TAPEMOVIE : UN ENVIRONNEMENT LOGICIEL POUR LA CREATION TEMPS REEL INTERMEDIA

Tom Mays
Didascalie.net
CICM – Université de Paris VIII
CNSM de Paris
contact@tommays.net

Renaud Rubiano
Didascalie.net
contact@renaudrubiano.com

RÉSUMÉ

Nous voulons dans ce texte présenter l'environnement logiciel *tapemovie*, programmé avec Max/MSPJitter). C'est un environnement pour la composition et l'improvisation musicale temps réel, les dispositifs audio/visuel interactifs, et l'art numérique dans le spectacle vivant. Il est construit sur les fondamentaux de modularité, d'optimisation, de souplesse de configuration et d'interconnectivité. Son architecture est un hub (tapemovie) qui agglomère des groupes de modules et centralise les configurations des sous environnements *tape* (audio), et *movie*, (vidéo/GL). Une gestion de projet permet de changer rapidement de configuration et de préférences utilisateur. La partie tape met en jeu des modules de traitements et d'analyses sonores connectés par une matrice. La partie movie implémente des sources et des traitements vidéos à la fois en traitement matriciel (jit.matrix)[1] et en module OpenGL[2]. Des mappeurs multi-destination permettent des connections expressives avec des contrôleurs externes ou des analyses internes. Enfin, pour les événements, l'event manager centralise le stockage des données, leur état, leur séquencement et leur évolution.

1. INTRODUCTION

Tapemovie (http://www.tapemovie.org) est un environnement logiciel pour la création temps réel intermédia. Il a été écrit dans Max/MSP/Jitter de Cycling'74 et existe à la fois sous forme de patch et sous forme d'application standalone fonctionnant uniquement sous MacOSX – dans les 2 cas sous licence libre LGPL[3]. Ses champs d'action sont : la composition et l'improvisation musicale temps-réel, les dispositifs intermédia interactifs, et l'art numérique pour le spectacle vivant. C'est un outil d'écriture et de contrôle de média en temps réel, construit par des praticiens afin de répondre à leurs propres besoins. Il vise un utilisateur prêt à acquérir une certaine compétence en informatique musicale et visuelle, et qui cherche un environnement solide et puissant lui permettant des réalisations exigeantes, sans avoir à programmer dans Max/MSP/Jitter. Il est le fruit de 20 ans de programmation temps réel de Tom Mays (http://www.tommays.net), de l'expérience intensive dans l'art visuel et le spectacle vivant interactif de Renaud Rubiano (http://renaudrubiano.com/), de la mise en situation très poussée et les documentations du réalisateur sonore Olivier Pfeiffer, et du soutien de la plateforme *didascalie.net* (http://didascalie.net/). Il est utilisé depuis 2007 dans les projets de ses auteurs, dans ceux de *didascalie.net* via les créations et les accompagnements des projets, ainsi que par divers étudiants en composition au CNSMDP[4]. Tapemovie participe également à l'évaluation des prototypes du projet de recherche Virage[5] autour des interfaces de contrôle et d'écriture.

2. HISTORIQUE

A partir des premiers environnements intégrés de contrôle et de traitement de signal, comme Max/FTS à l'Ircam pour la NeXT et les cartes ISPW au début des années 90, en passant par PureData, puis MSP qui se rajoute à Max en 1997 [1], chaque utilisateur expert cherchait petit à petit à faciliter et optimiser leur travail de développement en créant des bibliothèques de fonctions (abstractions et objets) et en réutilisant des structures de patches globaux. L'auteur, ayant fait de nombreuses créations et réalisations avec Max MIDI, Max/FTS à l'Ircam et Max/MSP depuis 1991, a commencé à formaliser de cette manière son environnement de traitement audio temps réel au début des années 2000. Il y avait déjà l'architecture modulaire, la séparation des moteurs de traitement et leurs interfaces, une matrice centrale qui permettait de connecter tout à tout, et un séquencement d'événements en forme de qlist[6]. L'arrivée de Jitter en 2003 [1] a permis de créer un environnement vidéo basé sur les mêmes principes.

A partir de 2005 ces environnements ont commencé à servir dans les créations des spectacles intermédia de la Compagnie *Incidents Mémorables* (devenue *didascalie.net* en 2008). Les membres de

[1] le format des données vidéo qu'utilise Jitter de Cycling74
[2] le calcul de l'image sur la carte graphique
[3] Lesser GNU Public Licence
[4] Conservatoire National Supérieur de Musique et Danse de Paris
[5] cf. site de Virage (http://www.virage-platform.org)
[6] cf. *Max Object Reference*, www.cycling74.com

l'équipe expérimentant sur divers logiciels ont voulu partir de ce travail existant pour réaliser un environnement commun répondant à leurs différentes pratiques.

