

Deal

Cette documentation est faite pour des humains, pas des techniciens. Si vous n'en comprenez pas une partie, la documentation est en cause pas vous. Par contre, si vous n'en dites rien, ce ne sera plus la documentation qui sera en cause, mais vous.

Donc, si vous lisez cette documentation, vous acceptez implicitement le deal suivant. Si une partie de ce document n'est pas claire ou vous semble erronée, vous devez écrire un courrier électronique à maa@lagraine.com en signalant l'objet de votre trouble et son emplacement. Les rédactions alternatives sont les bienvenues.

Merci pour toutes corrections, suggestions...

Des images viendront illustrer ce document, mais je voudrais qu'elles enrichissent le contenu texte et ne s'y substituent pas.

Mâa (22 Mars 2003).

Convention

param est l'abréviation de paramètre et sera abondamment utilisé dans ce document ainsi que params.

En **Gras** les noms ou portion de nom des params dans l'interface.

Si ... est accolé directement à un nom (**toto...**) cela signifie tout les noms commençant par toto.

Une expression du type **Pref_ui / Draw / draw_curve** tout en Gras (Y compris les /), signifie dans l'objet **Pref_ui**, on trouve un param **Draw** qui permet d'accéder au param **draw_curve**. Ces expressions seront plus ou moins longue suivant la profondeur du param concerné.

t signifie le temps

dt l'intervalle de temps entre deux rendus

* la multiplication (de l'influence du C sur les développeurs)

ALT désigne la touche Alt (en général, gauche de la barre d'espace).

CTRL désigne la touche Ctrl (en général, en bas à gauche du clavier ou deux touche à droite de la barre d'espace).

SHIFT désigne la touche Majuscule.

L-cliquer signifie cliquer avec le bouton de gauche (Left cliquer)

M-cliquer signifie cliquer avec le bouton du milieu (Middle cliquer)

R-cliquer signifie cliquer avec le bouton de droite (Right cliquer)

ALT-x-cliquer signifie cliquer avec le bouton x (Right, Middle ou Left) avec la touche ALT maintenue enfoncée.

« ex : » ou « Ex : » est l'abréviation de « par exemple : ».

« pour l'instant » doit être compris comme cela peut changer dans de futures versions de AAASeed.

Params

Un param dans AAASeed correspond à une ligne dans l'interface de AAASeed (éventuellement plusieurs dans le cas de param texte à plusieurs lignes).

Double cliquer sur un param ouvre l'éditeur correspondant.

Pour les curieux, il suffit de ALT-M-cliquer pour avoir des détails sur un param.

Les trax (voir ci-dessous) se connectent aux params. Des règles de conversions s'appliquent éventuellement d'où les précisions ci-dessous.

Les param peuvent être des types suivants:

- **REAL**

Un réel c'est à dire un nombre à virgule. Ex : 3,1416. C'est le type par défaut des nombres dans AAASeed, c'est vers ce type que sont convertis tout les nombres avant d'être traité par les trax.

- **INT32**

Un entier c'est-à-dire un nombre sans virgule.

En entrée un **REAL** est arrondi au **INT32** juste inférieur ou égal c'est à dire
1,5 donne 1
1,0 donne 1
-0,5 donne -1.

- **BOOL**

un booléen c'est-à-dire une valeur à 2 états possibles :

ON (1, dit vrai)

ou **OFF** (0, dit faux).

en sortie d'une trax une valeur supérieure à 0,5 donne **ON**.

- **SYMBOLIC**

un **SYMBOLIC** est en fait un **INT32** et se comporte de la même manière, mais à chaque valeur (de 0 à n) est associé une valeur symbolique (un texte) qui s'affiche à la place du chiffre.

- **UINT32_SYMBO_NEG**

ce type de param est en fait un **INT32** lorsque il est positif et prend des valeurs symboliques comme un **SYMBOLIC** lorsque il est négatif (de -1 à -n).

- **UINT32_SYMBO_ZERO**

ce type de param est en fait un **INT32** lorsque il est strictement positif et prend des valeurs symboliques comme un **SYMBOLIC** lorsque il est nul ou négatif (de 0 à -n). C'est la même chose que **UINT32_SYMBO_NEG** mais la valeur 0 est elle aussi symbolique.

- **STR**

il s'agit de texte ou de ce que l'on appelle en jargon une chaîne de caractère.

En entrée (plug in) d'une trax attendant une valeur numérique, ce param sera considéré comme 0. En sortie (plug out) le param **format** dans le groupe **Out** d'une trax détermine la conversion (format identique au printf() de la bibliothèque C)

- **FILENAME**

Ce type de param se comporte comme le type **STR**, mais il représente un nom de fichier. Lorsque l'on double clique dessus le dialogue de choix de fichier correspondant s'ouvre. Pour l'instant le branchement de la sortie d'une trax (Plug Out) sur ce type de param est déconseillé.

- **TIME_CODE**

C'est en fait un **REAL** mais qui représente un nombre de seconde. Mais il est représenté sous la forme mm:ss:ii où mm signifie minute, ss les seconde, et ii le numéro d'image (25 par seconde). Il s'agit d'une version non finalisé du time code vidéo.

- **BIT32**

Un **INT32** représenté sous sa forme binaire.
Saurez vous trouver un tel param.

- **GROUP**

- **GROUP_CLOSED**

Les params de ces types servent de tête de chapitre. Ils regroupent conceptuellement et visuellement des params. M-Cliquer dessus ouvre/ferme le groupe, de même qu'un appui sur la barre d'espace. Il n'y a pas de valeur associé et donc on ne peut pas y brancher de traxs.

- **NONE**

Ces params servent à accrocher des objets.
Il n'y a pas de valeur associé et donc on ne peut pas y brancher de traxs.

Que ce soit dans les fichiers ou l'interface la présence de majuscule dans le nom d'un param est un souci esthétique, mais n'est pas considéré pour distinguer les noms de param. Par exemple OuT ou out représente un même nom de param.

Lors de l'affichage des params sous forme de lignes successives, AAASeed ne réaffiche pas le début du nom s'il est identique à la ligne précédente. Cela allège et rend l'interface plus lisible, les vides donnant une représentation visuelle de point commun entre params. Il suffit de remonter aux lignes précédentes pour obtenir le début manquant du nom du param.

Lorsque le nom du param finit par **_trig** cela signifie qu'il s'agit d'un trigger (un bouton poussoir est un exemple de trigger). Le passage à **ON** (1) va déclencher une action (le démarrage d'un son par exemple), dès que l'état ON du param a été utilisé par l'objet correspondant, le param retombe de lui-même dans son état OFF. Ce fonctionnement permet de lancer facilement des actions à la souris dans l'interface.

Trax généralités

- Une trax (déformation de track, piste en anglais) doit être vue comme un block avec des entrées (parent) et des sorties (child) sur lesquelles on branche des params. La métaphore se résonne toujours par rapport à la trax :
 - Plug In : brancher un param en entrée de la trax,
 - Plug Out : brancher un param en sortie de la trax.

Todo Globales/locales

- Une trax n'a pas de notion d'ordre de ses connections autre que l'ordre où elle sont effectuées. Lors de la lecture d'un fichier l'ordre de création des connections n'est pas garanti. Le fonctionnement des traxs ne peut alors reposer sur la notion d'ordre des entrées. il n'y a pas donc pas de trax de soustraction/différence ou de division, il faut deux traxs pour effectuer ces opérations.

- Une trax peut être

1D (une dimension)

la trax écrit dans le param en sortie (Plug Out)

3D (trois dimension)

la trax écrit dans le param en sortie (Plug Out) et les deux emplacements mémoires suivants, donc prudence. Brancher en sortie (plug out) sur autre chose que **red/green/blue**, **x/y/z** ou **u/v/axe** est dangereux / indéterminé.

Certains params (**Min/Max/Offset** par exemple) s'appliquent de la même manière aux trois dimensions (pour l'instant).

On appelle donc trax **3D**, une trax qui une sortie 3D. Une trax **1D** peut avoir des entrées 3D. Ces entrées 3D ont les mêmes contraintes que les sorties 3D (voir juste ci-dessus), mais un mauvais branchement en entrées n'est pas dangereux juste indéterminé.

- Le traitement d'une trax se fait par étages, chaque étage comportant un certain nombre d'étapes, les params sont regroupés en partie dans cet ordre et cette logique.

What

Détermine le type de la trax et le traitement à effectuer.

Ce param n'apparaît en fait jamais car le type (voir **Fn** dans **What**) de la trax s'affiche à sa place pour rendre immédiatement le type lisible.

How

Définit le traitement exact à effectuer

si possible (en fonction du type de trax) le résultat est calibré entre 0. et 1.

Out

Transforme ce résultat avant de déposer cette information à l'endroit ad hoc.

Out Value

Matérialise les sorties et reroute les sorties des traxs 3D

Net Midi

Commande l'envoi et la réception des traxs au travers du réseau et/ou Midi

Draw

Définit comment si et comment est dessiné le signal de sortie d'une trax. Ce dessin est à nouveau visible (v0.69 Beta 57).

- Dans une trax circulent des valeurs numériques ou du texte. Dans ce dernier cas la plupart des traitements et des params ne s'appliquent pas et n'ont aucun sens.

- Dans chaque layers (Groupe de layer) se trouve un param **Traxs** (BOOL) qui regroupe les traxs du layers et commande globalement leur activation.
- Dans **Pref_ui/Traxs** ou trouvera des params qui agissent globalement sur les traxs.
- Les traxs sont mises à jour par ordre croissant.

CONFIDENTIAL

Trax param par Param

Active (BOOL)

dans l'état **OFF** aucun traitement n'est effectué.

