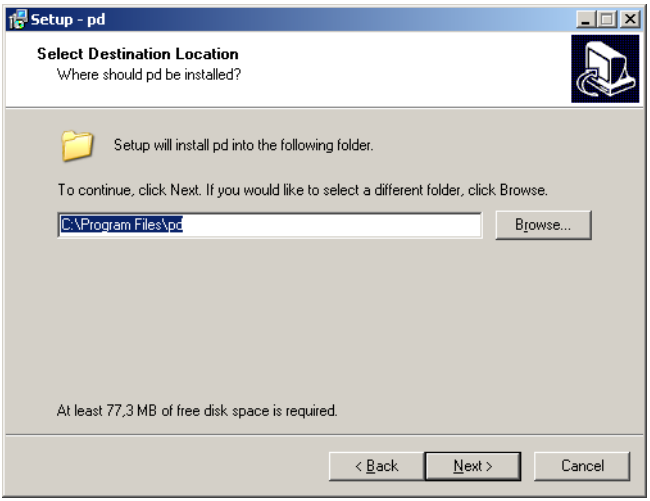
Télécharger et installer le logiciel

Le logiciel Pure Data Extended est disponible pour Microsoft Windows 2000/XP/2003/Vista/Windows 7 (32-bit)

1. Une fois le fichier Pd-0.42.5-extended-windowsxp-i386.exe téléchargé, double-cliquez sur celui-ci

2. Vous pouvez lire la Licence d'utilisation, puis cliquez sur « Accepter ».

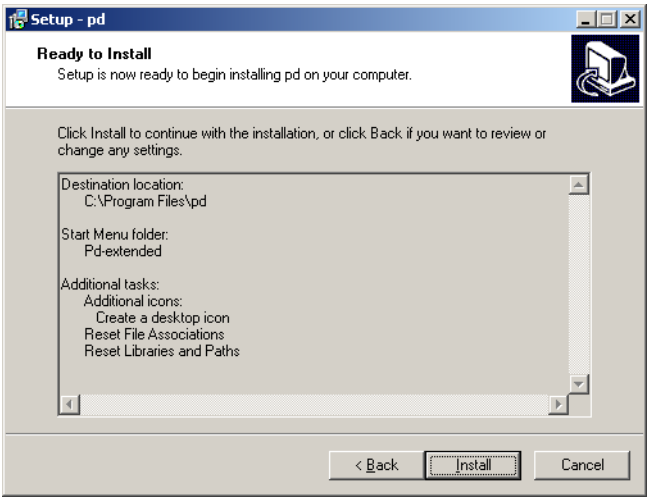
3. Choisissez le dossier d'installation.



4. Créez le raccourci dans le menu **Démarrer**.

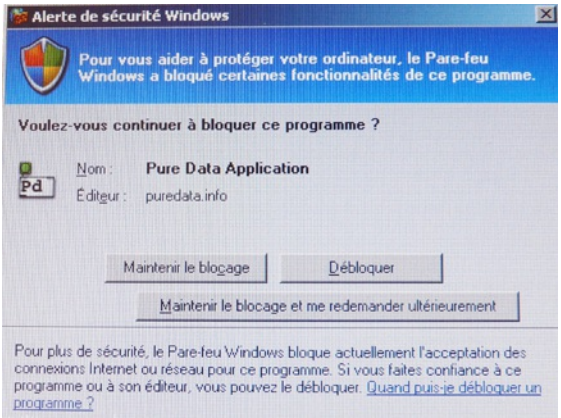
5. Cliquez sur "Create a desktop icon" pour créer un raccourci sur le bureau.

6. Lancez l'installation.



7. Lancer Pure Data Extended en cliquant deux fois sur l'un des raccourcis. **Une alerte de sécurité Windows apparait : cliquez sur Débloquer.**

Pd étant construit autour d'un client et d'un serveur qui communiquent entre eux au sein de l'ordinateur, il faut indiquer au pare-feu de Windows qu'il doit débloquer cette communication.



PD VANILLA

Nom du logiciel : Pure Data Vanilla

- **Page de téléchargement :** http://sourceforge.net/projects/pure-data/files/pure-data/0.43.0/pd-0.43-0.msw.zip/download
- **Version du logiciel utilisée ici :** Pure Data 0.43
- **Système d'exploitation utilisé ici :** Windows XP
- **Matériel minimum recommandé :** Processeur à 300 Mhz, 128 Mo RAM
- **Matériel pour un usage audio/vidéo intense :** Processeur à 2 Ghz, 1 Go RAM, Carte graphique supportant bien l'OpenGL (Nvidia, ATI...)

Télécharger et installer le logiciel

Le logiciel Pure Data Vanilla est disponible pour Microsoft Windows 2000/XP/2003/Vista/Windows 7 (32-bit)

1. Une fois le fichier pd-0.43-0.msw.zip téléchargé, l'extraire par exemple dans c:\Program Files ou tout autre répertoire de votre choix.

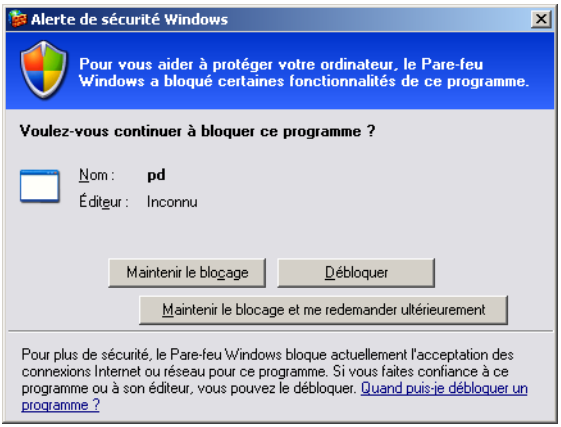
2. Un dossier « **Pd** » est automatiquement créé. Allez dans le répertoire "bin" et cliquez sur le fichier "pd.exe", faites un clic droit et choisissez *Envoyer vers > Bureau (Créer un raccourci).*

12b.png

3. Cela permet de lancer Pd depuis le bureau.

4. Double-cliquez sur le raccourci pour lancer Pure Data Vanilla.

5. Il faut cliquer sur **Débloquer** pour laisser le client et le serveur de Pd discuter ensemble.


Alerte de sécurité Windows
Pour vous aider à protéger votre ordinateur, le Pare-feu Windows a bloqué certaines fonctionnalités de ce programme.
Voulez-vous continuer à bloquer ce programme ?
Nom : pd
Éditeur : Inconnu
Maintenir le blocage
Débloquer
Maintenir le blocage et me redemander ultérieurement
Pour plus de sécurité, le Pare-feu Windows bloque actuellement l'acceptation des connexions Internet ou réseau pour ce programme. Si vous faites confiance à ce programme ou à son éditeur, vous pouvez le débloquer. Quand puis-je débloquer un programme ?

CONFIGURATION

9. CONFIGURATIONS AUDIO ET MIDI
10. CONFIGURATION : CHEMINS ET LIBRAIRIES

9. CONFIGURATIONS AUDIO ET MIDI

Chaque système informatique est spécifique, tout comme chaque utilisateur est différent. Si vous souhaitez utiliser Pure Data pour gérer du signal audio, il vous faudra préalablement configurer vos appareils audio. Si vous décidez de recevoir ou d'envoyer des notes et des contrôles MIDI, il sera également nécessaire de configurer vos appareils MIDI.

TESTEZ SI LE SON FONCTIONNE

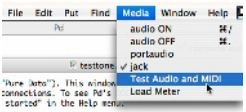
Tout d'abord, Pure Data a peut-être détecté automatiquement votre interface audio ! Pour vérifier si celle-ci fonctionne, choisissez l'item "Test Audio and MIDI" dans le menu "Media" de Pure Data et le patch "testtone.pd" apparaît.

Repérez les colonnes de carrés à gauche du patch sous le titre "TEST TONES". Ces carrés ou cases sont des choix multiples dont les options sont : "OFF", "60" ou "80". Pour entendre un joli La à 440 Hz, cliquez sur la case correspondant au choix "60" ou "80" et un carré noir apparaît dans la case que vous avez sélectionnée. Si vous entendez un son, cela signifie que Pure Data a détecté votre interface audio. Dans le cas contraire, il va vous falloir la configurer. Nous verrons comment faire par la suite. On peut changer le son du test par un bruit en choisissant l'option "NOISE" plutôt que "TONE" en cliquant dans la case correspondante.

CHOIX DU PILOTE AUDIO

Pour capturer et lire un son, Pure Data prend en charge de nombreux pilotes qui dépendent de votre système d'exploitation. Les pilotes servent à supporter des appareils audio. Ceux-ci sont listés dans le menu "Media".

Sur GNU/Linux, les pilotes supportés sont : OSS, ALSA ou JACK ; sur Mac OS X : portaudio ou JACK ; sur Windows : ASIO (via portaudio et qui est le seul supporté). Pour Windows, il est possible d'installer Asio4all (http://www.asio4all.com/).



Pour changer de pilote, choisissez l'un d'entre eux dans le menu "Media". Sur GNU/Linux, ALSA devrait fonctionner assez facilement. JACK, quant à lui, est essentiel si vous souhaitez utiliser une interface audio Firewire ou si vous désirez connecter l'audio de Pure Data à d'autres logiciels. Dans ce cas, vous pourrez être amenés à utiliser l'application QJackctl (qjackctl en ligne de commande) pour gérer JACK et ses connexions.

Une fois le pilote choisi, vous pouvez ajuster avec finesse votre configuration audio.

CONFIGURATION AUDIO

Pure Data fournit une fenêtre de dialogue qui permet de configurer le nombre d'entrées et de sorties audio du logiciel ainsi que son taux d'échantillonnage. Vous n'entendrez rien si les options de ce dialogue sont mal configurées.

Dans Pure Data, l'item "Autio settings..." se trouve dans le menu "Media" sous GNU/Linux et Windows et dans le menu "Preferences" sous Mac OS X.

Cliquez sur "Save all settings" avant de fermer cette boîte de dialogue si vous souhaitez appliquer les changements effectués.

Le taux d'échantillonnage

Ce chiffre peut prendre l'une des valeurs supportées par votre pilote et votre interface audio. Par défaut, il est de 44 100 Hz, ce qui correspond au taux d'échantillonnage des disques compacts. Une autre valeur couramment utilisée ici est 48 000 Hz.

Délai

Pure Data a besoin d'un peu de temps pour temporiser le signal audio qu'il reçoit et pour effectuer tous les calculs que vous lui soumettez avant de vous faire entendre des sons. À intervalle régulier, Pure Data enregistre le son dans un espace mémoire, puis il applique toutes les opérations sur le son et, enfin, donne le résultat de ces calculs à l'interface audio. La durée de cet intervalle est spécifiée dans cette boîte de dialogue (*delay*). Par défaut, il est de 50 millisecondes.

Les musiciens préfèrent quand ce nombre est petit. Toutefois, si vous le réduisez trop, Pure Data n'a pas le temps de tout calculer et vous entendrez des craquements indésirables. Pour trouver une valeur optimale, vous pouvez tenter de réduire ce délai progressivement jusqu'à entendre ces craquements. Vous pourrez ensuite remonter sa valeur jusqu'à ce que vous n'entendiez plus de parasites.

Si vous utilisez JACK, assurez-vous d'utiliser des blocs audio de la même taille que ceux de JACK. Pour cela, il vous suffit de créer un objet [block~] avec comme argument un nombre magique. Sous GNU/Linux, on obtient ce nombre en consultant sa configuration "QJackctl" (ou l'équivalent). Le nombre que l'on trouve sous "Frames/Period" est alors à multiplier par celui qui apparaît sous "Periods/Buffer".

Activation et nombres d'entrées et de sorties

Vous pouvez activer ou non les entrées et sorties de votre interface audio. Si elles sont inactives, sélectionnez-les. La plupart du temps, vous pouvez choisir la même interface pour les entrées et les sorties, avec du son mono ou stéréo. Si ce n'est pas le cas, il vous faudra choisir deux canaux en entrée et deux autres en sortie.

CHOIX DU PILOTE MIDI

Pour recevoir des notes et des contrôles MIDI de quelque instrument MIDI que ce soit, il faut d'abord choisir un pilote. Les pilotes audio sont listés dans le menu "Media" avec leur appellation. Sur GNU/Linux, vous pouvez choisir "Default" ou "ALSA". Le second devrait bien fonctionner.

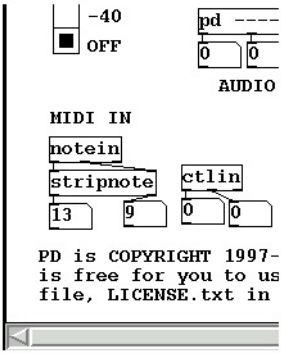
CONFIGURATION MIDI

Le dialogue de configuration MIDI est accessible par l'item "MIDI settings..." du menu "Media" ou "Preferences" sous Mac OS X. Cette boîte de dialogue vous permet de choisir quel appareil MIDI vous désirez utiliser ainsi que le nombre d'entrées et de sorties MIDI. Par exemple, si vous souhaitez brancher un clavier MIDI dans Pure Data, indiquez le chiffre 1 dans le nombre d'entrées MIDI.

Remarque : sur GNU/Linux, vous pouvez router le MIDI entre les applications via JACK.

TESTER LE MIDI

Le patch qui sert à tester l'audio sert également à tester le MIDI. Choisissez l'item "Test Audio and MIDI" du menu "Media". Si tout est bien configuré, les nombres affichés dans ce patch devraient changer lorsque vous appuierez sur les touches de votre clavier MIDI.



Vous pouvez également envoyer des notes MIDI à un appareil ou à un autre logiciel branché à la sortie MIDI de Pure Data en cliquant sur la case de l'interrupteur qui a pour titre "MIDI OUT". Une croix apparaît dans cette case signalant la connexion effective.

10. CONFIGURATION : CHEMINS ET LIBRAIRIES

Pure Data est un programme modulaire qui, au démarrage, charge les objets de base, puis des extensions (librairies d'objets externes et librairies d'abstractions). Mais pour pouvoir charger ces modules, Pure Data a besoin de connaître les chemins précis vers ces fichiers.

FICHIERS DE CONFIGURATION

La configuration de Pure Data est modifiable via son interface graphique et est enregistrée dans un fichier :

- **GNU/Linux** : ~/.pdrc, ~/.pdsetting, ~/.pdextended.
- **Mac OS X** : ~/.pdrc, ~/Library/Preferences/org.puredata.pd.plist
- **MS Windows** : Menu Démarrer > Exécuter une commande > Taper "regedit" : HKEY_LOCAL_MACHINE > SOFTWARE > Pd ou Pd-extended

Note : dans les chemins de fichiers, le signe tilde "~" désigne votre dossier utilisateur.

OPTIONS AU DÉMARRAGE

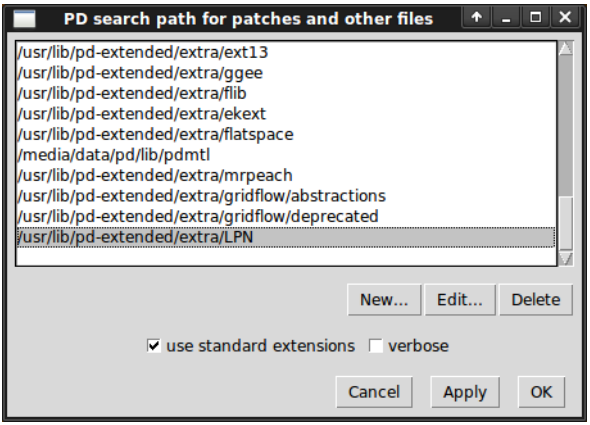
Beaucoup d'options sont modifiables. Par exemple, vous pouvez avoir besoin d'ouvrir un patch au démarrage de Pure Data ou de spécifier votre configuration audio et MIDI. Sous GNU/Linux, on peut démarrer Pure Data en ligne de commande, et spécifier ces options directement à ce moment. En voici un exemple :

```
pd -jack -r 48000 -inchannels 2 -outchannels 2 monpatch.pd
```

Vous pouvez aussi préciser ce genre d'options dans un des fichiers de configuration. Nous verrons comment un peu plus loin.

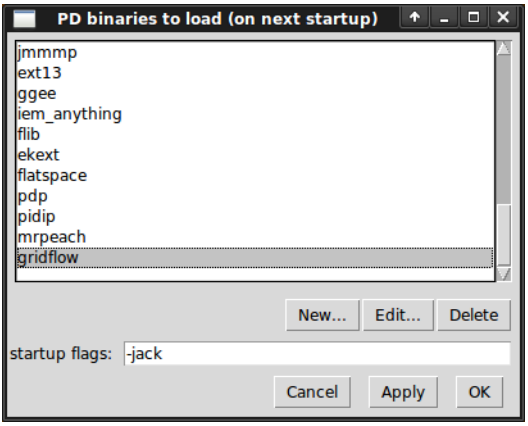
CHARGEMENT DE FICHIERS AU DÉMARRAGE

La boîte de dialogue des chemins de recherche de Pure Data indique au logiciel où chercher les extensions, les abstractions ou les polices de caractère, par exemple. Vous pouvez y ajouter les chemins des dossiers dans lesquels vous placez vos propres extensions. Pour afficher la boite de dialogue, choisir l'item "Path..." du menu "File". Sous Mac OS X, il se trouve sous *Pd-Extended > Preferences > Path...*.



CHARGEMENT DES LIBRAIRIES AU DÉMARRAGE

« Charger les librairies» signifie charger un ensemble d'objets externes. Il faut spécifier ces librairies à Pure Data en indiquant leur nom, qui se situe dans le répertoire : "./extra/". On ouvre cette fenêtre en choisissant l'item "Startup" sous le menu "File" (ou "Pd-Extended > Preferences > Startup" sous Max OS X).



Noter le bloc de texte éditable à droite de "startup flags", qui permet de spécifier vos options de démarrage. Chaque option est précédée d'un tiret "-".

CHARGER DES LIBRAIRIES UNE FOIS EN FONCTION

Charger les librairies seulement lorsqu'on en a besoin plutôt que toutes les charger au démarrage est plus léger pour le programme. L'objet [import] permet de charger une librairie depuis un patch une fois que Pure Data est déjà en marche.

ARBORESCENCE DES FICHIERS INSTALLÉS

Pour utiliser des extensions, il faut connaître leur nom et leur emplacement. Lorsqu'on installe l'application, un répertoire nommé "extra" est créé. Ce répertoire contient plusieurs extensions. Pure Data doit être configuré pour les chercher dans ce dossier lorsque vient le temps de créer un objet. Ce répertoire est situé dans "/usr/lib/pd/extra" ou "/usr/lib/pd-extended/extra". Sous Mac OS X, allez dans "/Applications/Pd-extended.app/Contents/Resources" et faites un clic-droit sur l'icône de l'application pour en afficher le contenu. Sous Windows, les fichiers de Pure Data sont situés sous "C:\Program files\pd".

Au même niveau que ce répertoire "extra" se trouve également un répertoire nommé "doc" qui contient des tonnes de patchs de documentation.

Les extensions pour Pure Data se nomment ".pd_linux" sous GNU/Linux, ".pd_darwin" sous Mac OS X et ".dll" sous Microsoft Windows.

NOTES SUR LES CHEMINS ABSOLUS ET RELATIFS

Pour indiquer à Pure Data un fichier (par exemple sonore ou vidéo) à ouvrir, on peut utiliser un chemin relatif ou un chemin absolu. Un chemin relatif se réfère au répertoire dans lequel se trouve votre patch. Ainsi, si on souhaite ouvrir l'image "bonjour.jpg" qui se trouve dans le même répertoire que notre patch, on doit simplement spécifier "bonjour.jpg". Un chemin absolu, lui, est relatif à la racine du système de fichiers. Par exemple, voici comment indiquer le chemin absolu vers le fichier "bonjour.jpg" qui se trouve sur votre bureau, si votre nom d'utilisateur est « martin » :

Sous GNU/Linux :

```
/home/martin/bonjour.jpg
```

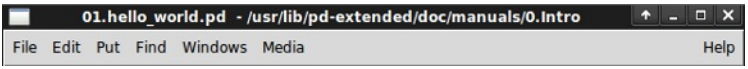
Sous Mac OS X :

```
/Users/martin/bonjour.jpg
```

Sous Microsoft Windows :

```
C:\Documents and Settings\martin\bonjour.jpg
```

Pour connaître le chemin absolu vers votre patch, voir dans la barre du haut de sa fenêtre :



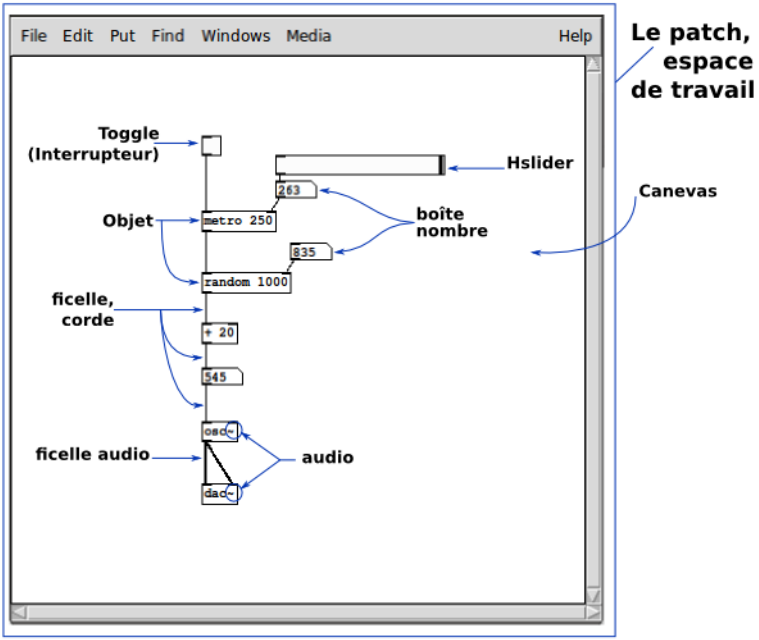
Ici, on peut voir que ce patch est dans le répertoire "/usr/lib/pd-extended/doc/manuals/0.Intro/".

PRISE EN MAIN RAPIDE

11. INTRODUCTION PAR L'EXEMPLE
12. L'INTERFACE UTILISATEUR
13. RÉSOLUTION DE PROBLÈMES
14. LE PATCH
15. LE FLOT DES DONNÉES
16. OBJETS GRAPHIQUES
17. QUELQUES OBJETS UTILES
18. MON PREMIER PATCH
19. ORGANISATION DES OBJETS

11. INTRODUCTION PAR L'EXEMPLE

Nous allons maintenant voir un exemple de patch conçu avec Pure Data. Il s'agit d'un patch qui produira des notes de musique aléatoires de façon ininterrompue.



Dans un patch, l'information s'écoule du haut vers le bas. La sortie de chaque boîte est connectée à l'entrée d'une autre afin d'y diriger de l'information. Celle-ci se transmet par impulsions, comme le sang dans les artères.

Ce patch contient un interrupteur, qui lorsqu'on l'active allume le métronome. Ce métronome bat toutes les 250 millisecondes. Il envoie une impulsion via sa sortie. Notez que la glissière horizontale qui se trouve en haut à droite de ce patch permet de varier la vitesse du métronome, car il est connecté à sa seconde entrée, qui sert à spécifier cet intervalle.

Plus bas, un objet [random] (par convention, "[...]" sert à décrire un objet dans le texte) reçoit les impulsions du métronome. Il produit un nombre au hasard entre 0 et 999 (car il offre mille choix). On peut changer le nombre de choix parmi lesquels il pioche au hasard en lui envoyant un nombre dans sa deuxième entrée.

Ensuite, on additionne 20 au nombre choisi au hasard. Pourquoi? Pour éviter de produire des fréquences trop graves, ce qui risquerait d'endommager vos enceintes audio. Mieux vaut être prudent. Le nombre obtenu est envoyé dans à l'objet [osc~]. Celui-ci attend un nombre qui lui dira à quelle fréquence osciller périodiquement. Cela produit une onde sonore.

Enfin, le signal produit par l'oscillateur [osc~] est envoyé aux deux sorties de votre interface audio. C'est là le rôle de l'objet [dac~] que d'être le point de sortie du son traité ou produit par Pure Data. Notez que les fils qui transportent un signal audio sont plus grasses que celles qui acheminent des impulsions de données.

12. L'INTERFACE UTILISATEUR

L'apprentissage de Pure Data passe tout d'abord par la maîtrise de l'espace de travail du logiciel, appelé aussi interface utilisateur. Actuellement, les messages et menus affichés dans cette interface sont disponibles uniquement en anglais. Malgré cela, son usage reste simple et il est très probable que les versions à venir proposeront une traduction en plusieurs langues, dont le français. Les nombreux atouts de ce logiciel multimédia le rendent assez incontournable pour apprendre à l'utiliser sans plus attendre.

POUR DÉMARRER PURE DATA

Selon votre système d'exploitation Pure Data se lance de différentes façons.

- Sur **GNU/Linux**, nous le trouverons dans le menu *Applications* > *Son et Vidéo* > *PureData* ou dans le menu *Applications* > *Multimédia* > *Pd-extended*
  .

- Sur **Windows**, vous le trouverez dans le menu *Démarrer* > *Programmes* > *Pd-extended*
  .
- Sur **Mac OS X**, double-cliquez sur l'icône présent dans le dossier « Applications ».
  .

Sous **GNU/Linux**, Pure Data (Vanilla) se lance également en entrant la commande "pd" dans un terminal :

```
pd
```

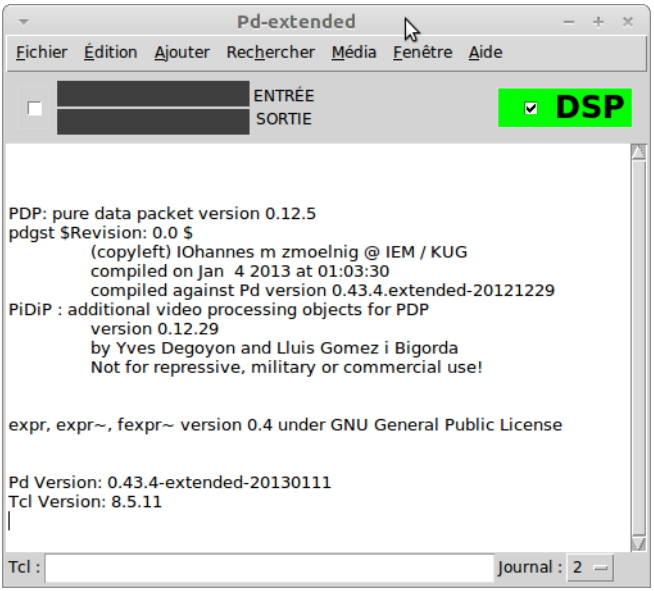
Le lancement de Pure Data Extended sur GNU/Linux dépend de votre méthode d'installation (paquets .deb ou gestionnaire de paquets). Essayez cette commande dans le terminal :

```
pdextended
```

FENÊTRE PRINCIPALE

Quand on ouvre Pure Data pour la première fois, on se trouve face à ce que nous appelons la **fenêtre principale**. Celle-ci contient diverses informations et surtout la **console**.

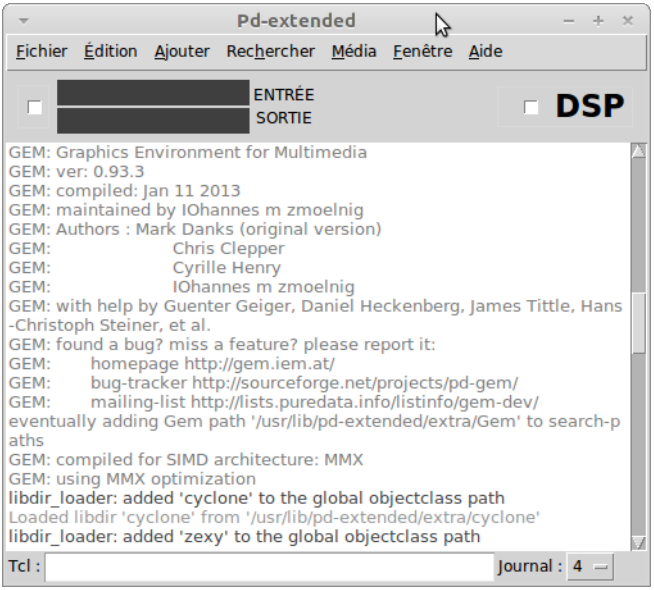
Tout d'abord, nous remarquons une succession de messages (des indications textuelles) qui, nous le comprendrons au cours de notre pratique, peuvent être des outils essentiels pour l'écriture des programmes. Ils le sont notamment pour la correction des erreurs éventuelles. On vérifie ainsi que le programme que nous venons de créer fonctionne tel que nous le souhaitons. Ces messages sont lisibles dans la fenêtre principale. Quand on envoie des messages dans l'objet [print], on peut afficher des informations dans la console.



Les messages textuels envoyés par Pure Data lors de son chargement sont également des indications importantes à propos du chargement des librairies externes (*externals*) et éventuellement des erreurs rencontrées lors de leur chargement.Pour voir tous ces messages, il faut changer le niveau de «verbosité» de l'interface à l'aide du menu déroulant «Journal : 2» en changeant cette valeur «2» par une une valeur plus grande pour voir plus d'informations.

Quelques-uns des messages affichés

Ces messages informent essentiellement sur la version de Pure Data utilisée. Ci-dessus, la version est la 0.43.4




```
GEM: Graphics Environment for Multimedia
GEM: ver: 0.93.3
GEM: compiled: Jan 11 2013
```

Ce message nous apprend que la librairie GEM (voir plus loin dans ce manuel pour de plus amples informations à son sujet) est présente dans sa version 0.93.3 compilée le 11 janvier 2012. En général s'ensuivent les noms des différents développeurs de la librairie.

Activer et désactiver l'audio

Une des options les plus importantes dans la fenêtre principale repose sur la possibilité de commuter l'audio. Pour ce faire, il suffit de cocher la case nommée "DSP" en haut à droite de la console.

LES MENUS

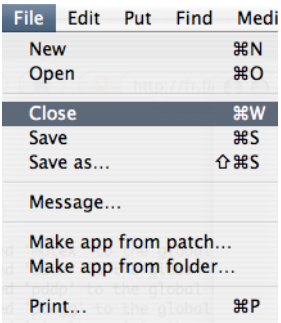
Au-dessus de la console des messages, on trouve toute une suite de menus déroulants. Voici quelques-unes de leurs principales caractéristiques.

Le "Pd-extended" contient essentiellement :

1. Des informations au sujet de la version de Pure Data installée sur votre système.
2. La possibilité de changer les "Preferences" par défaut (voir la partie installation pour plus de détail à ce sujet).
3. La possibilité de quitter Pure Data.

Dans le menu "File", nous allons pouvoir :

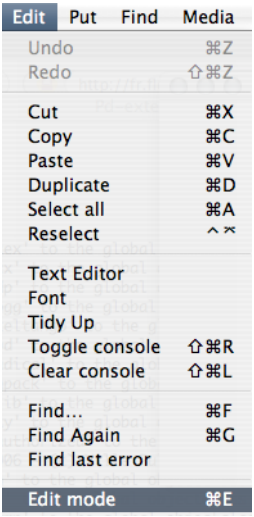
1. Créer un nouveau fichier.
2. Ouvrir un fichier existant.
3. Fermer le fichier courant.
4. Sauvegarder le fichier.
5. Sauvegarder le fichier sous.



MODE ACTION ET MODE ÉDITION

Dans le menu "Edit" se trouve l'état fondamental dans lequel nous évoluons avec Pure Data : soit nous éditons le patch, soit nous l'utilisons. En mode « action », nous agissons sur le programme, nous contrôlons les paramètres et nous actionnons des comportements pré-définis. En mode « édition », nous pouvons « écrire » ces comportements, définir les calculs, etc. Le mode « action » est utile lorsqu'on exécute par exemple une performance ou quand on utilise le résultat de notre patch.

Pour passer d'un mode à l'autre, faire "ctl + e" ou "pomme + e" : mode « édition » (curseur main) / mode « action » (curseur flèche).



Le mode dans lequel nous nous trouvons est indiqué tout en bas du menu face à l'item "Edit mode".

Note : agissez légèrement sur le curseur pour visualiser le changement.

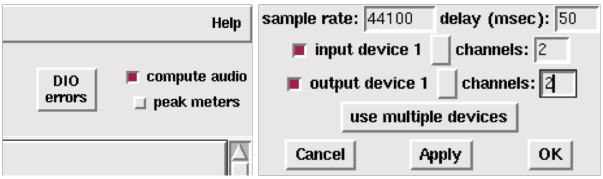
Savoir passer du mode « édition » au mode « action » est une habileté des plus importantes pour maîtriser Pure Data. Nous distinguerons facilement ces deux modes : en mode « édition » le curseur de la souris se transforme en petite main, tandis qu'en mode « action », les éléments placés sont grisés en permanence, ne peuvent être déplacés, ni effacés. En revanche, l'avantage du mode « action » est que nous pouvons alors cliquer sur les boîtes messages afin d'en activer la sortie, changer la valeur dans les boîtes nombre, cliquer sur les bang et utiliser les autres objets graphiques.

13. RÉSOLUTION DE PROBLÈMES

Il existe de nombreux petits détails qui pourraient faire en sorte que ça ne fonctionne pas dans l'immédiat. Voici quelques-uns des problèmes les plus courants que vous pourriez rencontrer.

IL N'Y A PAS DE SON

Dans la fenêtre de Pd, assurez-vous que la case compute audio est cochée. Ensuite, vérifiez que vous avez sélectionné la bonne carte son ainsi que les bons pilotes pour votre système, et que la carte son est bien connectée et fonctionnelle. Sur OS X, assurez-vous que les cases à cocher à côté de votre carte son ont été vérifiées dans réglages audio. Sur Linux ou OS X avec le pilote audio Jack, assurez-vous que le pilote est en marche. Sur toutes les plateformes, consultez le panneau de contrôle audio fourni avec votre système d'exploitation et vérifier que la sortie est autorisée et que son volume de lecture est mis en place. De plus, assurez-vous que vous utilisez le taux d'échantillonnage dans Pd qui correspond à celle de votre carte son.



À gauche : la case compute audio dans la fenêtre principale de Pd. À droite: les réglages audio de la boîte dialogue.

IL Y A CLICS ET DES CRÉPITEMENTS OU DES PARASITES DANS LE SIGNAL AUDIO.

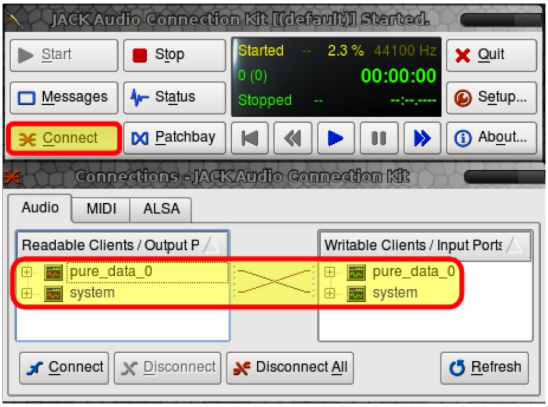
Il est plus que probable que vous avez choisi un temps de latence qui est trop rapide pour votre ordinateur et votre carte son à gérer. Retourner aux réglages audio et augmenter le délai. Sous Linux, il est également possible que d'autres processus en cours d'exécution sur votre ordinateur, ou même une carte graphique mal configuré ou trop lente, peuvent affecter les performances de Pd. Envisagez d'exécuter Pd avec l'option -rt activé drapeau (startup flag) (sur Linux uniquement!). Cela peut être fait à partir de la ligne de commande ou en ajoutant -rt pour les drapeaux dans le menu de démarrage. Sur Linux ou OS X, avec le pilote Jack il est possible de régler le temps de latence de l'application à une plus grande quantité et ainsi, de réduire les parasites (appelés xruns dans Jack) là aussi.

LE SIGNAL D'ESSAI EST DISTORDU

Il est possible que vous jouez le son trop fort pour votre carte son. Utiliser les commandes de votre carte son afin de réduire le volume de lecture. De plus, assurez-vous que vous utilisez le taux d'échantillonnage dans Pd correspond à celle de votre carte son.

JE NE VOIS AUCUNE ENTRÉE AUDIO

Peut-être que vous n'avez pas activé d'entrée audio. Sur OS X, assurez-vous que vous avez sélectionné les cases à cocher à côté de votre carte son dans réglages audio. En outre, certaines cartes avec un nombre impair de canaux in et out peuvent créer des problèmes de Pd. Essayez de définir le nombre de canaux de la même façon pour l'entrée que pour la sortie. Sur toutes les plateformes, consultez le panneau de contrôle audio fourni avec votre système d'exploitation et assurez-vous que la bonne entrée est activée et que le volume de l'enregistrement est bien mis en place.

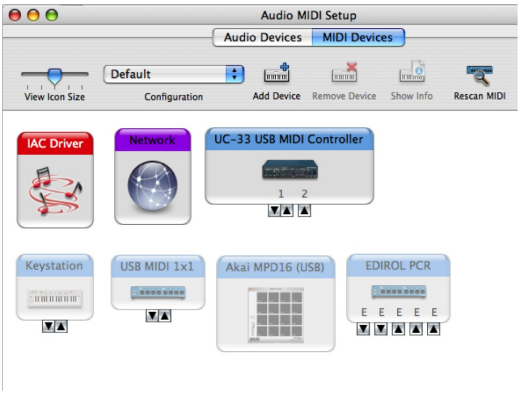


Sur Linux, l'application QJackCTL permet un routage facile des signaux audio entre les applications et la carte son, ou encore, entre les applications sur le même ordinateur.

JE NE VOIS PAS D'ENTRÉE MIDI

Vérifiez que vos périphériques MIDI et/ ou vos programmes font correctement l'envoi des données et que votre système d'exploitation envoi ces données à Pd. Sur OS X, vérifiez que vous avez sélectionné les bons périphériques MIDI, et que Audio MIDI Setup.app était en marche avant de démarrer Pd. Sous Linux, en utilisant les pilotes MIDI par défaut, vérifiez que vous avez sélectionné le bon périphérique MIDI au démarrage. Aussi, avec les pilotes ALSA-MIDI, assurez-vous que vous avez bien branché vos périphériques ou programmes MIDI de Pd. L'utilisation de Jack avec l'application QjackCRT est recommandée à cet effet. Sous Windows, vous pouvez utiliser une application comme MIDI Ox / MIDI Yoke Junction pour voir, analyser et gérer vos connexions MIDI.

La boîte de dialogue Réglages MIDI.



Sur OSX, il est nécessaire d'avoir l'application Audio MIDI Setup.app en cours d'exécution afin de connecter le matériel MIDI et les ports de logiciels à Pure Data.

Qjackctl permet également aux utilisateurs Linux de router ALSA MIDI entre applications et les ports MIDI.

Lorsque je tape le nom d'un objet, le message «... couldn't create » (ne peut créer) apparaît et il ya une ligne pointillée autour de mon objet La raison de cette erreur est que vous avez demandé à Pd de créer un objet qui n'existe pas. Il peut y avoir plusieurs raisons pour cette erreur, et la plus courante est l'orthographe. Les noms d'objets dans Pd doivent être correctement écrit, et ils sont sensibles à la casse. Par exemple, [Osc ~] ou [OSC ~] ne va pas créer [osc ~], pas plus que [osc] sans le tilde. Parfois, les utilisateurs combinent la création d'argument et le nom de l'objet, comme [+1] au lieu de [+ 1]. Un argument est la création complémentaire d'information donné à un objet et définit les paramètres de l'objet. Les nouveaux utilisateurs de Pd confondent souvent les boîtes objets avec les boîtes de messages, qui sont des types très différents d'éléments qui peuvent être placé dans le patch à partir du menu « Put ». Avec la fonction « Find last error » sous l'onglet « Find » du menu principal, il est possible de retrouver les objets qui n'ont pas créé. Pour plus de détails, consulter le chapitre « Interface utilisateur ».