Il leur fallait donc un nom. Le premier nom par défaut, *l'environnement de Tom*, à été remplacé en 2007 par l'introduction des noms actuels pour les deux parties, audio et vidéo (avec une touche d'ironie) : *tape* (Tom's Audio Patch Environnement) et *movie* (MOdular Video Environment). En 2008 les deux patches *tape* et *movie* et leurs préférences ont été réunis en tant que *plugins* à l'intérieur du patch *tapemovie* – en introduisant la notion de *projet* et celle de construction dynamique des instances des modules (fonction du *build*, cf. 3.3)

Tapemovie version 1.5, prévue pour une sortie officielle en avril 2010, est le sujet de cet article. Elle est programmée dans Max 4, bien qu'elle puisse tourner dans Max5. La version *standalone* permet un fonctionnement complet sans licence Max.

3. L'ARCHITECTURE GLOBALE

Que nous utilisions *tapemovie* dans sa version *standalone* ou dans sa version de *patch* le fonctionnement reste le même. Nous allons donc plutôt parler de la version *standalone*.

Quand on ouvre l'application *tapemovie*, on voit une fenêtre qui représente le *patch* principal (**Figure 1**).

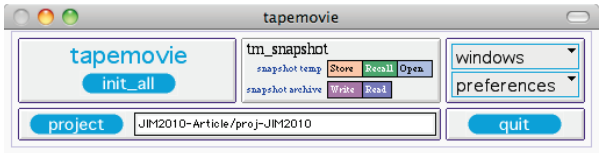

Figure 1. La fenêtre principale de tapemovie.

Ce *patch* est une sorte de « hub » qui sert à gérer le choix du *projet* (cf. 3.1 *le projet*), accéder aux réglages de toutes les configurations (cf 3.2 les configs), initier la construction par script des configurations chosies (cf 3.3 le build), faire un *snapshot* global de tous les paramètres, accéder aux diverses fenêtres de l'application, et également, quitter *tapemovie*. Ce patch est le niveau supérieur de l'arborescence. Il permet d'ouvrir des sous-environnements (plugins) *tape* et *movie*, ainsi que d'instancier divers modules communs aux deux environnements comme des contrôleurs externes, les objets de réseau, et l'event manager. (cf. 3.2.1 tm_config). On configure d'abord *tapemovie* puis *tape* et *movie* (si la configuration le nécessite), chacun ayant encore sa propre configuration avec ses propres modules audio et vidéo.

3.1. Le *project*

Le *project* est un dossier qui contient des sous-dossiers qui rassemblent toutes les préférences utilisateur et les données associées à un « projet » spécifique.

Le dossier */config* contient toutes les données des choix des modules pour l'environnement, plus les données spécifiques à certains modules de traitement, ainsi que les fichiers temporaires (*snapshots*). Le dossier */events* contient uniquement des fichiers des événements et des séquences. Le dossier */instruments* contient le patch *instruments* qui est un espace de programmation « libre » interfaçable avec le reste de l'environnement (cf. 3.5). Le dossier */media* contient tous les fichiers des images et des sons par catégorie.

Par le biais du *project* nous pouvons entièrement changer les caractéristiques de l'environnement de manière souple et dynamique sans quitter le programme. Un même *project* est compatible à la fois avec la version *patch* et la version *standalone* de *tapemovie*.

3.2. Les *config*

Les *config* sont les pages de configuration de chaque partie de l'environnement – *tapemovie*, *tape*, et *movie*. Ils sont accessibles par le menu *preferences* de tapemovie en choisissant *tm_config*, *t_config*, et *m_config* respectivement.

3.2.1. tm_config

Le *tm_config* est édité dans la fenêtre *tm_configeditor*. Nous l'utilisons principalement pour choisir la quantité de chaque module disponible que nous souhaitons avoir dans notre *tapemovie*, spécifique à un projet. Les modules sont organisés par catégorie : *maps*, *osc*, *devices*, *controls* et *plugins* (**Figure 2**).