Comment (STR)

Ce param est là pour que nous, utilisateurs, puissions annoter une trax comme bon nous semble. N'hésitez pas, c'est pour l'instant le moyen principal de retrouver « vos câbles » dans le plat de nouilles.

Groupe What

Ces params déterminent ce qui se passe à haut niveau dans une trax.

Fn (SYMBOLIC défaut **SINUS**)

détermine le type de la trax, un chapitre ci-dessous détaille le traitement pour chaque type.

Output (SYMBOLIC défaut **FN**)

altère le rôle de la trax sans en changer son type.

RECORDER_FN

A chaque boucle de rendu la valeur de la sortie de l'étage **What** est mémorisée. Deux types de trax **PLAYER** et **PLAYER_RAW** font le pendant en relecture.

Ne fonctionne que partiellement pour l'instant.

Todo **PLAYER** et **PLAYER_RAW** doivent devenir des outputs et pas des fns.

En fait ces fonctionnalités doivent occuper un param isolé.

FN_THRESHOLD

Après le traitement du **Gain/Bias** et du **Round**, avant le traitement du **Limit**, la valeur est comparée au **Threshold** (seuil en français).

Deux cas : la valeur est

- Inférieure au **Threshold** et devient 0 à cet étape.

- supérieure ou égale au **Threshold** et devient 1 à cet étape.

C'est un moyen simple d'effectuer un test, ou de transformer une trax analogique en **BOOL** en choisissant explicitement le seuil (au lieu de 0,5 par défaut en cas de conversion automatique en sortie).

FN_TRIGGER_UP_AND_DOWN

Il s'agit comme le nom l'indique d'un trigger (impulsion en français).

La valeur est comparée au **Threshold** comme ci-dessus, si le résultat est identique à la précédente exécution la valeur devient 1, sinon la valeur devient 0.

Autrement dit, lorsque la valeur passe au dessus (**UP**) ou en dessous (**DOWN**) du **Threshold** une impulsion est générée.

FN_TRIGGER_DOWN

Idem mais l'impulsion ne se produit que lors de franchissement descendant du seuil.

FN_TRIGGER_UP

Idem mais l'impulsion ne se produit que lors de franchissement montant du seuil.

FN

C'est le fonctionnement normal d'une trax tel que définit par le param **Fn**.

MIN

MIDDLE

MAX

La sortie de la trax devient constante. Le traitement est « court circuité ».

C'est-à-dire que le traitement est remplacé respectivement par 0, 0,5 et 1.

Toutes les étapes ne sont pas pour autant inactives. La valeur de la sortie est encoredéterminée par les params **Gain**, **Bias**, **Round**, **Min**, **Max**, **Offset**, **Limit** et **Filter** (dans cet ordre)..

C'est une aide permettant d'arrêter une trax pour régler sa sortie

La valeur du **Bias** intervenant dans la détermination de **MIDDLE**, cela permet de régler le point milieu.

IN

La traitement est ici aussi « court circuitée », comme dans le cas des constantes, s'il y a une entrée elle est prise comme constante, sinon la constante est 0. Il faut rappeler qu'une trax n'a pas de notion d'ordre garanti de ses entrées (voir généralités) et la « première » est prise comme constante.

Channel (INT32 de 1 à 128)

Control_id (INT32 de 1 à 1024)

AAASeed est souvent calé sur la métaphore MIDI: un contrôleur est désigné par son **Channel** et son **Control_id**. Ce type de fonctionnement est utilisé pour différents types de trax. Par exemple le **Channel** va désigner une bdd mocap et le **Control_id** un noeud (node) de cette bdd.

Si un type de trax supporte un nombre moins grand de **Channel** ou de **Control_id** ces valeurs seront limitées au minimum et au maximum supporté par le type de trax. Par exemple MIDI supporte 16 channels donc dans le cas MIDI toute valeur de **Channel** supérieure à 16 sera utilisée par la trax comme 16.

Name

Des types de trax (**NAME_REAL...** pour l'instant) utilisent un mécanisme différent des **Channel** et **Control_id** pour sélectionner des valeurs. Le param **Name** sert alors d'identificateur

Il n'y pour l'instant pas de limitation sur les caractères utilisables, mais il est recommandé de se servir uniquement des lettres (caractères accentués compris pour l'instant), des chiffres, de l'underscore « _ » et de l'espace.

Attention la case, c'est à dire minuscule/majuscule est prise en compte.

Dans le menu des params on trouve un item « Print all REAL_NAME » qui liste dans la console tout les noms définis, depuis le lancement de AAASeed.

Channel_bis (INT32 de 1 à 128)

Control_id_bis (INT32 de 1 à 1024)

Ce jeu supplémentaire est utile pour certains types.

Groupe How

Ces params précisent le traitement de la trax en complément de ceux du group **What**. Le rôle de cet étage est la description du traitement effectué en fonction du type de la trax (**Fn**), puis de premiers ajustements. L'ordre des params correspond à l'ordre des étapes dans le traitement de la trax.

Restart_trig (BOOL)

Si une trax a un état initial, l'activation de ce trigger remet la trax dans cet état. En particulier s'il s'agit d'une trax où le temps intervient (hors **Filter**), ce trigger remet la phase interne à 0.

La touche t qui remet le temps global à zéro provoque un restart de toutes les trax actives.

Ease_before Freq Phase Ease_after

Freq et **Phase** représentent comme leur nom l'indique la fréquence et la phase. Cela n'a de signification que pour certains types et cet usage est détaillé plus bas.

Certains types ont besoin d'information spécifiques autre que les **Channel** et **Control_id**. **Freq** puis **Phase** sont alors détournés de leur sens traditionnel. Autrement dit, un type de trax qui a besoin de valeurs numériques supplémentaires utilisera ces deux params à cet effet.

Ease_before et **Ease_after** seront utilisées aussi si une troisième et une quatrième valeur sont utiles.

En particulier **Freq** et **Phase** servent souvent à définir un intervalle, dans ce cas **Ease_...** peuvent servir à définir une zone de transition avant (**Ease_before**) et après (**Ease_after**) l'intervalle, d'où l'ordre de ces 4 params.

Ces usages spécifiques sont documentés dans la partie consacré aux différents types de trax.

Ease_before (REAL)

Dans le cas particulier où **Freq** et **Phase** servent à définir un intervalle, **Ease_before** sert, si nécessaire, à définir une zone de transition avant l'intervalle.

Ease_before peut servir de troisième valeur numérique pour certains types.

Freq (REAL)

Si ce param représente vraiment une fréquence, il interagit toujours avec la **Phase**. On ne peut comprendre le param **Freq** sans lire la documentation du param **Phase** (voir ci-dessous le param suivant).

Freq peut représenter une fréquence temporelle

1 signifie 1Hz, c'est-à-dire que une période de 1 seconde.

10 signifie 10Hz, c'est-à-dire que une période de 0,1 seconde.

Freq peut représenter une fréquence spaciale

1 signifie un facteur d'échelle de 1 unité d'espace.

10 signifie un facteur d'échelle de 0,1 unité d'espace.

Sinon sa signification est documentée avec le type de trax.

Eventuellement le rôle de **Freq** et **Phase** est explicité dans le nom (ex: **INSIDE_FREQ_PHASE**).

Phase (REAL)

La phase p dans une fonction périodique est en général :

$$p(t) = \text{Freq} * t + p(0)$$

le produit du temps par la fréquence + une constante.

Dans ce cas un changement, même infime, de fréquence provoque un saut, une discontinuité. C'est pourquoi AAASeed calcule ainsi :

$$p(t) = p(t-dt) + \text{Freq} * dt$$

la phase au temps précédent auquel s'ajoute le produit du temps écoulé par la fréquence.

Ainsi la continuité est possible, par contre la valeur de la phase $p(t)$ en fonction du temps est une valeur relative, qui dépend de l'historique de la trax. Cela implique que deux trax dont les params et les entrées sont identiques peuvent avoir une valeur différente en sortie (elles sont déphasées). Le param **Restart_trig** (ci-dessus) ou la touche t peut servir à « rephaser » deux trax.

En fait la phase utilisée par les trax est

$$p(t) + \text{Phase avec } p(t) = p(t-dt) + \text{Freq} * dt \text{ et } p(0) = 0.$$

On parlera de phase **incrémentale ou relative**.

Ainsi on peut travailler directement sur la phase de la trax, avec une **Freq** nulle on contrôle la **Phase**.

On peut aussi contrôler le déphasage entre plusieurs trax avec la **Phase**.

Si une trax a une d'entrée E connectée qui contrôle la phase (Ceci dépend du type). Le calcul de la phase est différent, elle ne varie plus avec le temps, mais avec l'entrée : $p(t) = \text{Freq} * E$.

Donc si une trax a une d'entrée E connectée, la phase utilisée par la trax est

$$\text{Freq} * E + \text{Phase}$$

On parlera de phase **absolue**.

Dans ce cas une variation du param **Freq** entraînera des discontinuités.

Le param **Restart_trig** (ci-dessus) ou la touche t n'ont dans ce cas aucun effet direct sur la phase et la trax.

En branchant en entrée un param reproduisant le temps on obtient une phase :

$$\text{Freq} * t + \text{Phase} \text{ et ainsi une phase absolue dépendant du temps.}$$

Etonnant non !

La phase est accessible voir **phase_internal** plus loin dans le Groupe **Out Value**.

Ease_after (REAL)

Dans le cas particulier où **Freq** et **Phase** servent à définir un intervalle, **Ease_after** sert, si nécessaire, à définir une zone de transition après l'intervalle.

Ease_after peut servir de quatrième valeur numérique pour certains types.

Gain (REAL entre 0, et 1, défaut à 0,5)

On peut comparer le **Gain** à un réglage de contraste.