JE REÇOIS LE MESSAGE «... COULDN'T CREATE » (NE PEUT CRÉER) QUAND J'OUVRE UN PATCH ET IL YA UNE LIGNE POINTILLÉE AUTOUR DE MON OBJET

Si vous obtenez ce type d'erreur lors de l'ouverture d'un patch et que vous êtes certain qu'il est fonctionnel autrement (c'est-à-dire, vous l'avez téléchargé à partir d'Internet ou l'avez créé lors d'une session Pd précédente), il est probable qu'il y ait un objet externe qui était disponible lors de la création du patch ne soit plus disponible maintenant. Via la fonction Find last error sous Find du menu, vous pourrez traquer les objets qui n'ont pas été créé. Pd préservera l'emplacement et les connexions d'un objet qui ne parvient pas à créer mais ne sera pas fonctionnel, alors que la plupart des librairies externes sont disponibles dans la distribution de Pd Extended, d'autres non, ou encore nécessitent une configuration supplémentaire du path et des paramètres de démarrage. Voir le chapitre « Configuration Pd ». Si l'externe n'est pas disponible en Pd Extended, vous devrez peut-être l'installer vous-même.

JE REÇOIS LE MESSAGE « ...ERROR: SIGNAL OUTLET CONNECT TO NONSIGNAL INLET (IGNORED) » (ERREUR: SORTIE DU SIGNAL CONNECTÉE À UNE ENTRÉE NON-SIGNAL - IGNORÉ) » QUAND J'OUVRE UN PATCH

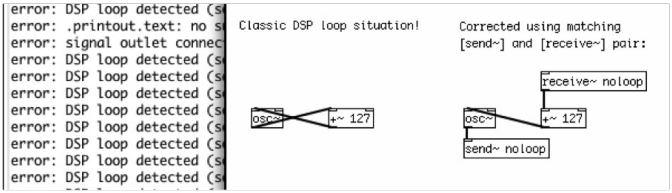
Tout comme l’erreur mentionnée ci-haut : « Je reçois le message «... couldn’t create [...] », cela signifie qu'un objet n'a pas pu être créé car il utilise un objet externe qui n'est pas disponible dans l'installation ou la configuration actuelle de Pd. Comme précédemment, via la fonction « Find last error » sous « Find » du menu, vous pourrez traquer les objets qui n'ont pas été créé. Pd va traiter des objets non créés en tant qu'objets DataFlow (flux de données) même si, à l'origine, ces objets étaient des objets audio, alors cette erreur suivra la précédente. Pour plus de détail, voir le chapitre « Configuration Pd ».

JE REÇOIS LE MESSAGE « ERROR: CAN'T CONNECT SIGNAL OUTLET TO CONTROL INLET » (ERREUR: IMPOSSIBLE DE CONNECTER LA SORTIE DU SIGNAL À L’ENTRÉE DE CONTRÔLE) ET JE NE PEUX PAS CONNECTER DEUX OBJETS ENSEMBLE

La sortie des objets audio (avec un tilde « ~ » dans leur nom) ne peuvent pas être connectés à des objets DataFlow (sans tilde « ~ » dans leur nom). Donc Pd ne permettra pas ces connexions à réaliser. Assurez-vous que vous utilisez la bonne combinaison d'objets.

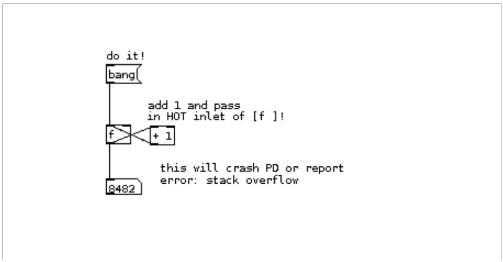
JE REÇOIS LE MESSAGE « ERROR: DSP LOOP DETECTED (SOME TILDE OBJECTS NOT SCHEDULED » (ERREUR: BOUCLE DSP DÉTECTÉE - CERTAINS OBJETS TILDE NE SONT PAS PRÉVU) QUAND JE CLIQUE SUR « AUDIO ON » ET LE SON NE FONCTIONNE PAS

Dans un système électronique analogique, vous pouvez connecter la sortie d'un mixeur à l'une des entrées, tourner le canal et obtenir une rétroaction. C'est parce que qu’à peu près tout, dans un système analogique, se passe simultanément. Les ordinateurs ne fonctionnent pas comme cela, donc vous ne pouvez pas demander un patch Pd de calculer des résultats sur la base de sa propre sortie simultanée. Pd travaille dans ce qu'on appelle des blocs (c'est-à-dire, un groupe d'échantillons, tel que le nombre par défaut de 64 échantillons) et tous les échantillons dans chaque bloc doivent être calculé avant leur sortie. Ainsi, une boucle DSP se produit quand un correctif a besoin d'informations qui est calculé à l'intérieur du même bloc afin de créer une sortie. Vous pouvez utiliser la fonction Find last error sous Find du menu pour traquer les objets qui sont à l'origine de la boucle DSP. La meilleure façon de contourner ce problème est de créer au moins un bloc de délais entre les objets reliés entre eux. Les objets [send ~] et [receive ~] sont utiles pour cela, puisqu'ils ont un retard intégré d'un bloc. Pour changer le numéro de d’échantillons dans chaque bloc, vous pouvez utiliser l’objet [bloc ~]



JE REÇOIS LE MESSAGE « ERROR: STACK OVERFLOW » (ERREUR: DÉBORDEMENT DE PILE) LORSQUE JE CONNECTE ENSEMBLE DEUX OBJETS DATAFLOW

Un débordement de pile survient lorsque vous avez demandé à Pd de calculer une opération récursive, et cette opération provoque une perte de surcharge de la mémoire de Pd qui peut occasionner un gel du programme. Un exemple courant d'une opération récursive est le compteur classique; en utilisant [float] et [+ 1]. L'objet [float] emmagasine un nombre à virgule flottante sur son entrée de droite et renvoie le résultat quand il reçoit un [bang]. Si la sortie de [float] est reliée à l'entrée [+ 1], et la sortie de [+ 1] est connecté à l'entrée de droite (entrée froide) de l'objet [float], alors le message [bang< envoyé à l'entrée de gauche (entrée chaude) du [float] va afficher un nombre qui augmente d'une unité chaque fois que le message est envoyé. Si toutefois la sortie de [+ 1] est connectée à l'entrée chaude de [float], puis le message [bang< à son entrée gauche, cela aura un effet différent. Il exécutera [float] et [+ 1] de façon à additionner des nombres ensemble aussi vite que l'ordinateur peut les faire. Cette opération va rapidement épuiser toutes les ressources de la mémoire de Pd résultant un débordement de pile. Vous trouverez plus d'informations dans la section « L'ordre des opérations » du chapitre « Le flots des données ».



JE REÇOIS LE MESSAGE D'ERREUR « CONNECTING STREAM SOCKET: NETWORK IS UNREACHABLE » (CONNEXION SOCKET DE FLUX: RÉSEAU INACCESSIBLE) QUAND J'OUVRE PD

Si vous utilisez le système d'exploitation Linux, et voyez ce message lorsque vous démarrez Pd, cela signifie que votre machine ne peut pas établir une connexion réseau lui-même. Vous devez configurer votre périphérique réseau de bouclage. Dans de nombreuses distributions Linux, vous pouvez le faire en répondant oui lorsque les outils de configuration du système demande si la machine sera un «réseau» (même si elle ne sera pas).

14. LE PATCH

L'ensemble structuré et organisé des instructions relatives à ce que l'on veut programmer se fait dans le patch.

Nous présenterons dans ce chapitre les composantes de base ainsi que les menus complémentaires de l'espace de travail de Pure Data.

COMPOSANTES DE BASES

C'est à partir du menu "*Put*"que nous allons pouvoir placer sur notre patch les éléments de base du traitement des données. Ce menu offre des choix intéressants seulement si le patch est ouvert. Afin de rendre ces premières indications plus concrètes, nous allons à présent les mettre en œuvre et créer un premier **patch**.

LE PATCH

Qu'est-ce qu'un patch ?

Ce que nous appelons communément un patch, c'est une feuille de travail (ou un espace de travail), au commencement vide, sur laquelle nous allons pouvoir « composer », « tisser» ou « écrire » notre programme.

Allez dans *Menu* > *File* > *New* ou faites *Ctrl+n*. Cela crée un patch :

Un nouveau patch, vraiment vide !

LA BOÎTE MESSAGE

Le message, dans son concept abstrait, est toute donnée qui transite à l'intérieur de Pure Data à travers les ficelles grises fines, à l'exception du son. Comprendre le traitement des messages dans Pure Data est une partie essentielle de sa prise en main complète. Cette partie spécifique sera traitée ultérieurement.

La boîte message, quant à elle, prend la forme concrète d'une sorte de drapeau, que nous pourrions comparer également à une « enveloppe ». Disons que c'est une sorte de « poche » de réception et d'envoi, très pratique pour traiter presque toutes les formes de données de natures différentes (nombres, symboles...) rencontrées dans Pure Data, sauf le son.

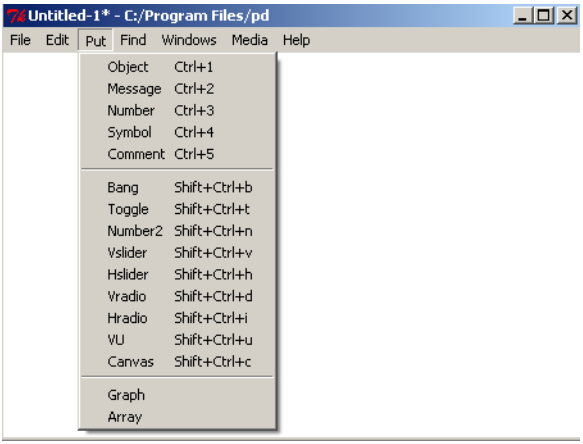
Sur cette feuille vide, dans le *Menu* > *Put* > *Message* (ou Ctrl+Maj+2), nous créons notre première boîte de message.

Ci-dessus, une boîte de message.

Veuillez noter que dans ce manuel "[...<" sert à décrire un message, comme dans [Bonjour Pure Data<.

LA BOÎTE OBJET

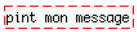
Nous allons ensuite créer un premier **objet**. Allez dans *Menu* > *Put* > *Object* (ou Ctrl+Maj+1).



Ce que nous nommons *objet* est ce genre de boîte rectangulaire, comme ci-dessous :

À l'intérieur du rectangle en pointillé, nous allons écrire le nom de l'objet que nous souhaitons utiliser, ici **[print]**. "print" sert à afficher dans la console les messages qui lui sont envoyés, sans les transformer, sans effectuer aucun calcul dessus.

Une fois créée, cette boîte se dessine soit sous forme de pointillés bleu ou rouge, soit de traits continus et gris foncé. Si les traits sont en pointillés, il y a une erreur et nous avons échoué à créer l'objet (erreur de frappe par exemple). Si les traits sont pleins, tout va bien et notre objet existe.

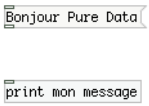


En programmation, chaque caractère, espace ou signe de ponctuation a son importance !

On pourrait comparer cet objet à ce que l'on nomme en informatique une **fonction**, c'est-à-dire un élément qui effectue une action précise quand on le lui demande. Dans le cas présent, nous voulons qu'il écrive un message dans la console. Pour que cet ordre devienne effectif, nous l'avons spécifié explicitement, ici en inscrivant le mot "print" dans la boîte. On appelle ce terme la classe de l'objet, ou plus simplement, le nom de l'objet.

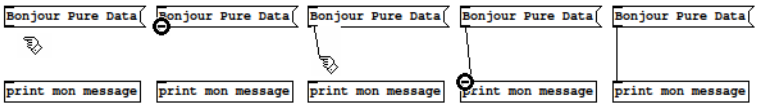
À la suite de certains noms d'objets, on peut ajouter un ou plusieurs arguments sous la forme de mots et/ou de chiffres séparés par des espaces. Certains objets acceptent différents types d'arguments. Si nous commettons une erreur, dans ce cas également l'objet peut ne pas être créé.

Dans le cas présenté ici, l'argument accepté par la fonction d'impression [print] est un en-tête ajouté devant le message envoyé. Cet en-tête sera utile pour distinguer plusieurs messages les uns des autres dans le cas où plusieurs objets [print] se trouvent dans le patch.



ÉTABLIR UNE CONNEXION

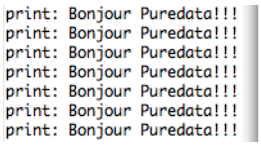
Ensuite, en mode édition (Ctrl+e) à l'aide du pointeur de la souris, nous allons nous positionner au-dessus du minuscule rectangle situé en bas à gauche de notre boîte message. Le pointeur se transforme en cercle noir gras. En cliquant sur le bouton gauche de la souris, nous allons tirer un câble du bas de la boîte, vers le petit rectangle situé en haut et à gauche de la boîte objet **[print]**. Lorsque notre curseur est au-dessus du petit rectangle en haut à gauche de l'objet en question, un cercle noir devrait apparaître à nouveau, nous signalant que, si nous lâchons le bouton de notre souris, nous allons effectuer une connexion, c'est-à-dire brancher nos deux boîtes ensemble !

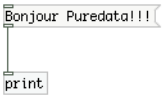


INLET - OUTLET

Ces rectangles situés en haut et en bas des boîtes messages, des boîtes objets, des boîtes nombres etc. sont les entrées et les sorties du flot des données. Nous les nommerons entrées (***inlet***) et sorties (***outlet***).

Lorsqu'on clique sur une boîte, le message qu'elle contient est envoyé à travers les fils connectés à sa sortie. Bien faire attention à passer en mode action (Ctrl+e) !

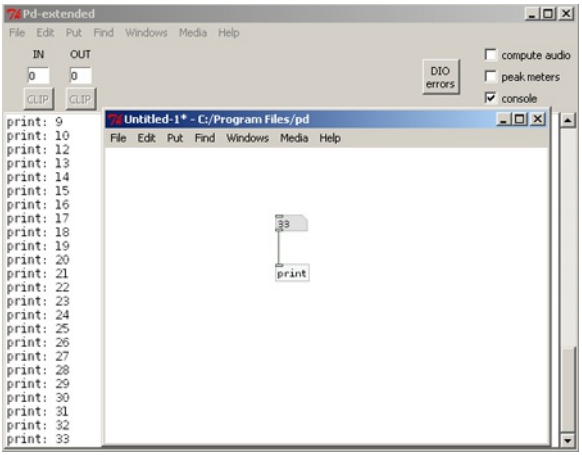




LA BOÎTE NOMBRE

La boîte nombre permet de recueillir, de lire ou de contrôler un flot de données de type "float" ou nombre décimal. Il est à noter que Pure Data ne travaille pas avec des nombres entiers, mais seulement avec des nombres à virgule. De plus amples détails à ce sujet seront développés dans la suite de ce manuel.

Pour créer une boîte nombre, faites Ctrl+Maj+3.



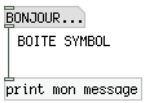
En mode action, nous pouvons soit cliquer et glisser le curseur de la souris de bas en haut ou de haut en bas pour faire évoluer les valeurs à l'intérieur de la boîte, soit cliquer directement à l'intérieur de la boîte nombre puis entrer une valeur au clavier.

LA BOÎTE SYMBOLE

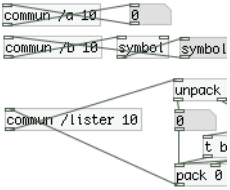
La boîte symbole aura un peu le même comportement que la boîte nombre, sauf qu'elle va nous permettre de visualiser ou entrer au clavier des chaînes de caractères. Notez que cette boîte ressemble à la boîte nombre. Elle est seulement plus allongée. Pour saisir une chaîne de caractères à l'intérieur, il faut cliquer sur la boîte en mode action, puis taper le mot souhaité.

Remarque : pour faire sortir le mot dans la console de Pure Data, en d'autres termes activer une action sur la boîte symbol, faire un retour à la ligne (touche *enter*).

Allez dans *Menu* > *Put* > *Symbol* (ou faites Ctrl+Maj+).



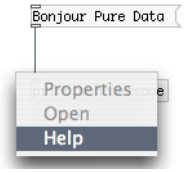
Toutes ces boîtes nombre, objet ou symbole peuvent être connectées entre elles.



LES RUBRIQUES D'AIDE

Il est à noter une particularité très intéressante de Pd offrant l'accès à des rubriques d'aide internes, écrites directement avec Pure Data. Ceci nous permet, par exemple, de tester les patchs pour les découvrir concrètement ou bien effectuer des copier/coller de portions de ces rubriques, d'expérimenter, de jouer avec, etc.

Pour accéder à ces rubriques pendant l'écriture du patch sans sortir de ce dernier, il suffit par exemple de placer le curseur de votre souris sur la boîte objet avec le nom de sa fonction et de faire un clic droit. Une fenêtre de dialogue doit apparaître comme ceci :



LES COMMENTAIRES

Un commentaire permet de laisser des informations lisibles pour soi-même ou pour d'autres, d'indiquer nos actions effectuées sur le patch, les modifications apportées ou le comportement de certaines portions du patch. C'est une bonne habitude à prendre lorsque l'on commence à programmer. Cela favorise la lisibilité, le travail en équipe et la réutilisation des patchs créés.

Faire Menu -> Put -> Comment ou Ctrl+Maj+5

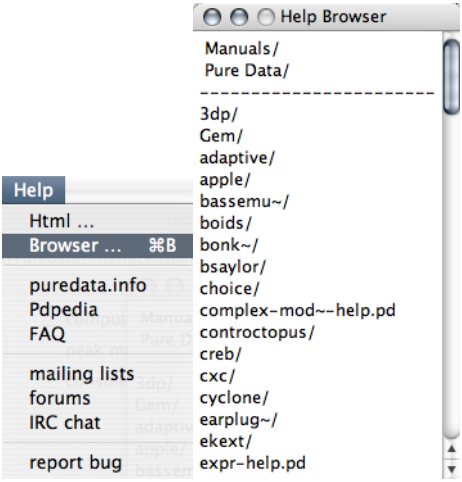
SUPPLÉMENTS AU SUJET DES MENUS.

Le menu "Find" permet de localiser les parties problématiques. Elles seront marquées en bleu. Si une erreur est détectée par Pure Data, elle sera signalée dans la console.

Le menu "Media" permet d'activer ou de désactiver l'audio, de configurer l'audio et le MIDI, de tester si tout ça fonctionne, et bien plus encore. Ce sujet est abordé en détail dans le chapitre sur la configuration audio et MIDI.

Le menu "Window" permet de vérifier et d'accéder aux différentes fenêtres ouvertes durant la session de travail.

Enfin le menu "Help" nous donne accès à de précieuses informations et exemples d'aides :



Dans les deux captures ci-dessus, nous avons à gauche la plage déroulée du menu "Help". *Browser* est sélectionné. À droite, le sous-menu "Help Browser", tel qu'il se présente dans la version 0.42.5 de Pd-extended (Mac OS X). Ce navigateur d'aide contient des tonnes d'exemples et de patchs d'aide pour de nombreux d'objets. Essayer de comprendre et d'utiliser un patch d'aide par jour est un excellent moyen de maîtriser rapidement Pure Data.

La rubrique *Manuals* permet d'accéder à divers manuels en anglais. Pure Data mène vers les tutoriels de base du cœur de Pure Data. Viennent ensuite la liste des librairies externes avec leurs exemples d'aide et autres informations intégrés par chacun de leurs développeurs.

15. LE FLOT DES DONNÉES

La programmation graphique avec Pure Data est relativement intuitive, mais certaines notions dans la manière dont les informations s'écoulent peuvent parfois dérouter et rendre la compréhension des patchs difficile pour l'œil néophyte. Nous allons donc introduire ici plusieurs particularités relatives au flot des données dans le langage de *dataflow* qu'offre Pure Data.

Pure Data fonctionne suivant la logique de la « chute d'eau », c'est-à-dire que les données partent du haut pour transiter vers le bas. Les entrées (*inlets*) sont donc sur le haut des boîtes, les sorties (*outlets*) sur le bas. Le transit des données est instantané : il n'y a pas de délai entre l'entrée d'une donnée et sa sortie dans la boîte suivante.

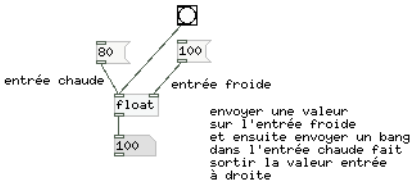
L'ORDRE DES OPÉRATIONS

L'ordre dans lequel les messages entrent et sortent des boîtes est important. L'ordre des opérations dans Pure Data est déterminé par les règles suivantes :

1. Entrées chaudes / entrées froides
2. Ordre des connexions
3. La priorité du parcours en profondeur des messages

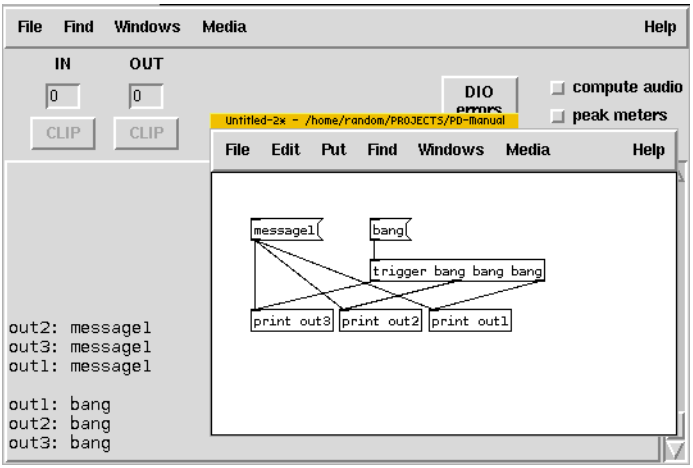
1. Entrées chaudes / entrées froides

Une distinction existe entre l'entrée de gauche et les autres entrées (à droite) d'un objet : l'entrée de gauche est appelée « entrée chaude » parce qu'un envoi de données dans cette entrée entraîne directement le déclenchement de l'opération et la sortie du résultat. L'envoi dans les entrées « froides » de droite, par contre, ne déclenche pas d'action sur l'objet. Cela permet par exemple de stocker des paramètres qui seront utilisés lors du prochain déclenchement de l'objet par l'arrivée de données dans l'entrée chaude (comme un "bang").



2. Ordre des connexions

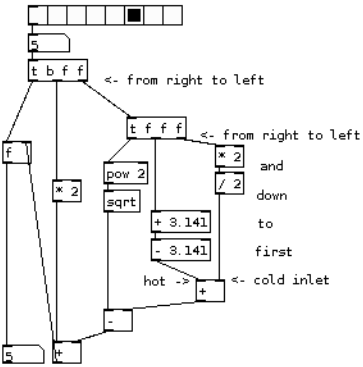
Quand un objet effectue une action, et si des messages doivent passer par plusieurs de ses « sorties », ce sera toujours la sortie la plus à droite qui produira un message en premier. Ses autres messages à envoyer le seront de droite à gauche jusqu'à la sortie la plus à gauche. Plusieurs connexions à une entrée (*inlet*) ne pose pas de problèmes sauf lorsque l'ordre des opérations est important. Lorsqu'une sortie a plusieurs connexions, nous ne pouvons pas déterminer l'ordre des opérations. Pure Data garde une mémoire de l'ordre dans lequel les connexions ont été créés mais ne permet pas de visualiser la priorité d'envoi des messages. L'objet **[trigger]** permet de contrôler l'ordre d'envoi des informations des sorties de droite à gauche et permet de convertir le type de donnée relayée (voir chapitre : « Quelques objets utiles »).



3. La priorité du parcours en profondeur des messages

Dans le cas de plusieurs connexions sortantes depuis un **[trigger]** ou d'une sortie (*outlet*) d'un autre objet, une opération n'est considérée "terminée" que lorsque toutes les suivantes qui appartiennent à la même branche le sont.

Dans cet exemple, suivez le chemin de l'ordre des opérations avec votre doigt. Les règles de l'ordre d'exécution de droite à gauche de l'objet **[trigger]** et celle du parcours en profondeur s'appliquent. Ici, le résultat des opérations sera toujours le même nombre qu'en entrée.



LES MESSAGES

Dans Pure Data, les objets communiquent les uns avec les autres en utilisant des messages partant généralement de la sortie (ou "outlet") d'un objet vers l'entrée (ou "inlet") d'un autre. Ces messages sont transportés à travers les lignes noires appelées « ficelles ». Ces messages sont généralement des « requêtes » c'est-à-dire qu'ils demandent à l'objet d'opérer une action comme « ouvrir un fichier », de faire un calcul (addition, soustraction, etc.) ou de stocker des données. Mis à part les signaux audio (objet + ~ avec ficelle épaisse) toutes les autres données en Pure Data sont des messages.

Les messages sont composés d'un ou plusieurs éléments de données appelés **atomes**.

Il existe plusieurs types d'atomes dans Pure Data, mais les plus importants sont les suivants :

- numériques - des nombres comme "3", "4.5", ou "5.55556e+06"
- symboliques - en général un mot qui ne peut être interprété comme un nombre, et qui ne contient pas d'espace, par exemple "blablah", "fichier02" ou "reset".

(N.B. : attention utiliser le point à la place de la virgule pour les nombres avec des décimales.)

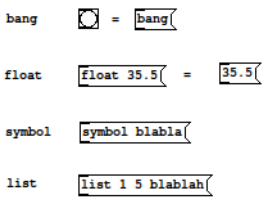
Les messages commencent par un atome symbolique appelé **sélecteur**, qui peut être suivi par zéro ou plus d'atomes, appelés **éléments** du message (tous séparés par des espaces). Hormis deux exceptions bien pratiques expliquées ci-dessous, les messages Pure Data adhèrent à la forme : *sélecteur élément1 élément2 élément3 (etc.).*

Le sélecteur du message permet à l'objet de savoir ce qu'il doit faire : quand un message arrive à l'une de ses entrées, l'objet vérifie le sélecteur et choisit l'action appropriée (ou **méthode**). Chaque objet a sa ou ses méthodes particulières. Un objet peut ainsi accepter un message constitué du sélecteur "set" suivi d'un atome numérique, ce qui permettra de sauvegarder cette valeur. Un autre objet peut accepter un message constitué du seul sélecteur "clear" (sans élément) comme méthode permettant « d'oublier » les données actuellement en mémoire. Pour savoir ce qu'un objet accepte comme messages, il suffit de faire un clic-droit sur l'objet et de choisir **help** depuis le menu contextuel.

Les objets de Pd impriment un message d'erreur dans la console à chaque fois qu'ils reçoivent une commande incomprise. [change] par exemple, n'accepte que les nombres et le symbole "set". Si on lui adresse le message [Nicolas est nerveux<, un message d'erreur nous fera savoir que le sélecteur "Nicolas" n'est pas compris, qu'il n'y a pas de méthode lui correspondant : "*error: change: no method for* '*Nicolas*'." Pour les entrées froides (celles de droite), l'erreur fera référence au sélecteur incorrect : "*error: inlet: expected* '*float*' *but got* '*Nicolas*'."

Types de messages standards

Différentes personnes créent des objets pour Pure Data, les désignant pour accepter des messages dont les sélecteurs sont formatés selon leurs besoins, ce qui rend souvent difficile pour l'utilisateur de savoir quel type de sélecteur l'objet accepte. Heureusement, il existe quelques types de messages standards :



- **message** float- le mot "float" (pour "*floating point numbers*" = « nombre à virgule flottante » - c'est à dire « nombre réel ») avec un élément numérique comme "float -5", par exemple.
- **message symbol** - le mot "symbol" suivit par un mot, par exemple "symbol blablah."
- **list** - le mot "list" suivit par un groupe de nombres et/ou symboles et/ou pointeurs.
- **bang** - le simple mot "bang" utilisé pour envoyer une impulsion.
- **pointer** - référence à des données enregistrées dans des structures de données graphiques.

Les messages standards ci-dessus simplifient les opérations communes de Pd. Par exemple, si un objet produit une opération arithmétique simple, on peut facilement en déduire qu'il acceptera des nombres en entrée.

Pour plus de facilité d'écriture, ces messages peuvent la plupart du temps se passer de sélecteur ; les objets les prendront et comprendront leur type de manière « implicite », permettant d'écrire des patchs beaucoup plus rapidement.

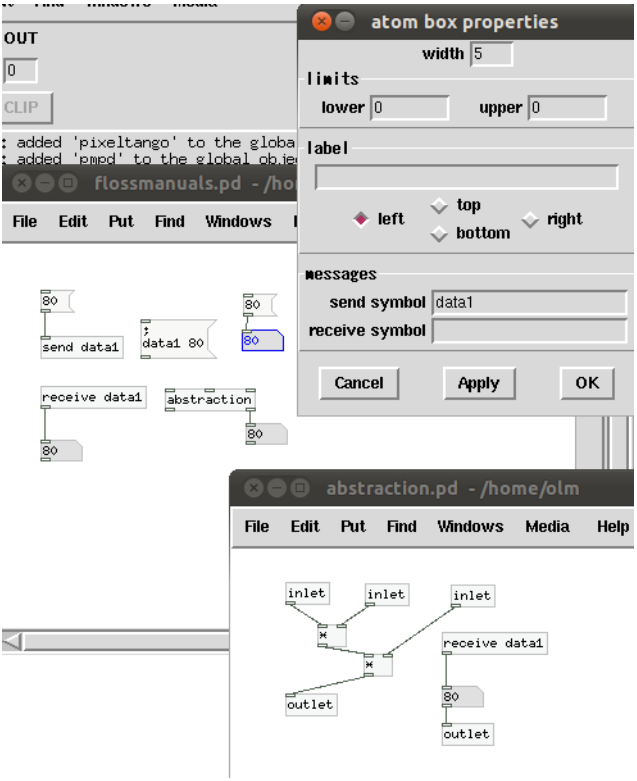
CONNEXIONS SANS FICELLE

Il est possible de connecter des composants ensemble et d'envoyer des messages sans utiliser de ficelle : on utilise alors la fonction des objets [send] et [receive] qui vont agir comme émetteurs et récepteurs.

Lors de la création des objets, on ajoute un mot-clé qui sera le nom du fil virtuel reliant l'émetteur et le(s) récepteur(s). Ceci fonctionne sur toute la session Pure Data en cours comme lien entre tous les patchs, les **sous-patchs** et les **abstractions** (voir le chapitre *Organisation des objets*) qui sont ouverts. Il faut donc faire attention au mot-clé employé pour être certain du chemin pris par les données.

Il est également possible de réaliser cette opération en utilisant les propriétés des composants (clic droit, "Properties") et en éditant les champs "send symbol" ou "receive symbol" avec l'étiquette désirée. Ceci supprime l'entrée/sortie concernée, et la remplace par un envoi sans fil. Cette méthode est pratique, mais nécessite plus d'attention lors de la relecture du patch, puisque les sources/destinations ne sont dès lors plus visibles dans la fenêtre.

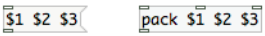
Un message peut également envoyer ses données sans ficelle, en utilisant une syntaxe spécifique : si un message contient un point-virgule + retour à la ligne, la suite du message est interprétée comme un [send] avec le premier argument comme étiquette et le second comme valeur. Plusieurs lignes précédées d'un ";" peuvent être combinées pour envoyer des messages à plusieurs [receive] différents.



LE SIGNE $

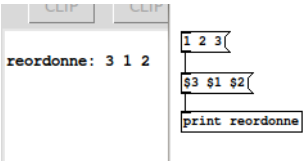
On veut parfois réserver une place dans les arguments d'un objet ou dans un message pour une valeur qui sera déterminée ultérieurement. Pour cela, on utilise le signe $ (dollar).

Ce signe a un usage différent selon qu'on l'utilise dans un objet ou un message.



Dans un message

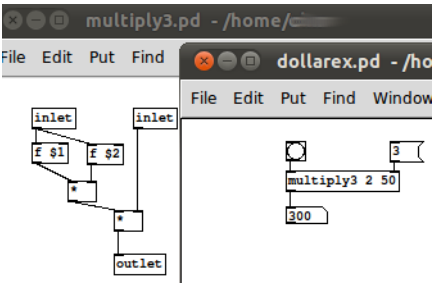
Une boîte message accepte les variables $n (avec n > 0) et les substitue par la valeur attribuée à l'index de l'atome dans une liste reçue. La substitution est purement locale et correspond uniquement à la liste reçue, contrairement à l'utilisation dans un objet où les $n correspondent à des arguments du patch.



Dans un objet

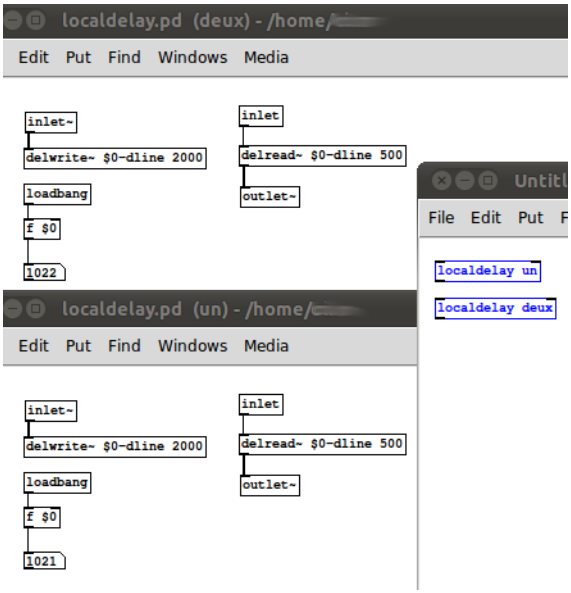
De la même manière qu'un objet accepte des arguments, une abstraction peut en accepter pour les utiliser à l'intérieur. Les arguments sont passés dans l'ordre, et sont accessibles par l'usage de $n avec n correspondant à la position de l'argument dans la déclaration de l'abstraction.

Par exemple, cette abstraction multiplie la valeur entrée à droite par les 2 arguments lorsqu'on lui envoie un bang à gauche :



Le signe dollar $n utilisé dans un objet fera donc toujours référence aux arguments de l'abstraction à laquelle il appartient, contrairement à l'usage dans les messages.

Alors que $1, $2, etc. représentent le premier, second, etc. argument de l'abstraction, il y en a un spécial qui est très utile dans Pure Data. **$0** est une variable qui est remplacée de manière interne par un nombre unique de 4 chiffres propre à chaque patch et instance d'abstraction. En d'autres termes, Pd fait en sorte que chaque instance d'abstraction reçoive un nombre unique mémorisé dans la variable $0. L'utilité en est démontrée dans l'abstraction suivante, qui utilise des lignes de délais, lesquelles ne peuvent avoir le même nom dans plusieurs instances de la même abstraction, sous peine de conflit.

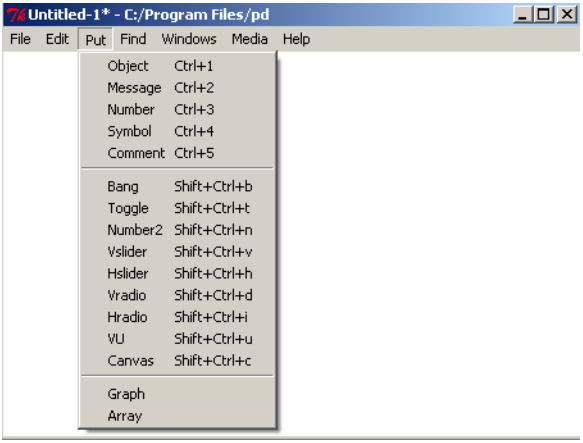


Les lignes de délais dans les instances de l'abstraction [localdelay] sont donc nommées ici 1021-dline et 1022-dline.

Bien qu'on appelle souvent une variable construite avec $0-quelquechose « variable locale », il est tout à fait possible d'y accéder depuis l'espace de variable globale simplement en trouvant par quoi a été remplacé $0, comme ci-dessus. C'est donc uniquement l'assignation qui est locale, pas la variable construite.

16. OBJETS GRAPHIQUES

Il existe dans Pure Data des objets qui peuvent nous aider à créer des interfaces graphiques en quelques clics. Ils illustrent des métaphores classiques de contrôle comme celles de boutons, de curseurs et d'interrupteurs. Il vous est possible de créer ces objets en choisissant un des items de la deuxième section du menu "Put".



Note : Il faut préalablement avoir créé un premier patch pour pouvoir accéder au menu déroulant "Put".

BANG! IMPULSIONS

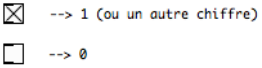
Certaines actions peuvent être déclenchées en cliquant avec la souris comme quand nous cliquons sur un lien Web ou sur le bouton « confirmer » d'un formulaire. Une action est alors aussitôt effectuée.

L'objet [bng] est un bouton qui émet une impulsion quand nous cliquons dessus. Pour créer un [bng], vous pouvez choisir l'item "Bang" du menu "Put". On peut aussi appuyer sur Majuscule + Contrôle + b.

Nous pouvons activer cet objet "Bang" en lui envoyant un message "Bang" dans son entrée et en cliquant sur ce message.

LES INTERRUPTEURS

Il est possible d'activer ou désactiver certaines fonctionnalités de notre patch, et ce au moyen d'un interrupteur.



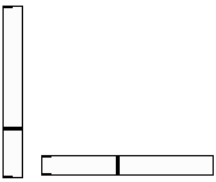
L'objet [tgl] est un interrupteur à deux positions. Il peut être soit allumé, soit éteint. Au moment où il change d'état, il envoie un chiffre à sa sortie. Par défaut, il envoie « un » (1) lorsqu'il est allumé, mais cette valeur peut être modifiée par l'intermédiaire de "Properties". Il envoie « zéro » (0) lorsqu'il est éteint. Vous pouvez créer un [tgl] en choisissant l'item "Toggle" du menu "Put". On peut aussi appuyer sur Majuscule + Contrôle + t.

Nous pouvons allumer ou éteindre un interrupteur en lui envoyant la valeur « un » (1) (ou une autre) ou "zéro" (0) par le biais d'un message "nombre".

LES GLISSIÈRES

Pour contrôler le volume d'un son ou l'intensité d'un effet, il est très facile de placer une glissière dans notre patch. Celle-ci expulse par sa sortie une suite de valeurs situées entre sa valeur minimale et sa valeur maximale. Ceci a lieu lorsqu'on déplace son curseur au moyen de la souris. Pour cela, cliquez sur le curseur, maintenez le bouton enfoncé et déplacez la souris.

On crée une glissière horizontale [hsl] en choisissant l'item "Hslider" du menu "Put", ou en appuyant sur Majuscule + Contrôle + h. Pour créer une glissière verticale [vsl], on peut choisir l'item "Vslider" du menu "Put" ou en appuyant sur Majuscule + Contrôle + v.

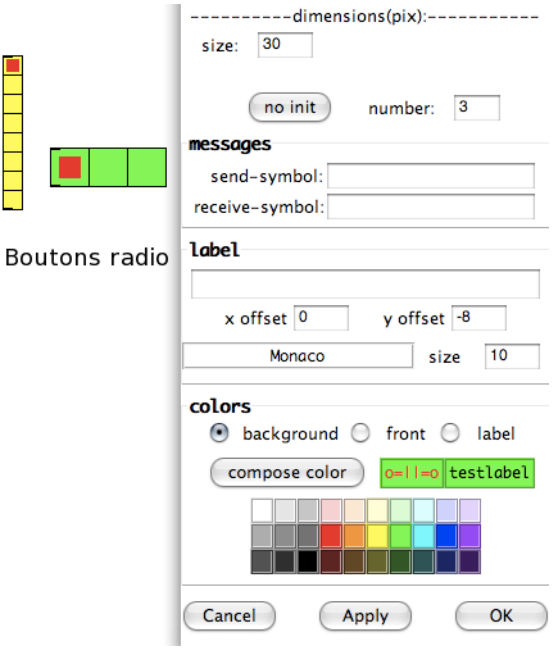


Il est également possible de déplacer le curseur de la glissière en envoyant un nombre dans son entrée. Pour changer la position du curseur sans pour autant que la glissière n'envoie quoi que ce soit à sa sortie, il faut lui envoyer le message [set 54< : ici "54" est la valeur que l'on souhaite lui donner. Choisir dans ses propriétés "*steady on click*" ou "*jump on click*" lui dicte d'atteindre la valeur souhaitée en déplaçant le curseur de la glissière, ou d'atteindre cette valeur par un clic de souris à l'endroit souhaité du curseur.