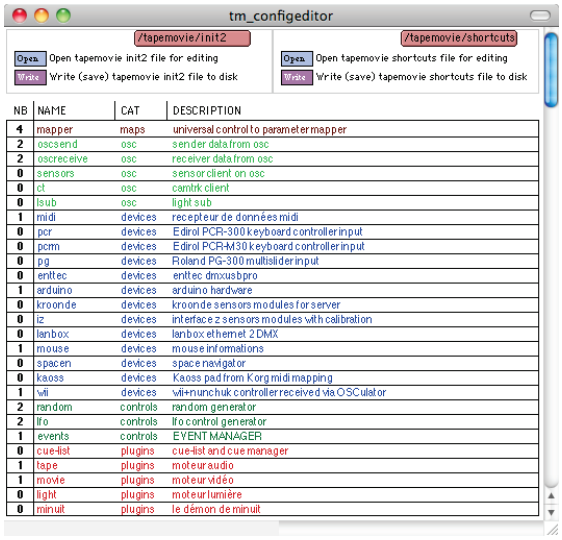

Figure 2. La fenêtre *tm_configeditor*.

Les *maps* sont les *mappeurs* centraux qui permettent de connecter n'importe quel contrôle (objet

send ou *forward* dans Max) à n'importe quel paramètre contrôlable en local ou sur le réseau (cf. 6). Les *osc* sont des modules d'envoi et/ou réception des messages OSC via UDP, comme le module minuit[7] qui permet la communication avec le séquenceur Virage[8]. Dans cette catégorie sont aussi les clients « sensors » et « camtrk » qui reçoivent et distribuent les données provenant d'une application de gestion et d'analyse de capteurs et de suivi de caméra. Les *devices* implémentés dans *tapemovie* sont des contrôleurs comme *midi*, *arduino*, *kroonde*, *kaoss*, *wii* et *mouse*. Les *controls* sont des générateurs de *lfo* ou de *random*, ainsi que l'*event manager* qui est capable de stocker et d'organiser l'ensemble des paramètres. Les *plugins* sont *tape*, *movie* et *light* (gestion de lumiere via DMX) On peut choisir jusqu'à 20 instances de modules normaux, mais uniquement une instance pour chaque *plugin*.

Dans le *tm_configeditor* nous pouvons également éditer une séquence d'initialisation personalisée qui s'appelle *init2*, puis aussi un fichier de définition des *shortcuts* permettant d'ouvrir les fenêtres par une touche du clavier de l'ordinateur.

3.2.2. t_config

Le *t_config* fonctionne exactement sur le même principe que le *tm_config*, mais pour les modules *dsp* du sous environnement *tape* (cf. 4 pour plus info sur *tape*). Il y a, pourtant, un paramètre de plus: le choix global du dispositif de spatialisation (**Figure 3**). Il existe actuellement 3 dispositifs : spat4 (4 haut parleurs en quadriphonie), spat41 (spat4 + un centre) et spat441 (4 plus 4 plus un centre). (cf. 4.4 La Spatialisation)

Figure 3. Choix de dispositif de spatialisation

3.2.3. m_config

Le *m_config* fonctionne aussi sur le même principe que le *tm_config*, mais pour les modules du sous environnement *movie* (cf. 5 pour plus info sur *movie*). Les catégories sont *sources* pour les modules générant de l'image, *effects* pour les modules permettant un traitement de l'image et *outputs* pour ceux qui affiche une image, en local ou via le réseau. Les modules de la catégorie effects peuvent être de deux types, *jmatrix* pour les modules qui *opèrent via le processeur de l'ordinateur (CPU)*, et *gl* pour les modules qui *opèrent via le* processeur graphique (*GPU*/*OpenGL*)[9]

[7] http://www.plateforme-virage.org/?p=1444
[8] cf. site de Virage (http://www.virage-platform.org)
[9] cf. documentation de *Jitter* de Cycling74 (www.cycling74.com)

3.3. Le *build*

Une fois que nous avons réglé les configurations, nous lançons la construction de l'environnement par *script*. Cette fonction s'appelle un *build* et elle est lancé en cliquant sur le bouton *init* dans la fenêtre *tapemovie*. Cet étape peut prendre de 10 secondes à plus d'une minute selon les configurations.

Durant le *build*, les fenêtres *tape* et *movie* s'ouvrent (dans le cas où ils étaient choisis dans le *tm_config*). Le *build* a fini lorsque « tapemovie ready » s'affiche dans la fenêtre « status » (*standalone*) ou « Max » (patch Max). Le patch a été créé automatiquement suivant les choix de configuration de l'utilisateur.