La sortie de l'étage précédent (déterminé par le type **Fn** et les params qui y sont associés) est transformée par une courbe de transfert dont la forme est conditionnée par la valeur du **Gain**. Si l'étape précédente n'a pas de sortie « canonique » (entre 0, et 1,), la situation est problématique. L'effet du **Gain** est indéterminé hors de cet intervalle..

Pour un **Gain** de 0,5 il ne se passe rien, la courbe est une droite.
Plus le **Gain** monte vers 1, plus les extrêmes sont favorisés et les valeurs autour de 0,5 repoussées vers 0, ou 1,. Un signal sinusoïdal entre 0, et 1, deviendra un signal carré pour un **Gain** de 1,. En terme d'image on passe d'une image à niveaux de gris vers une image noir et blanc.
Plus la valeur du **Gain** descend vers 0, plus des valeurs éloignées de 0,5 convergent vers 0,5. Un signal sinusoïdal entre 0, et 1, tendra vers un signal plat à 0,5 avec des pics vers 0, et 1, de plus en plus fins et de moins en moins marqués. En terme d'image on passe d'une image à niveaux de gris vers une image grise avec les zones les plus noires et blanches qui disparaîtraient en dernier.
En fait la courbe de **Gain** est composée de deux courbes symétrique de **Bias** avant et après 0,5 (voir **Bias**).

Si le **Gain** n'est pas utilisé, il est important de le laisser exactement à 0,5.
L'étape est sauté, AAASeed gagne en vitesse, on évite de propager une légère dérivation.

Bias (REAL entre 0, et 1, défaut à 0,5)

On peut comparer le **Bias** à un réglage de luminosité.
La sortie de l'étape précédente (déterminé par le type **Fn** et les params qui y sont associés) est transformée par une courbe de transfert dont la forme est donnée par la valeur du **Bias**. Si l'étape précédente n'a pas de sortie « canonique » (entre 0, et 1,) la situation est problématique. L'effet du **Bias** est indéterminé hors de cet intervalle.

Pour un **Bias** de 0,5 il ne se passe rien, la courbe est une droite.
Plus le **Bias** monte vers 1, plus les valeurs se déplacent vers le haut (1,). Un signal sinusoïdal entre 0, et 1, deviendra un signal aplati à 1, avec un pic de plus en plus fin vers 0,. En terme d'image on passe vers une image plus clair, le noir restant noir.
Plus la valeur du **Bias** descend vers 0, plus les valeurs se déplacent vers le bas (0,). Un signal sinusoïdal entre 0, et 1, deviendra un signal aplati à 0, avec un pic de plus en plus fin vers 1,. En terme d'image on passe vers une image plus sombre, le blanc restant blanc.
La valeur du **Bias** donne précisément la valeur de sortie lorsque l'entrée du **Bias** est à 0,5. Par exemple, un **Bias** à 0,75 donnera à une valeur 0,5 en entrée du **Bias** une valeur de 0,75 à sa sortie.

Si le **Bias** n'est pas utilisé, il est important de le laisser exactement à 0,5.
L'étape est sauté, AAASeed gagne en vitesse, on évite de propager une légère dérivation.

Round (REAL)

Arrondi en français.
En sortie du **Bias** la valeur est arrondie (en dessous) à la résolution donné par le **Round**. Par exemple une entrée à 0,35 avec un **Round** de 0,1 donnera une sortie à 0,3.
Si le **Round** est a 0, la valeur est intouché.

Threshold (REAL)

Seuil en Français.

Voir dans Groupe **HOW** les **OUTPUT** de type **FN_THRESHOLD**,
FN_TRIGGER_UP_AND_DOWN, **FN_TRIGGER_DOWN** et
FN_TRIGGER_UP.

CONFIDENTIAL

Groupe Out

Une fois l'étage **What** traversé, la valeur résultante est à nouveau transformée par cet étage dont l'utilité est principalement de donner l'amplitude (la dynamique) de la sortie. Cet étage traite ce qui sort de l'étage précédent et n'est pas lié au type de la trax (**Fn**).

Ici encore, l'ordre des params correspond à l'ordre de leur prise en compte dans le traitement de la trax.

Min (REAL)

Max (REAL)

Offset (REAL)

Une valeur E en entrée de cette étape donne en sortie une valeur S suivant la formule:

$$S = \text{Min} + E * (\text{Max} - \text{Min}) + \text{Offset}.$$

Au passage notons :

$$E = 0 \text{ donne } S = \text{Min} + \text{Offset}$$

$$E = 1 \text{ donne } S = \text{Max} + \text{Offset}$$

Pour une entrée calibrée entre 0, et 1, le rôle de ces params est donc évident.

Brancher sur l'**Offset** d'une trax la sortie d'une autre est un bon moyen d'effectuer une addition sans utiliser une trax supplémentaire pour cela.

Limit (SYMBOLIC)

Limit_min (REAL)

Limit_max (REAL)

En fonction de la valeur de **Limit** la sortie de l'étape **Min/Max/Offset** est à nouveau transformé.

No

Il ne se passe rien. Cet étage est désactivé.

CLAMP

Ecrêter en français. Clamper en français.

Si l'entrée E est inférieure ou égale à **Limit_min**, la sortie S sera **Limit_min**.

Puis si E est supérieur à **Limit_max**, S sera **Limit_max**.

Sinon S sera identique à E.

Autrement dit la sortie S est identique à l'entrée E tant que E se situe dans l'intervalle **Limit_min/Limit_max**, mais reste limité à ces valeurs pour les valeurs E extérieures à l'intervalle.

Pour les spécialistes du C pour l'instant:

$$S = (E \leq \text{min}) ? \text{min} : (E \geq \text{max}) ? \text{max} : a;$$

WRAP

Boucler/Replier en français. Warper en québécois.

Pour les technos il s'agit d'un modulo de (**Limit_max - Limit_min**).

Imaginez l'entrée E croissant à partir de la valeur **Limit_min**, la sortie S de cette étape est identique à E. Dès que E passe la valeur **Limit_max**, S repasse directement à la valeur **Limit_min**, puis continue ensuite à croître de manière identique à E. Et ce jusqu'à ce que S franchisse encore la valeur **Limit_max**, où le même phénomène se reproduit.

Il en est de même lorsque E diminue, le saut de bouclage se reproduisant à chaque franchissement de S d'une des valeurs limites.

Autrement dit : alors que CLAMP écrête un signal, WARP le réinjecte de l'autre côté de l'intervalle, blouclant/warpant ainsi l'espace.

Encore autrement dit: cela se passe comme si E correspondait à un point qui se déplace sur un fil. Et que ce fil est cousu en un point régulier dont la taille est donnée par (**Limit_max- Limit_min**). la sortie S correspond à la distance de E à un bord de la couture.

ABSOLUTE

ABSOLUTE_THEN_CLAMP

ABSOLUTE_THEN_WRAP

Il s'agit de trois cas symétriques au trois précédents, mais l'entrée E, devient d'abord la valeur absolu de E (noté ABS(E)), avant de passer ensuite par **NO**, **CLAMP** ou **WARP**.

La valeur absolue ABS(V) d'une valeur V est la distance au point zéro.

Autrement dit :

V est positif \Rightarrow $ABS(V) = V$

V est négatif \Rightarrow $ABS(V) = -V$.

En terme mathématique :

$ABS(V) = |V|$

ABSOLUTE

$S = ABS(E)$

ABSOLUTE_THEN_CLAMP

$S = CLAMP(ABS(E))$

ABSOLUTE_THEN_WRAP

$S = WRAP(ABS(E))$

Filter_type (SYMBOLIC)

Absent pour l'instant.

Le filtre par défaut est un filtre incrémental de type $(1+f)/(1-f)$.

Filter (REAL)

Définit l'influence d'un filtre temporel du type **Filter_type**.

0, il ne se passe rien

1, le signal est tellement filtré (lissé) qu'il ne change pas.

Cet étage permet de filtrer, d'atténuer voir de supprimer des variations brusques au cours du temps. Cela permet d'éliminer du bruit, mais cela donne du retard, de l'inertie aux variations du signal.

Le filtre c'est la Steady-Cam du signal.

Les usages principaux sont :

d'éliminer du bruit.

de lisser, d'adoucir des variations

ex : l'entrée son qui est très sensible.

d'éviter des sauts dû à une résolution trop faible d'un capteur.

ex : un slider MIDI n'a que 128 valeurs possibles, branché sur un

param de position d'une caméra, on déplace la camera. Il n'y a que 128 positions possibles, le point de vue saute alors de position en position.

L'utilisation du filtre rend ces déplacements continus au lieu de directs.

Le mouvement est amorti. Les 128 positions d'arrêt sont toujours là, mais la douceur des transitions le masque.

On atteint vite des valeurs de **Filter** au dessus de 0,7 voir proche de 1,. L'échelle n'est pas linéaire donc plus on est proche de 1, plus il faut contrôler finement la valeur de **Filter**. Ne pas oublier les touches CTRL et ALT qui modifient la sensibilité de la souris.

Format (STR)

Ce param détermine la conversion d'un nombre en texte.

Ce format est celui du printf() de la bibliothèque C. Trouvez un développeur et pompez ces neurones (Beurk). Une version simplifiée est exposée ci-dessous.

Ce param est de la forme « Texte avant %0x.yf texte après »

% est obligatoire ainsi que son pendant f. %f est le minimum.

0 est optionnel et force l'affichage de zéros devant
x est optionnel et impose le nombre de caractère pour le nombre
y est optionnel et impose le nombre de chiffres après la virgule
%% est nécessaire pour afficher le symbole %

Pour une sortie de 0,3

«%f»	donne	0,300000
«%0.2f»	donne	0,30
«%05.2f»	donne	00,30
«encore %.2f %% de plus»	donne	encore 0,30 % de plus

Lorsque ce param est vide %.2f est utilisé par défaut.