Particularité : le message [range 34< va permettre de redéfinir « à la volée » l'intervalle des valeurs des glissières. Dans cet exemple "34" va permettre d'attribuer à notre glissière un champ de valeurs qui ira de 0 à 89. Si nous avons créé cette glissière sans éditer ses valeurs par défaut (de 0 à 127) notre glissière ne pourra plus envoyer que des valeurs allant de 0 à 34 après avoir été réinitialisée par le message [range + (valeur choisie)<.

BOUTONS RADIO

Certains appareils radio ont des boutons servant à sélectionner directement des postes pré-définis. Dans Pure Data, ces sélecteurs à nombre fixe de choix sont constitués d'un groupe de carrés et un seul d'entre eux contient un carré noir plus petit. On crée un groupe horizontal de boutons radio [hradio] en choisissant l'item "Hradio" du menu "Put". De même, le groupe vertical se crée via l'item "Vradio". Leur fenêtre de propriétés permet de choisir la couleur du fond et celle de leur premier plan. Par l'intermédiaire de la fenêtre de dialogue "Properties", on peut sélectionner le nombre de boutons radio que l'on souhaite créer pour le [hradio] ou [vradio].

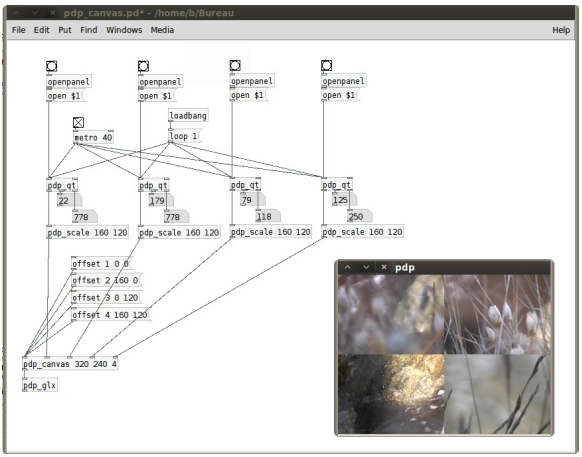


CANVAS

Le plus simple de ces gadgets est le "Canvas", qui n'est en fait qu'un rectangle de couleur. Cet objet peut être créé en écrivant [cnv] dans une boîte d'objet ou en choisissant l'item "Canvas" du menu "Put".

Nous pouvons changer certaines propriétés du "Canvas" en faisant un clic droit à l'intérieur du carré aux bordures bleues qui se trouve en haut à gauche du *canvas*.

Un menu spécifique au "Canvas" apparaît et vous pouvez choisir l'item "Properties" : ceci fait apparaître une fenêtre avec différentes propriétés, tel que sa couleur et sa taille. La caractéristique de pouvoir changer les propriétés d'un objet est valable pour la plupart des objets interactifs que nous trouvons dans la seconde section du menu "Put". Cette possibilité distingue les objets interactifs des boîtes objets de base vues précédemment.

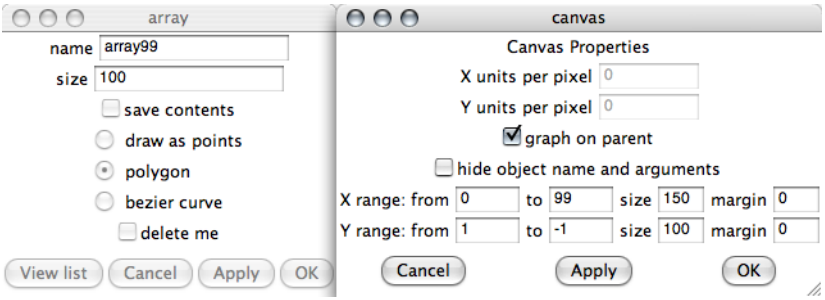


On peut également accompagner d'une étiquette de texte un patch ou un secteur du patch. Cela peut rendre les interfaces graphiques créées plus faciles à comprendre au premier coup d'œil. Le "Canvas" est alors bienvenu pour ce type d'utilisation.

Attention, un objet nouveau (ici le "Canvas") va venir se superposer à tout ce qui a été précédemment créé au même emplacement dans le patch. Le "Canvas" est souvent utilisé pour souligner une section particulière du patch, un objet, un commentaire, etc. ou pour mettre quelque chose en relief. Lorsque ces mises en relief interviennent après la création de la section que nous voulons pointer, le "Canvas" viendra recouvrir cette section. Dans ce cas, créez le "Canvas" en dehors de la zone concernée du patch, puis sélectionnez-le, coupez-le et collez la zone. De cette manière, le "Canvas" viendra se placer en dessous de la section à faire ressortir.

LES TABLEAUX

On doit souvent gérer de grandes listes de nombres, effectuer une seule et même opération sur tout un groupe de valeurs ou, encore, contenir les échantillons d'un son. En effet, un son numérique n'est qu'une série de chiffres indiquant la position de la membrane d'un micro ou d'une enceinte. Dans ce cas, on aura besoin de créer un tableau. Nous verrons ultérieurement une manière plus précise de les utiliser. Pour le moment, il s'agit de présenter une manière générale de les créer et de configurer certaines de leurs propriétés les plus courantes. Dans le menu "Put", sélectionnez et placez "Array". Un clic droit de souris permet d'accéder à ses propriétés ("Properties").

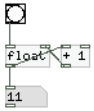

array
name
array99
size
100
save contents
draw as points
polygon
bezier curve
delete me
View list
Cancel
Apply
OK
canvas
Canvas Properties
X units per pixel
0
Y units per pixel
0
graph on parent
hide object name and arguments
X range: from
0
to
99
size
150
margin
0
Y range: from
1
to
-1
size
100
margin
0
Cancel
Apply
OK

17. QUELQUES OBJETS UTILES

Pour créer des objets, il suffit d'écrire du texte dans une boîte objet vide. Le texte est organisé en deux parties toujours séparées par un espace. Le premier mot définit le type d'objet et ceux qui suivent servent d'argument qui initialise l'objet en fonction de sa ou ses valeurs. Les exemples qui suivent illustrent ce principe.

[FLOAT] NOMBRES RÉELS

L'objet [float] enregistre un nombre réel avec une virgule flottante. Il envoie ce nombre via sa sortie quand une information lui est envoyée par son entrée gauche. Si un nombre lui arrive par son entrée droite, ce nombre change sa valeur interne mais sans l'évacuer par sa sortie. Il permet donc de stocker de manière temporaire une valeur pour envoyer cette valeur par ailleurs au moment souhaité.
L'exemple ci-dessous présente un compteur. Celui-ci commence à zéro, soit la valeur par défaut d'un [float], et il augmente de 1 chaque fois que nous cliquons sur le bouton "Bang".



[LIST] TRAITEMENT DES LISTES

Dans Pure Data, les messages peuvent être catégorisés en deux groupes :

- les messages contenant des données directes ("bang", nombres, symboles, listes) qui sont la principale façon de communiquer ;
- les messages contenant « tout le reste » et qui peuvent, par exemple, servir à changer des arguments de configuration des objets, lire et écrire des fichiers, quitter Pure Data, etc.

Ces derniers messages existent de sorte que des objets complexes ne nécessitent pas un grand nombre d'entrées et soient simples à configurer.

L'objet [list] permet de travailler des messages comprenant plus d'un élément en les ajoutant, les supprimant, les divisant dans la liste ou en extrayant le nombre d'éléments.

[LOADBANG] INITIALISATION DE PARAMÈTRE

L'objet [loadbang] génère un bang unique à chaque démarrage du patch, vers son unique sortie (*outlet*). Cela permet d'automatiser une action en envoyant un paramètre à un objet lorsque par exemple il ne permet pas d'inscrire celui-ci en argument. L'objet [loadbang] ne possède pas d'argument, il se déclenche une fois seulement.

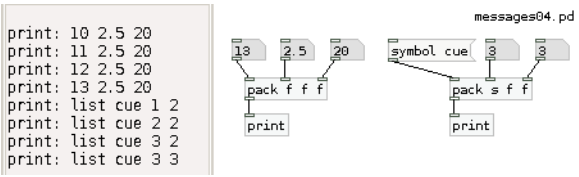
[METRO] MÉTRONOME

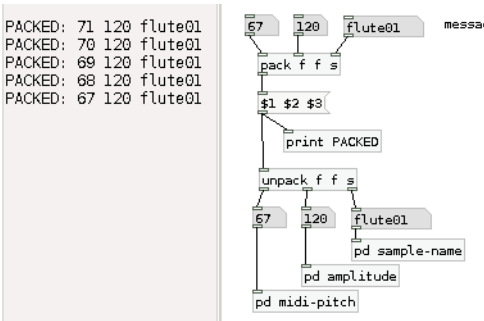
L'objet [metro] est un métronome dont nous pouvons contrôler le rythme. Il envoie un "bang" chaque x millième(s) de seconde. Nous spécifions sa longueur d'onde (= 1/fréquence) en argument. Par exemple, [metro 50] envoie une impulsion toutes les 50 millisecondes soit 20 impulsions par seconde.

Pour lancer le métronome, nous envoyons un "bang" ou un message [1< dans l'entrée de gauche du métronome. Pour modifier le rythme, l'argument est envoyé dans son entrée de droite. Cette opération arrêterait le métronome s'il fonctionnait. L'envoi du message [0< dans l'entrée de gauche a en général pour effet de l'arrêter, ou nous pouvons encore utiliser un [toggle] comme interrupteur.

[PACK] ET [UNPACK] TRAITEMENT DE LISTES DE MESSAGES

L'objet [pack] permet de rassembler des messages et [unpack] les sépare.





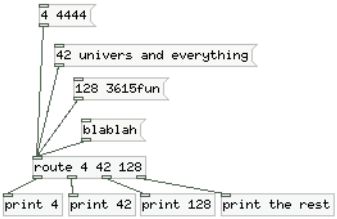
[PRINT] AFFICHER DU TEXTE DANS LA CONSOLE

Cet objet imprime les messages qui lui sont envoyés dans la fenêtre de la console et vous permet ainsi de voir l'ordre et le contenu des messages qui lui parviennent : ceci est très utile pour contrôler le bon déroulement des opérations. L'argument qui vient après "print" dans l'objet est une étiquette qui sera ajoutée en début de ligne lors de l'impression des messages, et il permet de différencier plusieurs messages imprimés dans la fenêtre de la console.

[ROUTE] AIGUILLAGE DE DONNÉES

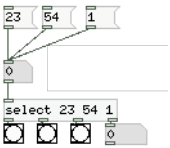
Cet objet permet de sélectionner le chemin emprunté par les messages en fonction du premier atome du message (c'est-à-dire la première partie des messages qui sont des nombres). Les arguments de "Route" déterminent le type des messages attendus et permettent de les comparer aux entrées.

Dans l'exemple ci-dessous, s'ils ne correspondent pas, ils sortent par la sortie de droite, sinon les chaînes de caractères qui suivent les atomes dans les messages sont imprimées dans la fenêtre de la console grâce à [print].

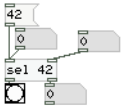


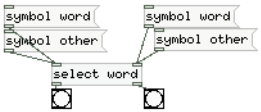
[SELECT] COMPARAISON DE VALEUR

[select] permet de comparer l'entrée d'une valeur dans l'objet à une valeur ou à un nombre donné de valeurs définies comme arguments. Si la valeur entrée est égale à l'une des valeurs argumentées dans "Select", un "Bang" est émis dans la sortie correspondante. L'ordre des "Bangs" suit l'ordre des valeurs argumentées dans "Select". Les valeurs qui ne correspondent pas à celles testées sortent non transformées dans la sortie de droite.



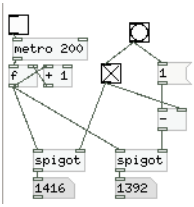
[select] peut prendre des symboles comme arguments. Le premier argument détermine le type de donnée qu'il attend (float ou symbol). Si [select] (qui peut être abrégé en [sel]) est créé sans argument ou avec un seul argument, l'entrée de droite de [select] sert à fixer cet argument avec un « nombre » et ceci permet de modifier l'argument en intervenant par ailleurs dans le patch.





[SPIGOT] INTERRUPTEUR DE FLUX

L'objet [spigot] permet de créer des interrupteurs qui bloquent ou laissent passer le message.



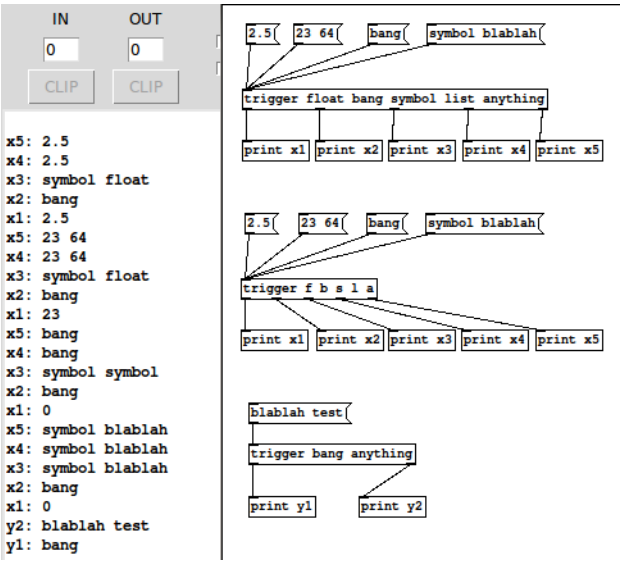
L'entrée de droite ouvre le canal lorsqu'il reçoit 1 et le ferme avec 0. L'exemple ci-dessus fait passer alternativement les données à droite ou à gauche lorsqu'on clique sur le "Bang".

[SYMBOL] CHAINES DE CARACTÈRE

L'objet [symbol] fonctionne de la même manière que [float] pour des symboles (chaînes de caractères).

[TRIGGER] DÉCLENCHEMENT CONTRÔLÉ

L'objet [trigger] est un objet très utile pour contrôler l'ordre des opérations que l'on doit effectuer. Celui-ci prend une valeur d'entrée, convertit cette valeur en fonction des arguments (soit les arguments : float, bang, symbol, list et/ou anyting) et sort les nouvelles valeurs dans les sorties, en commençant par sa sortie la plus à droite et en terminant par celle la plus à gauche. Il est à noter que le nombre de sortie (*outlet*) varie selon le nombre d'argument ajouté.




```
x5: 2.5
x4: 2.5
x3: symbol float
x2: bang
x1: 2.5
x5: 23 64
x4: 23 64
x3: symbol float
x2: bang
x1: 23
x5: bang
x4: bang
x3: symbol symbol
x2: bang
x1: 0
x5: symbol blablah
x4: symbol blablah
x3: symbol blablah
x2: bang
x1: 0
y2: blablah test
y1: bang
```

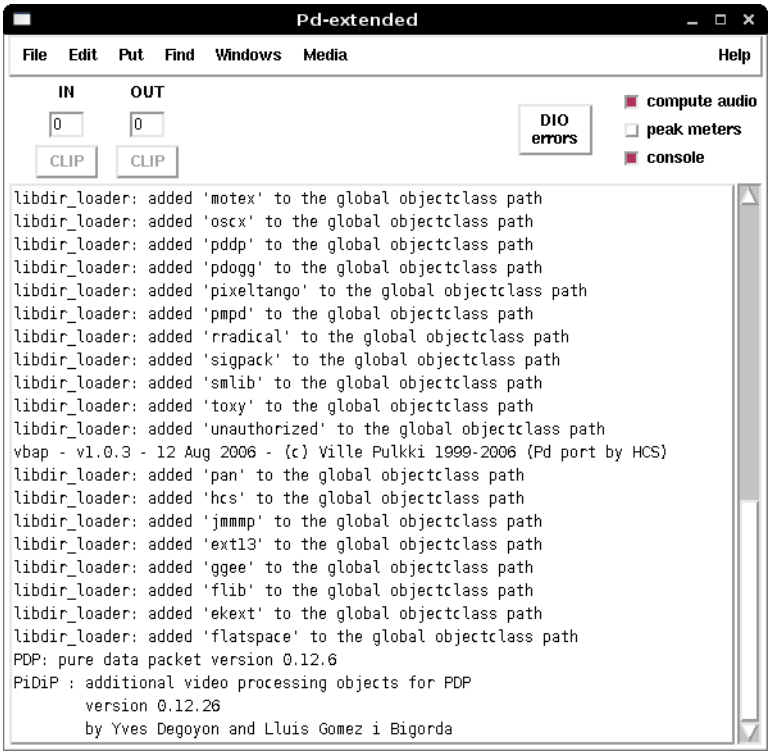
18. MON PREMIER PATCH

COMMENT CONSTRUIRE UN PREMIER PATCH DE SYNTHÈSE SONORE ALÉATOIRE ?

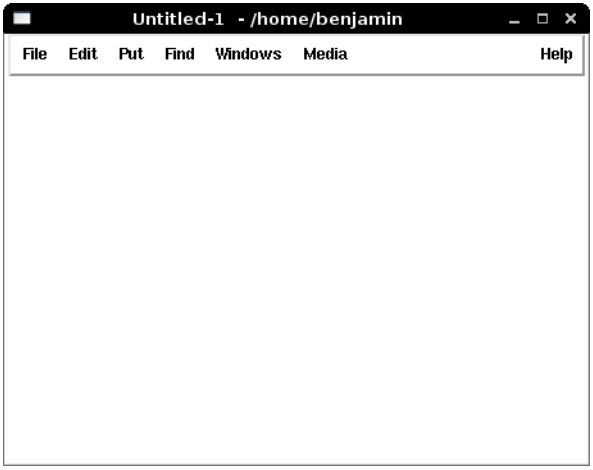
Nous allons à présent décrire pas à pas la construction d'un patch qui permettra d'émettre un son dont nous pourrons moduler le tempo avec un métronome et modifier la fréquence de façon aléatoire.

1. Lancez Pure Data.
2. Vérifiez que l'audio fonctionne bien : allez dans le menu "Media" de l'interface principale puis cliquez sur "Test audio and MIDI".
3. Un patch s'ouvre. Cliquez sous "TEST TONES" dans la case la plus haute du rectangle de gauche, à côté de "80". Vous devriez alors entendre un son continu. Si ce n'est pas le cas, référez-vous à la partie « Configuration de l'audio et du MIDI » de ce manuel.
4. N'oubliez pas de monter le volume dans la console audio de votre système d'exploitation.
5. Vérifiez que le bouton "Compute Audio" est bien coché, faute de quoi Pure Data ne produira pas de son.

Attention : Si vous êtes passés par l'étape "Test audio and MIDI", le bouton "Compute Audio" est normalement coché de façon automatique.



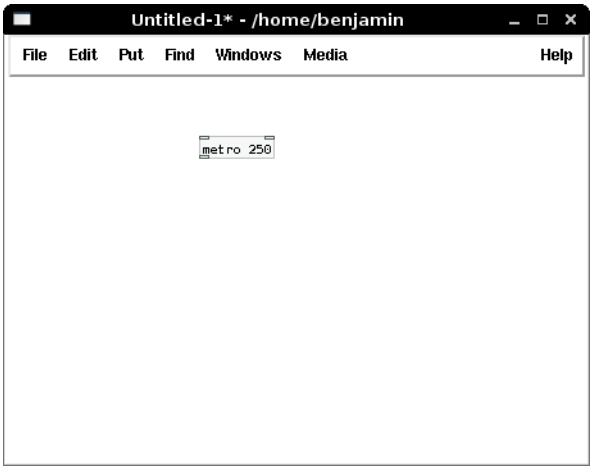
6. Pour créer un nouvel espace de travail (ou patch), allez dans le menu de l'interface dans "File" et cliquez sur "New". Un espace vierge s'ouvre alors, comme le montre l'exemple ci-dessous.



7. Pour y placer un premier objet, allez dans le menu principal de l'interface, puis dans "Put" et sélectionnez "Object". Un rectangle pointillé bleu apparaît sur l'espace de travail, cliquez une fois sur cet espace avec la souris pour déposer l'objet à l'endroit voulu. Tapez ensuite dans cette boîte "Object " : "metro 250" (metro-*espace*-250) puis cliquez une fois à l'extérieur de l'objet dans l'espace de travail blanc pour valider.

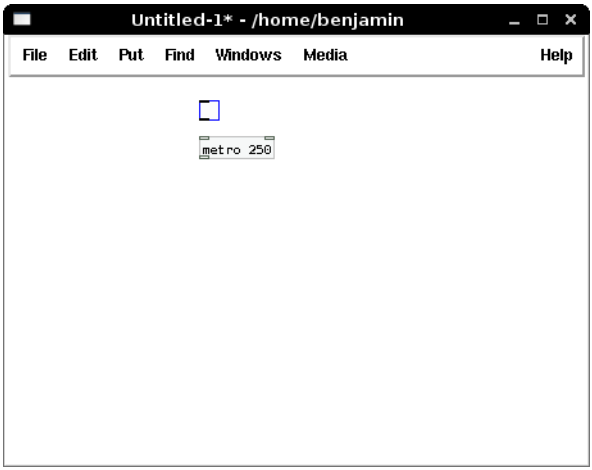
Si tout va bien, les pointillés bleus du rectangle disparaissent pour laisser place à un rectangle noir.

En cas de problème, les pointillés deviennent rouges. Cela signifie que Pure Data n'a pas réussi à interpréter ce qui a été tapé dans l'objet : il doit y avoir une erreur de syntaxe. Réessayez à partir du début de ce point 7.



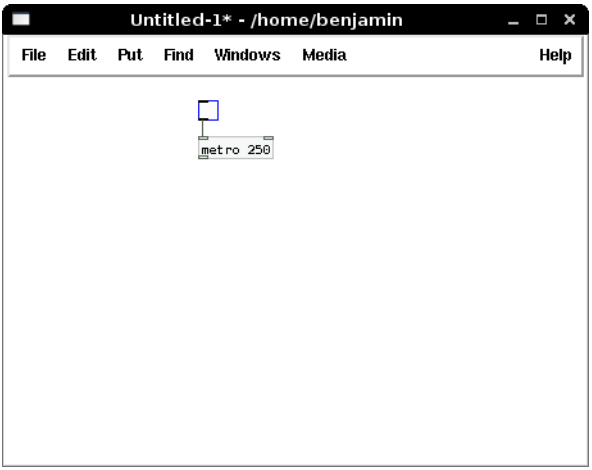
Cette boîte de l'objet [metro 250] a pour fonction d'envoyer une impulsion toutes les 250 millisecondes. Pour actionner le métronome, il faut l'allumer.

8. Aller dans le menu "Put" puis sélectionnez "Toggle". Placez le petit carré qui apparaît au-dessus de l'objet [metro 250] en cliquant une fois dans l'espace de travail à l'endroit voulu.



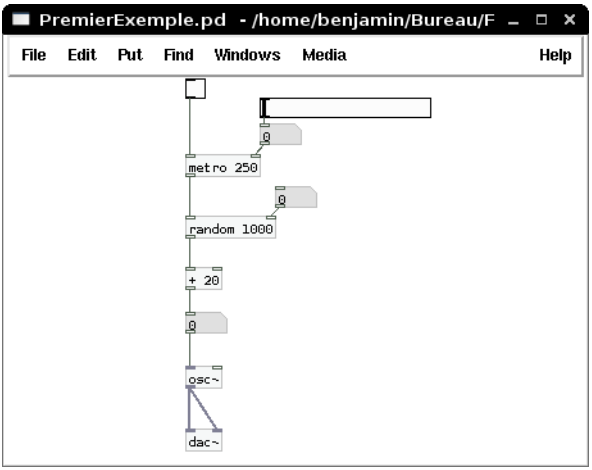
Cet interrupteur (*toggle*) va nous permettre d'allumer et d'éteindre le métronome : il faut donc les relier. Vous aurez certainement remarqué que la boîte interrupteur et la boîte de l'objet [metro 250] comportent de minuscules rectangles à leurs angles : ces mini-rectangles sont pleins pour l'interrupteur et vides pour le métronome. Il s'agit d'entrées (*inlets*) et de sorties (*outlets*) qui se relient en allant du haut vers le bas.

Placez la souris sur le petit rectangle noir en bas de l'interrupteur. Quand le curseur de la souris y est bien placé, il change d'aspect et fait apparaître un petit cercle au contour noir et épais (parfois barré sur certaines versions). Ce cercle indique que l'on va pouvoir tirer un fil. Sans bouger le curseur, faites un clic gauche sur ce petit rectangle noir, maintenez le clic jusqu'à rejoindre le rectangle évidé au sommet gauche de l'objet [metro 250] : vous devez voir un fil noir sortir de l'interrupteur. Lâchez le clic quand le curseur de la souris se transforme à nouveau en cercle au-dessus de l'objet [metro 250]. Un fil relie alors les deux boîtes.



9. Enregistrez votre patch. Allez dans le menu "File", puis sélectionnez "Save As". Choisissez un emplacement et un nom de fichier. L'extension par défaut du fichier enregistré est ".pd", qui est ajoutée automatiquement.

10. Reconstituez maintenant le patch ci-dessous. Il faudra notamment utiliser la glissière (*Hslider*) et la boîte nombre (*Number*) et créer des objets audio dont le nom se termine par le signe "tilde": "~" ("Alt Gr + 2" sur PC et "Alt + n" sur Mac OS X pour les claviers français).



Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/Prise_en_main/PremierExemple.pd

11. Pour tester ce patch, passez du mode "Edition" dans lequel nous nous trouvions (qui permet d'ajouter des boîtes et les relier par des fils) au mode "Action". Le curseur de la souris change alors d'aspect. En mode "Edition", il est représenté par une petite main ; en mode "Exécution", il redevient la flèche habituelle de la souris (voir chapitre "L'interface utilisateur).

Cliquer sur l'interrupteur tout en haut du patch.

Les enceintes de l'ordinateur doivent émettre un son.

Note: Pensez à vérifier de nouveau que "Compute Audio" est coché dans l'interface principale de Pure Data.

Vous pouvez ainsi vous exercer à changer les valeurs des boîtes « nombre ». Pour modifier les valeurs, cliquez dans la boîte en maintenant le clic et faites monter la souris verticalement, vers le haut ou vers le bas. Faites aussi glisser le curseur de la glissière pour voir quels changements ces actions provoquent.

COMMENT ÇA MARCHE ?

L'interrupteur permet d'enclencher le métronome, qui envoie une impulsion toutes les 250 millisecondes à l'objet [random]. À chaque impulsion, le [random] va sortir un nombre aléatoire entre 0 et 999 (car il offre 1000 possibilités). À ce nombre aléatoire, on ajoute la valeur 20 pour être sûr que le son produit sera dans le domaine de l'audible. La valeur résultante entre dans l'objet [osc~] qui va générer une onde sinusoïdale à la fréquence correspondante (l'oreille humaine entend normalement des sons compris dans une plage de fréquence de 20 Hz à 20 000 Hz).

À chaque impulsion du métronome, le nombre aléatoire tiré change, ce qui modifie aussi le son synthétisé.

En modifiant les deux boîtes « nombre » en haut du patch, la valeur par défaut indiquée à l'intérieur de l'objet (le "250" du [metro 250]) est modifiée par celle choisie à l'aide de la souris. Ces valeurs seront modifiées tant que le patch est ouvert, la valeur par défaut n'étant active qu'à l'ouverture du patch.

Notez que la boîte "nombre" en bas du patch ne peut être modifiée par la souris : des données lui sont en effet envoyées toutes les 250 millisecondes.

Attention : générer un son en dessous de 20 Hertz peut endommager vos enceintes, voire vos oreilles.

19. ORGANISATION DES OBJETS

Il existe plusieurs façons d'organiser ses patchs. Nous présentons ceux rencontrés fréquemment.

LES SOUS-PATCHS

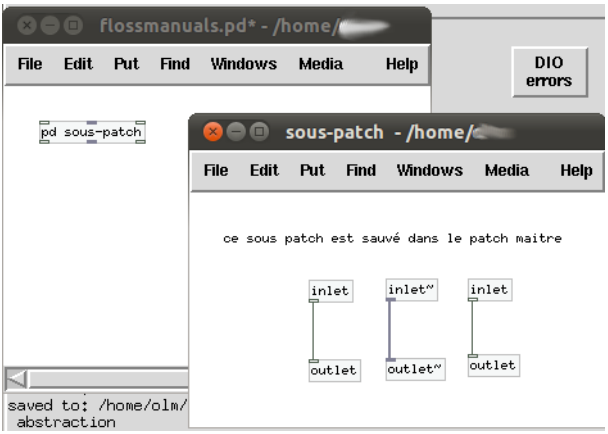
Lors de la rédaction d'un patch, il arrive souvent que la taille d'une seule fenêtre ne soit plus adaptée à la complexité du projet. Pour séparer les parties d'un système plus complexe, on peut dès lors créer des **sous-patchs**. Ce sont des boîtes ressemblant à des objets, mais qui contiennent une portion du patch qui s'ouvre dans une autre fenêtre (pensez aux poupées russes qui s'emboîtent les unes dans les autres). Le sous-patch appartient entièrement au patch principal. Le patch principal pourra être déplacé (dans un autre répertoire, partagé en ligne). Le ou les sous-patchs intégrés au patch principal seront également déplacés ou partagé en même temps (contrairement à l'abstraction).

Pour créer un sous-patch, il suffit de créer une boîte "object" dans laquelle on écrit : [pd nomdusous-patch].

Dans le sous-patch, si l'on veut créer des entrées et des sorties pour les données, il faut insérer des objets **[inlet]** et **[outlet]**. Pour faire transiter le son, on utilise les objets **[inlet~]** et **[outlet~]**. Il est également possible d'utiliser les messages sans ficelles. Il faudra alors prêter une attention particulière à la notion de variables globales et locales. Reportez-vous au chapitre « Flot des données » pour de plus amples informations à ce sujet.

Il est à noter que s'il est possible de dupliquer des sous-patchs, les modifications effectuées dans l'un ne se répercuteront pas dans les autres. Conséquence directe : les sous-patchs de même nom peuvent avoir des contenus différents. Par souci de clarté, il est alors recommandé de nommer différemment les sous-patchs au contenu différent.

Si un sous-patch est amené à être utilisé à l'identique plusieurs fois dans le même patch, il peut être souhaitable de créer une abstraction pour faciliter d'éventuelles modifications ultérieures.



LES ABSTRACTIONS

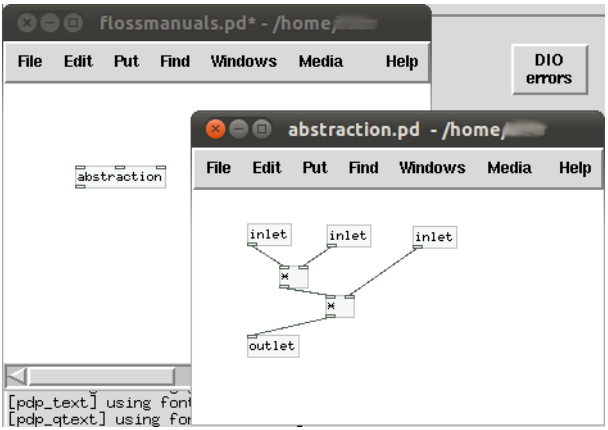
Les abstractions sont des patchs prévus pour être réutilisés comme raccourcis de programmation. Pratiquement identiques à une boîte objet (fonctions), on peut en revanche les « ouvrir », à la différence des objets natifs de Pure Data. En mode action, un clic de souris sur une abstraction suffit pour voir ce qu'elle contient.

Nous pouvons donc écrire nos propres abstractions. Ces abstractions sont ensuite appelées comme des objets ordinaires depuis le patch principal nommé aussi « patch maître ». À la différence d'un sous-patch, lorsqu'on partage ou déplace le patch principal dans un autre répertoire, on doit réitérer l'opération pour les abstractions qui lui sont attachées. Sinon, on obtient le même résultat que pour un objet qui refuse de « naître », c'est-à-dire des pointillés rouges entourant le nom de l'abstraction (objets non reconnus).

Pour simplifier l'écriture d'un projet, l'abstraction peut être soit sauvegardée dans un dossier spécifié de la configuration des chemins « regardés par Pure data » ou, à défaut, dans le même dossier que le « patch maître » qui appelle cette abstraction. Pour configurer les chemins, allez dans le menu "File", sélectionnez "Path", puis ajoutez le chemin où se trouve l'abstraction désirée (voir la section « Configuration des chemins »).

Une abstraction est donc indépendante du « patch maître » et peut être facilement réutilisée d'un projet à l'autre.

L'un des intérêts de l'abstraction réside dans le fait que toute modification effectuée dans l'une d'entre elles se répercutera, une fois l'abstraction sauvegardée, dans toutes les autres présentes dans le patch.

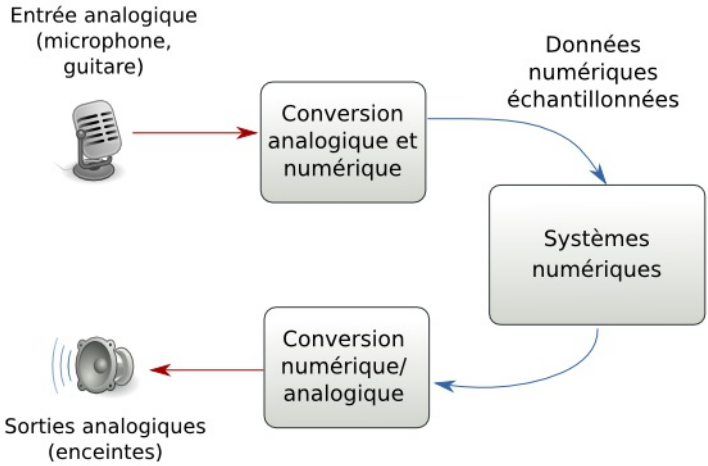

flossmanuals.pd* - /home/
File Edit Put Find Windows Media Help
DIO errors
abstraction
abstraction.pd - /home/
File Edit Put Find Windows Media Help
inlet
inlet
inlet
outlet
[pdp_text] using font
[pdp_qtext] using for

AUDIO

20. LES BASES DE L'AUDIO NUMÉRIQUE
21. L'AUDIO DANS PURE DATA
22. L'ÉCHANTILLONNAGE
23. LIRE ET ENREGISTRER DE L'AUDIO
24. LA SYNTHÈSE AUDIO
25. LES EFFETS
26. L'ANALYSE AUDIO

20. LES BASES DE L'AUDIO NUMÉRIQUE

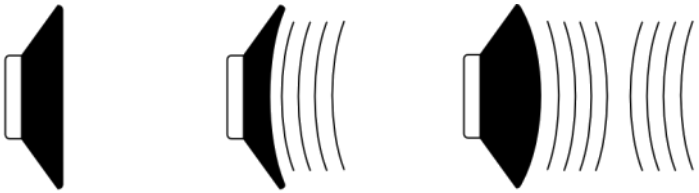
Pure Data traite le son de manière numérique, c'est-à-dire, comme une simple suite de nombre. La notion d'audio numérique est fondamentale pour travailler la matière sonore, et nous allons décrire ici ce que l'on entend par « son numérique ».



Ce diagramme montre comment le « son » passe par l'ordinateur. Les conversions de l'analogique vers le numérique et du numérique à l'analogique sont réalisées au sein de la carte son. Le cadre intitulé « Systèmes numériques » pourrait ici représenter Pure Data.

FRÉQUENCE ET GAIN

La membrane d'une enceinte acoustique, par exemple, fait vibrer l'air autour elle. Ces vibrations sont responsables du phénomène du « son ». La membrane vibre d'avant (figure 2. avec voltage négatif) en arrière (figure 3. avec voltage positif) depuis son état de repos (figure 1. membrane au repos). Le nombre de vibrations par seconde est la **fréquence** (la note, le ton ou la hauteur) du son que nous percevons et la distance que la membrane parcourt depuis son état de repos détermine le **gain** (le volume ou la puissance) du son. Normalement, on mesure la fréquence en **hertz** (Hz) et le volume/gain en **décibels** (dB).



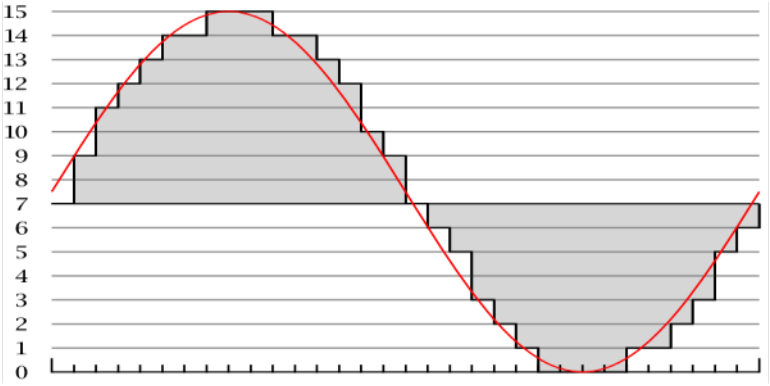
1. membrane au repos - 2. avec voltage négatif - 3. avec voltage positif

Un microphone fonctionne à l'inverse : l'air fait vibrer sa membrane et cette vibration modifie l'intensité du courant électrique. Lorsqu'on branche un microphone sur l'entrée de la carte son de l'ordinateur et que l'on commence à enregistrer, la carte son prend la mesure de ce courant électrique des milliers de fois par seconde et les enregistre comme des nombres.

FRÉQUENCE ET PRÉCISION D'ÉCHANTILLONNAGE

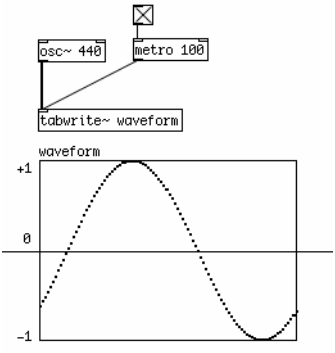
Pour que l'audio soit lisible par un lecteur numérique (comme pour un *Compact Disc*), l'ordinateur doit relever, par exemple, 44 100 mesures par seconde (selon la configuration de la carte son). Ces mesures sont nommées **échantillons** ou *samples*. L'ordinateur enregistre chacune de ces mesures comme un nombre sur **16 bits**. Un bit est une parcelle d'information qui peut prendre la valeur 0 ou 1, cette valeur est l'information la plus petite que l'ordinateur puisse stocker. En combinant les bits ensemble pour créer un échantillon, on obtient 2^{16} (ou 2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2 = 65536) valeurs possibles par échantillon. Donc la « qualité CD » a une fréquence d'échantillonnage de 44 100 Hz et une précision de 16 bits .

Par contraste, les enregistrements sonores professionnels peuvent être réalisés à une précision de 24 bits et une fréquence de 96 kHz (96 000 Hz) pour conserver un maximum de détails durant le mixage ; ces enregistrements sont ensuite réduits à 16 bits et 44 100 Hz pour être diffusés sur CD. Les anciens jeux vidéo sont connus pour avoir une précision de 8 bits donnant un « son » caractéristique. En augmentant la fréquence d'échantillonnage, on peut enregistrer des fréquences plus élevées. En enregistrant avec une précision plus grande, on augmentera la **dynamique** (la différence entre l'amplitude la plus faible et la plus forte).



Un exemple d'échantillonnage de 4 bits (en rouge) permet 16 valeurs donc la dynamique est très faible... Source: http://en.wikipedia.org/wiki/Image:Pcm.svg

Dans Pure Data, la valeur oscille entre -1 et +1. Ces seuils représentent la plus grande amplitude possible de notre enceinte théorique avec 0 comme état de repos.



Représentation graphique d'une onde sinusoïdale dans Pure Data.

Lorsque Pure Data joue le son du patch ci-dessus, il lit les échantillons et les envoie à la carte son. La carte son convertit les échantillons en courant électrique, lequel fait alors vibrer la membrane de l'enceinte. La membrane, quant à elle, fait vibrer à son tour l'air, de sorte que nous puissions entendre le son.

Si le taux d'échantillonnage est de 44 100 Hz et que l'on essaie de travailler avec des fréquences supérieures à la moitié de la fréquence d'échantillonnage, donc supérieur à 22 050 Hz (qui correspond au **nombre de Nyquist**), des fréquences parasites deviennent audibles.