3.4. La séparation des *moteurs* et des *éditeurs*

Un des principes forts dans *tapemovie* est la séparation des moteurs de traitement de leurs éditeurs qui ne sont pas essentiels. On peut donc ne pas les créer pour des raisons d'optimisation, si on souhaite lors d'un *build.*

Il est notre souhait dans le futur proche de s'inspirer de l'architecture modèle/vue/contrôleur pour améliorer ce système. « L'architecture *Modèle/Vue/Contrôleur* (MVC) est une façon d'organiser une interface graphique d'un programme [2]. Elle consiste à distinguer trois entités distinctes qui sont, le *modèle*, la *vue* et le *contrôleur* ayant chacun un rôle précis dans l'interface » [3]

On accède aux éditeurs par les menus *windows* de chaque partie de l'environnement.

3.5. Le patch *instruments*

Dans l'optique de faire de *tapemovie* un outil prêt à l'emploi bien que toujours ouvert à une utilisation experte, il fallait un « endroit » où l'on pouvait faire de la programmation Max librement. Le patch *instruments* se trouve dans le projet dans le dossier */instruments* (**Figure 4**). C'est un patch, pour ainsi dire, vide. On l'ouvre par le menu *windows* de *tapemovie*, ou bien automatiquement par le biais de l'init2 – l'init « user » (cf. 3.2.1).

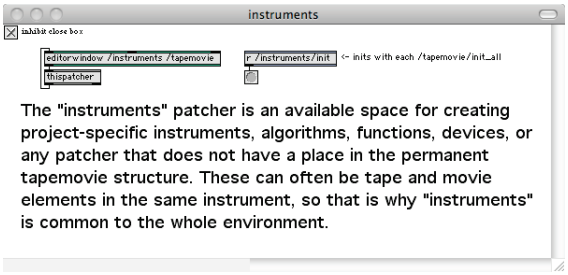

Figure 4. Le patch instruments

Il peut communiquer avec le reste de *tapemovie* par le moyens de *send/receive*[10] traditonnel, et en audio par les *send~/receive~*[11] associés aux modules *aux* dans *tape* (cf. 4.2). Ainsi nous pouvons profiter de l'architecture de *tapemovie* tout en gardant la liberté et l'efficacité de programmer en parallèle dans Max– pour ceux qui le souhaitent.

3.6. Le fonctionnement en réseau

Tapemovie incorpore les bases du protocol OSC[12] pour pouvoir contrôler les paramètres en dehors de l'application, ou bien en dehors de la machine. Chaque paramètre est capable également d'envoyer sa valeur en OSC par le réseau. Cet envoi de paramètres est très utile pour échanger des valeurs entre différents logiciels (compatible OSC) et aussi pour créer des interfaces sur des contrôleurs externes tels le Lemur[13]).

Il y a également une implémentation de Minuit[14] par le module *minuit*. Celui ci à été créé pour pouvoir tester le Séquenceur Virage[15].

3.7. Le SDK

Il est essentiel pour l'avenir de *tapemovie* qu'un utilisateur puisse incorporer ces patches dans l'environnement. Dans ce but, il existe un guide de développement qui documente le fonctionnement et la réalisation d'un module *tapemovie*. Ce SDK permet à un programmeur Max de créer ses propres modules *tapemovie*, *tape* ou *movie*.

4. LA PARTIE AUDIO : *TAPE*

Si le *plugin tape* a été choisi dans la *config* de tapemovie (cf. infra 3.2.1), une fois le *build* effectué, une fenêtre *tape* va s'ouvrir (**Figure 5**).

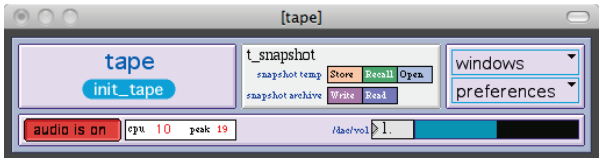

Figure 5. La fenêtre principale de *tape*.

Cette fenêtre ressemble beaucoup à la fenêtre *tapemovie*. Elle contient un bouton *init_tape* qui initialise ou construit uniquement la partie *tape*, un *snapshot* de tous les paramètres *tape*, un accès par menu (ou *shortcuts*) de tous les éditeurs *tape*, un menu pour accéder à la configuration de *tape* (*t_config*), un bouton pour allumer/éteindre l'audio, un affichage de la cpu utilisé, puis enfin un contrôle de volume global de l'audio.