AAASeed est pour l'instant par défaut en Français (locale french) pour ces informations de format, le point décimal est donc le, caractère « , » (virgule).

Groupe Out Value

phase_internal (REAL)

La phase utilisée dans le traitement de certaines traxs est ici « exportée ». Ceci peut permettre de mieux comprendre ce qui se passe ou bien de synchroniser (phaser) deux traxs.

Phase_internal prend en compte **Freq** et **Phase**.

out_0 (REAL)

out_1 (REAL)

out_2 (REAL)

Ces params permettent de voir les sorties d'une trax.

C'est aussi le seul moyen de pouvoir brancher les sorties d'une trax en entrée d'une trax, sans avoir à connecter d'abord la sortie à un param intermédiaire.

out_selector_0 (SYMBOLIC)

out_selector_1 (SYMBOLIC)

out_selector_2 (SYMBOLIC)

Ces params ne sont utilisés que pour les traxs 3D.

Ils permettent de raffiner la sortie 3D : ne pas utiliser toutes les dimensions, les permuter, les inverser.

Par défaut ils sont respectivement à **X Y** et **Z**.

Les valeurs possibles (ligne suivante) se passent de commentaires.

No X Y Z -X -Y -Z

Groupe Net Midi

Ces params déterminent si une trax reçoit ou envoie de l'information au travers de diverses connections.

Ceci n'est actif pour l'instant que pour les trax à une dimension (**1D**). Pour l'instant les trax **3D** n'envoient ni ne reçoivent.

Net implique une transmission réseau dans un protocole AAASeed.

Max implique un envoi vers le logiciel son Max sur Mac, ce format est aussi valable pour le logiciel pd (Pure Data).

Midi implique un envoi sur des contrôleurs Midi. Mais est désactivé pour l'instant et absent des params de la trax.

La valeur envoyé est pour l'instant la sortie avant l'étape **Filter**. Ainsi on peut filtrer différemment à la réception.

Dans l'objet Net (menu *Focus*) on peut voir si une application AAASeed est **Master** (Maître) ou **Slave** (Master à OFF, esclave). Ceci se change en éditant directement le fichier .net dans le dossier Pref de premier niveau.

net_out (SYMBOLIC)

Il s'agit ici d'une transmission dans un format interne à AAASeed.

Les valeurs envoyées le seront sur la **Net/Net Connection 0**.

La réception se fait avec des trax de type **NET_IN** (voir Fn **INPUT** plus bas).

L'envoi n'est pas fait, pour l'instant, pour les trax 3D.

No

Il ne se passe rien ni envoi, ni réception.

Send

La sortie avant l'étape **Filter** est envoyée.

Receive

Non actif pour l'instant, cette absence peut être considéré comme un bug, et sera donc corrigée très bientôt.

Master send Slave receive

Une même application peut être, en fonction de la machine et du fichier .net, **Master** ou **Slave**. Cette valeur du param bascule une même trax en émission (**Master**) ou en réception (**Slave**). Cette bascule de fonctionnement se fait au moment de la lecture des fichiers. Sur une application esclave, des params de la trax sont changées pour la transformer en simple trax de réception :

Fn	devient NET_IN
Offset	devient 0
Min	devient 0
Max	devient 1
output_type	devient OUTPUT_FN
channel_id	devient net_channel_id
control_id	devient net_control_id

Une application slave ne sauve pas ses fichiers, donc cette modification de la trax n'est pas sauvée.

Avec l'enrichissement des trax depuis deux ans, ceci pose maintenant des problèmes de symétrie et doit être corrigé avant Fin Mai 2003.

Ceci permet d'utiliser les mêmes fichiers d'environnement sur la machine Master et les machines Slave. La gestion d'un seul jeu de fichier dans pour une application multi écran fait partie du cahier des charges AAASeed.

Only Master send

Comme dans la cas **Send** pour une application **Master**, .

Only Slave receive

Cette fois-ci la bascule en réception dans la cas **Slave** se fait, mais dans le cas **Master** il n'y a pas d'envoi.

net_channel_id (INT32)

net_control_id (INT32)

Ces params déterminent comme pour un contrôleur Midi les numéros de **channel** et de **control_id** associés à la transmission de la valeur de la trax. En réception ces deux indices seront l'accès à la valeur transmise.

max_out (SYMBOLIC)

Il s'agit ici de déclencher des envois dans un format simple vers le logiciel son Max sur Mac, ce format est aussi valable pour le logiciel pd (Pure Data). maa@lagraine.com pour en savoir plus.

Les valeurs envoyées le seront sur la **Net/Net Connection** correspondant au param **Net/midi_destination**.

No

Il ne se passe rien pas d'envoi.

Send

La sortie avant l'étape **Filter** est envoyée.

max_channel_id (INT32)

max_control_id (INT32)

Ces params déterminent comme pour un contrôleur Midi les numéros de **channel** et de **control_id** associés à la transmission de la valeur de la trax.

midi_out (BOOL)

Désactivé et absent pour l'instant.

Groupe Draw

Le dessin des trax a été réactivé à partir de la version 0.69 Beta 57.

Pour qu'une trax soit dessinée il faut que :

- le viewport CURVE soit présent. Pour cela passer en plein écran avec la touche W, puis avec ALT-W changer la zone (Viewport) de rendu pour qu'elle ne soit pas plein écran. Le viewport CURVE se situe entre le zone de rendu et le bas de l'écran. Ce Viewport CURVE doit de plus être activé (touche V ou **Pref_ui/Draw/draw_curve**). Lorsque qu'il est activé, 8 lignes horizontales grises sont présentes.

draw (SYMBOLIC)

No

Le signal de la trax n'est pas dessiné.

FILTERED

Le signal de la trax après l'étape **Filter** est dessiné.

On a donc le signal filtré (FILTERED).

RAW

Le signal de la trax avant l'étape **Filter** est dessiné.

On a donc le signal brut (**RAW**).

Il est dessiné avec plus de transparence que de la couleur choisi.

ALL

Le signal de la trax avant et après l'étape **Filter** est dessiné.

Le signal **RAW** est plus transparent que le signal FILTERED ce qui les distingue.

draw_red (REAL)

draw_green (REAL)

draw_blue (REAL)

draw_alpha (REAL)

draw_line_size (REAL)

Faut-il un dessin ?.

Fns : tout les Types

Pour changer le param **Fn** le sous menu « **Set Trax Type** » est présent, il présente des sous menus regroupant les types par catégories. La description des types qui suit traite les types catégorie par catégorie. On peut changer le param type de manière traditionnelle, mais il faut savoir ce que l'on fait, ou désactiver la trax au préalable. En effet, le passage par certains type dangereux (ex : **COPY_BLOCK_REAL**) ou des types 3D peu provoquer des crashes.

Les premières trax **3D** (Sortie **3D**) commence avec la catégorie **3D Mocap**. La catégorie **3D Distance 1D** regroupe des traxs qui ont des entrées **3D** mais les sorties sont **1D**. Ce ne sont donc pas des traxs 3D et leur noms à été mal choisi, mais la compatibilité ascendante a un prix.

Les traxs qui utilisent une entrée et une seule voit leur noms dans la documentation suivi de (In). La mention sera (Ins) pour plusieurs et absente si l'entrée n'est pas utilisée.

Les noms de type suivi de (Freq/Phase), dans la documentation, sont des types qui utilisent les params **Freq** et **Phase** de manière directe (voir ci-dessus ses params en particulier **Phase**).

Si une entrée connecté est utilisé pour calculer la phase interne (cas phase absolu) le nom de type sera suivi de (In/Freq/Phase), dans ce cas la mention (In) seule ne sera pas là.

Si l'étage lié au type **Fn** à une sortie entre 0,(inclus) et 1,(inclus) on parlera de type canonique.

Le nom, dans la documentation, est alors suivi de [0,1].

Dans ce cas les **Gain/Bias** s'appliquent toujours sans problèmes.

Il existe un cas encore plus particulier, lorsque le résultat est **ON/OFF** c'est-à-dire 0, ou 1,.

Le nom, dans la documentation, est alors suivi de 0/1.

Si les **Gain/Bias** s'appliquent dans ce cas sans problèmes, ils ne servent à rien est il conseillé de les laisser à 0,5 pour éviter des effets de bords en cas d'approximations qui pourrait propager des erreurs de calculs.

Player

PLAYER_RAW

Non documenté

PLAYER

Non documenté

CONFIDENTIAL

Fn

SINUS (In/Freq/Phase) [0,1]

Il s'agit d'un sinus variant entre 0 et 1 au lieu de -1 et 1.

Contrairement à la trigonométrie traditionnelle, ce **SINUS** boucle à 1, et non pas à 360 degrés ou 2π radian. C'est un usage répandu dans AAASeed où les rotations sont comptées en tour.

Pour obtenir un COSINUS, imposez une phase de 0,25.

TRIANGLE (In/Freq/Phase) [0,1]

Comme un synthétiseur analogique, la sortie monte de 0 à 1 à l'image de la phase montant de 0 à 1 (exclus). Puis à 1 la sortie retombe à 0 et reprend sa progression, et ainsi de suite jusqu'à 2,

Les matheux parleront d'une fonction modulo 1.

On parle aussi de fonction en dent de scie.

La **Freq** modifie le temps de montée et donc la période de bouclage.

TRIANGLE_INV (In/Freq/Phase) [0,1]

Comme le TRIANGLE mais inversé. On part de 1 pour descendre vers 0. puis on remonte brusquement à 1.

TRIANGLE_UP_THEN_DOWN (In/Freq/Phase) [0,1]

Une période sur deux, on a un **TRIANGLE** suivi d'un **TRIANGLE_INV**. On monte puis on descend sans saut brusque de valeur.