VITESSE ET HAUTEUR

La « hauteur » d'une onde sonore est déterminée par sa fréquence. Plus la fréquence est élevée, plus le son semble aigu.

Lorsque les échantillons sont lus plus vite ou plus lentement que la vitesse originale, la fréquence d'échantillonnage est alors modifiée par ce changement de vitesse. Comme on peut le constater avec une bande magnétique ou un disque vinyle, les fréquences sonores sont modifiées par la vitesse de lecture. La hauteur perçue sera plus aiguë si la lecture de l'échantillon est accélérée et plus grave si elle est ralentie.

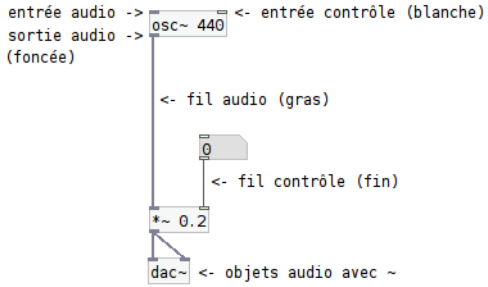
21. L'AUDIO DANS PURE DATA

Pure Data possède toute une gamme d'objets audio natifs (version de base de Pure Data) ou externes (librairies) qui permettent de créer des algorithmes sonores complexes. Avant d'aller plus loin, il faut comprendre comment l'audio est traité dans Pure Data.

LES OBJETS AUDIO

Les objets audio dans Pure Data se distinguent des autres objets par trois informations visuelles :

1. Le signe "tilde" - "~" est placé à la fin d'un objet.
2. Les rectangles représentant les entrées et sorties de l'objet sont noirs ou foncés.
3. Les ficelles de connexion portant du signal audio sont en gras.



Rappel : Pour faire le ~ sur un clavier français, sur PC : altGr + "2" + espace, sur MAC : alt + "ç".

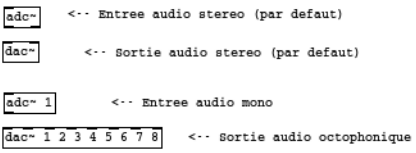
CADENCE DES OBJETS AUDIO~ ET CONTRÔLE

Les objets audio et les objets de contrôle ne fonctionnent pas à la même cadence. Les objets audio fonctionnent en continu (*stream*) à la cadence spécifiée dans les configurations audio (*Audio Settings*) ; quant aux objets de contrôle, ils fonctionnent beaucoup plus lentement. Ces différences de cadence peuvent générer des artefacts audio, des « clics », qu'il est préférable d'éviter. On peut donc contrôler les objets audio comme [*~] avec des signaux audio générés avec [line~] plutôt qu'avec des boîtes de message.

[ADC~] ET [DAC~] : ENTRÉES ET SORTIES SONORES

Dans Pure Data, l'objet permettant d'obtenir des signaux audio entrants (un microphone, par exemple) s'appelle [adc~] qui en anglais signifie *analog-digital converter*. Ce nom désigne la conversion du signal audio analogique, en provenance d'un micro par exemple, en signal numérique par l'interface audio utilisable dans Pure Data. De la même manière, l'objet qui envoie du son vers la sortie de l'interface audio s'appelle [dac~] soit *digital-analog converter*.

Attention : Dans Pure Data, il n'y a pas de limites au traitement du signal, qui peut alors dépasser les normes numériques du son comprises entre -1 et 1. Par conséquent, le volume sonore généré par Pure Data peut être plus puissant que la capacité de la carte son ou des haut-parleurs : il convient donc d'être prudent ! Vous pouvez intercaler d'autres objets pour contrôler ce signal, comme par exemple un filtre qui ne laissera passer que les fréquences supérieures à 20 Hz : [hip~ 20]. L'objet [clip~ -0.8 0.8], quant à lui, forcera le signal à rester dans des limites comprises entre -0,8 et 0,8, valeurs communément admises par la carte son.

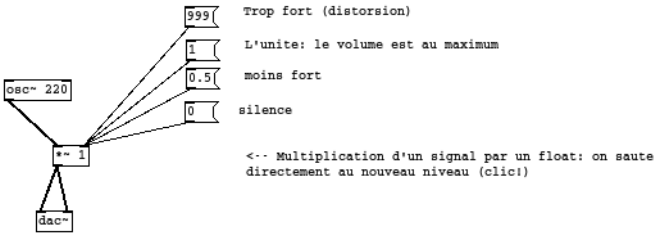


Les deux objets, [adc~] et [dac~], acceptent une liste de numéros de canaux audio. La succession des chiffres indique l'ordre par lequel on souhaite accéder aux canaux de l'interface audio. Si aucun argument n'est fourni, les deux objets sont par défaut les deux premiers canaux : ceci donnera du son en stéréo que l'on pourra recevoir et envoyer. Si l'interface audio possède plusieurs canaux, on peut spécifier le numéro d'un canal en particulier. Par exemple, si l'on donne le chiffre 10 comme seul argument, notre objet [dac~] possède une seule entrée qui enverra du son à la sortie numéro 10 de notre interface audio.

[*~] : MODIFIER LE VOLUME SONORE

Une onde synthétisée avec [osc~] a l'amplitude maximale pour la sortie audio de votre ordinateur. Elle a comme sommets les nombres -1 et 1. Si l'on souhaite mixer ensemble plusieurs sons, il faudra atténuer un peu chacun d'entre eux afin que l'ensemble ait un volume sonore moins élevé.

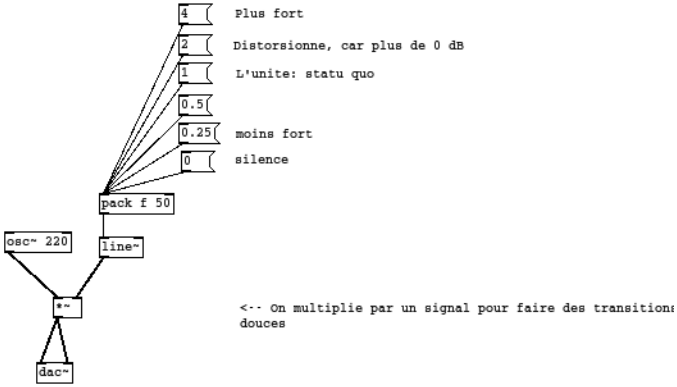
Pour augmenter ou baisser le volume d'un son, il suffit de le multiplier par un nombre avec l'objet [*~]. S'il est multiplié par 1, [*~ 1], son volume demeure inchangé, par 0, [*~ 0], il est réduit au silence. On peut ainsi changer l'amplitude de l'onde sonore en multipliant les échantillons par un coefficient.



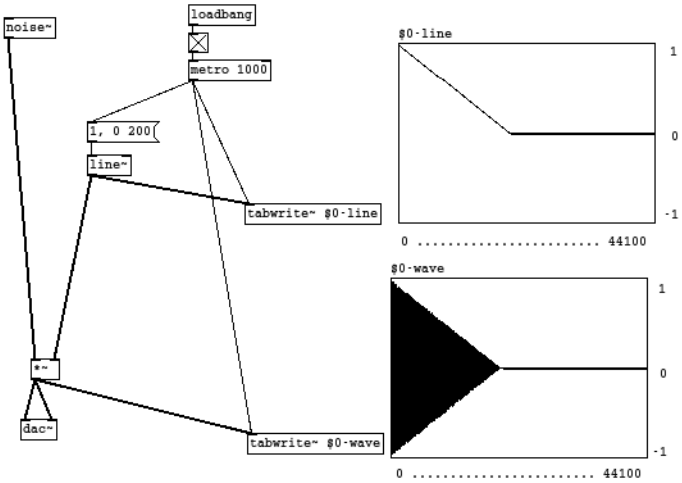
Rappel : ne jamais dépasser 1, cela pourrait endommager vos enceintes !

[LINE~] : ENVELOPPE SONORE

Sauter directement d'un niveau sonore à un autre est un peu brusque. On entend ainsi un « clic » lorsque le son change subitement. Il est donc préférable d'utiliser un signal audio branché dans la seconde entrée de l'objet [*~] pour obtenir des transitions plus fluides comme dans l'exemple ci-dessous, afin de moduler le volume à la vitesse du son.



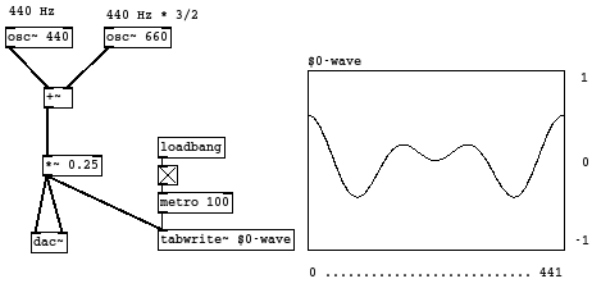
Pour faire des transitions audio, on augmente ou réduit graduellement le volume d'un son. L'enveloppe d'un son correspond à la forme de la courbe de variation de son volume dans le temps. Les parties qui nous intéressent le plus sont surtout le début et la fin de cette courbe. Si son volume sonore est soudainement très fort, on dit que son **attaque** est franche. Si son volume commence par le silence pour augmenter graduellement, son attaque est plus douce. On peut provoquer des interpolations linéaires de cette courbe au moyen de l'objet [line~]. Le signal qui en résulte peut être utilisé pour contrôler le volume d'un signal sonore.



L'objet [line~] permet de créer des rampes, c'est-à-dire l'interpolation linéaire d'un signal audio. On peut passer de n'importe quelle valeur à n'importe quelle autre dans un temps donné. Cet objet comporte deux nombres : une cible et une durée en millisecondes. Lorsque l'objet reçoit un message, le signal qui en sort part de sa valeur actuelle et se déplace progressivement vers sa cible en prenant la durée qui lui a été impartie. Pour le faire sauter directement à une valeur, sans transition, on lui envoie un seul nombre. La durée de l'interpolation est alors nulle. Une fois arrivé à destination, le signal y est maintenu jusqu'à ce que l'objet reçoive un nouveau message. Si le signal reçoit un message alors qu'il est en cours de route, il part de l'endroit qu'il a atteint pour suivre sa nouvelle trajectoire. Le signal qui résulte de cet objet peut être utilisé pour créer des enveloppes, mais aussi pour varier avec finesse la hauteur d'une note de musique, ou la position de lecture d'un fichier vidéo par exemple.

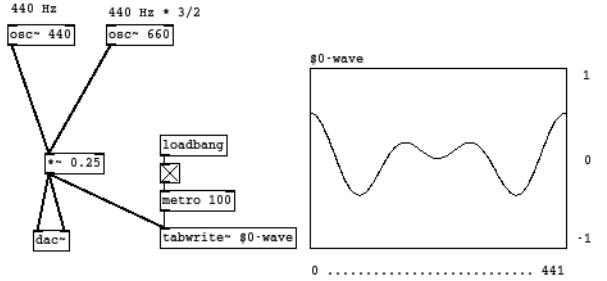
[+~] : Mixer plusieurs sons ensemble

Pour mélanger plusieurs signaux audio ensemble, il suffit de les additionner : c'est l'objet [+~] qui le permet. [+~] a deux entrées pour les valeurs à additionner, et sa sortie donne le résultat de cette addition.



Une autre manière d'additionner deux sons consiste à brancher plusieurs signaux sonores dans la même entrée d'un objet. Le résultat est le même que celui opéré par [+~].

Si l'on additionne deux signaux audio, il se peut que le volume sonore du résultat soit trop fort. Pour cette raison, il vaut mieux se garder une marge de manœuvre et les atténuer avant leur addition. Par exemple, vous pouvez brancher un [*~ 0.25] avant de les additionner. Placer [+~] après l'addition est tout aussi valable.



Lorsqu'un échantillon dépasse la valeur de -1 ou celle de 1, il est tronqué à la valeur maximale (-1 ou 1) et il en résulte un son distordu à cause de la perte d'informations et de l'apparition de fréquences parasites. Le son est dit « saturé ».

TAILLE DES BLOCS AUDIO

Pure Data travaille avec le son par paquets ou blocs. Un bloc représente le nombre d'échantillons audio qui sont traités en une fois avant d'être envoyés à la sortie audio ou au traitement numérique suivant. La taille des blocs par défaut est de 64 échantillons : ce qui signifie que chaque fois que sont lus les 64 échantillons, Pure Data fait le calcul nécessaire sur le son. Lorsque ce calcul est terminé, le patch produit le son en sortie. On peut modifier la taille des blocs avec l'objet [block~]. Cette opération est parfois nécessaire pour traiter le son et la vidéo en synchro.

22. L'ÉCHANTILLONNAGE

Pure Data permet de lire des fichiers sonores et d'en enregistrer de nouveaux. L'échantillonnage consiste à enregistrer et jouer ces sons. Dans Pure Data, on peut choisir de lire et d'enregistrer les sons directement sur le disque dur ou en *streaming* via le réseau, ou bien de les stocker dans des tableaux.

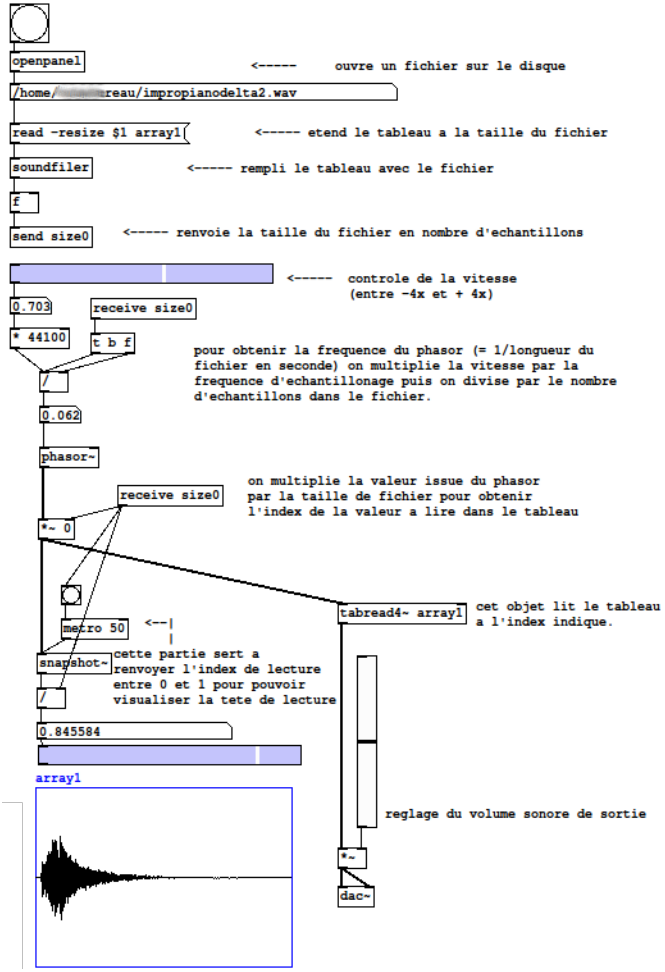
Les tableaux contiennent une liste de nombres que l'on peut consulter et modifier en cas de besoin. On peut voir le contenu de ces tableaux sous la forme d'un graphique rectangulaire. On peut même y dessiner avec la souris. Les nombres qui y sont stockés peuvent représenter n'importe quoi, mais le plus souvent ce sont des échantillons sonores. Dans le cas d'un stockage de son, l'axe horizontal représente le temps et l'axe vertical l'amplitude. En résumé, c'est la variation dans le temps de la membrane d'une enceinte audio que nous pouvons visualiser.

LES TABLEAUX

Dans Pure Data, les tableaux permettent de gérer des listes de nombres comme des échantillons sonores ou des enveloppes de contrôle. Pour créer un tableau, sélectionnez "Array" dans le menu "Put". Un dialogue apparaît pour aider à choisir notamment le nom du tableau et le nombre d'échantillons que l'on souhaite y stocker. En utilisant le message [resize], on peut aussi modifier la taille du tableau. Les tableaux sont inclus dans les "graphs", lesquels peuvent contenir plus d'un tableau. Dans ce cas, néanmoins, ils ne peuvent se réajuster d'eux-mêmes lorsqu'on modifie la taille de tableaux.

[TABREAD4~] : LIRE UN SON DEPUIS UN TABLEAU

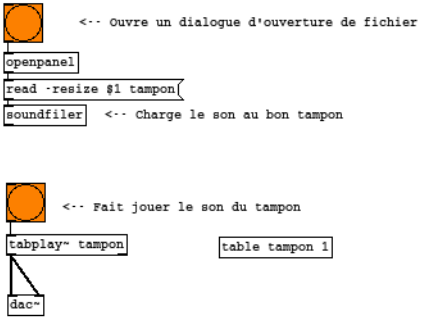
Voici un exemple de patch qui charge un fichier audio (.wav ou .aiff) et le lit en boucle à vitesse variable.



C'est l'objet [tabread~] qui lit le contenu du tableau et donne le signal audio qui en résulte.

[TABPLAY~] : JOUER SIMPLEMENT UN SON

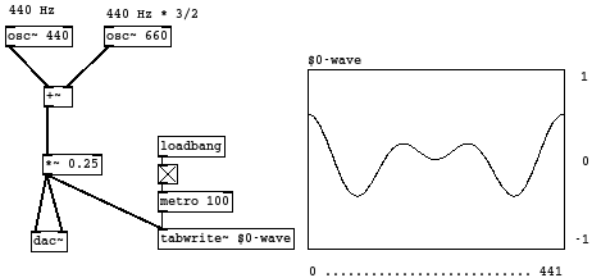
L'objet [tabplay~] offre un moyen très simple de jouer un son contenu dans un tableau du début à la fin. Il suffit de lui envoyer un "bang". On peut également commencer la lecture du tableau à partir de n'importe quel endroit.



Dans l'exemple ci-dessus, il importe de charger un fichier sonore dans le tableau avant de le lire. Pour ce faire, appuyez sur le bouton "bang" au-dessus de l'objet [openpanel]. Il faut que les taux d'échantillonnage du fichier sonore et celui de Pure Data soient égaux (44100 par défaut). Si ce n'est pas le cas, l'objet [tabplay~] jouera le tableau trop vite ou trop lentement.

[TABWRITE~] : ÉCRIRE DU SON DANS UN TABLEAU

On peut aussi écrire du son dans un tableau avec l'objet [tabwrite~]. Il suffit de lui donner le nom du tableau dans lequel écrire. Lorsqu'on lui envoie un simple "bang", il écrit dans le tableau à partir de son début et jusqu'à sa fin. Cela peut être utile pour visualiser l'onde que porte un signal.



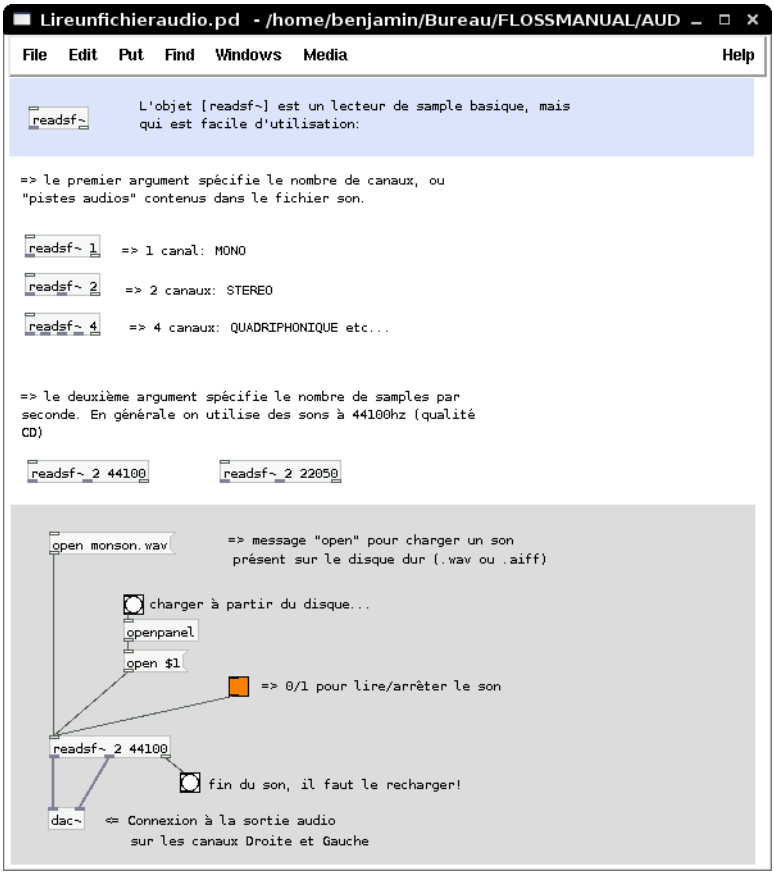
Dans le patch ci-dessus, on envoie un "bang" 10 fois par seconde à l'objet [tabwrite~] qui s'y trouve. Si le taux d'échantillonnage de l'application est alors de 44100 Hz et que le tableau a une taille de 441 échantillons, on y écrira exactement 1/10 de seconde d'onde sonore à chaque dixième de seconde.

23. LIRE ET ENREGISTRER DE L'AUDIO

Le travail sur les fichiers sonores est un sujet qui mérite que l'on s'y attarde, car il existe plusieurs formats de fichiers. Plusieurs objets permettent de lire chacun de ces formats. Parmi les formats audio numériques, notons ceux qui ne compressent par les données (.wav et .aiff), ceux qui les compressent avec perte de qualité sonore (Vorbis, MP3, etc.), et ceux qui les compressent, mais sans perte de qualité (FLAC).

LIRE UN FICHIER AUDIO AVEC [READSF~]

Pour lire un fichier audio enregistré au format .wav ou .aiff, on peut utiliser un objet qui charge un son à partir du disque dur. Cet objet permet très peu de manipulation de la « matière », uniquement la lecture. Des lecteurs audio complémentaires proposant plus de fonctionnalités existent dans des librairies externes (voir plus bas dans ce chapitre).



Patch tiré du tutoriel de Raphael Isdant
http://raphael.isdant.free.fr/puredata.htm

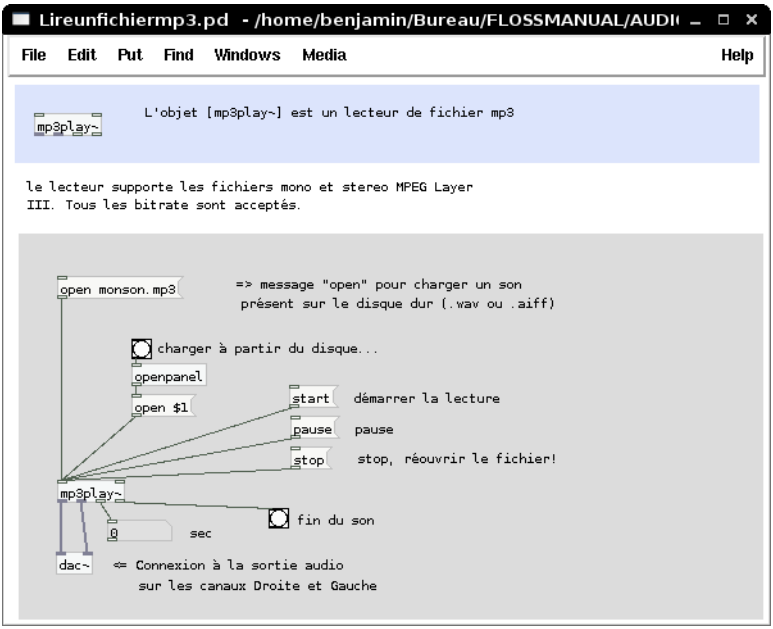
Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/audio/Lireunfichieraudio.pd

Pour le faire fonctionner :

1. Chargez un fichier à partir de votre disque dur en cliquant sur le "bang" situé au-dessus de l'objet [openpanel].
2. OU changez le chemin/nom du fichier présent dans le message [open monson.wav<
3. Cliquez ensuite sur l'interrupteur (*Toggle*) orange pour lire ou arrêter la lecture du son.
4. Un "bang" est expulsé par la dernière sortie de l'objet [readsf~] lorsque la lecture arrive à la fin du fichier.

LIRE UN FICHIER MP3 AVEC [MP3PLAY~] (LIBRAIRIE IEMLIB)

Même principe pour lire un fichier MP3 :



Patch tiré du tutoriel de Raphael Isdant
http://raphael.isdant.free.fr/puredata.htm

Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/audio/Lireunfichiermp3.pd

Pour le faire fonctionner :

Processus identique à celui décrit pour les fichiers .wav ci-dessus.

AUTRES LECTEURS EXTERNES

Il existe d'autres lecteurs audio dans Pure Data provenant de librairies externes, pour certaines intégrées à Pd Extended. Ces lecteurs proposent d'autres fonctionnalités ou acceptent plus de formats de fichier. Par exemple :

[oggamp~], **[oggwrite~]**, **[oggread~]**, **[oggcast~]** (Librairie pdogg):

- Pour lire, écrire et *streamer* du Ogg/Vorbis.

[sfplay 2] (Librairie Zexy) :

- Peut lire des fichiers au format raw.
- Tête de lecture déplaçable à un temps donné du fichier audio.

[readsfv~] (Librairie Moonlib) :

- Peut lire de gros fichiers son stockés sur le disque dur (direct-to-disk).
- Possibilité de faire varier la vitesse de lecture.

[sfread2~] (Librairie Moonlib) :

- Mêmes fonctionnalités que l'objet précédent, avec en plus une interpolation de la lecture en 4 points.
- Possibilité de mettre la lecture en boucle.

[readanysf~] (Linux et MacOS, à compiler ou à récupérer dans un obscur dépôt) :

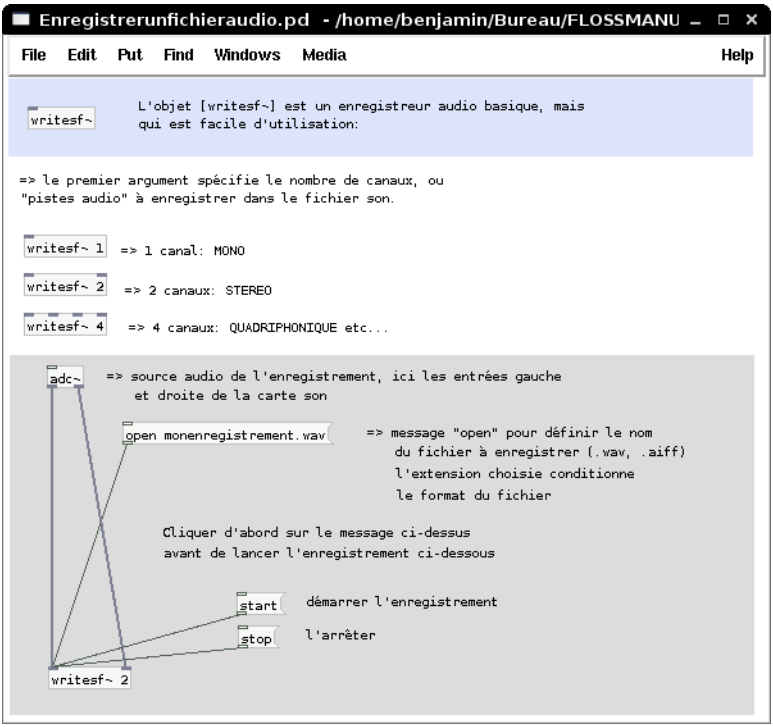
- Est capable de lire de nombreux formats de fichiers, y compris des flux audio en *streaming*, car il s'appuie sur gavl/gmerlin.
- Supporte les fichiers audio multicanaux.
- Permet de déplacer la tête de lecture sur le fichier à un temps donné.
- Possibilité de lecture en boucle précise au *sample* près.
- Lecture à vitesse variable.
- Voir http://aug.ment.org/readanysf/.

[mp3amp~], **[graphic-mp3amp~]**, **[mp3live~]**, **[mp3streamout~]**, **[mp3cast~]**, **[mp3streamin~]**, **[mp3fileout~]**, **[mp3write~]** (Librairie unauthorized) et **[mp3play~]** (Librairie iemlib) :

- Pour lire, enregistrer et *streamer* du son en format mp3, avec diverses option.

ENREGISTRER UN FICHIER AUDIO AVEC [WRITESF~]

Vous trouverez ci-dessous une méthode très simple pour enregistrer un fichier audio dans Pure Data. Il est bien évidemment possible de contraindre la durée de l'enregistrement (avec un objet [del 1000] par exemple) ou encore de réaliser des enregistrements en série en incrémentant les noms des fichiers enregistrés.



Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/audio/Enregistrerunfichieraudio.pd

Pour le faire fonctionner

1. Définissez le nom du fichier à enregistrer dans le message [open monenregistrement.wav<.
2. Sans plus de précisions, le fichier sera enregistré dans votre répertoire personnel, il est tout à fait possible d'ajouter un chemin pour « ranger » les fichiers dans un dossier.
3. Prenez bien en considération le fait que l'extension du fichier conditionne son format (.wav, .aiff, .snd).
4. Cliquez sur le message [open monenregistrement.wav<.
5. Cliquez sur le message [start< pour démarrer l'enregistrement.
6. Cliquez sur le message [stop< pour stopper l'enregistrement.

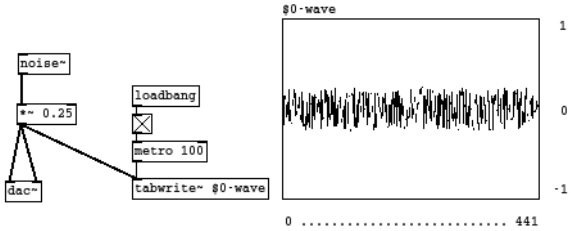
24. LA SYNTHÈSE AUDIO

Pure Data offre toute une série d'objets pour l'audio : ceux qui permettent de produire, de traiter ou d'analyser du son sont presque toujours nommés par la terminaison tilde (« ~ »). Par exemple, l'objet [osc~] produit une onde sinusoïdale. En fait, un signal sonore est représenté par un nombre dont la valeur change très rapidement, à la fréquence du taux d'échantillonnage audio de l'application.

Nous allons maintenant présenter quelques objets Pure Data permettant de réaliser de la synthèse audio. Pour chacun, nous verrons à quoi il sert et comment l'utiliser. Nous en profiterons pour expliquer d'autres notions de base de l'audio numérique (voir chapitre « Les bases de l'audio numérique »).

[NOISE~] : LE BRUIT

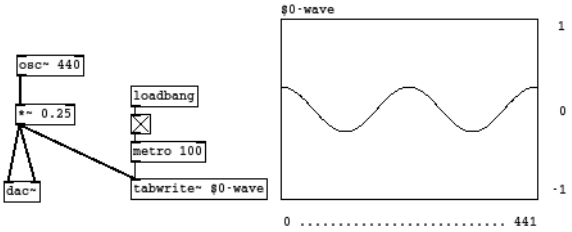
Un bruit, c'est un son dont la hauteur n'est pas déterminée. Son onde fait des sauts dans un désordre total. Un bruit blanc est un bruit à peu près uniforme, un son dont le spectre parcourt toutes les fréquences. L'objet [noise~] produit du bruit blanc. Avant d'essayer l'exemple qui suit, il est conseillé de baisser le volume sonore au minimum.



L'intérêt de cet objet se manifeste notamment dans la **synthèse soustractive**, ainsi que dans la création d'effets sonores.

[OSC~] : PRODUIRE UNE ONDE SINUSOÏDALE

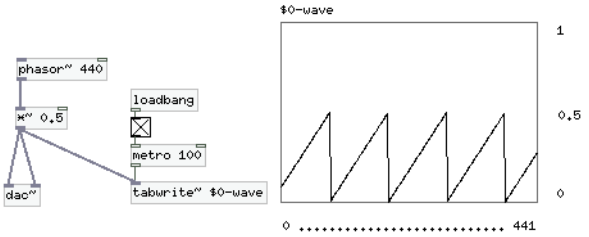
Une onde sinusoïdale oscille comme une pendule entre ses deux extrêmes selon une courbe sinusoïdale. Il s'agit d'une fonction sinusoïdale du temps. Si on lui donne une fréquence audible, par exemple 440 Hertz, cela donne un son pur, sans harmonique.



Le premier et le seul argument de cet objet est sa fréquence, en hertz. On peut modifier cette fréquence en envoyant un nombre dans sa première entrée. La seconde entrée sert à remettre sa phase à zéro.

[PHASOR~] : UNE ONDE EN DENTS DE SCIE

Si l'on souhaite synthétiser un son timbré à hauteur déterminée, créer des dizaines d'objets [osc] pour tous les additionner ensemble est un peu long. L'objet [phasor] génère une onde en dents de scie, c'est-à-dire une ligne qui va périodiquement de 0 à 1.



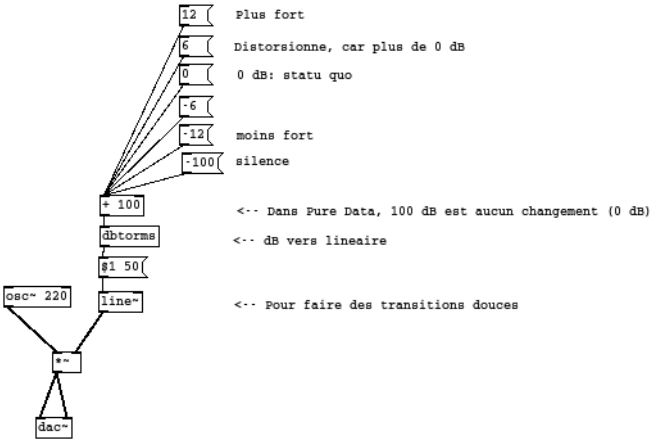
Si on en fait une analyse de Fourier, on peut voir qu'elle contient de nombreuses harmoniques. Sans entrer dans les détails, le son produit évoque celui d'un klaxon.

[DBTORMS] : LES DÉCIBELS

Les professionnels de l'audio utilisent le décibel (dB) comme unité de mesure pour l'atténuation ou l'amplification du volume d'un signal audio. Chaque fois que l'on baisse de 6 dB, l'amplitude diminue de moitié. Si on augmente de 6 dB, l'amplitude double. Une atténuation de -6 dB sera perçue comme donnant un son moitié moins fort que l'original. Un son qui aurait une amplitude de 1.0 se retrouverait avec une amplitude de 0.5.

En mixage audio, on se garde habituellement une marge de manœuvre de l'ordre de -12 ou -18 dB, pour s'assurer que l'amplitude de chaque source sonore ne dépasse pas environ 25% ou 12.5% de l'amplitude maximale possible en sortie.

Dans Pure Data, les objet [dbtorms] et [dbtorms~] aident à convertir des gains en dB vers un facteur linéaire. Il s'agit ici du facteur par lequel on multiplie un signal audio afin d'ajuster son volume. Ainsi, on bénéficie d'un contrôle fin sur le volume sonore. La perception de celui-ci n'est pas linéaire, mais logarithmique. Utiliser une échelle logarithmique étant assez difficile, on préférera utiliser des décibels.



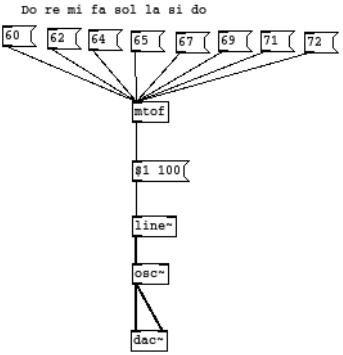
Dans Pure Data, l'échelle en dB va de 0 à 100, notation plus simple que d'aller de moins l'infini à zéro. Si ce n'était pas le cas, jusqu'à combien de dB dans le négatif faudrait-il descendre afin d'atteindre le silence ? Ainsi, dans Pure Data, à 100 dB (au lieu de zéro), le volume d'un signal n'est pas atténué. À 0 dB (au lieu de moins l'infini), c'est le silence absolu. À 106 dB, il est deux fois plus fort, et à 94 dB, il est deux fois moins fort.

Enfin, la différence entre la version sans tilde de [dbtorms], et celle avec tilde, c'est que la première fonctionne avec des nombres, et donc à une fréquence maximale d'une fois toutes les 5 millisecondes, alors que la seconde fonctionne avec des signaux audio, et donc à une fréquence égale à celle du taux d'échantillonnage de l'application. Par défaut, c'est 44 100 hertz.

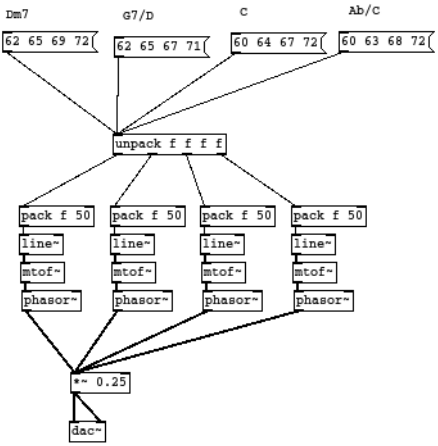
[MTOF] : NOTES MIDI ET FRÉQUENCE

La norme MIDI est un standard pour la musique numérique qui date des années 80. Elle permet de spécifier la hauteur des notes de musique, leur volume et différents contrôles, comme la pédale de soutien d'un piano. Les objets [notein] et [noteout], ainsi que [ctlin] et [ctlout] permettent de recevoir et d'envoyer des notes et des contrôles MIDI avec Pure Data. Ainsi, il est relativement facile de brancher un appareil MIDI dans Pure Data.

En MIDI, le Do central correspond au nombre 60. Chaque unité correspond à un demi-ton. Ainsi, le nombre 61 est un Do dièse, et 59 est la note Si. Il y a 12 demi-tons dans une octave. Ainsi, la note 72 est un Do d'une octave plus haute que le Do central.



Pour la synthèse audio, c'est la fréquence qui nous intéresse, pas la note MIDI. Si on veut contrôler la note à synthétiser au moyen de notes MIDI, il faut convertir ces notes en fréquences. C'est l'objet [mtof] qui nous le permet.

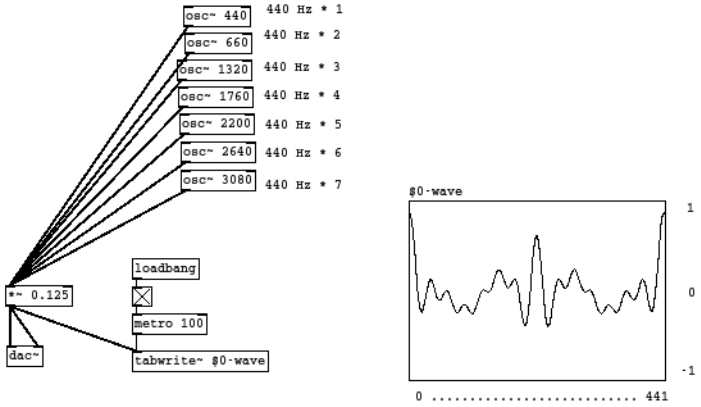


Il est assez facile d'utiliser des listes de nombres pour représenter des accords ou des gammes. Pour les transposer, il suffit d'additionner ou de soustraire à chacun des nombres dans la liste. Cet exercice est laissé aux soins du lecteur. La famille des abstractions musicales dans la librairie pdmtl offrent de telles fonctionnalités.

LA SYNTHÈSE ADDITIVE

En musique, un son à hauteur déterminée, dont la forme d'onde se répète périodiquement, peut être décomposé en une somme d'ondes sinusoïdales multiples de la fréquence fondamentale du signal. Cette fondamentale, c'est la note que nous pouvons identifier. Les autres ondes sont appelées harmoniques. Par exemple, un La 440 sur un violon aura 440 Hz comme fondamentale, puis plusieurs harmoniques, comme 880 Hz, 1320 Hz, 1760 Hz, et ainsi de suite. Le timbre est déterminé par la quantité relative de chaque harmonique partielle. Un timbre est plus nasillard lorsqu'il contient plus d'harmoniques aiguës.

Ainsi, on peut en théorie synthétiser tout son à hauteur déterminée en additionnant plusieurs ondes sinusoïdales. C'est ce que l'on appelle de la synthèse additive. À l'inverse, on peut décomposer tout son au moyen de l'analyse de Fourier, afin d'en tirer les harmoniques.