4.1. L'architecture

Les entrées et sorties audio (*adc, dac*) ainsi que tous les modules de traitement ou de synthèse, passent par une matrice centrale. Toutes les connexions et tous les paramètres sont manipulables par messages envoyés (cf. 7 event manager), ce qui permet une souplesse virtuellement infinie de configurations possibles.

4.2. Les modules audio

Les modules audio livrés en standard avec *tapemovie* représentent un certain choix de simplicité en se reposant sur le principe que nous pouvons construire des effets et des instruments complexes en connectant des modules entre eux par le biais de la matrice (**Tableau 1**. La liste des modules *tape*.), et en créant des mises en correspondance entre paramètres via les mappeurs (cf. 6).

adc	adc input
src	mono source
src2	stereo source
filt	standard filter
del	standard delay
fshift	freq shift/ringmod
harm	harmonizer module
gizmo	gizmo spectral domain harmonizer module
disto	tanh~ distortion module
mng	munger module
sfm	mono soundfile
sfst	stereo soundfile
sf4	quad soundfile
sf5	5 channel soundfile (L C R Ls Rs)
smp	mono one-shot sampler
smp4	mono 4 channel spat one-shot sampler with channel 5 to rev
vst	mono vst module
rev	reverb module based on gigaverb
ngate	noise gate
afol	amplitude follower
pt	pitch analysis
cntrd	spectral centroid analysis
zcross	zerocross analysis
delfib	8 tap delay with feedback based on fibonacci
fftfilt	fft filter 253 bands
fftx	fft generalized cross-synthesis - 2 ins 1 out
fftsf	fft source-filter cross-synthesis - 2 ins 1 out
freeze	mono freeze
gran	granular synthesis
reson	model-based resonating filter bank
sndmass	soundmass fm synthesizer
synth	simple polyphonic fm synthesis
csynth	click synthesizer
nsynth	noise synthesizer
rewire	8 tracks rewire mixer
aux	auxilary send/return (mono)
dac	dac output
Subb	dac output for sub

Tableau 1. La liste des modules *tape*.

4.3. La matrice

Une partie de la souplesse de *tape* réside dans sa matrice (éditeur: *mtrx*). Toutes les entrées audio (*adc*),

[10] cf. documentation de *Max* de Cycling74 (www.cycling74.com)
[11] *Ibid*.
[12] cf. site de « open sound control » (http://opensoundcontrol.org)
[13] cf. site de JazzMutant (http://www.jazzmutant.com)
[14] cf. le site du plateforme-virage et la page spécifique à *minuit*. (http://www.plateforme-virage.org/?p=1444)
[15] cf. site de Virage (http://www.virage-platform.org)

les modules de traitement et d'analyse, ainsi que les sorties audios (*dac*) y passent. L'éditeur de la matrice affiche le nom d'entrée, la valeur en dB et le nom de sortie dans chaque case. Dans **Figure 6**, nous voyons que adc.1 est connecté à filt.1 avec une valeur de -5 dB, puis les autres connections sont toutes éteintes, à une valeur de -127 dB.

adc.1 -127 src2.1R	adc.2 -127 src2.1R	src.1 -127 src2.1R
adc.1 -5 filt.1	adc.2 -127 filt.1	src.1 -127 filt.1
adc.1 -127 filt.2	adc.2 -127 filt.2	src.1 -127 filt.2

Figure 6. Vue d'une petite partie de l'éditeur de la matrice

4.4. La spatialisation

Chaque instance de chaque module de la configuration choisie peut accéder à la spatialisation. Ceci est rendu possible grâce au passage de tous les modules par la matrice, qui peuvent ainsi rejoindre les sorties (dacs) et la réverberation. Le calcul des amplitudes envoie automatiquement les bonnes valeurs à la matrice pour effecuer une spatialisation. Pour un exemple d'un éditeur de spat, dans un disositif de 441 (cf. 3.2.2 t_config), regardons dans le spat du module de l'adc (**Figure 7**).

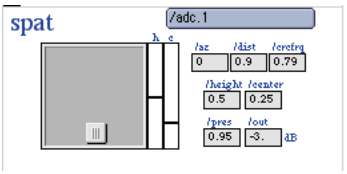

Figure 7. l'éditeur d'un spat441.