Il faut noter que la fréquence est le double de ce qu'elle devrait être mathématiquement. Mais ainsi la pente est la même qu'un **TRIANGLE** ou un **TRIANGLE_INV**, pour une même valeur de **Freq**.

min_TO_MAX_LINEAR (In/Freq/Phase) [0,1]

min_TO_MAX_SINUS (In/Freq/Phase) [0,1]

MAX_TO_min_LINEAR (In/Freq/Phase) [0,1]

MAX_TO_min_SINUS (In/Freq/Phase) [0,1]

La sortie des types **A_TO_B...** passe de A à B lorsque la phase passe de 0 à 1, reste à A pour une phase négative et à B pour une phase supérieure à 1.

Dans le cas ...**_LINEAR** la montée/descente se fait suivant une droite, avec une rupture de pente (discontinuité de tangente pour les matheux) au départ et à l'arrivée.

Dans le cas ...**_SINUS** la montée/descente se fait suivant un sinus, la transition au départ et à l'arrivée est alors douce (continuité de la tangente).

En fait à cette étape min est 0, MAX est 1, c'est l'étape de sortie qui recale à min/MAX.

SQUARE (In/Freq/Phase) [0,1]

Il s'agit du fameux signal carré, qui passe brutalement de 0 à 1.

Le **Bias** est utilisé de manière spécifique ici, il contrôle le rapport entre la partie haute (1) et basse (0) du signal, le rendant asymétrique.

Un **Bias** à 0,25 donne
de 0 à 0,75 le signal est à 0,
de 0,75 à 1(exclu) le signal est à 1.

RANDOM_LINEAR (In/Freq/Phase) [0,1]

RANDOM_GAUSS (In/Freq/Phase) [0,1]

RANDOM (Hasard en français) change la sortie à chaque fois que la phase franchit une valeur entière (ex : -1,0,1,2...) en croissant ou en décroissant. Sans connexion en entrée cela veut dire que le signal de sortie change aléatoirement de valeur tout les 1/**Freq**.

RANDOM_LINEAR donne une probabilité identique à toute les valeurs entre 0 et 1. C'est ce qu'on appelle du bruit blanc.

RANDOM_GAUSS donne une probabilité plus forte aux valeurs autour de 0,5 et une probabilité plus faible aux valeurs vers 0 ou 1. C'est ce qu'on appelle du bruit gaussien ou bruit rose.

TURBULENCE (In/Freq/Phase) [0,1]

FRACTAL SUM (In/Freq/Phase) [0,1]

Ce type exploite ce que l'on appelle le bruit de Perlin, invention de Ken Perlin NYU, hello and thanks Ken.

Le bruit de Perlin est une fonction de bruit continu qui fait correspondre à chaque point d'un espace 3D une valeur entre 0 et 1. La fréquence de variation (en terme de spectre pour les spécialistes) est environ d'une unité.

Si la trax n'a pas d'entrée, la trax calcule le bruit de Perlin pour un point de coordonnées (0, phase, 0), Autrement dit un point se déplace sur l'axe des y, à une vitesse déterminé par la **Freq**, et c'est à ce point qu'est calculé le bruit.

Si la trax a une entrée, c'est une entrée 3D. C'est au point correspondant à cette entrée (en fait les coordonnées sont multipliées par **Freq**) que sera calculé le bruit de Perlin.

En fait, ce type de bruit intègre la notion d'harmonique. Pour chaque harmonique la Fonction de Perlin est calculé avec une fréquence spatiale double, puis ajouté au résultat précédent. On superpose du bruit de plus en plus rapide (et de moins en moins fort), pour chaque harmonique, ajoutant ainsi du détail à chaque fois. C'est un processus similaire à ce qu'on peut faire avec du son.

Le nombre d'harmoniques est donné par **Ease_before**, mais ne descend jamais en dessous de 1.

TURBULENCE est basé sur le même calcul que **FRACTAL SUM**, mais fait intervenir en plus une valeur absolue (pour chaque Harmonique), repliant ainsi autour de zéro les variations. **TURBULENCE** est asymétrique : les variations sont plus brusques en bas (rebonds à 0) et plus douces en haut (1). **FRACTAL SUM** est symétrique et plus doux.

Math

MIN (Ins)

Le minimum (la valeur la plus faible) de toutes les entrées.

MAX (Ins)

Le maximum (la valeur la plus forte) de toutes les entrées.

COPY (In)

La « première » entrée est reproduite. Toutes les étapes suivantes peuvent alors modifier cette valeur.

En particulier ceci permet de changer la dynamique d'un signal (**Min/Max/Offset**).

ADD (Ins)

La somme de toutes les entrées.

Il n'y a pas de soustraction, car il n'y a pas d'ordre dans les connections (voir Traxs généralités). Utilisez d'abord une trax **COPY** avec un max à -1.

AVERAGE (Ins)

La moyenne de toutes les entrées (Barycentre), c'est-à-dire la somme de toutes les entrées divisée par le nombre d'entrées.

INTERPOLATE (Ins) (présent pour l'instant doit disparaître ?)

Interpolation entre deux entrées.

Ceci ne devrait pas exister car il n'y a pas d'ordre garanti dans les connections.

En fait si l'on connaît la structure de fond de AAASeed on peut connaître parfois l'ordre ; ex, à la lecture d'un objet values (à la condition qu'il ai été sauvé par AAASeed), les connections des différentes values sont effectués dans l'ordre.

Pour l'instant, ce type est maintenu pour les cas d'urgence et des problèmes de compatibilité. On peut fabriquer l'équivalent avec plusieurs traxs.

Le param **Freq** contrôle l'interpolation entre les deux « premières » entrées A et B suivant la formule $(1 - \text{Freq}) * A + \text{Freq} * B$. Comme souvent dans AAASeed l'interpolation n'est pas limité à l'intervalle [0,1]. Pour **Freq** à 0 la sortie est donc A, et B pour **Freq** à 1.

MUL (Ins)

La multiplication de toutes les entrées.

Il n'y a pas de division, car il n'y a pas d'ordre dans les connections (voir Traxs généralités). Utilisez d'abord une trax **OVER ONE**.

OVER ONE (In)

Une entrée E devient 1/E.

LOGARITHM (In)

Calcule le logarithme de l'entrée E.

LOGARITHM 10 (ln)

Calcule le logarithme en base 10 de l'entrée E.

...		
0,01	donne	-2
0,1	donne	-1
1	donne	0
10	donne	1
100	donne	2
...		

EXPONENTIAL (ln)

Calcule l'exponentielle de l'entrée E.

POWER BY FREQ (ln)

Calcule l'entrée à la puissance **Freq**.

POWER OF FREQ (ln)

Calcule **Freq** à la puissance de l'entrée.

Multiple

Ces traxs ont du sens uniquement lorsque le (ou les) groupe de couches (**Layers** esclave) est appelé par une **bdd** rendue dans le mode **multiple LAYERS** (appelé ici **bdd** maître). Le rendu utilise alors le **Layers** esclave de manière répétitive, changeant certaines valeurs à chaque itération.

Faire du rendu multiple avec une **bdd_text2d** avec les traxs ci-dessous branchés sur le param **text**, est un bon moyen de visualiser le fonctionnement de ces traxs.

MULTIPLE_INDEX

A chaque itération cette valeur augmente de 1 en partant de 0. La dernière boucle génère la valeur de **MULTIPLE_NB-1**.

Dans le cas où la **bdd** maître est un système de particule (**bdd_part**), les choses se passent différemment. A la création d'une particule un id lui est attribué, elle le conserve tout au long de sa vie. C'est une valeur provenant de cette id, qui est transmise par cette trax quand la particule est dessinée en multiple. Ainsi à chaque rendu d'une particule, cette trax sera identique en valeur.

MULTIPLE_REAL [0,1]

Cette trax est très similaire à la précédente mais si l'on part de 0, on arrête à 1, augmentant la valeur de $1/(\text{MULTIPLE_NB}-1)$ à chaque itération (on appelle cette valeur le pas).

On peut ainsi brancher directement la sortie de cette trax, sans avoir à modifier le **Max** si le nombre de rendu multiple change. Seul le pas change.

MULTIPLE_NB

C'est le nombre total de rendus multiple, que la bdd Maître effectue.

MULTIPLE_INDEX_U

MULTIPLE_REAL_U [0,1]

MULTIPLE_NB_U

MULTIPLE_INDEX_V

MULTIPLE_REAL_V [0,1]

MULTIPLE_NB_V

Certaines bdds (**bdd_grid**, **bdd_sphere**, ...) sont dites **bdd_uv**. Elles permettent de contrôler le nombre d'itérations sur deux axes. Elles incluent forcément les params **nb_u** et **nb_v** (qui peuvent d'ailleurs avoir la valeur 1).

Ces traxs se comportent de manière similaires aux précédentes mais traitent uniquement les itérations sur les axes **u** et **v**.

MULTIPLE_INDEX_W

MULTIPLE_REAL_W [0,1]

MULTIPLE_NB_W

Certaines bdds (**bdd_grid**,...) permettent de contrôler le nombre d'itérations sur trois axes. Elles incluent forcément les params **nb_u**, **nb_v** et **nb_axe** (comprendre **nb_w**).

Ces traxs se comportent de manière similaires au précédentes mais traitent uniquement l'itération sur l'axes **w**.

MULTIPLE_PARAMETER

L'objet **multiple** comporte 16 params **parameter_...** (de **00** à **15**) destiné à transmettre 16 valeurs du layer maître au layers esclave. Les programmeurs parlerons de passage de paramètres.

control_id-1 sélectionne lequel de ces params est transmis par cette étape.

Test

Toutes ces traxs définisse une condition. la sortie d'étage est à 1 si la condition est vrai, à 0 sinon.