Notez que certains sons contiennent des ondes périodiques qui ne sont pas des multiples de la fondamentale. On les appelle alors des partielles.

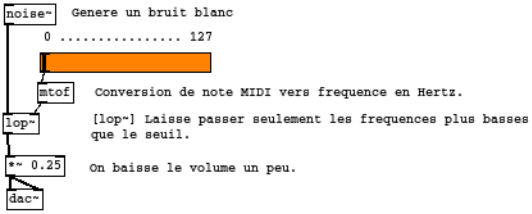
25. LES EFFETS

Une fois que l'on sait lire et synthétiser des sons, on peut vouloir modifier leur timbre.

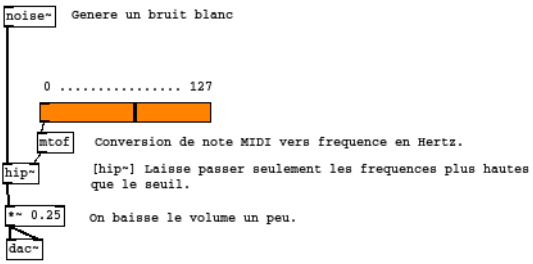
FILTRES SPECTRAUX

D'une manière générale, les filtres spectraux permettent d'agir sur le spectre du son en choisissant certaines plages de fréquences et d'éliminer les autres. Il existe principalement quatre grandes familles de filtres : les filtres passe-bas (*low pass*), les filtres passe-haut (*high pass*), les filtres passe-bande (band-pass), les filtres en peigne (*comb*). Ces filtres permettent de colorer le timbre des sons, et sont donc très utiles pour des applications de synthèse sonore.

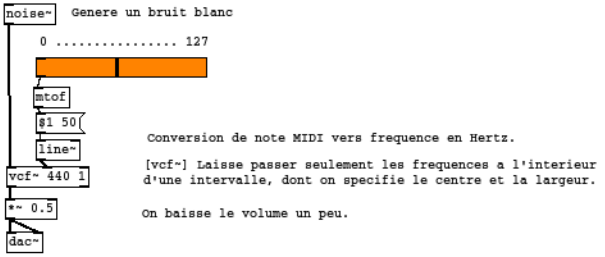
Un filtre passe-bas laisse passer les fréquences basses, et atténue donc les fréquences plus hautes que son seuil. Le résultat sera plus rond, plus chaleureux, moins sec. Avec l'objet [lop~] (*low-pass*), il s'agit simplement de spécifier la fréquence seuil. Nous utiliserons [mtof] pour convertir les notes MIDI reçues du gradateur en des fréquences en hertz.



Un filtre passe-haut laisse passer les fréquences hautes, et atténue donc les fréquences plus basses que son seuil. L'objet [hip~] (*high-pass*) fonctionne de manière semblable à [lop~] : il laisse donc passer les sons plus aigus au lieu des plus graves.

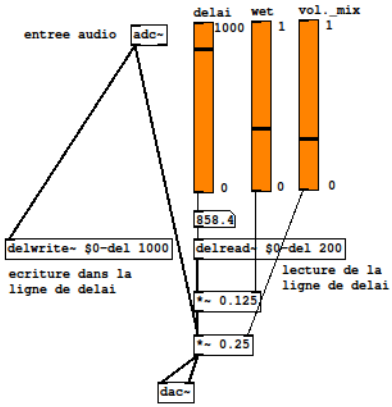


Un filtre passe-bande laisse passer les fréquences se situant à l'intérieur d'une certaine plage. Lorsqu'on en utilise une, on doit spécifier une fréquence centrale et la largeur de la bande à laisser passer. On peut choisir l'objet [bp~] si la fréquence centrale ne change pas. Si elle change, on utilisera plutôt l'objet [vcf~], car il est contrôlé via un signal audio, ce qui évite les clics.



EFFETS TEMPORELS

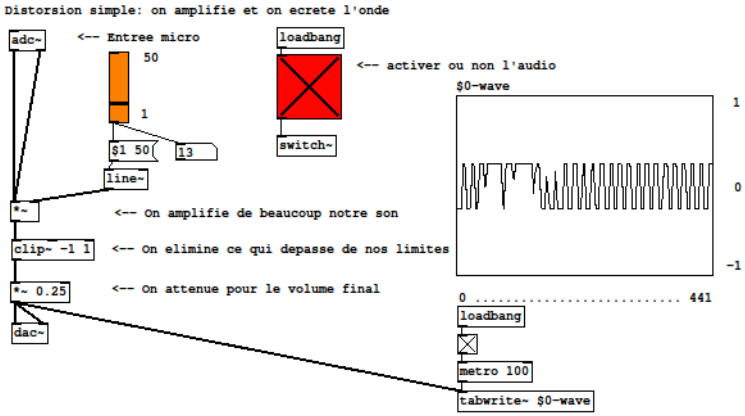
Outre le domaine spectral, on peut aussi agir sur le domaine temporel du son. Il est très simple d'ajouter un délai sur un signal sonore. Cela a pour effet de retarder le son.



Si on injecte le résultat d'un délai dans le son original, mais un peu moins fort que l'original, cela résulte en une forme simple de réverbération. On peut produire toutes sortes de sonorités en variant la durée du délai et son volume sonore. Pour écrire dans la ligne de délai on utilise [delwrite~], avec comme argument le nom (unique) de la ligne de délais et sa longueur, pour lire la ligne de délais on utilise [delread~] avec comme argument le nom et le délai souhaité.

DISTORSION

Pour imiter une pédale de distorsion de guitare électrique, le moyen le plus simple est de rapprocher la forme de notre onde sonore d'une onde carrée. Pour cela, on amplifie beaucoup notre signal, puis on l'écrête. On entame les parties les plus élevées de l'onde. L'objet [clip~] empêche un signal de dépasser des valeurs extrêmes.



AUTRES EFFETS UTILES

Parmi les procédés couramment utilisés en audio numérique, notons surtout la compression et le limiteur. Ces techniques visent à éviter de dépasser les limites de volume avant la sortie. Il existe des librairies externes pour le faire, et il est aussi possible de le faire en combinant des objets internes, mais cela dépasse la portée de ce manuel. Pour en savoir plus à ces sujets, vous pouvez vous reporter aux patchs d'aide fournis avec Pure Data.

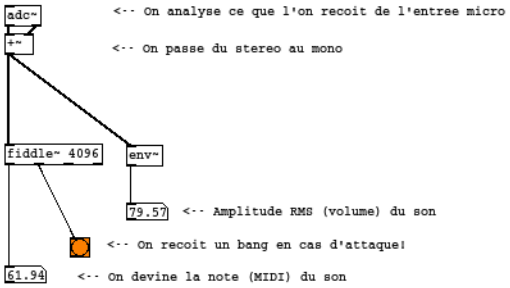
Bien d'autres effets sont possibles, et souvent disponibles sous une forme plus ou moins avancée dans des externes. Là encore, la rubrique d'aide est la source la plus précieuse pour découvrir et comprendre leurs usages et possibilités.

26. L'ANALYSE AUDIO

Pure Data est conçu pour l'interactivité. On peut retirer d'innombrables informations à partir d'un signal audio. On peut par exemple mesurer la hauteur d'un son (sa fréquence fondamentale), ou encore détecter les attaques, c'est-à-dire les moments durant lesquels le volume monte subitement, comme lorsqu'on frappe un instrument percussif. On peut aussi analyser tout le spectre sonore d'un son. Ces informations varient en permanence, car le son est une vibration qui change dans le temps. Le résultat de ces analyses nous fournit des informations musicales, bien sûr, mais aussi des moyens potentiels de contrôler par exemple de la vidéo, des effets sonores, ou le déclenchement d'appareils électroniques branchés à l'ordinateur. Ainsi, l'interactivité par le son vient enrichir la palette d'expression que Pure Data offre aux artistes et designers de l'interaction.

[FIDDLE~] ET [ENV~] : LA NOTE QUE L'ON CHANTE ET SON VOLUME

L'objet [fiddle~] offre beaucoup d'informations sur un signal audio. On peut facilement obtenir la note d'un son musical, en format MIDI. Cet objet détecte également les attaques, et peut nous fournir des renseignements supplémentaires sur les ondes partielles composant un son.



Ici, on utilise aussi l'objet [env~] pour mesurer l'amplitude du signal. Notez que l'amplitude mesurée n'est pas l'amplitude maximale (*peak-to-peak*), mais plutôt sa valeur efficace (*RMS*) en décibels dans le style de Pure Data : 100 étant l'amplitude maximale possible et 0 correspond au silence. On peut estimer l'amplitude sommet à sommet en ajoutant environ 3 dB.

Si l'on souhaite détecter des fréquences graves, il faut indiquer à [fiddle~] d'utiliser une fenêtre qui soit assez grande. La fenêtre, c'est le nombre d'échantillons qui sont accumulés afin d'analyser les courbes qui s'y trouvent, et donc la fréquence des ondes qui composent ce signal audio. Plus cette fenêtre est grande, plus on peut détecter des fréquences graves, car leur longueur d'onde est plus grande que celle des fréquences aiguës.

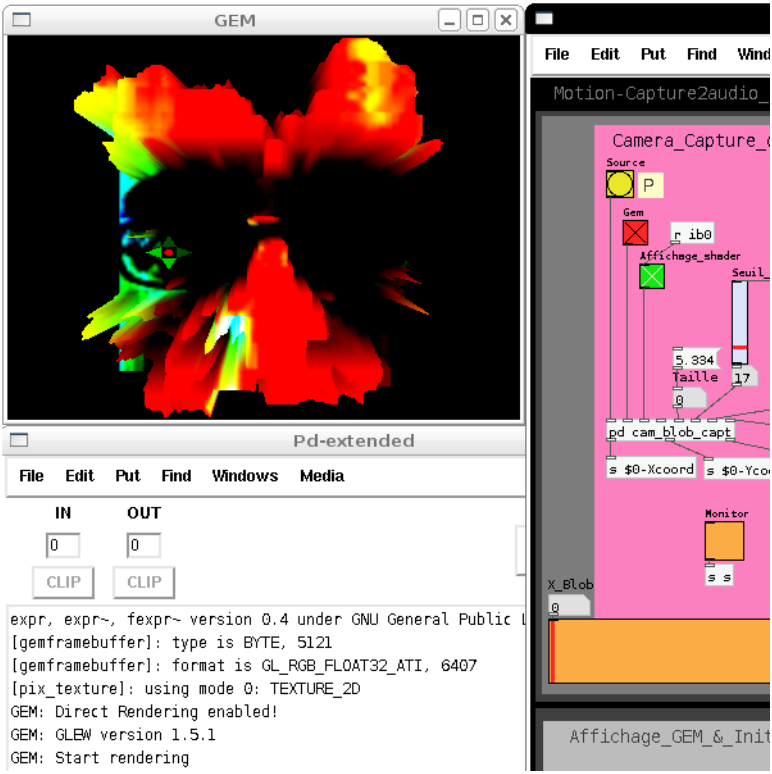
Il existe d'autres objet intéressant comme [sigmund~] (un équivalent à [fiddle~]) et l'objet [bonk~] pour la détection d'impact.

VIDÉO (GEM)

27. GEM
28. GÉRER L'AFFICHAGE
29. AFFICHER UNE FORME GÉOMÉTRIQUE
30. MANIPULER DES IMAGES FIXES
31. MANIPULER DE LA VIDÉO
32. UTILISER UNE CAMÉRA VIDÉO
33. APPLIQUER DES EFFETS SUR LES PIXELS
34. MIXER DES TYPES DE CONTENUS
35. INTÉGRER DU TEXTE
36. OPÉRER UNE ANALYSE DE MOUVEMENTS
37. OPÉRER UNE DÉTECTION DE COULEURS

27. GEM

La librairie **graphique GEM** est une extension externe très utilisée dans Pure Data pour manipuler des flux ou des fichiers vidéo, des images fixes ou des objets 3D. Au cours de cette brève introduction, nous aborderons ses principes ainsi que ses origines, pour appréhender ensuite différents aspects de son utilisation.



Extrait d'un patch utilisant GEM pour réaliser une analyse de mouvement et en générer des notes MIDI, tout en appliquant un effet sur le flux de la caméra.

GÉNÉRALITÉS

GEM signifie « Environnement Graphique pour le Multimédia » (*Graphic Environment for Multimedia*). GEM n'est pas une application indépendante. C'est une librairie qui se charge au moment de l'ouverture de Pure Data à condition d'avoir préalablement installé et correctement configuré l'environnement. **C'est un outil de création graphique en temps réel**. Il permet avec facilité et simplicité de construire un programme pour gérer et générer des images fixes, des images en mouvements, des textes, des images graphiques en deux et en trois dimensions, ou encore de réaliser une analyse de mouvements ou de couleurs. Les utilisateurs peuvent ainsi combiner audio et visuels en contrôlant l'un et l'autre simultanément, et ainsi créer, par exemple, une relation forte entre l'élément sonore et l'élément visuel.

Dans Pd-extended, la librairie GEM est incluse par défaut et opérationnelle. Pour plus d'informations à ce sujet, voir la rubrique « Configuration: Chemins et Librairies » de ce manuel.

ORIGINES

GEM a initialement été développée par Mark Danks. Certains des développeurs passés et présents sont : Iohannes Zmölnig, Chris Clepper, James Tittle (tigital) et Cyrille Henry.

GEM, UNE LIBRARIE QUI S'APPUIE SUR L'OPENGL

GEM est basé sur l'OpenGL (*Open Graphics Library*) cela implique que votre ordinateur dispose d'une carte graphique qui gère bien ce jeu d'instructions, exécuté directement par le processeur de la carte graphique. Il est ainsi utile de se renseigner avant l'achat d'un matériel dédié à cet usage. On peut aussi utiliser directement dans GEM des instructions tirées de l'OpenGL, ce qui étend ses possibilités de façon exponentielle.

L'OpenGL est une librairie professionnelle très utilisée pour les jeux vidéo et la 3D. En utilisant la librairie GEM dans Pure Data, **on peut dessiner des formes géométriques 2D/3D sur une scène 3D vue à travers la fenêtre d'affichage**. On peut faire tourner ou déplacer ces formes, changer leur couleur ou plaquer sur ces formes de la vidéo ou des images fixes sous forme de texture. On appelle ainsi « texture » les images qui viennent « recouvrir » ces formes géométriques dans un contexte de rendu OpenGL.

Pour dessiner des formes dans une fenêtre d'affichage, on doit placer des objets de la librairie GEM dans des chaînes où tous les objets sont connectés les uns aux autres. On les appelle des **« chaînes GEM »**. Tout en haut de ces chaînes, il y a toujours un objet [gemhead], et on trouve la forme à dessiner tout en bas. Entre les deux, on place les transformations que l'on veut appliquer sur la forme à dessiner.

Cette manière de procéder diffère quelque peu du principe habituel de « chute d'eau » propre à Pure Data. On peut donc lire les patchs réalisés avec GEM du bas vers le haut. L'objet [gemhead] est responsable des géométries, des textures et des transformations qu'il devra rendre, alors qu'il est au sommet de la chaîne. Pour rendre ce concept moins abstrait, il est important d'étudier les exemples proposés dans les autres chapitres de ce manuel consacrés à GEM.

RÉFÉRENCE SUR OPENGL

Pour aller plus loin sur OpenGL, vous pouvez notamment consulter l'ouvrage de référence suivant :

NEIDER J., WOO M., DAVIS T., SHREINER D. & CAMPULLO V. (trad.). (2006). « *Guide officiel : Le guide officiel pour l'apprentissage et la maîtrise d'OpenGL 2.0* ». Éditions CampusPress, Collection Référence, 788 p. ISBN : 978-2744020869

28. GÉRER L'AFFICHAGE

Dans ce chapitre, vous trouverez quelques-unes des notions essentielles de la création graphique avec GEM. Pour commencer de façon simple, nous découvrirons comment créer une fenêtre de base et y afficher un carré blanc.

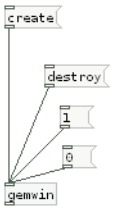
LES BASES DE L'AFFICHAGE

OpenGL est une librairie de calcul mathématique de graphisme en deux ou trois dimensions, mais elle ne gère pas l'affichage. C'est un concept important qui va nous permettre de comprendre les bases de la restitution *(rendering)* d'images ou de formes géométriques avec GEM.

Lorsque les images et les formes (ou tout autre objet visuel) sont affichés à l'écran, nous appelons cela la restitution, l'affichage, ou encore le rendu; sous ces termes se cache un seul et même concept. L'ordinateur va préparer une grille de pixels pour les afficher à l'écran. Il procède à partir d'une scène en deux ou trois dimensions dans laquelle différents acteurs (formes, images animées ou fixes, etc.) prennent place.

Pour créer une fenêtre contenant une scène, nous allons indiquer sur notre écran la zone destinée à la recevoir. À l'aide de l'objet [gemwin], nous allons spécifier cette zone, qui par défaut est de 500 par 500 pixels, et est positionnée en haut à gauche de l'écran. Ensuite, nous utiliserons l'objet [gemhead] pour indiquer quoi afficher dans cette fenêtre.

[GEMWIN]



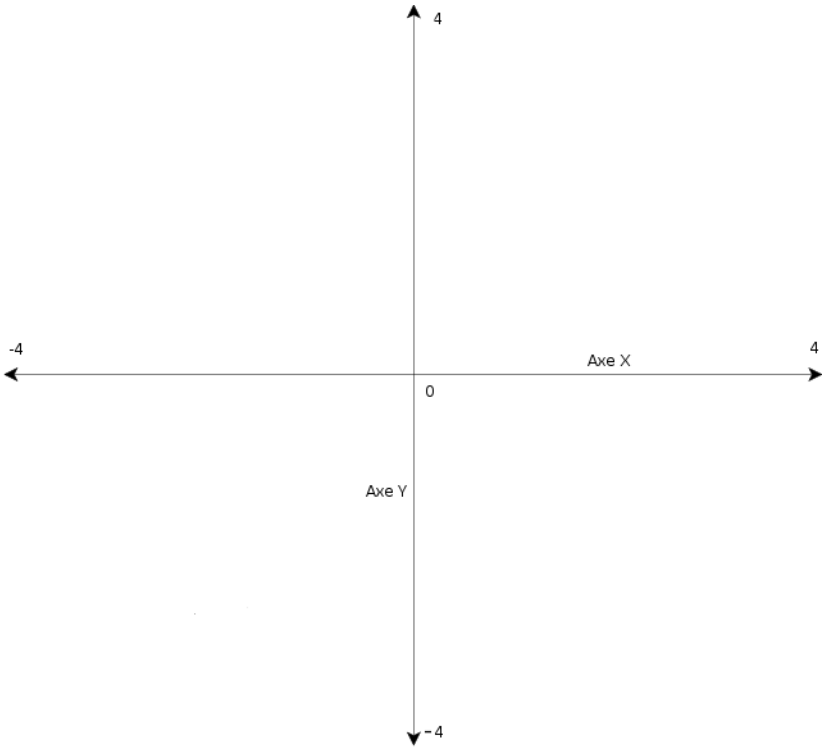
[gemwin] est l'objet GEM qui sert à créer la fenêtre dans laquelle vos graphiques seront dessinés. Cet objet contrôle à quel rythme chaque image de la scène sera rendu. En écrivant seulement [gemwin], la valeur par défaut votre taux d'images par seconde sera de 20. En le spécifiant explicitement, à l'aide d'un argument [gemwin 25] ou d'un message [frame 50<, vous pourrez modifier ce taux d'images par seconde.

Par défaut, [gemwin] nettoie également la fenêtre après chaque trame. Si ce n'était pas le cas, les objets laisseraient des traces derrière eux lorsqu'ils se déplacent dans la scène, car leurs images passées ne seraient pas effacées. Cet objet permet aussi d'afficher une couleur d'arrière-plan. On exprime les couleurs au moyen d'une liste de trois ou quatre nombres, correspondant aux valeurs de rouge, de vert, de bleu, et au niveau d'opacité. Dans le cas de l'arrière-plan, il n'y a pas d'opacité : par défaut, l'arrière-plan est noir.

Les différents messages possibles envoyés à [gemwin] peuvent être de nature à changer la taille de la fenêtre, démarrer et arrêter le processus de restitution, changer de point de vue la scène en 3D et permettre de contrôler de nombreux autres aspects de la fenêtre tel que l'anticrénelage, par exemple. Pour voir en détail toutes les propriétés et types de messages possibles, faites un clic droit sur l'objet [gemwin] afin d'accéder au fichier d'aide.

Le message qui va permettre de créer votre première fenêtre est [create<.

Ci-dessous, l'espace d'affichage et les repères tels que GEM les comprend. Cet exemple est valable pour votre fenêtre nouvellement créée.



Par exemple, un message [color 1 0 0< envoyé à [gemwin] va afficher un fond rouge. Ainsi, nous chargeons la couleur d'arrière-plan, noire par défaut. Il est à noter que les valeurs de chaque canal d'une couleur devront être comprises entre 0 et 1, contrairement à d'autres systèmes dans lesquels les canaux s'échelonnent de 0 à 255. Toutes les valeurs décimales seront possibles entre ces deux valeurs seuil. Les trois paramètres acceptés par le message [color< apparaîtront dans l'ordre suivant : rouge, vert, bleu.

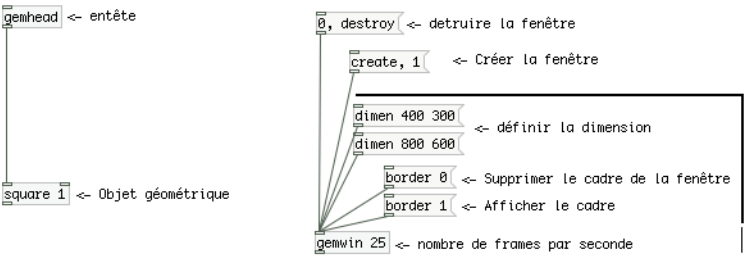
[GEMHEAD] : AJOUTER UN ACTEUR DANS LA SCÈNE

Il existe l'objet [gemhead], qui va nous permettre de faire un pas de plus vers de formidables animations en trois dimensions.

Il est possible d'afficher un simple carré blanc dans notre fenêtre en reliant [gemhead] à un nouvel élément [square 1]. Il est à noter que les scènes sont calculées en fonction de proportions et non pas d'une taille définie en pixels. Ici, en ajoutant l'argument 1 à l'objet [square], on demande à [gemhead] de calculer proportionnellement la taille d'un carré dans la fenêtre, dans ce cas, de 1/4.

Afin d'activer le calcul de la scène dans la fenêtre nouvellement créée, on a besoin de le spécifier explicitement par l'envoi d'un message supplémentaire, à savoir [1<. Une autre manière possible est d'écrire ceci dans une seule boîte message comme ceci : [create, 1<. Cela envoie en fait deux messages à l'objet [gemhead], mais l'un après l'autre, très rapidement.

[create, 1< va automatiquement créer la fenêtre et ensuite (bien noter ici la présence importante de la virgule entre *create* et 1) exécuter la restitution de la scène à afficher.

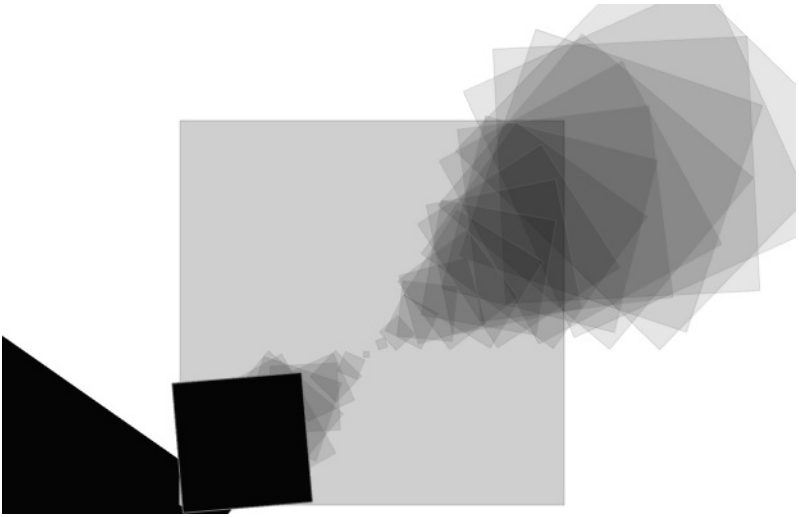


Dans l'image ci-dessus sont précisés quelques exemples simples de messages envoyés à [gemwin]. Tous les messages situés en dessous du cadre noir devront être envoyés avant le message [create, 1<. En d'autres termes, nous ne pourrons pas changer l'apparence de la fenêtre GEM de manière dynamique, à l'aide de ces seuls messages. Après avoir défini les dimensions en longueur et en hauteur : [*dimen (longueur) (hauteur)*<, l'absence ou la présence de la bordure : [*border*< (1 pour oui, 0 pour non), nous pourrons envoyer le message [*create, 1*< et voir le résultat de nos choix s'afficher via 1.

PRINCIPES D'AFFICHAGE AVEC GEM

Il est important de préciser qu'avec GEM, il n'est pas (encore) possible de créer plusieurs fenêtres d'affichage avec une seule instance de Pure Data. Même en utilisant plus d'un objet [gemwin] dans un seul ou plusieurs patchs ouverts en même temps, vous ne pourrez afficher qu'une seule zone de création de rendu. Il est préférable d'éviter de surcharger vos patchs de [gemwin].

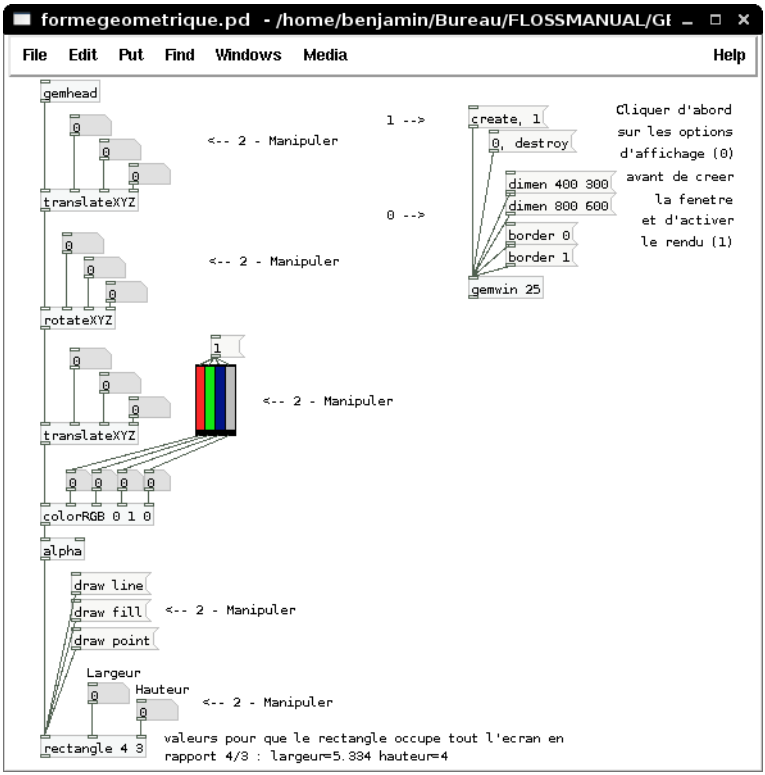
Ainsi, toutes vos créations visuelles vont venir se superposer les unes aux autres (ou se mélanger) dans votre fenêtre. Afin de mieux maîtriser ces possibilités d'intégration visuelle, vous pourrez consulter le chapitre « Mixer des types de contenus », en particulier la section « L'ordre de rendu ».



Ce sont les toutes premières valeurs envoyées à l'objet [gemwin] qui garderont « la main » sur les créations suivantes (après avoir détruit la fenêtre). Pour réinitialiser ces paramètres, il faudra soit envoyer les nouveaux paramètres choisis à l'aide des messages dans [gemwin] ou mieux, utiliser le message [reset< pour réinitialiser les valeurs par défaut, suivi des nouveaux paramètres.

29. AFFICHER UNE FORME GÉOMÉTRIQUE

Le patch ci-dessous permet **d'afficher un rectangle, d'en changer la position, l'angle, la couleur et l'opacité.**



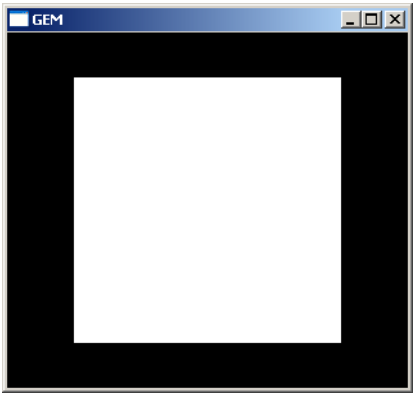
Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/formegeometrique.pd

Pour le créer (de bas en haut de la chaîne)

1. Créer un [rectangle];
2. Réaliser la transparence avec l'objet [alpha];
3. Donner une couleur avec l'objet [colorRGB];
4. Déplacer avec l'objet [translateXYZ];
5. Faites tourner avec l'objet [rotateXYZ];
6. Déplacer à nouveau (toujours en fonction des axes locaux de l'objet 3D considéré);
7. Afficher avec [gemhead] qui gère le rendu.

Pour le faire fonctionner

1. Créer d'abord la fenêtre d'affichage (1) en ayant préalablement cliqué sur les options d'affichage (0);
2. Manipuler ensuite les paramètres de translation selon l'axe des X, Y et Z;
3. Constater que le premier objet [translateXYZ] permet de décaler l'axe de rotation, alors que le deuxième [translateXYZ] opère selon ce repère XYZ modifié;
4. Tester l'affichage de l'élément géométrique sous forme de lignes et de points;
5. Changer la couleur, l'opacité (4e curseur)



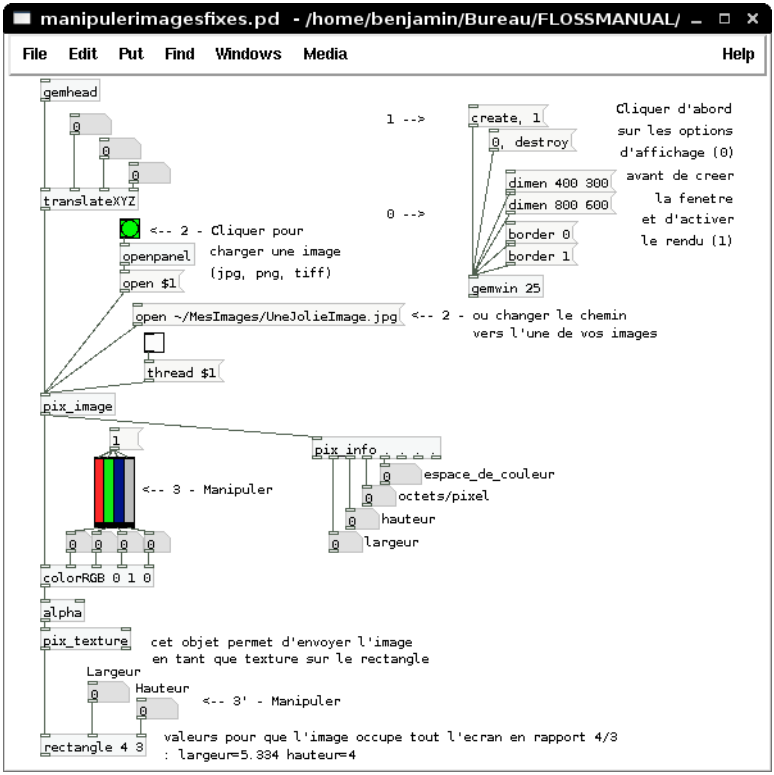
Il faut préciser que l'objet [alpha] permet de rendre modifiable l'opacité de l'objet. Les valeurs limites de la translation, lorsque l'écran est dans un rapport largeur/hauteur de 4/3, sont de -5.33 <> +5.33 pour l'axe des X, -4 <> +4 pour l'axe des Y. Les valeurs limites de la rotation sont de 0 à 360 (degrés) et celles des couleurs sont de 0 à 1.

Enfin, il est possible de remplacer l'objet [rectangle] par d'autres formes géométriques intégrées à GEM : *cube, sphere, circle, cylinder, curve3D, teapot*. Essayez ces différents objets et faites un clic droit afin d'afficher l'aide correspondante et connaître ses paramètres spécifiques.

30. MANIPULER DES IMAGES FIXES

Après nous être familiarisés avec quelques-uns des concepts de base de l'affichage dans GEM et de la manipulation d'objets géométriques dans l'espace en trois dimensions, nous allons aborder la notion de texture. Voyons comment on peut charger des fichiers d'images bitmap (formées de pixels) dans une texture afin de l'afficher sur une forme géométrique.

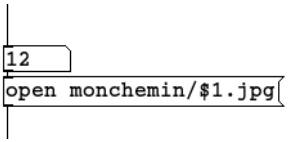
Le patch ci-dessous permet de **plaquer une image sur un rectangle, d'en changer la position et l'opacité**. Pour le faire fonctionner :



Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/manipulerimagesfixes.pd

1. Créer la fenêtre d'affichage (1) en ayant d'abord cliqué sur les options d'affichage (0);
2. Charger un fichier image (JPEG, PNG, TIFF) soit depuis votre disque dur en cliquant sur le bang vert, soit en indiquant le chemin de celui-ci dans le message;
3. Manipuler la couche alpha (glissière grise) ou les paramètres du rectangle ou de la translation.

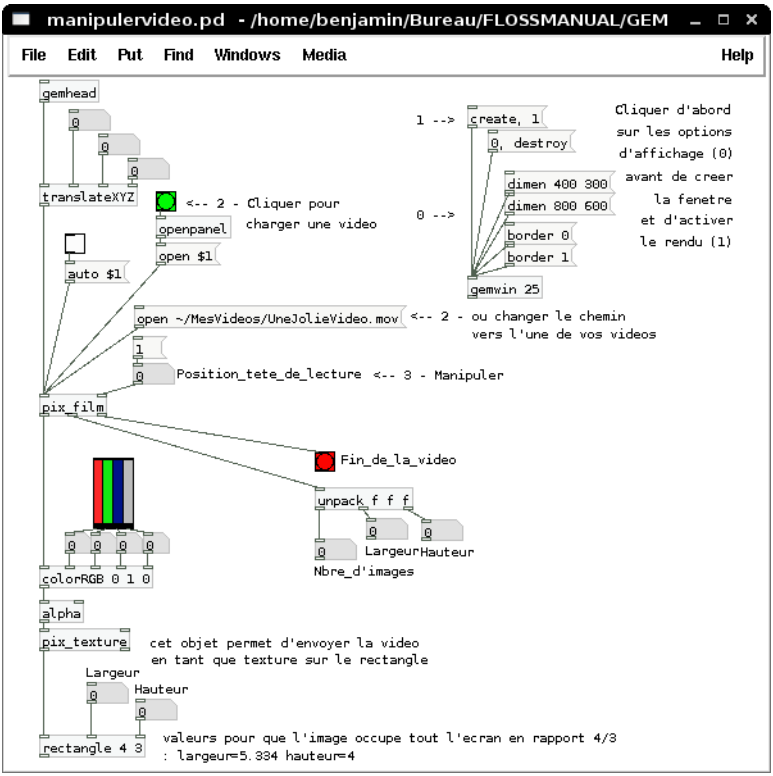
Il est à noter que lorsqu'on charge une image, elle prend toute la surface du rectangle. Dans le cas où l'image n'est pas dans un rapport 4/3, comme dans notre exemple, elle sera déformée. L'objet [pix_info] nous renseigne sur la taille de l'image chargée : on peut donc imaginer adapter la taille du rectangle en fonction des informations données par [pix_info] avec une division [/]. On peut aussi remplacer les deux derniers objets de la chaîne [pix_texture] et [rectangle] par un seul objet [pix_draw], et s'affranchir des contraintes de taille, mais l'objet [pix_draw] est beaucoup plus lent que l'exemple ci-dessus. Il est possible aussi de plaquer l'image comme texture sur tous les objets géométriques de GEM (*Sphere, Cube, Cylinder*, ...) plutôt qu'un rectangle à tester. Par ailleurs, pour réaliser un diaporama ou charger des images à la volée, il est possible de changer le message indiquant le chemin vers l'image par une variable [open $1< que l'on modifiera à volonté en utilisant un message, ainsi que d'utiliser des arguments variables ($1) pour générer le chemin adéquat. L'exemple ci-dessous permet de charger des images 0.jpg, 1.jpg, 2.jpg, ... en modifiait simplement la valeur de la boîte nombre.



Enfin, le message [thread 1< permet, sous Linux, de charger les images dans un autre fil d'exécution que le principal de *Pd*, ce qui évite de créer une micro-coupure dans le rendu audio.

31. MANIPULER DE LA VIDÉO

Avec GEM, on peut afficher des images fixes, mais aussi des clips vidéo. Un clip vidéo n'est qu'une succession d'images fixes dont la vitesse d'affichage donne l'illusion du mouvement. Les images défilent à une rythme fixe, selon le standard utilisé. Par exemple, le standard européen PAL implique une cadence de 25 images par seconde alors que le standard américain NTSC utilise plutôt 29,97 images par seconde. Le patch suivant montre comment lire un clip vidéo afin de l'afficher sur un rectangle.



Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/manipulervideo.pd

Le patch ci-dessus permet **de charger et lire une vidéo, d'en changer la position, la couleur et l'opacité**

Pour le faire fonctionner :

1. Créer d'abord la fenêtre d'affichage (1) en ayant préalablement cliqué sur les options d'affichage (0);
2. Charger un fichier vidéo (.avi, .mov, voir le prochain paragraphe à propos des codecs) soit depuis votre disque dur en cliquant sur le bang vert, soit en indiquant le chemin de celui-ci dans le message;
3. Manipuler la tête de lecture pour « scratcher » avec la vidéo.

Il faut noter qu'il est possible de cocher l'interrupteur relié au message [auto $1< pour que la vidéo avance automatiquement. Autre méthode : utiliser un compteur cadencé (ou non) au rythme de l'affichage (25 images / secondes = une image toutes les 40ms). Pour cela, on peut utiliser un [metro 40] ou bien intercaler dans la chaîne GEM un objet [t a b], un bang est ainsi envoyé à chaque fois qu'une image « passe » dans le fil. Lorsque la vidéo arrive à sa dernière image, un bang est envoyé dans la troisième sortie de l'objet [pix_film]. Pour mettre la lecture en boucle, il suffit de relier ce bang au message [1<. La tête de lecture sera ainsi replacée. La deuxième sortie de [pix_film] indique quant à elle le nombre d'images contenues dans la vidéo (et donc potentiellement sa durée, si elle est à 25 images par secondes = Nbre Images x 40 ms), ainsi que la largeur et la hauteur de celle-ci. Enfin, lorsque l'on charge une vidéo, elle prendra toute la surface du rectangle. Dans notre exemple, donc, si elle n'est pas dans un rapport 4/3, elle sera déformée. Les informations de largeur et de hauteur données par l'objet [pix_film] permettent d'adapter dynamiquement la taille du rectangle avec une petite division [/].

LES CODECS

Les **compresseurs-décompresseurs** méritent une partie à part entière, car il faudra souvent adapter l'encodage de la vidéo utilisée en fonction des librairies de décodage présentes sur son ordinateur, mais aussi l'usage que l'on souhaite faire de la vidéo (scratch de la tête de lecture ou lecture continue d'une longue vidéo).

Il faut tout d'abord rappeler que l'extension de fichier (.avi, .mov, .mkv, ...) ne révèle absolument pas avec quel codec la vidéo a été encodée, il s'agit simplement d'un conteneur qui cache la réelle « moulinette ». Une vidéo au format DV pourra, par exemple, avoir pour extension .mov, .mpg ou .avi.

Très grossièrement, il y a 2 types de codecs vidéo : ceux qui compressent chaque image séparément et ceux qui considèrent une suite d'images et vont tenter de diminuer le poids de la vidéo en trouvant des similitudes dans cette suite d'images (compression « interframe »). À cela, on peut ajouter la notion « d'images clefs », qui constituent un repère pour le lecteur vidéo. À titre d'exemple, si l'on veut scratcher avec la tête de lecture sur une vidéo, il faudra plutôt utiliser un codec qui compresse chaque image et les considère chacune comme une image clef.