Il y a les paramètres azimuth et distance, plus un moteur de rotation (crcfrq = fréquence de circle), avec aussi un paramètre de hauteur (height) pour la quadriphonie haut, et centre pour le haut-parleur de centre. Le paramètre « pres » est pour la présence qui va effectuer le dosage du son direct et du son réverbéré.

4.5. Les modules d'analyse et le contrôle

Parmi les modules de *tape* il y a des modules d'analyse permettant le contrôle de paramètres. Il y a des modules indépendants ainsi que des modules dédiés aux adc. Par exemple, chaque module adc contient un détecteur de niveau (*noise gate*), une détection de hauteur (*pt*), un suivi d'amplitude (*afol*), un suivi du centroide, (*cntrd*), et un zérocross pour la détection du bruit (**Figure 8**).

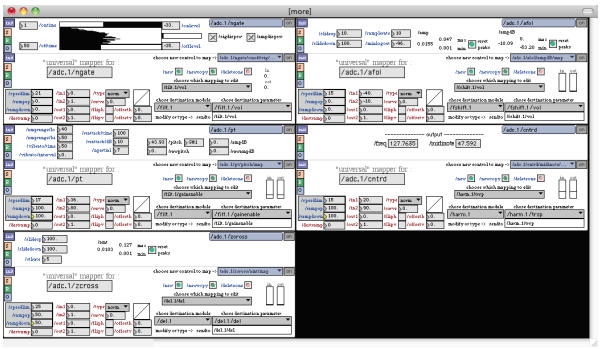

Figure 8. La fenêtre des modules d'analyse audio.

5. LA PARTIE VIDEO ET OPEN GL : *MOVIE*

Si le *plugin movie* a été choisi dans la *config* de tapemovie (cf. 3.2.1), une fois le *build* effectué, une fenêtre *movie* va s'ouvrir (**Figure 9**).

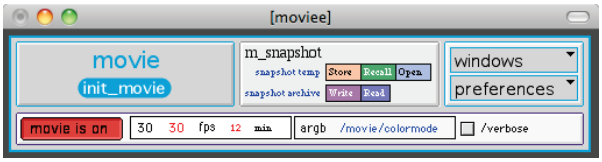

Figure 9. La fenêtre principale de *movie*.

Cette fenêtre ressemble également à la fenêtre *tapemovie*. Elle contient un bouton *init_movie* qui initialise ou construit uniquement la partie *movie*, un *snapshot* de tous les paramètres *movie*, un accès par menu (ou *shortcuts*) de tous les éditeurs *movie*, un menu pour accéder à la configuration de *movie* (*m_config*), un bouton pour allumer/éteindre le moteur des « frames », un affichage du *framerate* actuel, puis un affichage du *colormode*.

5.1. L'architecture

L'architecture est un peu plus complexe que *tape* car la chaîne de traitement vidéo passe entre calcul par *jit.matrix* (cpu) et calcul par *jit.gl.slab*[16] (GPU). Il en résulte que les connections entre les modules se font par *textures* plutôt que via une matrice centrale. A part ce fait, l'architecture de *movie* fonctionne sur les mêmes bases que *tapemovie* ou *tape* permettant plusieurs espaces de rendu. Ainsi nous pouvons diriger les images en fin de chaîne vers de multiple projections ou écrans (cf. 5.4 *output*).

5.2. Les modules vidéo et GL/GPU

Les modules vidéo livré en standard avec *movie* (**Tableau 2**. La liste des modules *movie*.) représentent le même choix d'efficacité que dans *tape*. La

[16] cf. documentation de *Jitter* de Cycling74 (www.cycling74.com)

différence étant que par la nature *gl.slab* (OpenGL) de *movie*, le matriçage se fait par *textures* (cf. 5.3).