Tous ces types ne sont pas nécessaires, l'utilisation de **Min** et **Max** permettrait d'en diminuer le nombre, mais rendrait la lecture des traxs plus difficile.

Cette pratique n'est donc pas recommandé, sauf si des traxs connectées aux **Min** et **Max** servent à changer le rôle des test.

EQUAL (Ins) 0/1

Teste l'égalité des deux « premières » entrées.

NOT_EQUAL (Ins) 0/1

Teste l'inégalité des deux « premières » entrées.

EQUAL_FREQ (In) 0/1

Teste l'égalité de la « première » entrée et de **Freq**.

NOT_EQUAL_FREQ (In) 0/1

Teste l'égalité de la « première » entrée et de **Freq**.

MORE_THAN_FREQ (In) 0/1

Teste si la « première » entrée est supérieure strictement à **Freq**.

MORE_EQUAL_THAN_FREQ (In) 0/1

Teste si la « première » entrée est supérieure ou égale à **Freq**.

LESS_THAN_FREQ (In) 0/1

Teste si la « première » entrée est inférieure strictement à **Freq**.

LESS_EQUAL_THAN_FREQ (In) 0/1

Teste si la « première » entrée est inférieure ou égale à **Freq**.

INSIDE_FREQ_AND_PHASE (In) 0/1

Teste si l'entrée est à l'intérieur (égalité comprise) des valeurs de **Freq** et **Phase**. **Freq** n'est pas contraint à être inférieur à **Phase**, AAASeed les permute de manière interne en cas de nécessité.

OUTSIDE_FREQ_AND_PHASE (In) 0/1

Teste si l'entrée est à l'extérieur (égalité exclue) des valeurs de **Freq** et **Phase**. **Freq** n'est pas contraint à être inférieur à **Phase**, AAASeed les permute de manière interne en cas de nécessité.

Test Real

Ces traxs ressemblent au traxs de **Test** mais leur résultat peut passer de 1 à 0 de manière continue, la zone de cette transition est définie avant par **Ease_before** et après par **Ease_after**. Ces valeurs à 0 ont retrouve une transition directe.

INSIDE_FREQ_PHASE_LINEAR (In) [0,1]

INSIDE_FREQ_PHASE_SINUS (In) [0,1]

Si l'entrée est à l'intérieur (égalité comprise) des valeurs de **Freq** et **Phase** la sortie de l'étape est 1. Avant l'intervalle à une distance supérieure à **Ease_before** et après l'intervalle à une distance supérieure à **Ease_after** la sortie de l'étape est 0. Dans les deux zones restantes la transition se fait suivant une droite (...**LINEAR**) ou un sinus (...**SINUS** avec continuité de tangente).

Freq n'est pas contraint à être inférieur à **Phase**, AAASeed les permute de manière interne en cas de nécessité.

PROXIMITY_FREQ_LINEAR (In) [0,1]

PROXIMITY_FREQ_SINUS (In) [0,1]

Ces traxs correspondent exactement au deux traxs précédentes lorsque **Freq** et **Phase** sont identiques.

Change

CHANGE_LINEAR (In) (Freq/Phase)

CHANGE_SINUS (In) (Freq/Phase)

Ces traxs sont faites pour lisser les variations d'une entrée E qui passe d'une valeur constante A à une autre B.

Lorsque E change de valeur, la phase interne est remise à 0, puis elle croit jusqu'à 1 en un temps $1/\text{Freq}$. Cette phase interne pilote une droite (**CHANGE_LINEAR**) ou un sinus (**CHANGE_SINUS**) qui raccorde la sortie de A à B.

Autrement dit lorsque l'entrée change brutalement, la sortie suit cette variation mais de manière continue, la rejoignant en un temps $1/\text{Freq}$. La variation est alors une droite (**CHANGE_LINEAR**) ou un sinus (**CHANGE_SINUS**) qui va de A à B.

IN_IS_CHANGED (In) 0/1

Comme son nom l'indique la sortie détecte un changement en entrée. Si l'entrée est identique à celle de l'exécution précédente la sortie est 0.

Si il y a eu un changement la sortie est à 1.

Data

Ces types de traxs sont regroupés ici, sous le vocable data, car il s'agit des mécanismes centraux de AAASeed pour faire circuler des informations de layers en layers. En effet, chaque layer a son objet traxs qui regroupe les objets trax (trax locale). Ces traxs ne peuvent être connecté qu'aux objets locaux (ceux qui dépendent du même layer)

Si l'on veut faire circuler des informations à l'extérieur d'un layer on peut utiliser des pistes globales, mais dans ce cas les layers deviennent dépendant de la structure globale, et il est alors difficile de transporter des layers d'une application à l'autre, de les dupliquer... Il est donc vivement recommandé de se servir de ces types.

Pour aller plus loin, à long terme, la notion même de trax globale va se modifier, voir disparaître. Alors que ces mécanismes resteront.

Ces traxs désignant des valeurs par un mécanisme d'indexation (**Channel**, **control_id**, **control_id_bis**) ou de nom, correspondent en fait à ce que les programmeurs appellent des variables (des variables globales pour être précis).

Ce mécanisme symbolique (on désigne une valeur par un groupe de trois chiffre ou par un nom) doit être utilisé de manière systématique dans tout environnement AAASeed autre que temporaire.

En particulier, lorsque une application AAASeed utilise des capteurs, la sortie de ces capteurs doit passer par ce mécanisme (objet capteur connecté avec **CONTROL_SET** ou **NAME_REAL_SET**). Le processus commandé par ces capteurs récupère alors les commandes sous une forme symbolique (**CONTROL** ou **NAME_REAL**). Ainsi on peut changer de capteur sans avoir à toucher au processus. Ex : j'ai fabriqué un environnement qui se commande avec un dispositif exotique, mais je fais une demo avec un joystick, je n'ai qu'à brancher des traxs **CONTROL_SET** ou **NAME_REAL_SET** sur le joystick et à désactiver celles de mon dispositif exotique... Passer ensuite d'un type de capteur à un autre devient juste une affaire d'activation/désactivation limité au objet capteur.

Pour pousser le vice, ce concept d'isoler des processus au travers de valeurs symboliques est le moyen de créer des blocks de haut niveaux, avec des entrées et des sorties (symboliques cette fois). C'est l'extension du principe de base de AAASeed (des blocs avec des entrées et des sorties) mais cette fois-ci dans vos mains, plus dans les miennes.

Avant de travailler sur un gros environnement je vous encourage donc à nommer les choses. Cela permet aussi de travailler en équipe, chacun sur son bout d'abord, puis de réunir ces bouts facilement, au simple prix d'un conventions symboliques (Ex : On_A, next, A_finished...).

CONTROL (Channel, Control_id,Control_id_bis)

Le type **CONTROL** est strictement identique au type **File** (lire trax **FILE** plus bas avant d'aller plus loin).

Channel_id désigne la datagrid avec pour l'instant un décalage de 1. Ex :

Channel_id à 1 désigne la **datagrid_2d_0**. L'objet **Datagrid_2d** désigne à son tour un fichier.

Control_id précise la ligne de ce fichier (1 pour la première ligne)

Control_id_bis précise lui la colonne dans cette ligne (1 pour la première colonne).

CONTROL_SET (In) (Channel, Control_id,Control_id_bis)

Si **CONTROL** lit, **CONTROL_SET** est son pendant en écriture. Cette trax écrit l'entrée dans la « Datagrid ». **CONTROL_SET** attend un nombre en entrée, si c'est du texte il sera converti.

CONTROL_SET est une trax de type **COPY** qui écrit sa sortie dans la « datagrid ».

Si la trax n'a pas d'entrée la valeur de Freq est utilisée.

Seul une valeur numérique peut pour l'instant être utilisée en entrée.

NAME_REAL

Ce type de trax utilise le param **Name** (voir doc param **Name** plus haut) pour retrouver une valeur (numérique pour l'instant).

NAME_REAL_SET (In)

Ce type de trax utilise le param **Name** (voir doc param **Name** plus haut) pour associer à ce nom une valeur (numérique pour l'instant).

NAME_REAL_SET est une trax de type **COPY** qui donne un nom symbolique à sa sortie.

Value

VALUE fait référence aux objets **Values**, soit celui attaché à dans chaque groupe/layers (**VALUE**), soit celui attaché à l'objet **App/Values** (**VALUE_GLOBAL**).

VALUE (Control_id)

VALUE_GLOBAL (Control_id)

SET_VALUE (In) (Control_id)

SET_VALUE_GLOBAL (In) (Control_id)

Control_id à x désigne la valeur du param **value_x** de l'objet **Values**.

Il faut donc noter que le param **value_00** est difficilement atteignable. En fait

Control_id boucle, c'est-à-dire que si l'objet **Values** à 128 params (de 0 à 127) **Control_id** à 128 (ou $128*n$) désigne **value_00**, **Control_id** à 129 (ou $128*n+1$) désigne **value_01**...

Pour les traxs ...**_SET**, l'étage de sortie écrit à l'emplacement désigné.

Les autres passent en sortie de l'étage la valeur du param **value_x** de l'objet **Values**.

MORE_THAN_VALUE_INDEX (In) (Control_id,Control_id_bis)

MORE_THAN_VALUE_GLOBAL_INDEX (In) (Control_id,Control_id_bis)

Lire **VALUE_INDEX MORE_THAN IN**.

L'entrée est comparée aux **value_x** pour x allant de **Control_id** à

Control_id_bis compris. Cet étage renvoie l'index du premier param

value_x supérieur ou égal à l'entrée. Cet index est relatif : $x - \text{Control_id}$. Si aucun param ne satisfait la condition, l'étage renvoie (pour l'instant)

Control_id_bis - Control_id + 1.

Si **Control_id** est supérieur à **Control_id_bis**, l'étage renvoie 0 (pour l'instant).