GEM s'appuie sur les librairies de décodage du système d'exploitation, ce qui conditionne le type de vidéos qui peuvent être lues :

- **Mac OS X** : [pix_film] devrait être capable de décoder des vidéos QuickTime en .mov.
- **Win32** : [pix_film] devrait être capable de décoder des vidéos .avi (tous les codecs installés sur le système. Avec de la chance, la version de GEM utilisée a été compilée avec le support QuickTime, si vous avez le lecteur QuickTime installé vous pourrez lire des vidéos en .mov.
- **GNU/Linux** : ce cas est plus complexe, car pour GNU/Linux, il n'y a pas de format vidéo « natif ». La lecture de vidéo dépend donc des librairies de décodage installées sur le système et de la façon dont GEM a été compilé. Dans le meilleur des mondes linuxiens, GEM sera capable de lire des fichiers QuickTime/MOV, des fichiers en MPEG1 et 2, des AVI encodées en DivX, au format DV, des vidéos encodées en Theora, de la vidéo en HD, voire du flash et du stream. Fondamentalement, c'est en théorie le système d'exploitation GNU/Linux qui est capable de lire la plus grande diversité de formats. En pratique, il est parfois délicat de parvenir à cette exhaustivité. Quoi qu'il en soit, lorsque vous créez un objet [pix_film], l'interface principale de Pure Data (la console) présente les librairies de décodage associées à GEM sur son système.

```
pix_film:: quicktime support
pix_film:: libmpeg3 support
pix film:: libaviplay support
```

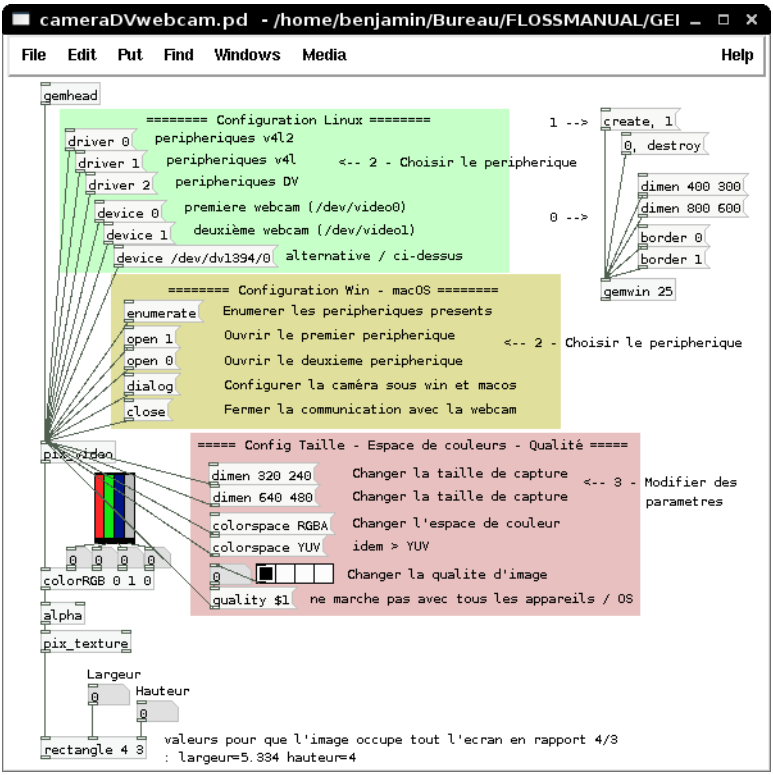
Il faut aussi considérer que plus une vidéo est compressée, plus elle demandera de travail au processeur de l'ordinateur pour être lue. *A contrario*, une vidéo peu ou pas compressée prend beaucoup d'espace disque et nécessitera donc un bon débit du disque dur et de la carte mère pour pouvoir être lue de façon fluide. C'est donc encore une fois un compromis à trouver selon l'usage que l'on souhaite faire de ces vidéos.

Un format générique fonctionnant dans de nombreux cas sur les 3 systèmes d'exploitation est le MJPEG (Motion JPEG).

Le processus d'optimisation des vidéos est facilité par l'usage de logiciels dédiés tels FFMPEG et MENCODER. Ces deux outils de compression libres réalisent un ré-encodage précis et efficace tout en proposant des fonctionnalités d'automatisation bien utiles lorsque l'on a un nombre important de fichiers à convertir et à optimiser (traitement par lot massif). FFMPEG et MENCODER sont disponibles sur Linux ainsi que sur Mac OS X et Windows (pour ces derniers, la prise en main des logiciels s'effectue à l'aide d'une interface graphique).

32. UTILISER UNE CAMÉRA VIDÉO

Le patch ci-dessous permet **d'utiliser dans GEM un périphérique vidéo de type webcam, caméra DV ou carte d'acquisition vidéo analogique**, et de manipuler les images fournies par celui-ci. Il est ensuite possible d'appliquer des effets, d'opérer une analyse de mouvement ou de couleurs, de mélanger ce flux d'images avec d'autres éléments (textes, vidéos préenregistrées, images fixes, etc.). Au besoin, on peut utiliser **simultanément** plusieurs périphériques vidéos dans un même patch, la seule limite théorique étant la capacité de l'ordinateur à gérer ces flux.



Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/cameraDVwebcam.pd

POUR LE FAIRE FONCTIONNER

1. Créez d'abord la fenêtre d'affichage (1) en ayant préalablement cliqué sur les options d'affichage (0) ;
2. Choisissez le périphérique s'il n'apparaît pas immédiatement ;
3. Pour les webcams, changer la dimension du flux d'images en fonction des résolutions proposées par le périphérique.

CONSEILS D'UTILISATION POUR GNU/LINUX

- Il faut **choisir le pilote en fonction du périphérique** : s'agit-il d'une webcam v4l (librairie de capture vidéo en voie d'obsolescence supportant les « vieilles » webcam), v4l2 (nouvelle librairie de capture vidéo) ? Si vous avez un doute, le plus simple est d'essayer les 2 ou de chercher sur le Net, selon votre modèle de webcam et votre système d'exploitation, quelle sera la librairie de capture la plus adaptée.
- Attention ! la correspondance entre le numéro de pilote et la librairie de capture peut changer selon votre système d'exploitation ([driver 1< peut par exemple correspondre au DV si vous n'avez pas v4l). Fort heureusement, en créant l'objet [pix_video], la console de Pure Data donne cette indication :

```
[pix_video]: video driver 0: video4linux v4l
[pix_video]: video driver 1: v4l2
[pix video]: video driver 2: ieee1394 dv4l dv
```

- **Guvcview** est une application destinée à tirer partie d'une webcam compatible avec le très bon pilote uvcvideo. Elle permet notamment d'**opérer aux réglages habituels** (luminosité, contraste, exposition auto, ...) avec une interface graphique, pour voir les cameras compatibles : http://www.ideasonboard.org/uvc/

- **Pour les caméras DV**, il peut être nécessaire de s'ajouter au groupe « vidéo » de son système GNU/Linux. Pour cela, avec l'interface graphique d'Ubuntu : *Administration > Utilisateurs et groupes > Paramètres avancés > Privilèges utilisateur,* puis cocher : « utiliser des périphériques vidéo » et redémarrer l'ordinateur. Pour réaliser l'opération dans un terminal :

```
sudo addgroup nom utilisateur video
```

CONSEILS D'UTILISATION POUR MICROSOFT WINDOWS

- **Le message [enumerate<** envoie dans la console de *Pd* la liste des périphériques vidéos détectés. Il est ensuite possible de les ouvrir en choisissant [open num_de_mon_periphérique<.
- Les réglages de la webcam sont accessibles en cliquant sur **le message [dialog<**. L'interface habituelle de réglage des caméras de windows s'ouvre alors.
- **Pour les caméras DV**, il faut utiliser l'objet **[pix_videoDS]** et non pas [pix_video]. Se reporter à l'aide dans Pure Data pour les messages spécifiques.

CONSEILS D'UTILISATION POUR MAC OS X

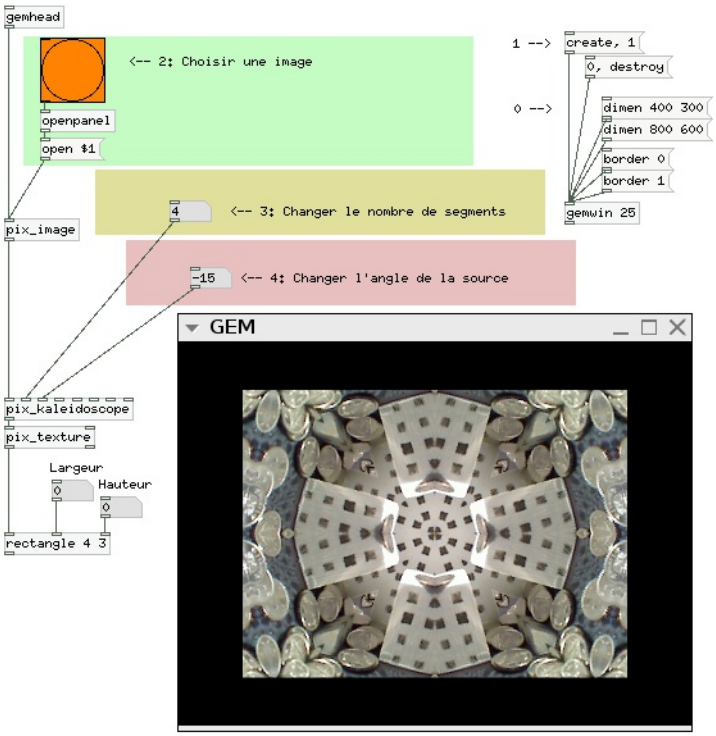
- **le message [dialog<** permet d'accéder à la boite de dialogue habituelle de Mac OS X pour gérer les caméras et webcam.
- Il existe l'application macam qui **rend compatibles la plupart des webcams** et autres entrées vidéo USB sur Mac OS X : http://webcam-osx.sourceforge.net/
- **Pour les caméras DV**, il peut être utile d'ajouter des paramètres de taille à l'objet : [pix_video 720 576], de jouer avec le message [quality $1< si l'image est pixelisée, et de cliquer sur les messages de réglage des espaces de couleur si l'image est bleutée.

33. APPLIQUER DES EFFETS SUR LES PIXELS

GEM offre plusieurs moyens d'altérer les pixels des images et des clips vidéo que l'on y affiche. Il offre notamment une panoplie d'objets dont le nom commence par "pix_". Ces objets manipulent les pixels des images qu'ils reçoivent.

[PIX_KALEIDOSCOPE] : JEU DE MIROIRS

Voici un exemple d'utilisation du très simple, mais ô combien amusant objet [pix_kaleidoscope]. Il imite le fameux effet du kaléidoscope, un jouet optique qui invite à regarder un paysage à travers un tube de miroirs. Cet objet ne prend aucun argument, mais accepte différents types de messages pour contrôler le kaléidoscope.

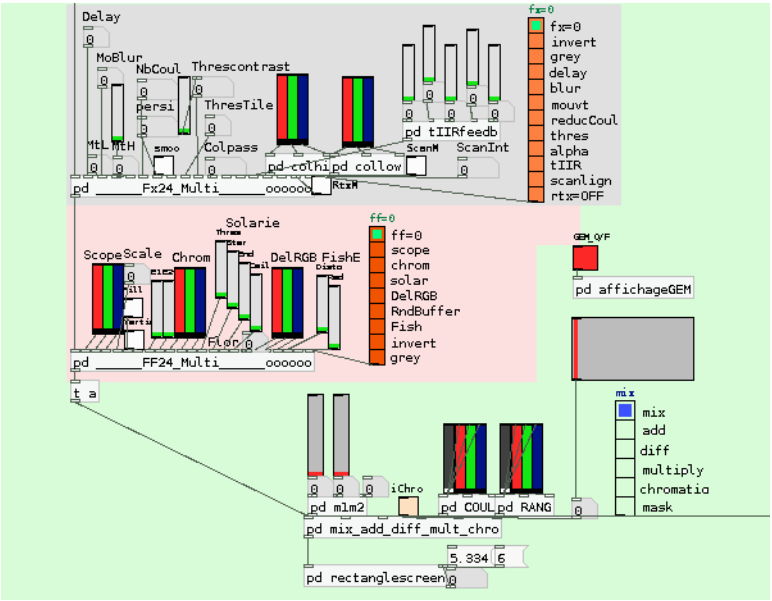


Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/effets_pixels_kaleidoscope.pd

Les effets des objets "pix_" peuvent vous aider à atteindre une esthétique appropriée à vos besoins. Toutefois, ces objets utilisent le processeur de l'ordinateur pour effectuer leurs calculs. Pour plus de performance et alléger la charge de votre ordinateur, il existe une méthode plus performante pour réaliser ce genre d'effets sur les pixels. Il s'agit des *shaders* (ombreurs), des programmes écrits dans des langages comme le GLSL pour ajuster avec finesse la manière dont les images et les formes sont rendues avec OpenGL. Si ce sujet vous intéresse, nous vous invitons à étudier les fonctionnalités proposées par les objets [glsl_program], [glsl_fragment] et [glsl_vertex] de GEM ainsi que leurs exemples d'utilisation.

34. MIXER DES TYPES DE CONTENUS

Cette section aborde **différentes approches pour réaliser des mélanges de contenus** : mixage ou superposition de vidéos ou d'images fixes, interactions combinées son <> image.



Extrait d'un patch de mixage vidéo

MIXER DES VIDÉOS

Il existe plusieurs façons de mélanger des vidéos (ou des images fixes) en opérant une combinaison de pixels selon différents modes. Considérons ici que l'on tente de **mélanger deux sources devant être de taille identique**. Il est possible d'utiliser l'objet [pix_resize] mais qui est fort gourmand en ressources. Cette remarque est valable pour tout ce sous-chapitre à propos du mixage. Certains de ces objets nécessitent de travailler dans l'espace colorimétrique RGBA (rouge, vert, bleu, transparence), il faudra alors intercaler l'objet [pix_rgba] sous une source qui ne serait pas dans ce mode. N'hésitez pas à consulter l'aide de chaque objet pour plus d'informations (bouton droit sur l'objet > *Help*).

[pix_mix] : le mélange par opacité

L'objet [pix_mix] accepte deux sources en entrée et permet de les mélanger comme une console de mix vidéo analogique en jouant sur l'opacité de l'une et l'autre des sources. À noter que ces sources doivent être de taille identique.

[pix_add] : le mélange par addition

Comme son nom l'indique, l'objet [pix_add] additionne la valeur de couleur de chaque pixel des deux sources. Par exemple, si l'on additionne une image colorée avec une image dont le fond est noir, ce fond noir n'apparaîtra pas du tout dans le mélange (le noir a une valeur de couleur en RGB de 0 0 0). Ainsi, un pixel gris moyen (0.5 0.5 0.5) avec un autre pixel gris clair (0.8 0.8 0.8) donneront un pixel blanc (1 1 1) puisque la valeur d'une couleur ne peut dépasser 1.

[pix_diff] : le mélange par différence

L'objet [pix_diff] réalise la différence des couleurs de chaque pixel des deux sources en soustrayant la valeur de l'un avec celle de l'autre.

[pix_multiply] : le mélange par multiplication

L'objet [pix_multiply] réalise la multiplication des couleurs de chaque pixel des deux sources.

[pix_chroma_key] : le mélange par élimination d'une plage de couleur

On indique à l'objet [pix_chromakey] une couleur de référence (prenons par exemple le vert pur) ainsi qu'une tolérance. Celui-ci retire alors de la seconde source la couleur en question et les couleurs « voisines » pour les remplacer par une zone transparente. L'usage typique à imaginer est la substitution de l'arrière-plan d'une vidéo tournée sur fond vert dans lequel on va pouvoir incruster une autre source (par exemple, « Superman » filmé en studio et que l'on place ensuite dans un ciel filmé en avion).

[pix_mask] : le mélange par soustraction des pixels noirs

L'objet [pix_mask] va transformer les pixels noirs de la deuxième source en pixels transparents.

[pix_composite] : le mélange par la couche Alpha

L'objet [pix_composite] mélange les deux sources vidéos ensemble en se basant sur la valeur de la couche Alpha de la première source.

SUPERPOSER DES VIDÉOS ET DES OBJETS GÉOMÉTRIQUES

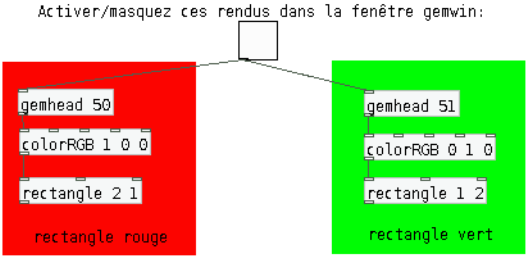
On a déjà vu dans les chapitres précédents comment afficher dans GEM des vidéos, un signal issu d'une webcam ou encore des objets géométriques OpenGL (carré, rectangle, cylindre, etc.). On a également évoqué, dans la partie dédiée à l'affichage, que si l'**on ouvre avec la même instance de Pure Data plusieurs patches contenant des objets GEM, ces objets se retrouvent tous « empilés » dans la même fenêtre**, le dernier objet affiché apparaissant au-dessus des autres. Cet ordre de rendu prend alors toute son importance, et nous détaillerons son utilisation dans le prochain point de ce chapitre.

La superposition offre notamment la possibilité de mixer deux vidéos en les plaçant chacune sur des textures rectangles singulières, l'une par-dessus l'autre. L'effet de mixage entre les deux vidéos est obtenu en jouant sur la transparence de la vidéo qui est positionnée au dessus de l'autre. Pour faire apparaître la vidéo « du dessous », il suffira de baisser l'opacité de la vidéo « du dessus », tel qu'indiqué dans le chapitre « Manipuler de la vidéo ».

Il faut également définir quelle vidéo est en dessous et quelle vidéo est au dessus.

LE BON ORDRE DE RENDU

Lorsque l'on créer plusieurs chaines, les [gemhead] peuvent avoir un argument. Il s'agit d'une valeur qui indiquera "l'ordre" dans lequel le rendu sera effectué. Par defaut, cet argument est de 50 Si on diminue cette valeur, l'objet sera alors dessiné AVANT les autres et apparaîtra "dessous":



Ici le rectangle vert sera calculé après (51) le rectangle rouge (50). Il en résultera que le rectangle vert sera affiché au-dessus du rectangle rouge.

Intervertissez les arguments 50 et 51 dans les objets [gemhead] pour renverser cette relation.

Patch issu des tutoriels de Raphaël Isdant
http://raphael.isdant.free.fr/puredata.htm

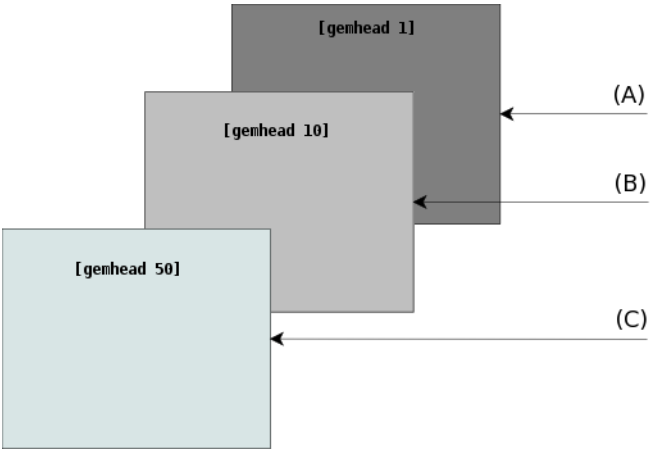
L'ORDRE DE RENDU

L'objet [gemhead] accepte un argument pour déterminer à quel moment il recevra la commande de rendu (ou d'affichage). La valeur de l'argument de [gemhead] par défaut est 50. Nous pourrions également changer la valeur de l'argument de manière spécifique, par un message précédé de set, par exemple : [set 55<

Plus faible est la valeur de l'argument, plus tôt le [gemhead] recevra la commande d'affichage. Plus la valeur est proche de zéro, plus tôt sera réalisé l'affichage.

GEM affichera les objets géométriques (ou de texte), en prenant pour règle que le premier affiché sera recouvert par le suivant, qui sera recouvert à son tour par le suivant, etc.

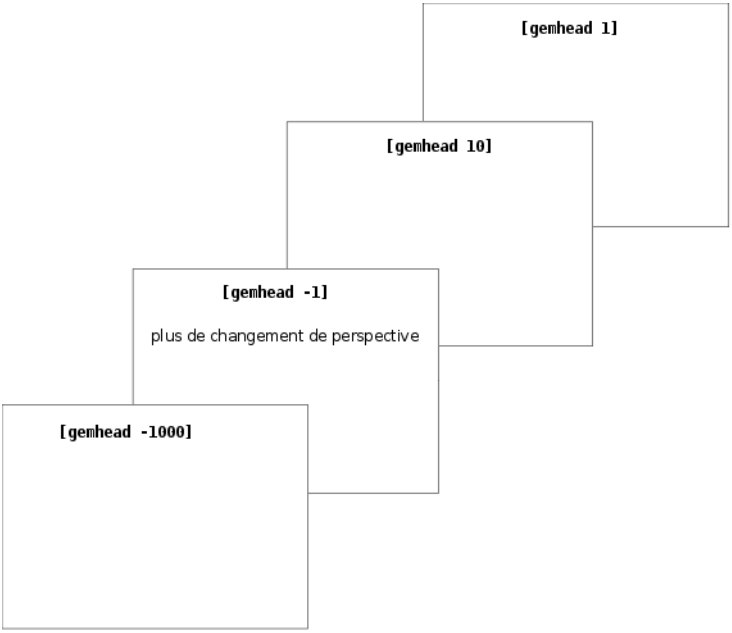
Voir 14.RenderOrder.pd dans
/usr/lib/pd/doc/examples/Gem/02.advanced/



(A) [gemhead 1] créé en premier

(B) [gemhead 10] recouvre le précédent [gemhead]

(C) [gemhead 50] valeur par défaut, recouvrira les précédents.



Le schéma ci-dessus présente le principe d'affichage progressif couche par couche du rendu, l'ordre de rendu des champs dépendant de valeurs positives et négatives. Le premier rectangle visible à l'avant-plan correspond au dernier à être calculé.

Les valeurs d'ordre d'affichage peuvent également être négatives. Les valeurs négatives des [gemhead] seront affichées après toutes les valeurs positives. Les primitives (carré, cercles, etc.) raccordées aux [gemhead] négatifs ne sont pas concernées par les changements de vue affectant la caméra (avec le message [view<). En revanche, elles seront concernées par un changement de perspective (avec le message [perspec<).

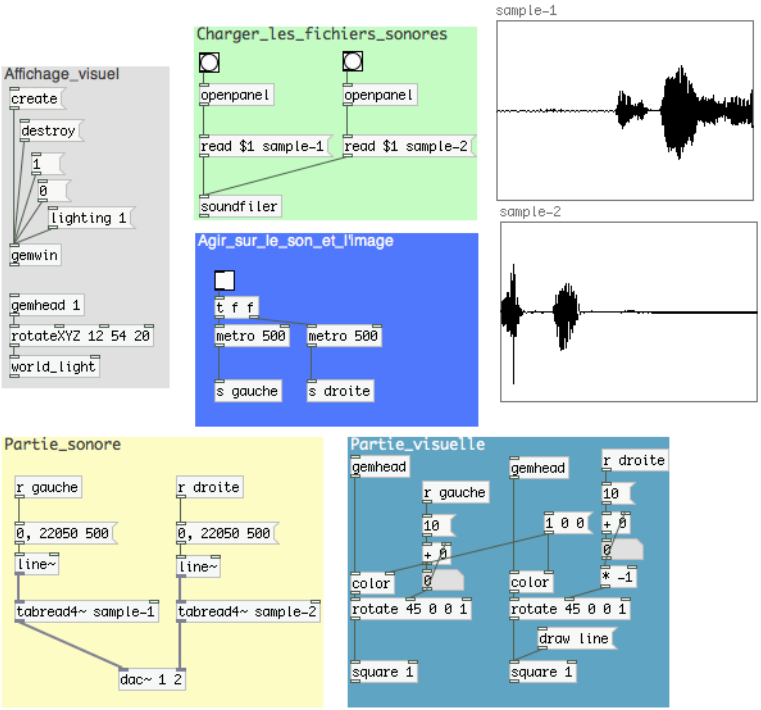
Remarques :

- La valeur -3, par exemple, sera affichée avant la valeur -10.
- Les [gemhead] avec des valeurs négatives ne seront pas affectés par les changements de point de vue (perspective).
- Toutes les valeurs négatives seront affichées APRÈS toutes les valeurs positives, en partant de la plus grande valeur négative (-1) vers la plus petite valeur négative, par exemple -1000.

On peut utiliser ce procédé pour réaliser des sous-titres, des masques, des avant-plans divers sur une scène dont on contrôle le point de vue.

FAIRE INTERAGIR LE SON ET L'IMAGE

Nous allons à présent rapidement évoquer une des nombreuses possibilités d'interaction son/image tout en abordant l'éclairage de la scène 3D de GEM (objet [world_light]). À partir de l'envoi d'une impulsion périodique, le patch ci-dessous va provoquer une interaction à la fois sur le son et l'image en synchronisant la lecture d'un échantillon sonore et la rotation d'un objet de forme carrée.



Fonctionnement

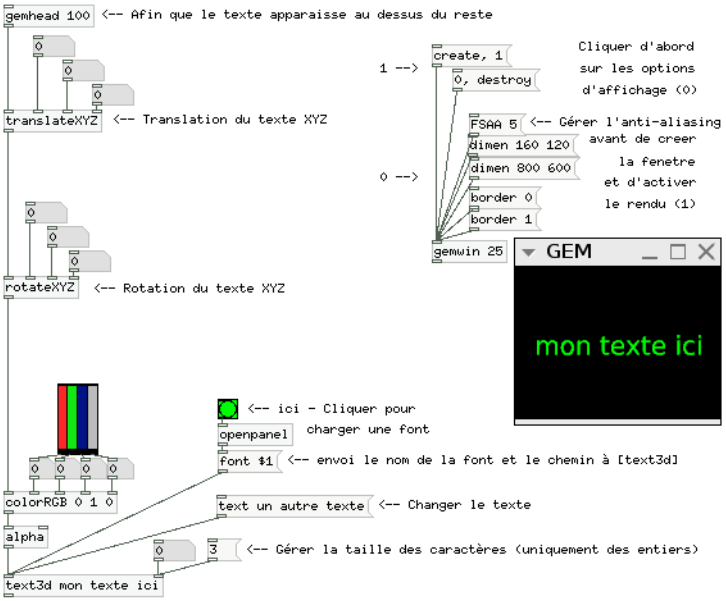
1. Choisissez et chargez les fichiers sonores par l'intermédiaire des deux bang dans la partie « Charger les fichiers sonores »
2. Créer la fenêtre GEM et afficher le rendu en envoyant les messages à partir de la zone « Affichage visuel »
3. Déclenchez les métronomes de la partie « Agir sur le son et l'image ». Attention, n'oubliez pas d'activer la sortie sonore générale de Pure Data (cocher "Compute Audio" dans l'interface principale de *Pd*).
4. Agissez sur les contrôles de la partie visuelle. Le message [1 0 0< va envoyer une valeur de couleur (rouge) à l'objet [color]. Le message [draw line< va demander à l'objet [square] de ne dessiner que les contours de la forme géométrique du carré.

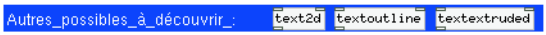
Remarques

- La partie sonore et la partie visuelle vont recevoir simultanément le même type d'information, c'est-à-dire une simple impulsion, ce qui induira une relation implicite entre le son et l'image.
- L'objet [world_light] va permettre un nouvel éclairage de la scène. Pour cela, il vous faudra d'abord activer le message [lighting 1<, qui enverra à l'objet [gemwin] l'ordre de calculer l'éclairage spécifique de la scène 3D. Vous pourrez contrôler l'orientation de l'éclairage grâce à l'objet [rotateXYZ]. Pour plus d'informations sur les propriétés de l'objet [world_light], veuillez consulter son patch d'aide.

35. INTÉGRER DU TEXTE

Dans ce chapitre, nous allons voir comment intégrer des éléments textuels dans les visuels GEM. On peut alors notamment déplacer le texte que l'on dessine dans la fenêtre de GEM, le faire tourner ou encore changer sa couleur.





Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/manipuler_texte.pd

Dans le patch ci-dessus, on affiche du texte grâce à l'objet [text3d]. Le fichier d'aide de cet objet va nous permettre d'accéder à quelques-unes des possibilités de l'objet.

Pour le faire fonctionner :

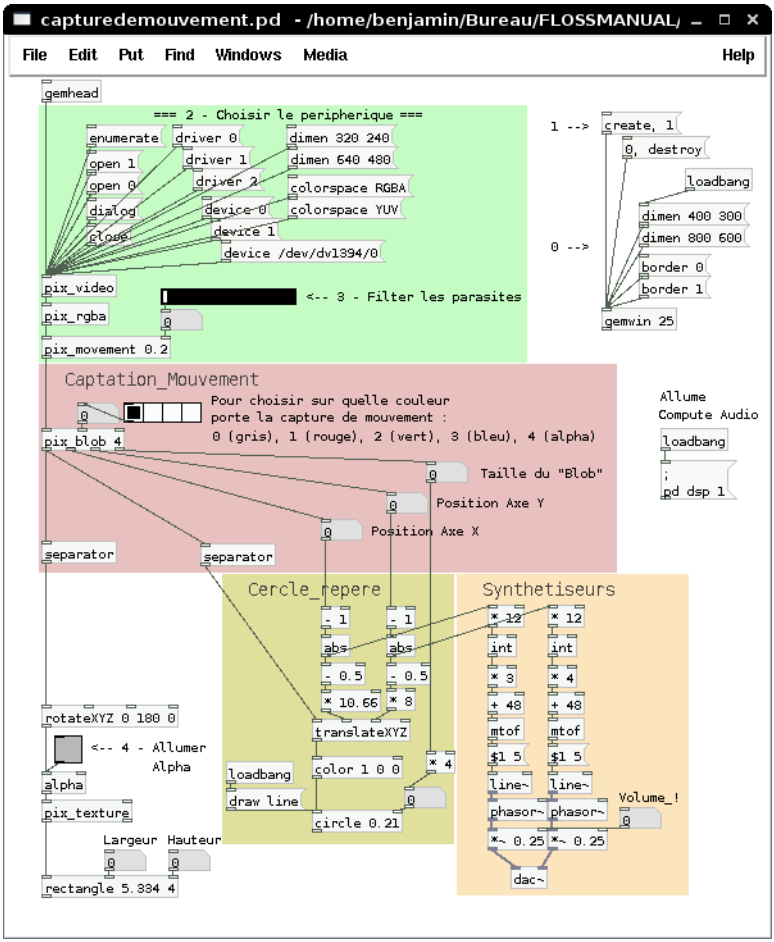
1. Choisissez les caractéristiques de la fenêtre gemwin. Remarquez ici la possibilité de définir un facteur d'anti-crénelage *(anti-aliasing)* avec le message **[FSAA <facteur> <**. Ce message peut être utilisé avec tous les objets géométriques, mais devient crucial pour traiter les caractères typographiques.
2. Créez la fenêtre et afficher aussitôt le rendu avec le message [create, 1<
3. Choisissez une police typographique ou adopter sa valeur par défaut *(vera.ttf)*
4. Testez les valeurs de couleur, d'alpha, de translation ou de rotation du texte.
5. Choisissez la taille du texte. Il est à noter que pour cet exemple, la taille du texte ne sera gérée que par des valeurs entières.

Remarque : pour faire apparaître le texte au-dessus d'un objet géométrique particulier de votre chaîne de calcul, il faudra faire attention à l'ordre du rendu des divers éléments en présence dans votre fenêtre d'affichage.

36. OPÉRER UNE ANALYSE DE MOUVEMENTS

Qu'est-ce qu'un « Blob » ? Ce n'est ni un blog enrhumé ni un cafard sans ailes, mais un terme technique utilisé par les spécialistes de **la vision par ordinateur** pour désigner le contour d'une zone de couleur détectée dans une image. Cette zone peut être produite par le contact d'un doigt sur une vitre, ou créée par des opérations sur l'image, notamment pour repérer des éléments en mouvement dans un flux vidéo, exemple que l'on traitera dans ce chapitre. En vision par ordinateur, on tente d'estimer au plus juste les contours et les coordonnées des formes.

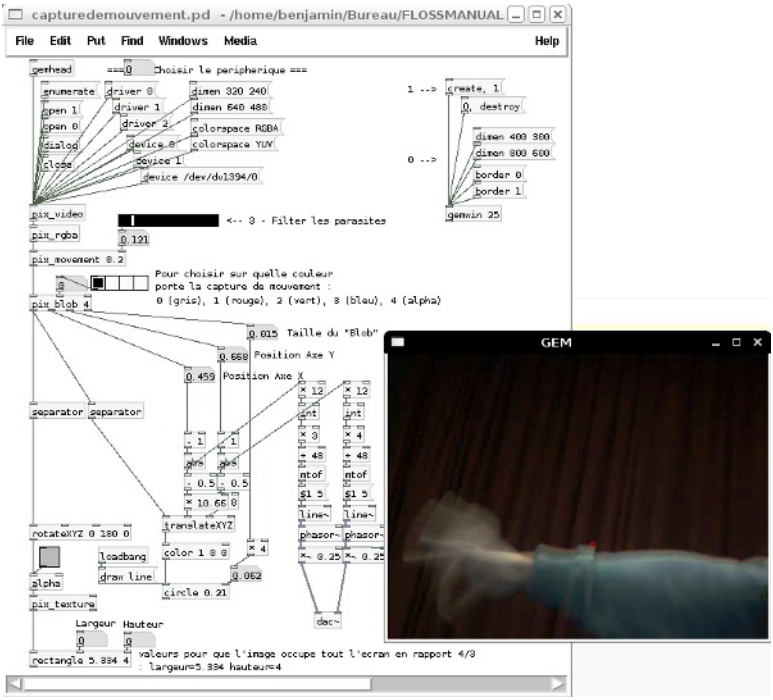
La librairie GEM propose plusieurs approches pour l'analyse du mouvement. On se limitera ici à une opération « monoblob », c'est-à-dire qui définira une seule zone de mouvement dans toute l'image et donnera donc comme information la position en X et Y de cette zone ainsi que sa taille. Ces trois informations obtenues en temps réel permettent déjà d'imaginer de nombreuses créations, la caméra ou la webcam s'avérant être des capteurs performants et relativement peu coûteux (vous pouvez même envisager de détourner la webcam présente sur certaines consoles de jeux). **Le patch proposé ici transforme ces coordonnées X et Y en notes de musique en fonction des mouvements** effectués devant la caméra.



Lien vers le patch : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/capturedemouvement.pd

Pour le faire fonctionner :

1. Créez d'abord la fenêtre d'affichage (1) en ayant préalablement cliqué sur les options d'affichage (0).
2. Choisissez le périphérique s'il n'apparaît pas immédiatement, changez éventuellement la dimension du flux d'images en fonction des résolutions proposées par le périphérique afin d'améliorer la qualité de la détection.
3. Modifiez le paramètre de l'objet [pix_movement]. Cet objet réalise un filtre en comparant les deux dernières images du flux vidéo et en ne « gardant » que la différence, c'est-à-dire ce qui bouge.
4. Pour voir la vidéo « normale » sans l'effet du filtre, cliquez sur l'interrupteur "Toggle".
5. En bougeant devant la caméra, vous devriez entendre un son de synthèse changer de tonalité, voir un cercle rouge vous suivre et s'agrandir avec l'accélération de vos déplacements.



Il serait tout à fait possible de décliner l'exemple ci-dessus en ne considérant par exemple qu'une portion de l'image **pour accomplir plutôt des détections par zone** (voir l'objet [pix_crop]).

Une autre possibilité serait de **privilégier une gamme de couleurs plutôt qu'une autre**. L'objet [pix_blob] peut être configuré dans ce sens (voir patch), mais il faut toujours considérer que la couleur blanche est « vue » d'un point de vue informatique comme la somme des luminosités du rouge, du vert et du bleu. En essayant de ne capter que la position d'éléments rouges, l'ordinateur prendra également en considération les éléments blancs. Pour éviter ce phénomène, il faudra mettre en place un système de filtrage soit optique soit numérique en amont de la captation de mouvement.

GEM propose **d'autres objets de captation de mouvement**, notamment [pix_multiblob] qui permet d'identifier plusieurs zones de mouvement dans l'image, ou encore [pix_data] ou [pix_mean_color] pour analyser les changements de couleur dans une portion de l'image (voir dans l'exemple ci-dessous.)

QUELQUES CONSEILS

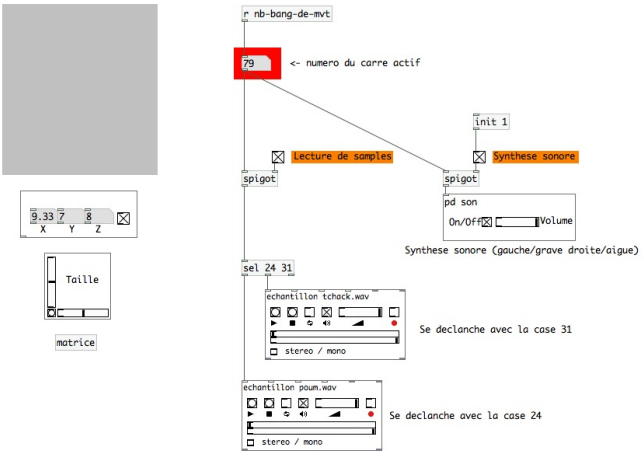
La captation de mouvement est un art qu'il n'est pas toujours aisé de maîtriser. Il faut **être vigilant sur l'éclairage et la qualité de la prise de vue** en cherchant parfois des astuces pour faciliter le travail de l'ordinateur. Cela peut passer par l'utilisation d'une caméra infrarouge lorsqu'on cherche le mouvement de personnes noyées dans une vidéoprojection, ou l'utilisation de décors contrastés avec ce qui est supposé être mobile, voire l'apposition de leds infrarouges sur les vêtements d'un danseur pour grandement améliorer la précision.

Les algorithmes de vision par ordinateur ont beaucoup progressé depuis quelque temps, grâce entre autres au développement de librairies comme **OpenCV** (initiée par Intel), sous licence libre, qui propose des objets pour réaliser de la détection de visages, de formes complexes. Cette librairie est intégrée à Pure Data et GEM dans une extension disponible ici :
http://www.hangar.org/wikis/lab/doku.php?id=start:puredata_opencv

Les périphériques de détection de mouvement conçus initialement pour certaines consoles de jeux comme la « **Kinect** » de Microsoft ou la « **Wiimote** » de Nintendo peuvent constituer des pistes intéressantes à explorer. Depuis leur apparition, de nombreux artistes les ont détournés de leur usage premier grâce à l'utilisation de pilotes (*drivers*) libres qui permettent de les faire interagir directement avec un ordinateur.

AUTRE EXEMPLE

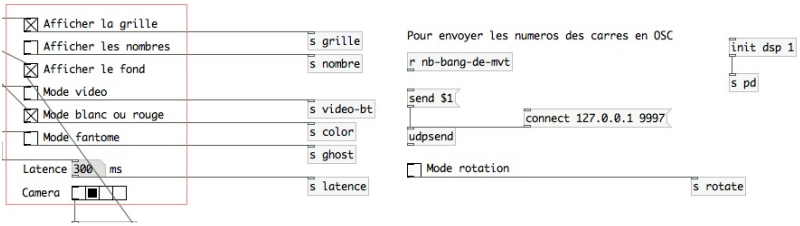
Voici un patch permettant la détection de mouvement avec l'objet [pix_mean_color], qui analyse le changement de couleur RGBA par zone. Quand il y a un trop grand changement de valeur colorimétrique dans une zone, elle devient rouge et son numéro est retourné à l'utilisateur. Vous y trouverez certains des objets présentés dans l'exemple précédent, tel que [pix_crop] pour la détection par zone et [pix_movement].