mov	jmatrix	qt movie player
cam	jmatrix	live video input
pict	jmatrix	picture player
noiz	jmatrix	noise generator
net	jmatrix	network video input
jrec	jmatrix	record matrix video to fichier Quicktime
rec	gl	record gl video to fichier Quicktime (EN COURS)
jnetsend	jmatrix	output jmatrix video to network (EN COURS)
grid	gl	gridshape module
handle	gl	handle give you position and rotation access of openGL outputs direct on screen with the mouse
boids	gl	boids generation
nurbs	gl	nurbs module
plane	gl	plane for video projection
liner	gl	lines moving with physics
mdl	gl	display 3D models in .obj format
mdlgrp	gl	display 3D models groups
mesh	gl	like a plane with crop/keystone - works with depthbuffer 1
render	gl	render space
jblur	jmatrix	blur effect
jbrcosa	jmatrix	brightness contrast and saturation corrections
jcroper	jmatrix	crop
jcrop	jmatrix	crop
jframediff	jmatrix	absolute frame difference
jrota	jmatrix	rotation correction
jslide	jmatrix	slide effect
jwake	jmatrix	wake effect
brcosa	gl	brightness contrast and saturation corrections
blur	gl	blur effect
buf	gl	dynamic video delay buffer
chromakey	gl	chroma keyer
composite	gl	compositing
crop	gl	crop and zoom
dilate	gl	dilate effect
emboss	gl	emboss effect
gaussian	gl	gaussian blur
keystone	gl	projection mapping
lumadisplace	gl	lumadisplace effect
mix	gl	binaries operators
outline	gl	outline maker
radial	gl	distorsion radial
rota	gl	rotation corrtrction
sharpen	gl	sharpen effect
slide	gl	slide effect
texture	gl	texture
tllslo	gl	tllslo special module - a little bit like a channel
mask	gl	threshold maker
wobble	gl	wobble effect

Tableau 2. La liste des modules *movie*.

5.3. La matriçage par texture

Il n'y a pas de représentation globale des connections entre les modules. En revanche, chaque éditeur de module de traitement movie contient un menu permettant de choisir une texture comme entrée (**Figure 10**). Nous créons ainsi l'enchaînement de modules nécessaire en choisissant la source comme paramètre pour chaque module.

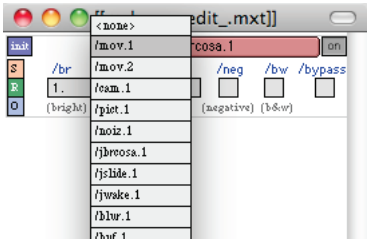

Figure 10. Le menu *texture* dans le module *brcosa*

5.4. Output

Le module de l'espace de rendu s'appelle render et il crée un espace de rendu (render) dans une sortie (out) qui est la fenêtre de sortie de l'image que nous pouvons placer sur un écran ou dans une vidéoprojection. (**Figure 11**).

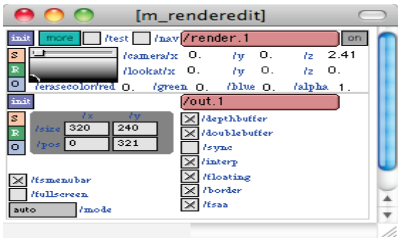

Figure 11. L'éditeur du module *render* + *out*.

6. LE CONTRÔLE ET LES *MAPPEURS*

Tous les *contrôleurs*, qu'ils viennent des entrées OSC ou MIDI, ou bien depuis les analyses internes (comme les suivi de hauteur ou d'amplitude se trouvant dans les *adc*) ont la possibilité de faire un « multi-mapping » (*one to many*) pour chaque paramètre de contrôle. Ces *mappeurs* ont été conçus et éprouvés par la pratique.

Ils sont situés à deux niveaux. Le premier est au niveau des modules *tapemovie*. Ce sont les *mappeurs centraux* et nous en choisissons le nombre dans la configuration pour *tapemovie* (**Figure 12**). Dans ces *mappeurs centraux* nous déterminons un contrôleur d'entrée (à *mapper*) puis un paramètre de destination. Dans les deux cas nous avons l'aide de menus listant les modules de contrôle ou de traitement disponible, puis un sous-menu avec les contrôleurs ou paramètres de ce module.

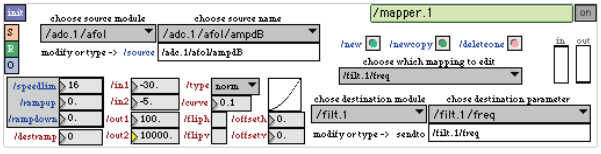

Figure 12. L'éditeur d'un *mappeur* central

Nous réglons évidemment l'échelle d'entrée et de sortie de chaque *mapping* mais également les choix de courbes entre exponentiel-linéaire-logarithmique,

symétrique autour de zéro et bosse/creux (*up/down*). Des fonctions de *invert* et *offset* permettent de rapidement configurer la courbe selon l'échelle.

Quand nous avons un *mapping* il suffit de cliquer sur *new* ou *new copy* pour en créer un autre avec le même contrôleur – vers une destination différente. En principe il n'y a pas de limite au nombre de *multi-mappings*.