Cette trax est très utile pour avoir une série de valeurs croissantes (ex : des temps successifs) et déterminer ainsi dans quel intervalle se situe l'entrée.

SORT_VALUE_INDEX_FROM_MIN (Control_id,Control_id_bis)

SORT_VALUE_INDEX_FROM_MAX (Control_id,Control_id_bis)

SORT_VALUE_GLOBAL_INDEX_FROM_MIN (Control_id,Control_id_bis)

SORT_VALUE_GLOBAL_INDEX_FROM_MAX (Control_id,Control_id_bis)

Tri (Sort) en français.

Pour x allant de **Control_id** à **Control_id_bis** compris, les **value_x** sont classés dans une liste (par ordre croissant pour **MIN**, et décroissant pour **MAX**).

Cet étage renvoie l'index relatif ($x - \text{Control_id}$) du param **value_x** à la position **Freq** dans la liste ordonné.

Si **Control_id** est supérieur à **Control_id_bis**, ces valeurs sont permutées.

Dans le traitement **Freq** inférieur à 0 est ramené à 0, **Freq** supérieur à **Control_id_bis - Control_id** est ramené à **Control_id_bis - Control_id**.

Ces traxs sont le seul mécanisme de tri existant (pour l'instant).

Pour ramener la valeur et non l'index, il faut alors connecter la sortie (ne pas oublier un offset de la même valeur que **Control_id**) sur le param **Control_id** d'une trax **VALUE**.

Trax

TRAX (Control_id)

Cette trax renvoie la valeur de sortie de la trax sélectionné par la valeur de **Control_id** dans le même groupe de trax (**Traxs**).

Input

MIDI CONTROLLER (Channel, Control_id) [0,1]

MIDI CONTROLLER RELATIVE (Channel, Control_id)

MIDI VELOCITY (Channel, Control_id) [0,1]

Channel et **Control_id** viennent en fait de MIDI, la correspondance est donc ici directe.

Voir **MIDI** dans dans **pref** et **MIDI** dans le menu **Focus**.

traxs_channel_offset et **traxs_control_offset** dans le layers correspondant sont ajoutées respectivement à **Channel** et **Control_id**. C'est un moyen rapide de déplacer globalement une interface de contrôle MIDI.

MIDI CONTROLLER renvoie la valeur du contrôleur MIDI.

MIDI CONTROLLER RELATIVE considère le contrôleur MIDI comme une valeur contrôlant la Freq d'une piste de type TIME. Le contrôleur au minimum l'étage décroît de 1 par seconde, au maximum le contrôleur croît de 1 par seconde. Il existe une zone neutre au centre.

MIDI VELOCITY renvoie la valeur de vitesse pour la note **Control_id**. Une valeur à 0 est l'équivalent d'un note Off MIDI.

NEAT (Channel, Control_id) [0,1]

Il s'agit de la lecture des convertisseurs analogiques/numériques Thing 2 (**TNG 2**) et Thing 3 (**TNG 3**) développé par Dave Warner (www.pulsar.org qu'il soit remercié ici).

Voir **Neat** dans dans **pref** et **Neat** dans le menu **Focus**.

TOASTER (Control_id) [0,1]

Il s'agit de la lecture des convertisseurs analogiques/numériques sans fils développés par La Kitchen (www.lakitchen.fr)

Voir **Toaster** dans dans **pref** et **Toaster** dans le menu **Focus**.

NET_IN (Channel, Control_id)

Cette trax est le pendant en réception de l'étage **net_out** (Voir **Groupe Net Midi** plus haut). **Channel** et **Control_id** doivent correspondre à ceux utilisés dans l'envoi.

FFT (Channel)

FFT signifie Fast Fourier Transform, en français la Transformé de Fourier Rapide. Elle est ici appliquée au son, il s'agit d'une analyse spectrale.

Autrement dit cette trax renvoie l'énergie du son dans une bande de fréquence.

Channel sélectionne le canal son : **Channel** à 1 Gauche, **Channel** à 2 Droite.

Freq et **Phase** déterminent l'intervalle de fréquence de manière logarithmique : 0 pour 20Hz, 1 pour 20KHz, 0,5 pour environ 1400Hz.

Keyboard & Mouse

SLIDE_INDEX

Cette trax reproduit une fonctionnalité similaire à ce que l'on peut trouver dans PowerPoint pour passer d'écran (slide) en écran.

L'appui sur la barre d'espace incrémente de 1 la sortie de cet étage, l'appui sur Backspace décrémente de 1. **Restart_trig** remet ce compteur à 0.

ALPHABET (Control_id,Control_id_bis)

ALPHABET_LAST (Control_id,Control_id_bis)

Ces trax permettent de détecter l'enfoncement d'une touche de a(1) à z(26) (sans distinction de majuscule/minuscule et sans Ctrl ou Alt). Cet étage renvoie la lettre enfoncée sous forme d'un chiffre correspondant à la lettre **Control_id** et **Control_id_bis** détermine l'intervalle actif : ex **Control_id** à 2 et **Control_id_bis** à 5 font que la piste ne renvoie que les touches b(2),c(3),d(4) et e(5).

Pour **ALPHABET** la sortie de l'étage prend la valeur correspondante à la touche seulement lors de l'enfoncement de la touche et retombe à 0 au rendu suivant.

ALPHABET_LAST est identique à **ALPHABET** mais garde la dernière valeur au lieu de retomber à 0. Pour ce type **Restart_trig** force la remise à 0.

Ces trax sont utiles pour installer des commandes au clavier.

Le mode **Edit** (Ctrl-E et **pref_ui/Interface/Edit**) désactivé empêche les touches d'être envoyées à l'interface traditionnelle de AAASeed de même que le param et **pref_ui/Interface/keyboard_alphabet_for_ui** à OFF. Le param **pref_ui/Interface/keyboard_alphabet_for_trax** contrôle lui l'envoi des touches à ces trax.

ALPHABET_ONE (Control_id) 0/1

ALPHABET_ONE ne teste qu'une (One) lettre (désigné par **Control_id**), la valeur de sortie de l'étage est 0 ou 1 suivant l'enfoncement de cette touche.

MOUSE_X

Position horizontale de la souris dans le viewport de rendu.

0 à gauche, 1 à droite.

La valeur peut sortir de l'intervalle [0,1] si la souris sort du viewport. Utilisez les params **Limit...** si vous souhaitez rester canonique : **Limit** à CLAMP, **Limit_min** à 0 et **Limit_max** à 1.

MOUSE_Y

Position verticale de la souris dans le viewport de rendu.

0 en bas, 1 en haut.

La valeur peut sortir de l'intervalle [0,1] si la souris sort du viewport. Utilisez les params **Limit...** si vous souhaitez rester canonique metez : **Limit** à CLAMP, **Limit_min** à 0 et **Limit_max** à 1.

MOUSE_CLICK_LEFT 0/1

Etat du bouton de gauche (left) de la souris.

0 relâché, 1 enfoncé.

MOUSE_CLICK_MIDDLE 0/1

Etat du bouton du milieu (middle) de la souris.
0 relâché, 1 enfoncé.

MOUSE_CLICK_RIGHT 0/1

Etat du bouton du milieu (right) de la souris.
0 relâché, 1 enfoncé.

KEYBOARD_SHIFT 0/1

Etat de la touche Majuscule shift) du clavier.
0 relâchée, 1 enfoncée.

KEYBOARD_ALT 0/1

Etat de la touche Alt du clavier.
0 relâchée, 1 enfoncée.

KEYBOARD_CTRL 0/1

Etat de la touche Ctrl du clavier.
0 relâchée, 1 enfoncée.

File

Dans le menu Focus on trouve un des objets **Datagrid 2d x** où x peut varier de 0 à 7 (7 pour l'instant). Ces objets servent d'intermédiaire avec le fichier désigné par le param **Datagrid_2d/data_filename**.

Ce fichier, de type csv (Comma Separated Values qui peut donc être lu et sauvé par Excel) est composé de valeurs (nombre ou texte) séparé par des virgules ou des points virgules suivant l'état de **Datagrid_2d/comma_is_separator**.

La première fois que le param **Datagrid_2d /active** est ON le fichier s'il est présent est lu. Ensuite si le param **Datagrid_2d /check_for_change** est ON, à chaque frame la date de modification du fichier est vérifiée, si elle a changé le fichier est relu.

Lorsque l'environnement est sauvé si le param **Datagrid_2d /save_data** est ON, les données seront sauvées. Les trax de type **CONTROL_SET** (voir les Trax **Data** plus haut) permettent en effet de modifier ces données.

Ces fichiers peuvent donc agir chacun comme « une grille de donnée à deux dimensions » (**Datagrid_2d**).

Dans les types suivants, **channel_id** désigne la datagrid avec pour l'instant un décalage de 1. Ex : **Channel_id** à 1 désigne la **datagrid_2d_0**. L'objet **Datagrid_2d** désigne à son tour un fichier.

Control_id précise la ligne de ce fichier (1 pour la première ligne)

Control_id_bis précise lui la colonne dans cette ligne (1 pour la première colonne).

Pour résumer on désigne un élément (case dans la suite) avec **channel_id-1 (datagrid_2d)**, **control_id** (ligne) et **control_id_bis** (colonne).

FILE_IS_CHANGED (Channel) 0/1

Passes à 1 lorsque un fichier a changé et qu'il vient donc d'être relu.

FILE (Channel, Control_id,Control_id_bis)

Retourne le contenu de la case, ce peut être un nombre ou du texte.

FILE_STRING_LEN (Channel, Control_id,Control_id_bis)

Retourne la longueur du texte (nombre de caractères) présent dans cette case, ou 0 s'il s'agit d'un nombre.

FILE_LETTER (Channel, Control_id,Control_id_bis)

Retourne un texte long d'un d'un caractère.