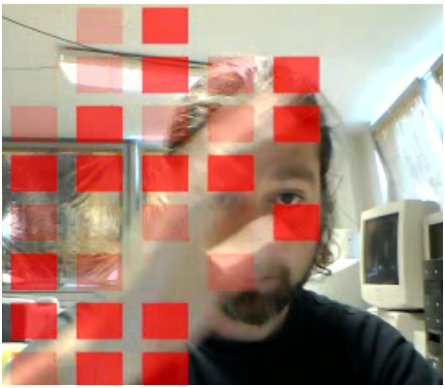


Lien vers le patch : http://yamatierea.org/papatchs/#detc-mvt

Pour le faire fonctionner :

1. La fenêtre d'affichage s'allume d'elle-même au moment de l'ouverture du patch.
2. En Cliquant sur l'interrupteur "lecture de samples", ce la permet de faire activer des échantillons audio par le mouvement effectué dans certaines zones. Vous pourrez changer les cases affectées (dans ce cas, 31 et 24), en ajouter, ainsi que déterminer d'autres effets qui pourront être actionnés.
3. Dans le patch "control" vous pourrez modifier plusieurs paramètres, tels que l'affichage de la grille, des nombres, du fond, ou encore allumer ou éteindre le mode vidéo, activer le mode blanc ou rouge, le mode fantôme, la latence, sélectionner le périphérique de caméra et pour l'utilisation en **OSC** (se référer au chapitre "Communication").
4. En bougeant devant la caméra vous verrez les zones s'activer en rouge (ou selon les paramètres choisi précédemment) et entendrez le son de synthèse changer de tonalité selon les zones activées (gauche = grave, droite = aiguë).



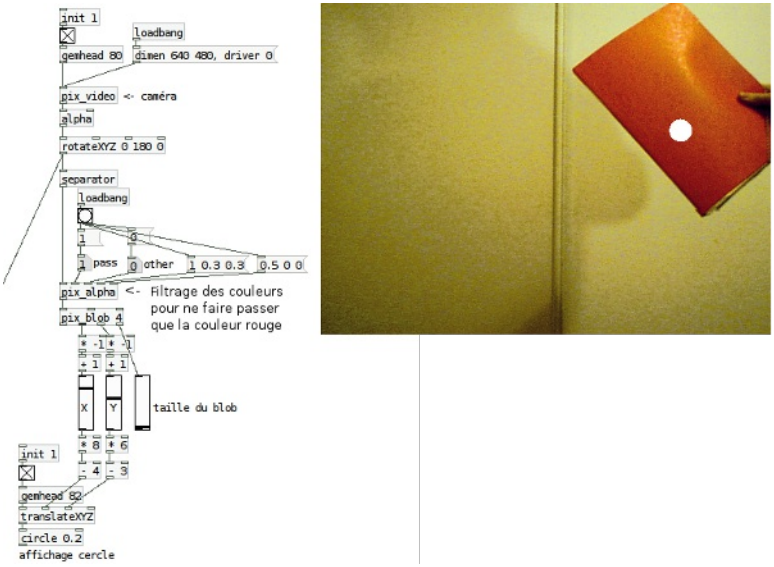


Ce patch à été tissée par Olivier Baudu au sein du Labomedia avec l'aide de Benjamin Cadon.

37. OPÉRER UNE DÉTECTION DE COULEURS

Continuons de visiter les techniques de vision par ordinateur. Cette fois, il s'agit simplement de repérer une zone de couleur (un « blob » en langage technique), et de récupérer ses coordonnées X et Y.

[PIX_ALPHA] : FILTRER LES COULEURS



Extrait du patch disponible ici : http://gitorious.org/flossmanuals-fr/pure-data/blobs/raw/master/patchs/GEM/detectiondecouleurrouge.pd

Dans cet exemple, le cercle blanc suit la chemise rouge. Nous pourrions donc réutiliser ces coordonnées pour manipuler d'autres types de médias ou d'effets !

L'astuce, récupérée sur le forum puredata.hurleur.com, consiste à utiliser l'objet **[pix_alpha]** pour filtrer la couleur que l'on souhaite, en combinaison avec l'objet [pix_blob] déjà traité plus haut. En général, les couleurs détectées sont des couleurs primaires.

Les valeurs passées à [pix_alpha] sont deux seuils de détection (*pass* et *other*) avec leurs valeurs en couleurs rouge-vert-bleu. Ainsi les messages [1 0.3 0.3< et [0.5 0 0< signifient que nous détectons la composante rouge entre une intensité moyenne (0.5) jusqu'à son maximum (1). Les autres composantes vertes et bleues sont atténuées, car nous ne prenons en compte que leurs intensités comprises entre 0 et 0.3.

Pour détecter la couleur verte, nous pourrions utiliser les messages [0.3 1 0.3< et [0 0.5 0<.

VIDÉO (PDP)

38. PDP : PURE DATA PACKETS

38. PDP : PURE DATA PACKETS

PDP est une extension gérant des flux vidéos, développée par Tom Shouten.

Il existe également une extension de PDP dénommée PIDIP proposant de nombreuses fonctionnalités supplémentaires (effets vidéo, dont EffectTV et FreiOr, texte, streaming, enregistrement directement vers le disque, etc.). Celle-ci est développée principalement par Yves Degoyon avec la collaboration de Lluis Gomez i Bigorda et Tatiana de la O.

Ces librairies sont incluses dans la version 0.42.5 de Pd-extended, et sont disponibles séparément pour Pure Data Vanilla (mais incluse dans la distribution Puredyne http://puredyne.org).

On peut noter quelques caractéristiques de PDP :

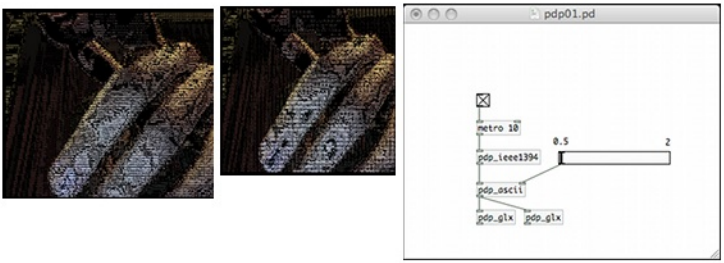
- fonctionne sur les systèmes POSIX : GNU/Linux et Mac OS X (si X11 est installé) ;
- utilise un système de gestion de flux d'information correspondant à celui du son en « chute d'eau », du haut vers le bas ;
- autorise le multi-fenêtrage avec plusieurs systèmes de gestion de fenêtres (sorties XV, GLX et SDL) ;

Le principe de PDP fonctionne sur le traitement de matrices. On travaille toujours avec des matrices de même taille et de même type de données. Il faut dès lors s'assurer que les images utilisées sont de la même dimension pour pouvoir les combiner.

Il y a beaucoup d'objets différents dans ces librairies : effets vidéo de transformation et de mixage, ajout de texte, détection et poursuite de mouvement, analyse de couleur, ainsi que des outils de streaming depuis et vers un serveur Icecast, comme Giss.tv.

EXEMPLE D'ARCHITECTURE DES FLUX

Voici un exemple de patch vidéo avec trois objets propres à PDP/Pidip.



- [pdp_ieee1394] : lit la sortie d'une webcam ou d'une caméra FireWire et la transforme en série de paquets PDP.
- [pdp_ascii] : transforme chaque paquet (ou image PDP) en image-texte (*ascii-art*).
- [pdp_glx] : envoie les images pour les afficher dans une fenêtre.

Lecture du patch : on appuie sur l'interrupteur. Cela lance le métronome qui cadence alors le flux vidéo, autrement dit l'envoi des « paquets » (ici cent fois par seconde, car un il y a cent fois dix millisecondes dans une seconde) depuis la caméra vidéo. Un effet transforme l'image en art ASCII et l'image est envoyée sur deux fenêtres d'affichage (ici de type GLX) créées automatiquement dès que les objets pdp_glx reçoivent une image. La glissière horizontale permet de régler la taille des caractères.

Note : si l'on désactive l'interrupteur, les images restent. Chaque nouvelle image remplace la précédente. Si on cesse d'envoyer de nouvelles images, c'est la dernière qui reste.

LES FONCTIONNALITÉS DE BASE

Sorties vidéo

PDP accepte trois types de fenêtres de sortie, basées sur des méthodes différentes de contrôle de l'écran. Chacune a ses avantages et inconvénients, liés au système sur lequel elles sont utilisées.

- [pdp_xv] sortie X11 (sur GNU/Linux avec XVideo) : utilise le rendu direct de X11 (gestionnaire de l'interface graphique).
- [pdp_glx] sortie X11 (sur GNU/Linux et Max OS X) : utilise le rendu OpenGL (accélération matérielle du rendu par la carte graphique).
- [pdp_sdl] sortie SDL (sous GNU/Linux) : rendu qui permet de se passer de X11 (moins utilisé).

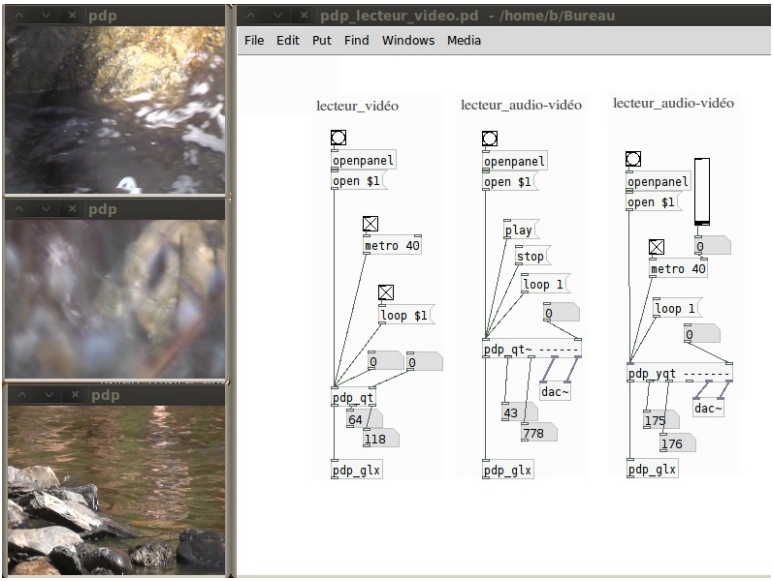
L'avantage de PDP par rapport à GEM est notamment la possibilité de sortir le signal dans plusieurs fenêtres indépendantes, et donc de faire du multi-écran de manière très simple.

LECTEURS VIDÉO

PDP offre plusieurs objets pour lire des fichiers vidéo.

- [pdp_qt] lecteur vidéo simple basé sur quicktime4linux.
- [pdp_qt~] lecteur audio-vidéo simple quicktime4linux.
- [pdp_yqt~] lecteur audio-vidéo Pidip (modification de pdp_qt~).

Ces lecteurs permettent de lire des fichiers vidéo compris par la librairie quicktime4linux ou Quicktime (sur Mac OS X).



Pour lire des fichiers OGG/Vorbis/Theora, on utilisera [pdp_theorin~].

Capture vidéo

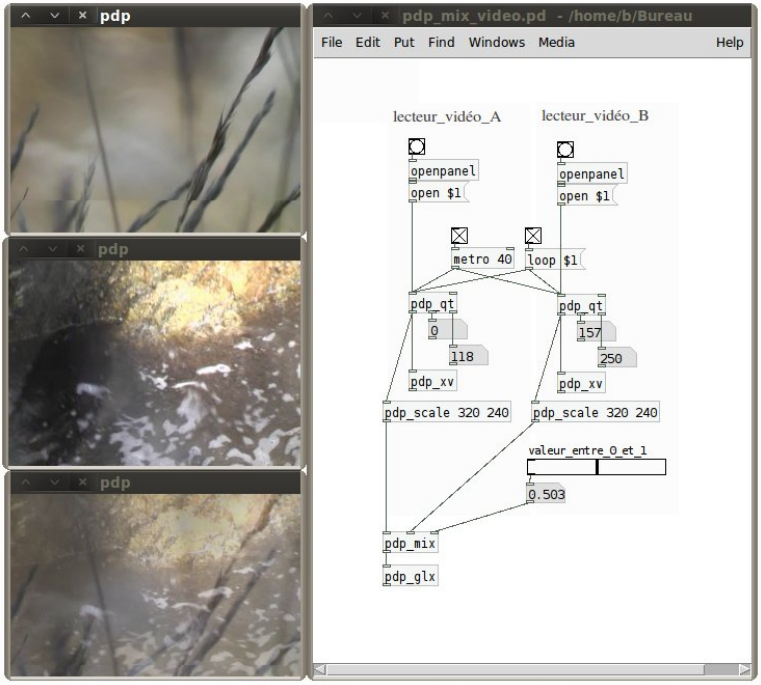
PDP peut capturer de la vidéo en temps réel à partir de sources externes diverses : webcam, cartes d'acquisition analogique, ou caméra DV.

- [pdp_v4l] acquisition webcam / carte tuner (ancien pilote).
- [pdp_v4l2] acquisition webcam / carte tuner (nouveau pilote, à préférer).
- [pdp_i1394] capture DV.

Ces objets ne redimensionnent pas les images capturées, il faut donc toujours s'assurer qu'elles sont de même taille que les images avec lesquelles on souhaite les mixer.

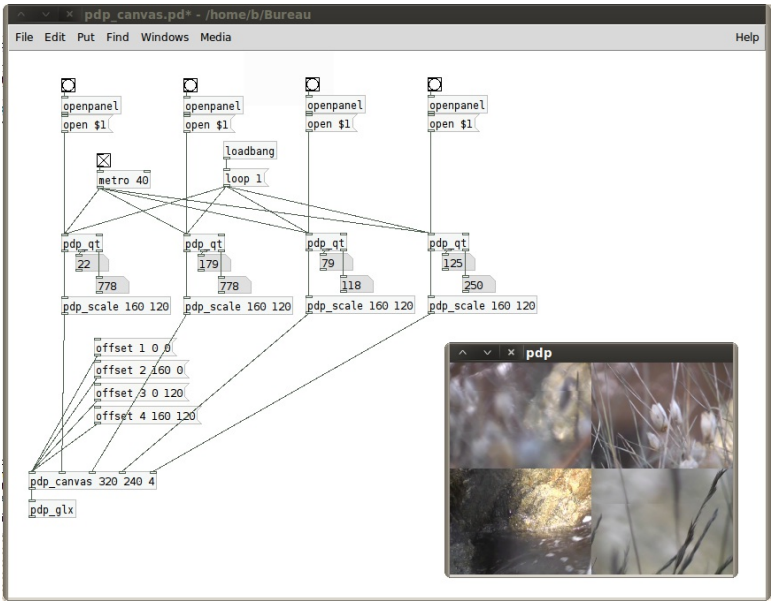
Les effets vidéo de base

- [pdp_mix] compose deux signaux vidéo, suivant un coefficient (0 = entrée gauche, 1 = entrée droite).
- [pdp_scale] redimensionne une vidéo (attention : étape à utiliser avec précaution, assez lourde pour le processeur).



- [pdp_gain] applique un gain de luminosité ou de couleur.
- [pdp_add] additionne deux flux vidéos.
- [pdp_mul] multiplie deux flux vidéos.
- [pdp_zoom] resserre le cadre et agrandit une vidéo.
- [pdp_canvas] superpose/juxtapose des vidéos dans un écran.

Exemple combinant quatre fichiers vidéo dans une grille :



Analyse d'images

Voici quelques objets permettant de faire une analyse de l'image pour en ressortir des caractéristiques, et/ou créer une synthèse sonore.

- [pdp_scan~] synthèse sonore à partir de l'analyse angulaire du plan de luminosité de l'image.
- [pdp_scanyz~] synthèse sonore à partir de l'analyse plane du plan de luminosité de l'image.
- [pdp_hue] analyse la couleur globale de l'image et sort les coordonnées RGB.
- [pdp_ctrack] traque un « blob » de pixel d'une couleur donnée, et en ressort la position X et Y ainsi que la taille de la zone suivie.

Il existe une extension à PDP et GEM intégrant la librairie OpenCV d'analyse d'image, permettant un travail très poussé dans ce domaine : http://www.hangar.org/wikis/lab/doku.php?id=start:puredata_opencv

Enregistrements

- [pdp_rec~] enregistre l'audio et la vidéo dans un fichier Quicktime en YUV420P.
- [pdp_theorout~] enregistre l'audio et la vidéo en OGG/Vorbis/Theora.
- [pdp_loop] enregistre une série d'images dans une boucle de taille variable (comparable à une ligne de délais).

Connections Réseau (*streaming*)

- [pdp_netsend] et [pdp_netreceive] permettent l'échange de paquets PDP entre des machines distantes, en UDP ou TCP. L'envoi est non compressé, et nécessite donc une bande passante importante. Cette méthode permet de conserver la qualité de l'image au maximum et de minimiser le délai (en général de l'ordre d'une image sur un réseau local).
- [pdp_theonice~] et [pdp_icedtheo~] permettent d'envoyer et recevoir du flux audio-vidéo vers et depuis un serveur Icecast2 et Giss.tv (remarque : selon la version de libtheora utilisée pour compiler ces objets, ils peuvent se révéler non fonctionnels sur certaines distributions... à tester avec prudence donc).

COMMUNICATION

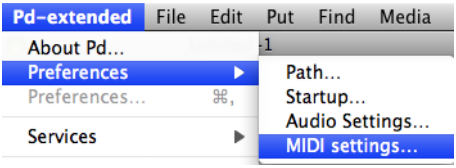
39. LE MIDI
40. OSC
41. ARDUINO ET PURE DATA

39. LE MIDI

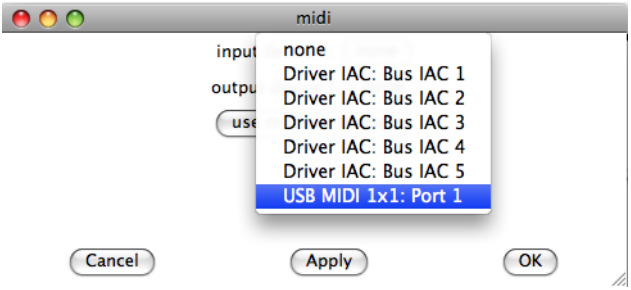
Ce chapitre traite de l'utilisation du MIDI afin de recevoir des notes depuis un clavier ou des interfaces MIDI standards, ainsi que des façons d'envoyer des informations en MIDI vers d'autres programmes ou des appareils MIDI.

MISE EN ROUTE

Avant de commencer à utiliser le MIDI, il faut sélectionner le périphérique dans la boîte de dialogue *Preferences > MIDI Settings*.



Sélectionnez le périphérique souhaité comme entrée/sortie MIDI. Par défaut, aucun périphérique n'est activé. Si le périphérique n'apparaît pas, débranchez-le, puis reconnectez-le au port USB de l'ordinateur ou au port MIDI de la carte son, puis relancez Pure Data.



Note pour les utilisateurs de Mac OS X : dans certains cas, vous devrez vérifier et activer le périphérique dans l'application "*Audio midi Setup*", qui se trouve dans dossier Utilitaires (voir exemple ci-dessous).

CANAUX ET PORTS

Une fois le périphérique MIDI sélectionné, il faut s'assurer qu'il fonctionne convenablement. Pour cela, mieux vaut en savoir un minimum sur le concept de canaux MIDI (*MIDI channels*).

Les canaux MIDI sont utilisés pour identifier les périphériques, afin de recevoir et envoyer des notes vers des interfaces physiques ou virtuelles spécifiques. En général, un périphérique utilise un seul canal pour envoyer et recevoir ses informations. Traditionnellement, il y a 16 canaux MIDI au total.

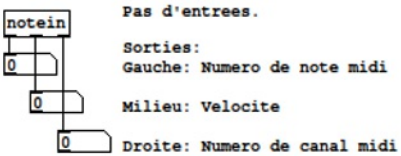
Dans Pure Data, certains messages ont un canal qui est plus grand que 16. Pure Data nous indique ainsi le numéro du port MIDI de Pure Data sur lequel chaque appareil est branché. En effet, Pure Data ajoute 16 fois le numéro du port au chiffre du canal de l'appareil.

Utiliser plusieurs périphériques

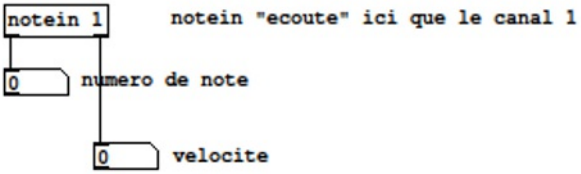
Pure data peut aussi utiliser plusieurs périphériques MIDI en simultané. Il suffit pour cela de cliquer sur "*use multiple devices*" dans le dialogue "*MIDI Settings*" (voir ci-dessus), et d'ajouter le périphérique sur n'importe quel port libre.

Il est aussi possible de filtrer les messages MIDI en spécifiant quel port ou canal doit être « entendu », en utilisant un argument dans l'objet MIDI. Les objets qui reçoivent ou envoient des notes MIDI sont respectivement [notein] et [noteout].

Voici un extrait de l'aide interne de Pure Data pour l'objet [notein]. C'est l'objet à utiliser pour connecter un clavier MIDI (ou n'importe quel autre type de périphérique MIDI) à Pure Data.



L'objet [notein] lit les notes MIDI reçues par Pure Data et envoie le numéro de chaque note reçue, sa vélocité (la force à laquelle elle est jouée) et son numéro de canal. Lorsqu'on rencontre une note qui a une vélocité de zéro, cela signifie que la note est relâchée. C'est la fin de cette note.



APPAREILS MIDI

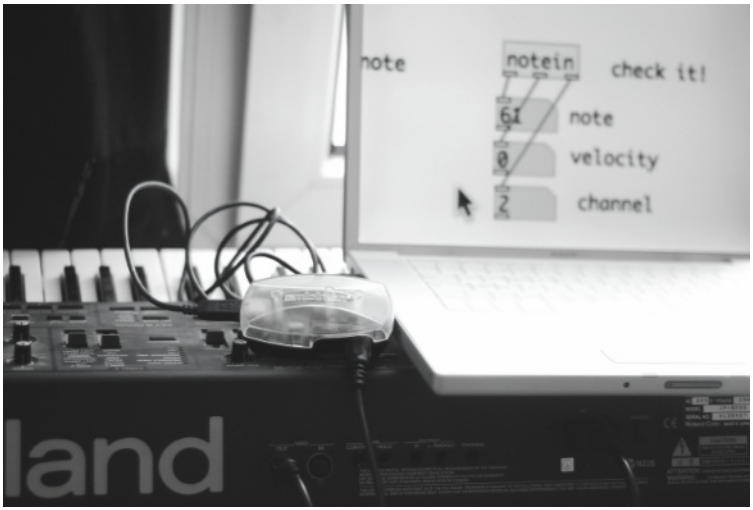
Depuis que le MIDI a été développé dans les années 1980, on trouve de nombreux périphériques compatibles, utilisables sous Pure Data. Synthétiseurs, claviers, interfaces de contrôle, interfaces audio, etc.

Le cable MIDI original au format DIN 5 est toutefois de plus en plus délaissé, au profit de contrôleurs USB *plug and play*, qui permettent de se passer d'une réelle interface MIDI.

Attention aux erreurs de branchement ! Les câbles MIDI standards étant dotés de connecteurs mâle/mâle, et les sorties/entrées sont indifféremment munies des connecteurs femelles.

Pour connecter un clavier MIDI à Pure Data, il faut simplement relier la sortie MIDI du clavier à l'entrée MIDI de l'interface audio ou de l'interface MIDI ou USB.


and
OUT
MIDI
IN
PEDAL
CONTROL HOLD


note
notein
check it!
61
note
0
velocity
2
channel
land

Ceci est une configuration Clavier MIDI > interface MIDI > ordinateur.

Dans le dialogue de configuration MIDI (*Preferences* > *MIDI Settings*), il faut sélectionner l'interface en tant qu'entrée MIDI (*MIDI input*).

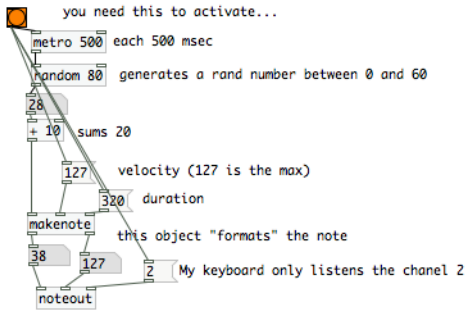
Ensuite, en créant un objet [notein] dans Pure Data, on récupère les notes jouées sur le clavier, leur hauteur, leur valeur et le canal sur lequel elles transitent. Ici, on peut voir que le clavier transmet uniquement sur le canal 2. La plupart du temps, le canal utilisé par un périphérique peut être modifié dans sa configuration interne, soit par un menu interne au périphérique, soit par un programme de configuration fourni par le fabricant.

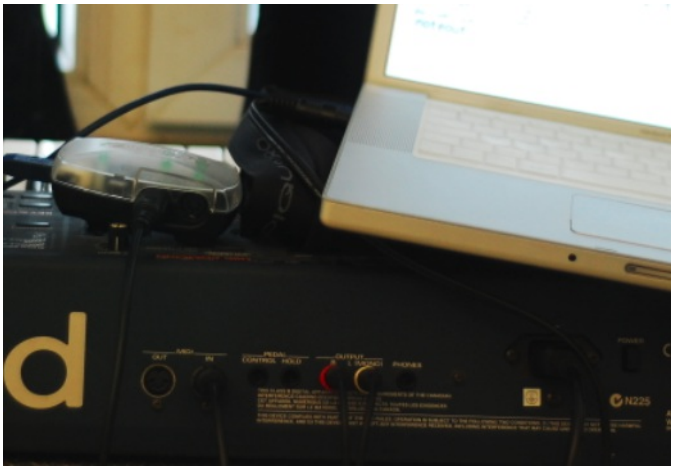
JOUER AVEC LES NOTES MIDI

Une note MIDI standard est un message composé d'un nombre qui détermine la hauteur de la note (*pitch*), sa vélocité (son volume sonore) et un message de fin (*note off*), qui contient la même hauteur de note avec cette fois une vélocité de 0. Pour envoyer des notes de Pure Data vers un périphérique ou une application extérieure, on utilisera l'objet [noteout].

Le patch ci-dessous génère des nombres aléatoires, puis convertit ces nombres en notes MIDI (hauteur + vélocité + message de fin), et les envoie vers l'objet [noteout].

Si le même port (*MIDI bus*) est sélectionné à la fois en entrée et en sortie, on peut alors envoyer des signaux MIDI entre différents patches Pure Data.





Ceci est une configuration Pure Data > Interface MIDI > Synthétiseur matériel.

Il faut alors sélectionner l'interface MIDI comme un périphérique de sortie (*output device*) dans le dialogue de configuration MIDI de Pure Data. (*Preferences > MIDI settings)*

Pour adresser des notes MIDI à l'appareil, envoyez le chiffre 2 dans l'entrée droite (froide) de l'objet [noteout], puis envoyez un bang dans son entrée gauche (chaude).

Si on ne connaît pas précisément le canal par défaut du matériel utilisé, on peut se reporter à la documentation fournie par son constructeur, ou bien essayer les différents canaux disponibles via Pure Data, jusqu'à ce que l'appareil reçoive une information.

CONTRÔLEURS MIDI

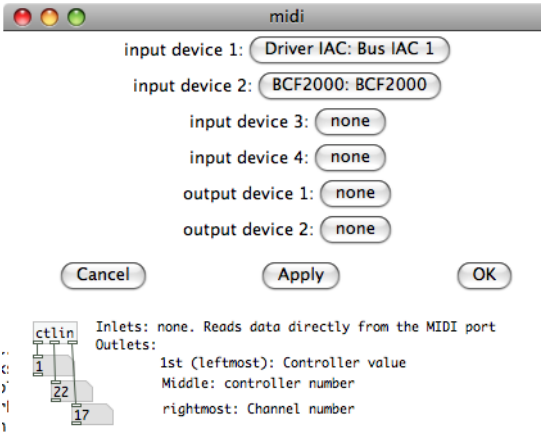
Les contrôles MIDI proviennent des pédales, glissières, gradateurs, encodeurs, potentiomètres et autres boutons des appareils MIDI. Ils servent à contrôler des paramètres divers. Pour recevoir ces contrôles dans des patches Pure Data, il faut utiliser l'objet [ctlin], qui possède trois sorties.

La sortie de gauche de [ctlin] nous donne la valeur (*controller value*) du contrôle. Cette valeur ne peut pas dépasser l'intervalle de 0 et 127, inclusivement.

La sortie du milieu de [ctlin] nous donne le numéro du contrôle (*controller number*). Ce chiffre peut être n'importe quel nombre entre 0 et 127, inclusivement. Il est en général unique pour chaque contrôle d'un appareil.

La sortie de droite nous informe sur le numéro de canal MIDI et le port de l'appareil.

Dans l'image ci-dessous, on peut voir la sortie de l'objet [ctlin] quand on tourne un des encodeurs d'un contrôleur, connecté au port 2.



Quand on crée un objet [ctlin] sans définir aucun argument, il « écoute » tous les numéros de contrôle et tous les canaux. Cela peut être utile en tant qu'outil d'analyse pour savoir quel numéro de contrôle et quel canal sont attribués aux différents boutons/potentiomètres du contrôleur utilisé.

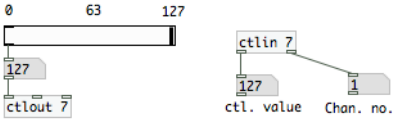
Avec certains contrôleurs MIDI, les faders/boutons/potentiomètres utilisent le même numéro de contrôle que d'autres boutons/potentiomètres, mais sur un canal MIDI différent.

On ajoute alors les arguments cc (*control change*) et un numéro de canal/port à l'objet [ctlin] pour le faire « écouter » uniquement le cc et le canal spécifié. Dans la mesure où les contrôleurs sont dotés de nombreux boutons, faders, potentiomètres, etc., il est important qu'à chacun soit attribué un usage particulier.

Quand on analyse chaque fader/potentiomètre/bouton, on peut créer un objet [ctlin] spécifique pour écouter un seul son. C'est là un moyen de créer une abstraction pour avoir tous les numéros de contrôle du contrôleur MIDI assignés à une sortie spécifique.

ENVOYER DES CONTRÔLES MIDI VERS D'AUTRES PROGRAMMES

On peut utiliser Pure Data pour envoyer des notes vers un autre programme lancé sur le même ordinateur. Il faut alors sélectionner le même port MIDI (*MIDI bus*) comme périphérique de sortie dans Pure Data (*MIDI output device*) et comme périphérique d'entrée (*MIDI in*) dans le programme que l'on souhaite utiliser.



Dans le patch ci-dessus, on envoie les valeurs d'une glissière (*slider*) vers le contrôle numéro 7 sur le canal 1.

AUTRES OBJETS MIDI

Il y a de nombreux autres objets MIDI dans Pure Data, et la plupart d'entre eux sont documentés dans l'aide interne du programme.

[pgmin] / [pgmout] : ces objets reçoivent et envoient des changements de programme (*program changes*), qui sont utilisés pour des sons d'instrument, ou des rythmes, par exemple.

[bendin] / [bendout] : ces objets reçoivent et envoient des variations sur la hauteur de toutes les notes (*pitchbend*). C'est utile pour le vibrato.

Tous les objets vus jusqu'à présent constituent les bases pour l'utilisation du MIDI dans Pure Data. D'autres tels que [midiin] ou [sysexin] ne fonctionnent que sur GNU/Linux. Enfin, il existe d'autres objets tels que [touchin] ou [polytouchin], qui ne sont pas encore documentés. Pd-extended propose dans ses librairies (Maxlib et Cyclone, par exemple) un certain nombre d'autres objets.

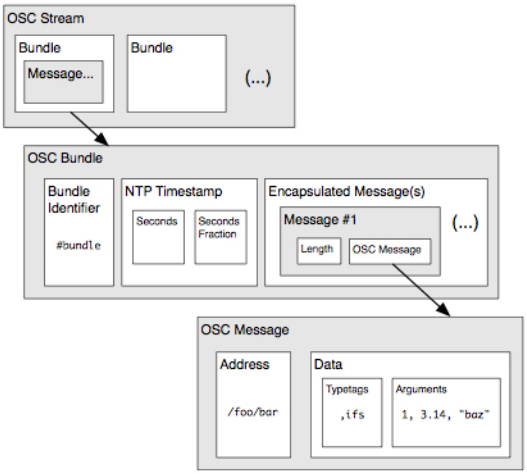
40. OSC

Il existe de multiples protocoles de communication entre applications, permettant le contrôle et l'échange de données via le réseau. Le protocole *Open Sound Control* (OSC) est probablement le plus répandu actuellement dans les logiciels de musique et pour les arts numériques, et figure comme un des successeurs du MIDI. Il permet non seulement d'envoyer des chiffres, mais aussi du texte et d'autres types de données.

Avec OSC, il est facile de créer son propre protocole spécialisé et personnalisé pour chaque projet. Cela fait, les différentes applications dialoguent entre elles selon des conventions OSC. Par exemple, plusieurs applications supportent le TUIO (http://www.tuio.org) spécifique aux interfaces tangibles multi-points.

DÉFINITION

OSC est un format de messages qui permet d'envoyer des informations entre des synthétiseurs, des applications et des appareils spécialisés. Pour être transmis, ces messages peuvent utiliser les protocoles réseaux TCP ou UDP. L'UDP est le plus utilisé grâce à sa rapidité et sa fluidité. Toutefois, il est possible que certains messages se perdent en cours de route, contrairement au TCP, où les messages sont surveillés afin d'arriver à bon port.



Quelques références anglophones :

- Références : http://opensoundcontrol.org/
- Spécifications : http://cnmat.berkeley.edu/OpenSoundControl/OSC-spec.html

ANATOMIE D'UN MESSAGE OSC

Un message OSC est composé d'un chemin et d'arguments. Le chemin est un texte qui contient des mots séparés par des barres obliques suivis par une liste d'arguments de différents types. Parmi ces types, on trouve nombres naturels, nombres réels, texte et couleur. Les adresses sont composées de mots séparés par des barres obliques et créent des hiérarchies de nœuds comme dans les branches d'un arbre ou encore le système de fichiers d'un ordinateur.

Exemple de message OSC : /message/mon_message ,ifs 4 98.456 bla

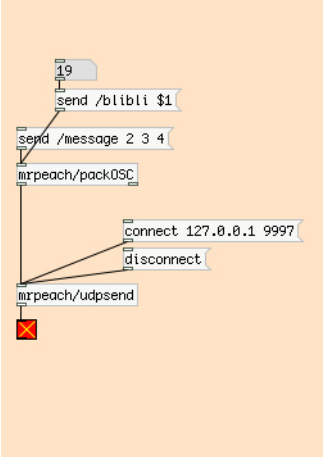
On peut noter la présence d'une série de lettres précédées d'une virgule. Ces lettres se trouvent entre le chemin et les arguments. Elles indiquent le type des arguments du message comme ",ifs". Les types des arguments sont : ***int*** (nombre entier), ***float*** (nombre réel) et ***string*** (texte).

PURE DATA ET OSC

Plusieurs extensions pour Pure Data offrent des outils permettant de communiquer via OSC. La librairie "mrpeach" réalisée par Martin Peach, représentée ci-dessous, est une des références.

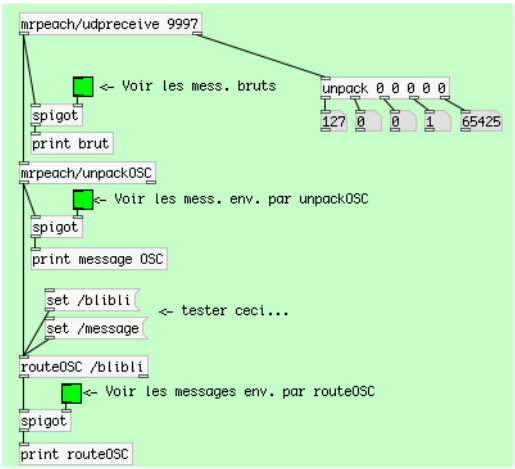
Voir les patches d'exemples dans le menu *Help > Browser > 5.reference/mrpeach*

Ci-dessous, une première partie d'un patch d'exemple (téléchargez ici) :



Dans cet exemple, nous demandons à Pure Data d'établir une connexion locale ou *localhost,* c'est-à-dire « avec l'ordinateur lui-même ». L'adresse Internet 127.0.0.1 représente toujours l'ordinateur sur lequel on travaille. À partir du moment où la connexion est établie, la petite croix dans l'interrupteur sera cochée. Nous utilisons ici le protocole réseau UDP, qui est le choix par défaut. Le chiffre qui suit correspond au numéro du port. Dès lors, nous pouvons envoyer les données par l'intermédiaire du message [send /blibli $1< ou [send message 2 3 4<.

Pour recevoir ces messages, il nous faut un second patch :



Ici, on reçoit les messages OSC envoyés par le premier patch. Ils vont s'afficher dans la console de Pure Data, si on ouvre les robinets [spigot] placés avant les objets [print].

Remarque :

Il est possible que certaines librairies ne se chargent pas correctement dans Pd-extended. Dans ce cas, l'objet apparaîtra comme inexistant et sera représenté en pointillés rouges. Dans la plupart des cas, on peut régler ce problème en ajoutant simplement le nom de la librairie appartenant à la librairie. Dans notre cas représenté plus haut, la mention "mrpeach/" avant le nom des objets spécifie son emplacement dans la librairie.

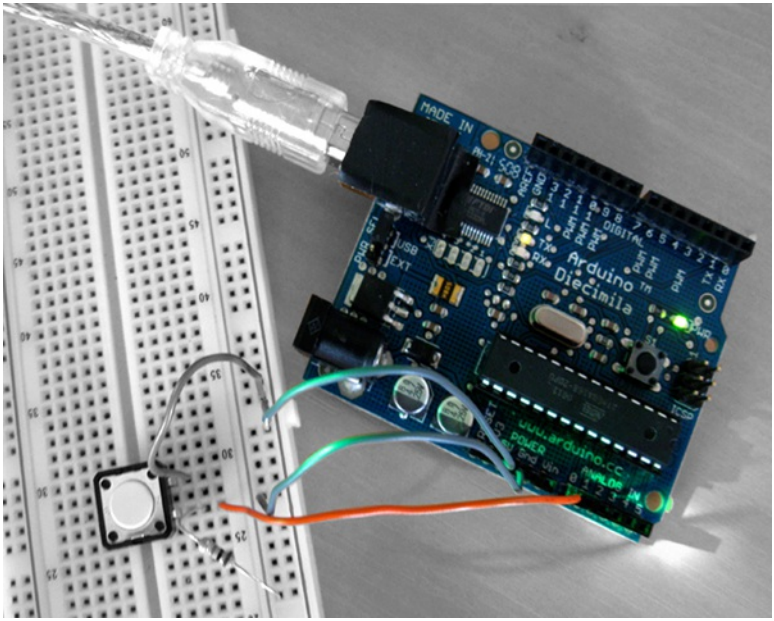
41. ARDUINO ET PURE DATA

Comment introduire le monde réel dans nos applications ? Comment interagir avec nos créations numériques dans le monde concret qui nous entoure ? Une solution relativement facile à mettre en œuvre consiste à utiliser la plateforme de développement matériel **Arduino** et de l'interfacer avec **Pduino** ou **Msg**.

ARDUINO

L'Arduino est une carte électronique en matériel libre destinée notamment à la création artistique interactive. Elle permet de servir d'interface entre le « monde réel » et l'ordinateur, en y branchant par exemple des capteurs ou des actionneurs (moteurs, lumières, etc.), en la reliant en USB à un ordinateur et en la pilotant à l'aide d'un logiciel de gestion d'événements multimédias interactifs comme Pure Data. Elle peut aussi être utilisée sans ordinateur de façon autonome (possibilité que nous n'étudierons pas ici).