Le deuxième niveau des *mappeurs* est le niveau asocié aux contrôleurs directement. Dans chaque éditeur de contrôleur ou module d'analyse, il y a un éditeur unique de *mappeur* permettant de choisir dans un menu le contrôle spécifique du module en question. Le réglages de ces *mappeurs* se font de la même manière que les *mappeurs centraux*.

7. L'EVENT MANAGER

Pour stocker des états de paramètres en créant et en organisant des événements, on choisi au minimum une instance du module « évents » dans le *tm_config*. Ce module permet de créer un événement du groupe *tapemovie* (tous les paramètres), du groupe *tape* (paramètres d'audio) ou du groupe *movie* (paramètres du traitmement de l'image). Les « events » sont stockés dans le dossier *events* du projet en tant que fichier texte, mais on peut aussi créer des sous-dossiers pour les différencier. Une fois créé, on peut faire « play », « edit », « rename » ou « delete » d'un événement. (**Figure 13**).

Pour en exécuter un, on le sélectionne dans le menu *play*. A ce moment, le fichier de l'événement sera lu depuis le disque. Puisqu'ils sont relus avant chaque lecture, on peut les modifier avec n'importe éditeur de texte à l'extérieur de tapemovie, et l'event manager va relire la version modifié automatiquement depuis le disque.

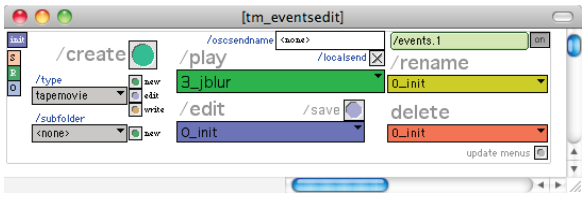

Figure 13. l'éditeur des *events*.

8. LA PÉRENNITE DES CRÉATIONS

Une des questions la plus importante dans la création avec les outils informatiques temps réel est comment rejouer l'oeuvre dans le futur. Les pièces du répertoire doivent subir des réactualisations (portages) régulièrement, sinon elles risquent de ne plus pouvoir être jouées à cause des supports de stockage, l'obsolescence des standards techniques, l'absence de soin dans la préservation des parties électroniques des œuvres, les conditions pratiques de la performance, et la confection souvent empirique des programmes [4].

Tapemovie aborde le sujet de la pérennité des créations par son recours aux fichiers texte qui définisent les événements. Chaque événement est une description d'un état ou une action incluant tous les paramètres nécessaires à son exécution. Ces fichiers texte sont lisibles en dehors de l'application par un éditeur de texte.

Si on combine la lisibilité des événements avec la documentation de l'environnement de *tapemovie* on peut dire que dans plusieurs années il serait encore possible d'étudier la documentation de tapemovie puis déchifrer les configurations et événements d'un *project* pour, avec les fichiers sons, re-créer l'oeuvre dans un environnement futur – sans jamais avoir à faire tourner *tapemovie*.

9. CONCLUSIONS

Tapemovie est utilisé dans la composition, l'improvisation et le spectacle vivant. Il peut servir à la fois le créateur individuel et toute une équipe avec des ordinateurs en réseau. Il a une capacité d'extension par sa nature open-source et son utilisation par différentes équipes dans des spécialités variées. En servant une pluralité de projets, tapemovie s'augmente aussi en terme de configurations possibles.

Les réflexions entre les développeurs ont commençé pour les futures versions crées en Max 5. La prochaine est en préparation en collaboration avec les développeurs de Jamoma[17]

Tapemovie est un environnement de création qui tend la main vers une utilisation intuitive, tout en gardant la spécificité d'un outil expert, solide et ouvert.

10. RÉFÉRENCES

[1] Manning, Peter, Electronic and Computer Music, Oxford University Press, New York, 2004, p. 361-374

[2] Reenskaug, Trygve, 2003, The Model-View-Controller (MVC) Its Past and Present, article publié par INTEGRATED DOMAIN SERVICES, http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf

[3] Le site du *Laboratoire d'Informatique Algorithmique*, la page sur l'architecture MVC : (http://www.liafa.jussieu.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/mvc.html)

[4] Alain Bonardi et Jérôme Barthélemy, 2008, «Le patch comme document numérique : support de création et de constitution de connaissances pour les arts de la performance», CIDE [En ligne], CIDE 10, Session Document culturel, mis à jour le : 16/09/2008, URL : http://172.16.128.67:50010/cide/index.php?id=298

[17] cf. site officiel Jamoma (www.jamoma.org)