Si la case sélectionnée est du texte, ce caractère est celui à la position indiqué par **Freq** :

0 (en fait strictement inférieur à 1.) désigne le premier caractère,

FILE_STRING_LEN-1 désigne le dernier caractère,

FILE_STRING_LEN ou plus renvoie un caractère Nul (ascii 0).

Si l'information sélectionnée est un nombre, ce caractère est celui de la table ascii correspondant à la valeur de ce nombre.

Time

Ces types concernent tous le temps.

Le type **TIME** donne un temps AAASeed.

Les autres de ces types suivent un temps « absolu », calé sur le temps de la machine. Il s'agit de « nombre de », c'est-à-dire qu'avant les étages de traitement suivants, la sortie de cet étage est un entier. Ce sont les composants de base pour fabriquer tout types d'horloge, ou déclencher des événements à des dates précises.

Il s'agit du calendrier « Chrétien », reposant sur la librairie C.

A part dans **TIME** la **Phase** est ajouté dans l'étage qui traite les **fns**.

TIME (Freq/Phase absolu)

Il s'agit du temps AAASeed (**app/time**) en secondes depuis la dernière remise à zéro du temps (démarrage de AAASeed ou touche t).

De plus **Restart_trig** remet à zéro la trax (on définit ainsi une origine locale de temps).

Les param **Freq** et **Phase** interviennent ici mais en absolu (pas de dt) la sortie de cette étape est donc **Freq * (t-t0) + Phase**, t0 étant l'origine locale.

SECOND (Phase)

Nombre de secondes après la minute, de 0 à 59.

MINUTE (Phase)

Nombre de minutes après l'heure, de 0 à 59.

HOURL (Phase)

Nombre d'heures depuis minuit, de 0 à 23.

DAY (Phase)

Jour du mois, de 1 à 31.

MONTH (Phase)

Mois de l'année, de 1(Janvier) à 12 (Décembre).

YEAR (Phase)

L'année du calendrier, 2003 cette année

DAY_OF_WEEK (Phase)

Nombre de Jours depuis Dimanche : 0(Dimanche), 1(Lundi) ... 6(Samedi).

DAY_OF_YEAR (Phase)

Nombre de jours depuis le début de l'année, de 0 à 365.

Image

Ces traxs vont chercher dans une image (image **control_id**-1 dans la bank) une information au point (u v) défini par (**Freq Phase**) sachant que (0 0) désigne le coin inférieur droit et (0,9999999 0,9999999) le coin supérieur gauche.

Pour l'instant, u et v « boucle » à 1. donc (2 1,5) désigne le point (0 0,5) c'est-à-dire le milieu du bord gauche de l'image. (1 1) désigne donc le point (0 0) d'où les 0,9999999. Des suggestions sont attendues des utilisateurs.

IMAGE RED (Control_id) [0,1]

Rouge.

IMAGE GREEN (Control_id) [0,1]

Vert.

IMAGE BLUE (Control_id) [0,1]

Bleu.

IMAGE ALPHA (Control_id) [0,1]

Si une image n'a pas d'alpha, 1 est retourné par cet étape.

IMAGE GREY (Control_id) [0,1]

Pour l'instant le gris est calculé comme la somme red+green+blue divisé par 3. C'est une approximation qui sera corrigée.

IMAGE SIZE X (Control_id)

La taille horizontale de l'image en pixel.

IMAGE SIZE Y (Control_id)

La taille verticale de l'image en pixel.

IMAGE SIZE RATIO (Control_id)

La taille horizontale de l'image en pixel divisé par la taille verticale de l'image en pixel.

Video

Ces traxs fonctionnent comme les traxs image : elle ramène une information au point (u v) défini par (**Freq Phase**).

Pour l'instant, et ceci est un bug, la vidéo choisie est la vidéo utilisée dans le dernier objet **tex_video** mis à jour avant la trax. Ceci implique que l'objet **tex_video** est référencé dans un groupe de couches (**layers**) précédents.

VIDEO RED [0,1]

VIDEO GREEN [0,1]

VIDEO BLUE [0,1]

VIDEO ALPHA [0,1]

VIDEO GREY [0,1]

Text

Ces traxs vont chercher le text correspondant à **control_id** dans la bank (voir le menu text). Il est à noter que le premier texte (d'indice 0) ne donc pas être ainsi atteint.

Si control_id est plus grand que le nombre de texte dans la banque, pour l'instant, le dernier texte (celui d'indice le plus grand) sera utilisé.

TEXT (Control_id)

TEXT_LEN (Control_id)

TEXT_LINE_NB (Control_id)

Bdd Tri

BDD_TRI_NAME (Channel, Control_id)

Non documenté

Net Host Id

L'objet de ces traxs est de spécialiser une application AAASeed en fonction de la machine sur laquelle elle s'exécute. Ainsi un même environnement se comportera différemment sur des machines différentes.

NET_HOST_ID

Renvoie le **Host_id** issu de l'objet **Net**.

NET_HOST_ID_EQUAL_FREQ 0/1

NET_HOST_ID_NOT_EQUAL_FREQ 0/1

NET_HOST_ID_MORE_THAN_FREQ 0/1

NET_HOST_ID_MORE_EQUAL_THAN_FREQ 0/1

NET_HOST_ID_LESS_THAN_FREQ 0/1

NET_HOST_ID_LESS_EQUAL_THAN_FREQ 0/1

Ces traxs se comportent comme les traxs de test (voir plus haut **Test**), sauf qu'elles ne testent pas l'entrée mais le **Host_id** issu de l'objet **Net**.

Il s'agit de raccourcis évitant des branchements supplémentaires.

Dangerous

COPY_BLOCK_REAL (In)

Non documenté

Experimental

CELL_SIZE

Non documenté

3D Distance 1D

3D_DISTANCE (Ins)

Non documenté

3D_DISTANCE_MOCAP (Channel, Control_id,Channel_bis,Control_id_bis)

Non documenté

3D_DISTANCE_CAMERA (In)

Non documenté

3D Fn

Ces traxs sont le pendant **3D** des traxs **Fn** (Voir plus haut).
Voir la documentation des types de trax **Fn** sans le préfix **3D_**.

3D_RANDOM_LINEAR (In/Freq/Phase) [0,1]

3D_RANDOM_GAUSS (In/Freq/Phase) [0,1]

3D Math

Ces traxs sont le pendant **3D** des traxs **Math** (Voir plus haut).
Voir la documentation des types de trax **Math** sans le préfix **3D_**.

3D_COPY (In)

3D_ADD (Ins)

3D_AVERAGE (Ins)

3D_INTERPOLATE (Ins)

3D Change

Ces traxs sont le pendant **3D** des traxs **Change** (Voir plus haut).
Voir la documentation des types de trax **Change** sans le préfix **3D_**.

3D_CHANGE_LINEAR (In) (Freq/Phase)

3D_CHANGE_SINUS (In) (Freq/Phase)

3D Data

3D_CONTROL (Channel, Control_id,Control_id_bis)

3D_CONTROL_SET (In) (Channel, Control_id,Control_id_bis)

Ces traxs sont le pendant **3D** des traxs **CONTROL** (Voir plus haut).
Voir la documentation des types de trax **CONTROL** sans le préfix **3D_**.

Channel_id désigne la datagrid avec pour l'instant un décalage de 1. Ex :
Channel_id à 1 désigne la **datagrid_2d_0**. L'objet **Datagrid_2d** désigne à son tour un fichier.

Control_id précise la ligne de ce fichier (1 pour la première ligne)

Control_id_bis précise lui la colonne dans cette ligne (1 pour la première colonne).

Trois emplacements de la même ligne sont utilisées au lieu de un
(**Control_id_bis**, **Control_id_bis+1** et **Control_id_bis+2**).

3D Value

3D_VALUE (Control_id)

3D_VALUE_GLOBAL (Control_id)

Non documenté

CONFIDENTIAL

3D Mocap

3D_PATH_POS (Channel, Control_id, In/Freq/Phase)

Non documenté

3D_PATH_TO_WORLD (Channel, Control_id, In/Freq/Phase)

Non documenté

3D_PATH_SCALE (Channel, Control_id, In/Freq/Phase)

Non documenté

3D_MOCAP (Channel, Control_id)

Non documenté

3D_MOCAP_AXE_X (Channel, Control_id)

Non documenté

3D_MOCAP_AXE_Y (Channel, Control_id)

Non documenté

3D_MOCAP_AXE_Z (Channel, Control_id)

Non documenté

3D_MOCAP_TANGENT (Channel, Control_id)

Non documenté

3D_MOCAP_SCALE (Channel, Control_id)

Non documenté

3D_MOCAP_FEED (Channel, Control_id)

Non documenté

3D Image

3D_IMAGE_COLOR (Control_id) [0,1]

Comme les traxs **Image** (Voir plus haut), La sortie est **3D**: rouge, vert et bleu (**red**, **green** et **blue**).

3D Video

3D_VIDEO_COLOR [0,1]

Comme les traxs **Video** (Voir plus haut), La sortie est **3D**: rouge, vert et bleu (**red**, **green** et **blue**).

3D Bdd Tri

3D_BDD_TRI_CENTER (Channel, Control_id)

Non documenté

3D_BDD_TRI_BARYCENTER (Channel, Control_id)

Non documenté

3D_BDD_TRI_MIN (Channel, Control_id)

Non documenté

3D_BDD_TRI_MAX (Channel, Control_id)

Non documenté

3D_BDD_TRI_SIZE (Channel, Control_id)

Non documenté

3D_BDD_TRI_TEX_CENTER (Channel, Control_id)

Non documenté

3D_BDD_TRI_TEX_SIZE (Channel, Control_id)

Non documenté

3D Experimental

3D_CELL_POS

Non documenté