Pour davantage d'information sur Arduino, nous vous invitons à consulter les ressources francophones et anglophones disponibles sur Internet. **Le site officiel du projet Arduino** est accessible à l'adresse http://www.arduino.cc



La carte Arduino est connectée à l'ordinateur par un port USB. L'actionneur est un bouton (*button) sur une plaque d'essai. Quand on appuie sur le bouton, la LED intégrée dans la carte s'allume.*

L'Arduino est une carte basée sur un micro-contrôleur (mini-ordinateur) Atmel ATMEGA8, ATMEGA168 ou ATMEGA328. Elle dispose dans sa version la plus basique de 1 Ko de mémoire vive et 8 Ko de mémoire flash pour stocker ses programmes. La carte basique dispose de 13 entrées ou sorties numériques et de 6 entrées analogiques sur lesquelles on peut brancher capteurs et actionneurs. Il existe de nombreuses autres versions et des déclinaisons de l'Arduino. La carte dispose d'un logiciel système interne (modifiable) et des programmes utilisateurs.

***Pour relier Pure Data et une carte Arduino**, nous allons présenter ici une méthode simple, mais qui offre néanmoins de nombreuses possibilités. Cette méthode est basée sur l'utilisation d'une librairie qui implémente un protocole de communication entre la carte et l'ordinateur-hôte. En termes plus simples, ces librairies permettent de faire dialoguer facilement la carte et l'ordinateur en permettant de modifier directement depuis l'ordinateur la configuration de la carte sans avoir à modifier le code que l'on y a chargé au départ. Deux librairies existent, l'une appelée **Firmata** et l'autre **Msg**. Firmata est un projet de Hans-Christoph Steiner et outre la librairie qui doit être installée sur l'Arduino, c'est avec un patch Pure Data dédié que l'on va pouvoir décider si les connecteurs de la carte se comportent comme des entrées ou sorties. Dans Msg, c'est une application séparée qui donne une représentation graphique de la carte Arduino où la configuration des broches se fait. Msg, de Thomas Ouellet Fredericks , permet d'utiliser cette même interface avec d'autres environnements de programmation car il utilise le protocole OSC.*

FIRMATA

Étape 1 : installer Firmata sur une carte Arduino

Installer la plateforme de développement Arduino :

- *Dans les dépôts de Debian et Ubuntu : installez le paquet « arduino » via le gestionnaire de paquets, ou tapez la ligne de commande "sudo apt-get install arduino" dans le terminal.*
- *Sinon, téléchargez l'application sur http://www.arduino.cc (voir configuration et notes d'installation sur ce site).*

- *Depuis l'application Arduino, sélectionnez la bonne plateforme correspondant au modèle de votre carte Arduino :* ***Tools > Boards >... (modèle de la carte)***
- *Sélectionnez le port USB qui communiquera avec la carte Arduino (normalement ttyUSB0) : Tools > Port > ttyUSB0*
- *Chargez le Firmata en ouvrant l'exemple proposé par l'interface Arduino :* ***File > Exemple > Firmata > StandardFirmata***
- *Vérifiez (compilez) le programme StandardFirmata en cliquant sur l'icône "play"*
- *Chargez le programme StandardFirmata dans la carte Arduino en cliquant sur l'icône "upload"*

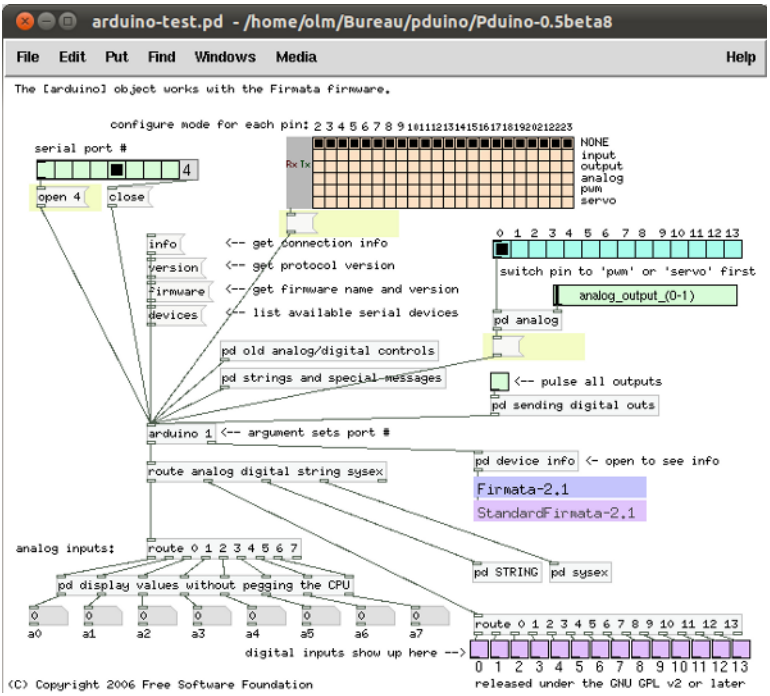
Normalement, tout s'est déroulé sans accroc, et vous pouvez quitter l'application Arduino IDE.

Étape 2 : connectez Pure Data à Arduino

- *Chargez dans Pure Data le patch "arduino-test.pd"*
- *Vous trouverez ce patch sur le lien : http://at.or.at/hans/pd/objects.html en téléchargeant "Pduino-0.5beta8.zip"*
- *Cliquez sur le message "device" pour voir dans la console quel port est utilisé par la carte Arduino :*

```
[comport]: available serial ports:
    0    /dev/ttyS0
    1    /dev/ttyS1
    4    /dev/ttyUSB0    <-- arduino
```

Choisissez le port correspondant dans les bouton radio "serial port#" (ici 5e bouton, le premier = 0)



USAGE

Le principe est simple : l'objet [arduino] accepte des messages qui constituent des commandes pour l'Arduino, permettant ainsi de configurer les ports à la volée.

Le patch arduino-test permet d'explorer les fonctionnalités qu'offre l'objet [arduino]. Voici quelques pistes :

Entrées Analogiques

Pour capturer les informations provenant de senseurs branchés sur un port analogique, il faut envoyer le message [pinMode analog n< (avec n le numéro du port) à l'objet [arduino]. Le patch ci-dessus permet de le faire simplement en choisissant le mode dans les boutons radio. Attention cependant à choisir les bons, le nombre de ports et leurs fonctionnalités changent d'un modèle à l'autre.

Entrées Numériques

Les entrées numériques se configurent de la même façon, en utilisant [pinMode digital n< pour les ouvrir.

MSG

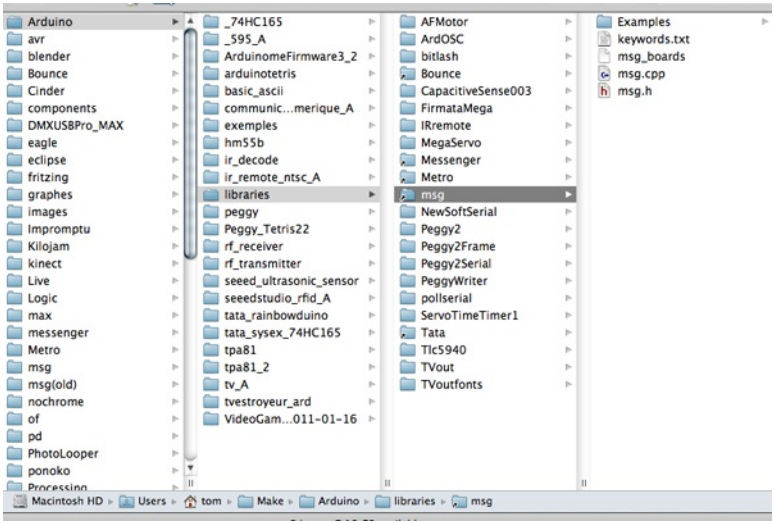
INSTALLATION

Installer l'application Msg :

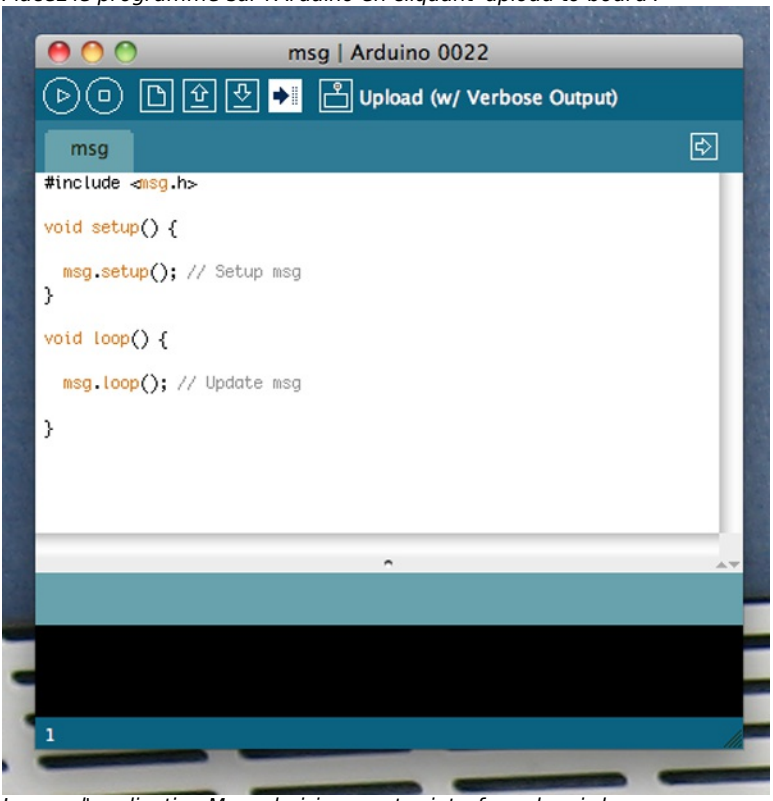
- *Téléchargez l'archive : http://wiki.t-o-f.info/index.php?n=Msg.Msg*
- *Dépaquetez l'archive*
- *Pour OSX, déplacez Msg dans le dossier 'Applications'*

Installer la plateforme de développement Arduino :

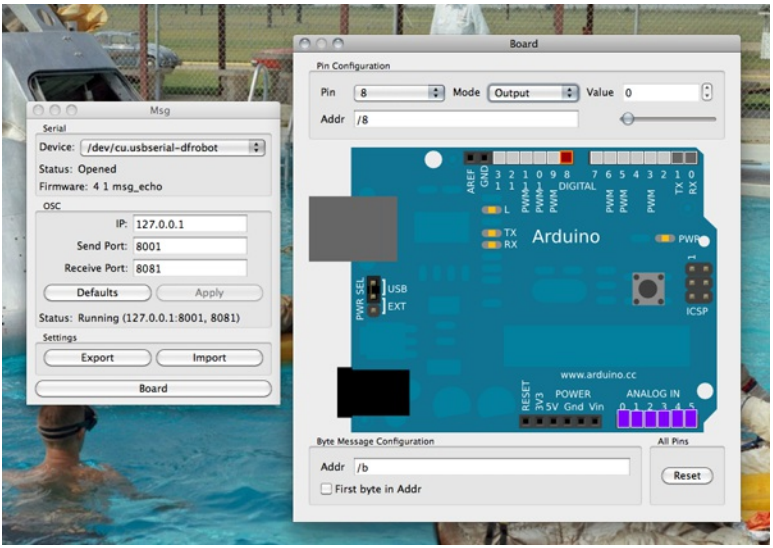
- *Pour Linux, dans les dépôts de Debian et Ubuntu : installez le paquet « arduino » via le gestionnaire de paquets, ou tapez la ligne de commande "sudo apt-get install arduino" dans le terminal.*
- *Pour OSX et Windows, téléchargez l'application sur http://www.arduino.cc (voir configuration et notes d'installation sur ce site).*
- *Placez la librairie 'Msg' sous le dossier 'librairies' de Arduino.*



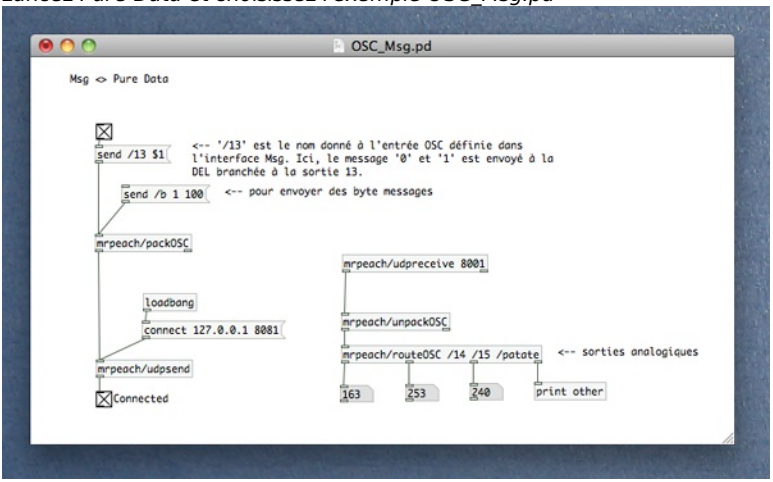
- *Depuis l'application Arduino, sélectionnez la plateforme correspondant au modèle de votre carte Arduino : Tools > Boards >... (modèle de la carte)*
- *Sélectionnez le port USB qui communiquera avec la carte Arduino (normalement ttyUSB0) : Tools > Port > ttyUSB0*
- *Chargez Msg en ouvrant l'exemple proposé par l'interface Arduino : File > Exemple > msg > msg.pde*
- *Placez le programme sur l'Arduino en cliquant 'upload to board'.*



- *Lancez l'application Msg, choisissez votre interface depuis la console de menu et utilisez l'interface graphique pour choisir l'utilisation de vos entrées et sorties.*



- *Lancez Pure Data et choisissez l'exemple OSC_Msg.pd*



CONCLUSION

42. DOCUMENTATION
43. GLOSSAIRE

42. DOCUMENTATION

Il existe de nombreux documents d'aide et ressources sur Pure Data, certains accessibles directement depuis le logiciel (liens internes) et d'autres accessibles en ligne (liens externes).

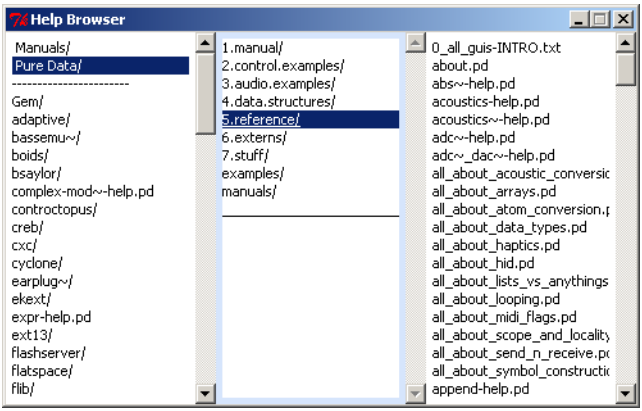
LIENS INTERNES

Les différentes ressources concernant Pure Data sont généralement anglophones. Elles sont disponibles dans le menu *Help* ou encore dans l'arborescence des fichiers.

Pour avoir les documents officiels en **langage HTML**, il suffit d'aller dans la barre des menus du *Patch*, puis sur *Help* enfin sur HTML...

On peut aussi avoir accès à la liste des **objets natifs**. Il suffit dans ce cas de faire un clic droit dans la zone blanche du patch, puis sur *Help*. On y trouvera ce qui suit : http://www.umatic.nl/workshop/objects.txt

Il existe aussi des **listes d'objets** dans la librairie du menu *Help*. Pour y accéder, il suffit de rentrer dans le menu *Browser*, puis de cliquer sur *Pure Data/*, et enfin sur *5.reference/*



Dans le menu *Help* puis dans *Browser*, nous trouvons les notions de base de Pure Data dans *2.control.examples/*. Des patches complémentaires se trouvent dans *Pure Data > 5.reference > all_about_**

Pour une documentation **audio**, toujours dans *Browser,* il faut cliquer dans *3.audio.examples/* ou bien dans *Manuals > 1.Sound*. La documentation sur **l'image, la 3D et la vidéo** se trouve dans *Manuals > 2.Image ou Gem > examples* ou *gridflow > doc > tutorials* . Enfin, la documentation **Réseau** est dans *Manuals > 3.Networking*

On peut aussi communiquer avec une **interface physique**, toujours dans le menu *Help > Browser*, ensuite cliquer sur *Manuals*, enfin sur *4.Physical.*

LIENS EXTERNES

En plus du présent manuel consultable en ligne (http://fr.flossmanuals.net), il existe plusieurs ressources sur la Toile. En voici une liste non exhaustive :

- Forum francophone : http://codelab.fr/pure-data
- Site officiel de Miller Puckette : http://www-crca.ucsd.edu/~msp/
- Site de la communauté (actualités, téléchargements, listes de diffusion, dépôts de patches, annonces...) : http://puredata.info/
- Forum anglophone : http://puredata.hurleur.com/
- Manuel anglophone : http://en.flossmanuals.net/PureData
- Conventions Pure Data : http://puredata.info/community/projects/convention04/
- "Designing Sound" d'Andy Farnell : http://aspress.co.uk/sd/

- Tutoriel Jérôme Abel : http://impala.utopia.free.fr/pd/
- Tutoriel Raphaël Isdant : http://raphael.isdant.free.fr/pure_data/
- Tutoriel Ben Bogart, traduit par Olivier Heinry : http://puredata.info/Members/oli4444/
- Tutoriel Johannes Kreidler : http://www.pd-tutorial.com/
- Tutoriel Labomedia : http://wiki.labomedia.org/index.php/Liens_ressources_tutoriaux_Pure_Data
- Tutoriel Adriano Castaldini : http://adrjork.altervista.org/puredatatutorials.html
- Patches de Cyrille Henry : http://www.chnry.net/ch/?011-Software
- Interface- Z : http://www.interface-z.com/ + Didacticiels : http://www.interface-z.com/patchs/index.htm#did
- CRAS : http://www.mainsdoeuvres.org/rubrique90.html

43. GLOSSAIRE

(Les termes en **caractères gras** font référence aux définitions du glossaire).

Abstraction
: Une abstraction est une boîte **objet**, par exemple : [ici_mon_abstraction]. L'abstraction enferme un bloc de code écrit par nous, avec Pure Data, relativement indépendant du reste du **patch**. L'abstraction est utile pour simplifier la lecture d'un patch, sa conception, ainsi que l'écriture, l'utilisation et la réutilisation d'une fonction particulière (la tâche à exécuter par l'abstraction). En réglant les paramètres des propriétés à ***Graph on Parent***, il est possible d'ouvrir une abstraction en clickant dessus pour que les **éléments d'interface graphique GUI** s'affichent même lorsque l'abstraction est fermée. Les **entrées** et **sorties** peuvent être utilisées pour envoyer [send] et recevoir [receive] de l'information de/à une abstraction, ainsi que les paires [send] et [receive].

ADC
: *Analog to Digital Converter* - est l'entrée de la carte son. L'**objet** est symbolisé par [adc~].

ADSR
: **Attack**, **Decay**,**Sustain** et **Release** (Temps d'attaque, de déclin, de maintien et de relâchement) – les points de changement commun (ou point de rupture) dans l'**enveloppe** d'une **note**.

ALSA
: Advanced Linux Sound Architecture – réglage par défaut du **pilote audio** pour le système d'exploitation Linux.

Anything
: Mot-clef dans certains objets qui correspondent à un atome ou des séries, parfois abrégé par [a] ou [any].

Arduino
: C'est une carte électronique en matériel libre pour la création artistique interactive, un langage et un environnement de développement. Arduino permet de faire l'interface entre le monde physique et le monde virtuel. Reliée à l'ordinateur, il devient alors possible d'établir une communication entre des logiciels (Pure Data, par exemple) et divers éléments matériels tels que lumières, moteurs à courant continu ou servo-moteurs et beaucoup d'autres choses intéressantes. Arduino peut bien sûr être utilisé seul, pour fabriquer des objets interactifs indépendants.

Argument
: Un élément d'information envoyé à un **objet** qui définit un paramètre de cet objet. Les arguments peuvent être envoyés sous forme de **message**, ou décris de la création dans la boîte de l'objet. Les arguments sont également utilisés pour remplacer les **variables** (souvent représentés par une succession de signes dollars) dans les messages et les objets. En utilisant l'objet [paquet], plusieurs arguments peuvent être envoyés dans un message.

Array

Il agit dans un tableau de valeurs X / Y, ce qui signifie que vous pouvez lui demander une information par l'envoi d'une valeur représentant un emplacement sur l'axe des X (horizontal), et il rendra la valeur de retour sur la position de l'axe Y (vertical). Les tableaux sont souvent utilisés pour charger les fichiers sons qui sont affichés à l'écran sous forme de **graphique**.

ASIO

Audio Stream Input/Output – est un **pilote audio** développé par la compagnie de logiciels audio Steinberg et disponible pour plusieurs types de cartes son compatibles avec le système d'exploitation Windows.

Atome

Un mot-clé correspondant à l'élément le plus fondamental des données. Peut-être un nombre ou une chaine de caractère (sans espace).

Attaque (*Attack*)

L'attaque est le commencement d'une **note** déclenchée soit en appuyant sur une touche, un clavier ou par une **séquenceur**. Une attaque lente signifie que le son émit prend plus de temps avant d'atteindre son plein volume qu'une attaque plus rapide. Voir la définition d'**enveloppe**.

Bang

Un objet-bouton qui émet une impulsion lorsqu'on clique dessus. On le crée à partir du menu « put » ou avec le **message** [bang<. Il existe un **objet** [bang] qui peut aussi être abrégé en [b] et renvoie une impulsion lorsqu'il reçoit un message quelconque. Elle est généralement liée à une **entrée chaude** d'un objet ou à d'autres messages.

Canvas (Canevas)

Le canevas (*canvas*) est une zone de pixels dans le **patch** qui est utilisée pour ajouter de la couleur ou modifier la configuration graphique pour le patch. Généralement utilisé pour se souvenir et mettre au clair les éléments ajoutés dans le patch. Il doit être placé avant de placer les **objets** par-dessus. Ensuite, il suffit de couper puis coller les objets un par un par-dessus.

Comment (Commentaire)

Élément de texte dans un **patch** qui explique et commente les éléments du patch. Elle sert aussi à donner des informations additionnelles au programmeur ou toute autre personne qui ouvrira le patch plus tard. Les commentaires (*comment*) n'ont pas d'incidence sur la fonction du patch.

Création d'argument

Information additionnelle donnée lorsque qu'un **objet** est crée. Par exemple, l'objet osc~ 440 crée une **oscillation** (nom de l'objet) avec une fréquence de départ de 440 Hz (étant la création d'argument). Voir aussi **argument**.

Crénelage (*Aliasing*)

Chaque fois qu'un son est rejoué ou synthétisé, et dont la fréquence est au dessus du **nombre de Nyquist** (la moitié du taux d'échantillonnage actuel), une deuxième fréquence sera entendue, incrémenté vers le bas le nombre Nyquist à l'équivalence en Herz. Par exemple, si le taux d'échantillonnage est 44 100 Hz, le nombre de Nyquist serait 22050. Si on a tenté de jouer un son à 23050 Hz, une tonalité supplémentaire de 21 050 Hz (qui est la différence entre les deux fréquences soustrait

du nombre de Nyquist) sera entendue.

DAC
: *Digital to Analog Converter* - est la sortie de la carte de son vers Pd. L'**objet** est symbolisé par [dac~].

DataFlow
: Flux de données.

Debugging
: Déverminage ou débogage. Activité de régler les problèmes de code.

Decay (Déclin)
: La quantité de temps que prend un son pour passer du volume maximal vers le niveau de ***sustain*** (dans le cas d'une **enveloppe**), ou à aucun son du tout (dans le cas d'un **délai**).

Delay (Délai)
: La quantité de temps entre un événement et un autre. Dans un effet audio, le signal sonore entrant prend un temps de **latence** d'une certaine durée. En cas de mélange avec le son original, un écho se fait entendre. En renvoyant le signal retardé dans le délai (en général après avoir abaissé son **gain**), on obtient des échos multiples avec une décroissance. Les **objets** Pure Data pour créer un retard sont nommés [delwrite ~] et [delread ~], qui de pair doivent être attribué la même **création d'argument** afin de leur permettre de communiquer (par exemple, [delwrite~ rastaman] et [delread~ rastaman]). Tout comme un paramètre de Pd, l'objet [delay] change la latence du programme pour permettre un temps de réponse plus rapide mais cela au détriment de **glitches** sonores et vice et versa.

Distorsion
: Une distorsion se produit lorsqu'un signal audio est changé de volume au niveau des échantillons et produit des fréquences non présentes dans l'original (par *clipping*).

Dollar ($)
: Le **symbole** $ accompagné d'un chiffre est utilisé pour représenter une **variable** soit dans un **message** soit dans un **argument**. Il sert à réserver une place dans les arguments d'un **objet** ou dans un message pour une valeur qui sera déterminée plus tard.

Échantillonneur (*Sampleur*)
: Un *sample* désigne un échantillon audionumérique. C'est un bloc de chiffre binaire dont la taille dépend de la résolution qui est donnée en bits. Ces échantillons sont assemblés (encapsulés) en fichiers audio, formatés en .aiff, .ogg, .wav, etc. Un échantillonneur (*sampleur*) enregistre ou lit ces fichiers.

Élément d'interface graphique (GUI)
: Graphical User Interface - les parties visibles du **patch** Pd qui sont utilisées pour le contrôle par la souris ou pour afficher des informations, telles que des curseurs, des boutons radio, les franges, des boutons, boîtes de numéro, VU-mètres, toiles, graphiques, tableaux, **symboles**, etc.

Entrée (**Inlet**)
: Entrée pouvant être chaude ou froide d'un **objet**, d'un **GUI**, d'un **message**, d'une **abstraction** ou d'un **sous patch** recevant de l'information d'une **sortie** (*outlet*).

Entrée chaude et froide
: Dans les composants de Pure Data, le point le plus à gauche d'un **objet** est appelé "entrée chaude", cela permet de réceptionner

l'information dans l'objet et de déclencher son fonctionnement. Toutes les autres entrées de l'objet sont considérées comme froides, dans ce cas l'information est stockée dans l'objet jusqu'à recevoir à nouveau une nouvelle information/impulsion dans l'entrée chaude.

Enveloppe

Terme utilisé pour décrire les changements d'un son au cours du temps. Traditionnellement, cela est utilisé pour synthétiser différent son instrumental avec ***Attack***, ***Decay***, ***Sustain*** et ***Release*** (ou voir **ADSR**) qui sera déclenché au tout début d'une note. Un violon, par exemple, aura une attaque lente au moment de la vibration des cordes, tandis qu'un piano aura une attaque plus rapide (ou « percutante ») qui distinguera son timbre d'un autre instrument.

Extended

Version étendue de Pure Data.

Extensions ou Externes (*Externals*)

Les extensions sont soit des groupes d'abstractions, soit des fonctions codées en C/C++ qui étendent les possibilités de Pure Data. **Objets** externes par opposition aux objets natifs qui font partie de Pd-Vanilla. C'est un objet de Pure Data qui n'était pas inscrit dans le programme de base par l'auteur Miller Puckette. Les extensions sont créées et maintenues par la communauté de développement Pure Data, et compte pour beaucoup de fonctions supplémentaires de Pd, y compris la capacité de manipuler la vidéo et la 3D ainsi que des flux MP3 et beaucoup d'autres. Les extensions sont habituellement chargées par une **librairie externe** au début d'une session de Pd en les incluant dans le menu. Mais certains peuvent être chargées comme des objets uniques en précisant l'endroit où l'externe est sauvegardé sur le système d'exploitation et répertorié dans le paramétrage du chemin de Pure Data.

Flags

Étiquettes ou options que l'on peut ajouter au démarrage.

Float

Nombre à virgule flottante, avec décimale (ex: 1.342) qui peut être positif ou négatif et qui se situe entre -8388608 et 8388608. Une notation spéciale est utilisée pour les très petits ou très grands nombres à virgule flottante, puisque Pd utilise seulement jusqu'à 6 caractères pour représenter un nombre à virgule flottante. Par conséquent, "1e +006» est un nombre à virgule flottante qui représente "1000000" (ou 1 avec 6 décimales après lui), tandis que "1e-006" représente "0,0000001" (ou 1 avec 6 décimales devant lui).

Gain

Exprime en décibel la force d'un signal audio. L'ampleur du gain est logarithmique, car il exprime le rapport physique du pouvoir entre un son et un autre. Le gain est mesuré en systèmes audio numériques comme la quantité de décibels en dessous de 0 dB, ce qui est le point d'écrêtage (-10 dB, -24 dB, etc.)

GEM

Graphical Environment for Multimedia – qui signifie en français environnement graphique pour le multimédia. Il est écrit pour supporter la production d'images et de compositions audiovisuelles en temps réel, dont la manipulation d'objet de synthèse, d'images et de vidéos.

Glitch
: Une erreur sonore survient lorsque l'ordinateur n'a pas assez de temps pour traiter l'audio entrant ou sortant d'une application audio avant de l'envoyer à la carte son. Ce résultat est d'avoir un trop faible temps de **latence**, de sorte que les tampons de la carte son ne se remplissent aussi vite pour que la carte son est le temps de les jouer, ce qui entraîne une perte temporaire, mais audible du son. Les glitches peuvent se produire lorsque d'autres processus interrompent ce processeur avec des tâches différentes (telles que l'actualisation de l'affichage sur l'écran, la lecture ou l'écriture d'un disque dur, etc).

Glissière (Ascenseur / Gradateur linéaire)
: Composant d'interface graphique du menu Pure Data. La glissière sert à contrôler le volume d'un son, ou encore l'intensité d'un effet. L'élément expulse par sa **sortie** une suite de valeurs situées entre sa valeur minimale et sa valeur maximale déterminée par la position du curseur ou sert aussi à afficher les nombres reçus par son **entrée**. La glissière peut être vertical (Vslider) ou horizontal (Hslider) et il est possible de modifier, via les propriétés, l'étendu **MIDI** qui est de 0 à 127 par défaut.

Graph
: Un *graph* est un conteneur graphique qui peut contenir plusieurs tableaux. Un tableau fait appel à un graphique. A chaque création d'un tableau à partir du menu, son emplacement soit dans un graphique nouvellement créé ou dans un graphique déjà existant.

Graph on Parent
: Une propriété des **éléments d'interface graphique** des **sous patchs** et des **abstractions** visibles dans le **patch** principal. Cela facilite l'utilisation des patchs plus compliqués.

Integer
: Nombre entier sans décimale pouvant être positif ou négatif. Voir **float**.

Latence
: La quantité de temps nécessaire pour traiter tous les **échantillons** provenant des applications audio sur votre ordinateur et l'envoyer à la carte son pour la lecture, ou pour recueillir des échantillons de la carte son pour l'enregistrement ou de traitement. Une latence plus courte signifie que vous allez entendre les résultats plus rapidement, donnant l'impression d'un système plus sensible, ce que les musiciens ont tendance à apprécier lors de la lecture. Cependant, avec une latence plus courte vous courez plus de risque de **glitches** audio. C'est parce que l'ordinateur pourrait ne pas avoir assez de temps pour traiter le son avant de l'envoyer à la carte son. Un plus long temps de latence signifie moins de glitches mais au prix d'un temps de réponse plus lent. La latence est mesurée en millisecondes.

Librairie
: Ensemble d'objets, de classe, de fonctions, indépendantes du programme central.

Librairie externe
: Une collection d'externes écrite pour Pd. Pris d'une **librairie**, les externes peuvent être chargés au démarrage d'une session Pd en les incluant dans les paramètres de démarrage.

Liste

Un type de message contenant une collection de données. Plus précisément, une liste est une série de 2 ou plus **atomes** dont le premier atome est le sélecteur de la liste. Ou encore, une série de 2 ou plusieurs atomes dont le premier atome est numérique.

MIDI

Musical Instrument Digital Interface - est un protocole de communication et de commande standardisé permettant l'échange de données entre des instruments de musiques électroniques et des ordinateurs.

Message

Le message est un élément d'information envoyé aux **objets** d'un **patch**, en utilisant souvent l'élément message GUI. Les messages disent aux objets quelle fonction effectuer et comment, ils peuvent être numérique, inclure un texte qui décrit la fonction à modifier ou même contenir d'autre information telle que l'emplacement de fichier sur l'ordinateur.

Mode action

Le mode action sert à tester et faire fonctionner le **patch**. Voir **mode édition**.

Mode édition (Mode utilisateur)

En mode édition on peut modifier, créer, déplacer des **objets**, des **messages**, des **commentaires**, des **éléments d'interface graphique**, etc. Du mode édition il est possible de passer au mode action via le menu edit (ou ctrl + e sur Mac) et vise versa.

Nombre Nyquist (*Nyquist Frequency*)

Un nombre qui représente la moitié du taux d'**échantillonnage** de l'application qui est utilisée, et représente la fréquence la plus élevée possible qui peut être lu sans **crénelage**. Le nombre de Nyquist est exprimé en Herz. Exemple: si le taux d'échantillonnage est 44 100 Hz, le nombre de Nyquist serait 22050. Si on a tenté de jouer un son à 23 050 Hz, un son crénelage supplémentaires à 21 050 Hz (la différence entre les deux fréquences soustrait du nombre de Nyquist) serait entendue.

Objet

Fichier écrit en C ou C++, compilé, pour être utilisé ensuite dans Pure Data. C'est aussi le bloc de construction les plus élémentaires d'un **patch**. Dans le vocabulaire de Pure Data, les objets portent un nom qui détermine leur fonction. Pour voir le fichier d'aide de documentation de tout objet, faites un clic droit avec la souris, ou utilisez la touche « Ctrl » (ou « pomme »).

OpenGL

Open Graphics Library - est une librairie largement utilisée, standard de fonctions graphiques 2D et 3D.

OSC

Open Sound Control - est un format de **messages** permettant d'envoyer des informations entre différentes applications et différents systèmes d'exploitation et matériel à travers un réseau informatique IP.

Oscillateur

Un générateur audio qui produit une constante forme d'onde répéter. Un oscillateur de cosinus [osc ~] produit une onde sinusoïdale pure sans harmoniques, tandis que d'un oscillateur en dent de scie ou d'une rampe [phaseur ~] produit un son plus riche en harmoniques nombreuses. Autres formes d'un signal

d'onde carrée comprennent celle de l'impulsion ou de triangle. Chaque forme d'onde est définie par une fonction mathématique et chaque forme a son propre spectre harmonique.

Patch
: Espace de travail où se lient les objets ensemble. Il est le document par lequel il est possible de construire les structures à l'intérieur de Pd. Un patch peut contenir plusieurs **objets**, **commentaires**, **éléments d'interface graphique**, **messages**, **sous patchs** et **abstractions**. Si un autre patch est sauvegardé dans le même répertoire de travail ou dans tout autre répertoire listé dans les paramètres du patch, alors il est possible de l'utiliser dans le patch principal ou patch parent comme une abstraction.

Patchage dynamique (*Dynamic patching*)
: Créer des **objets** dans un **patch** et modifier leur apparence de façon dynamique à l'aide d'un programme conçu à cet effet, directement avec Pure Data ou avec d'autres langages (Python, lua, etc.)

Patching
: Travailler sur un patch.

Path
: Est un paramètre de Pd qui détermine deux choses. Le premier est les répertoires sur votre ordinateur que Pd recherche pour charger les **externes**, et le second est le répertoire où Pd recherche pour trouver des **abstractions** utilisées dans les patchs. **Path** peut être réglée avec des drapeaux de démarrage, ou en entrant les répertoires dans les paramètres de démarrage en utilisant la fenêtre principale de Pd.

PDP
: **Pure Data Packe** - est une extension de Pd permettant de traiter des paquets de données dédiés au traitement de la vidéo.

PiDiP
: Extension de PDP ajoutant des fonctionnalités d'effet, d'**entrée** et **sortie** en divers formats.

Pilote Audio (**Audio driver**)
: Fournis un système d'**entrée** et de **sortie** entre la carte son et son application. Plus efficace sera le pilote audio, plus courte sera la **latence** du système audio. Exemples de pilote audio : **MME** et **ASIO** pour Windows, CoreAudio pour Mac OS X, **OSS**, **ALSA** et **JACK** pour Linux.

Prototypage
: Pure Data permet de faire du prototypage, qui consiste à réaliser un prototype pour tester rapidement ses idées.

Release (Relâchement)
: La quantité de temps nécessaire pour que le **gain** d'une puisse atteindre zéro après la touche du clavier a été relâchée. Voir aussi l'**enveloppe**.

Ring Modulation
: L'utilisation d'un signal audio pour moduler en amplitude un autre signal audio.

Sampleur
: Voir **échantillonneur**.

Send and Receive (envoyer et recevoir)
: Une méthode de communication entre les **objets** dans un **patch** sans les câbles de raccordement. Les objets [send] et [receive] sont utilisées avec une **création d'argument** partagée qui

définit le «canal» qu'ils transmettent, par exemple sur [send volume] et [receive volume]. Les noms des objets peuvent être abrégés en [s] et [r] et une paire pour les signaux audio existe également ([send ~] et [receive ~] ou [s ~] et [r ~]).

Séquenceur MIDI
: Un appareil capable de mémoriser puis de rejouer des séquences d'instructions contrôlant des instruments de musique électronique suivant la norme **MIDI**.

Sortie (*Outlet*)
: Sortie d'un **objet**, d'un **GUI**, d'un **message**, d'une **abstraction** ou d'un **sous patch** qui envoie de l'information vers l'**entrée** (**inlet**).

Sous patch
: Boîte, s'assimilant à un **objet**, qui contient un patch qui s'ouvre dans une autre fenêtre, permettant d'alléger le patch principal visuellement. Tout comme pour les abstractions, il est possible de les ouvrir en cliquant dessus et les **éléments d'interface graphique** s'affichent. Les **entrées** et **sorties** peuvent être utilisées pour envoyer [send] et recevoir [receive] de l'information de/à un sous patch, ainsi que les paires [send] et [receive].

Spigot
: L'**objet** [spigot] joue le rôle d'une porte logique qui, soit laisse passer (1) ou bloque (0) un flux de donnée.

Stream
: Flux en continu, par opposition aux événements asynchrones.

Sustain (Maintien)
: Le niveau de **gain** que détient une note après l'**attack** le **decay**. La note tient ce niveau de gain jusqu'à ce que la touche soit relâchée. Voir aussi l'**enveloppe**.

Symbole
: Une chaîne de caractères sans espace, qui n'est pas interprétée comme un nombre. [makefilename] ou d'autres **externes** permettent de gérer de manière plus complexe les chaines de caractères plus complexes (pour par exemple écrire des noms de fichier). Des mots uniques, imprimables et sans espaces sont des symboles communs, mais il est possible de construire des symboles non imprimables, des symboles avec des espaces blancs échappés ou des symboles qui ressemblent à un certain nombre mais qui ne comprennent que des caractères numériques avec des objets tels que [makefilename] ou quelques autres externes. Ces symboles ne seront pas sauvegardés correctement dans un fichier .pd et ils ne peuvent pas être créées en modifiant manuellement une boîte de message. En interne un symbole est défini comme un atome de type "t_symbol" dans Pd.

Synthèse AM
: **Amplitude Modulation Synthesis** ou **AM Synthesis** – est un type de synthèse sonore, où le **gain** d'un signal est modulé, par exemple, par le gain d'un autre signal. Le signal dont le gain est modulé est appelé le « transporteur », et le signal de responsables de la modulation est appelé le « modulateur ». En modulation d'amplitude classique, ou synthèse AM, autant le modulateur que le transporteur sont des oscillateurs, mais le support peut aussi être un autre type de signal, comme un instrument ou une entrée vocale. La modulation d'amplitude via

un modulateur de fréquence très basse est appelée **Tremolo** et l'utilisation d'un signal audio à moduler en amplitude un signal audio est appelée « **Ring Modulation** ».

Variable
: Un type d'espace réservé, souvent dans un **message** et écrit comme un **signe de dollar** qui est destiné à être remplacé par d'autres informations. Par exemple, dans le message [open $1 $2 $3<, il y a trois variables qui doivent être remplacés par des informations réelles.

Working Directory (Répertoire de travail)
: Dans Pd c'est le répertoire où le **patch** avec lequel vous travaillez a été enregistré. Toutes **abstractions** utilisées dans ce patch doivent être soit enregistrés dans ce répertoire, ou le répertoire dans lequel ces abstractions ont été sauvées doit être ajoutés au path dans les préférences de démarrage.