PdTl

Une TimeLine pour Pure Data

28 avril 2014
mis à jour 15 déc 2015

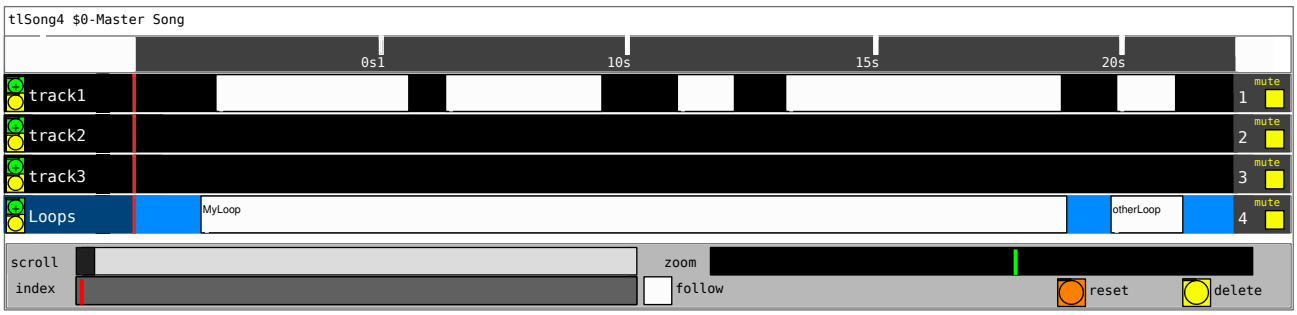


Table des matières

1) Introduction

PdTl est un éditeur temporel multipiste ; elle permet d'organiser dans le temps et sur plusieurs lignes l'apparition et la disparition d'événements.

Les événements ont pour uniques attributs leur durée et leur position dans le temps (et optionnellement un nom). Ces attributs sont édités et visualisés dans PdTl d'une manière entièrement graphique : le déplacement dans le temps ou la modification de la durée d'un événement s'opère en l'attrapant à la souris.

Les contrôleurs *zoom* et *scroll* permettent de se déplacer finement sur l'échelle du temps, offrant alternativement une vue d'ensemble et une grande précision temporelle.

Un ensemble d'objets (basés sur la bibliothèque **AutoPreset**) assure ensuite l'association entre une piste d'événements et les valeurs d'un groupe de paramètres ; ainsi chaque événement de cette piste mémorisera la valeur de chaque paramètre du groupe. Il est également possible d'interpoler la valeur d'un paramètre tout au long de la durée de l'événement.

L'écriture de **PdTl** par Antoine Rousseau a commencé en 2012, lors de la réalisation de l'installation « DeadEnd » de Théodora Barat, alors en formation au Fresnoy (Studio National des Arts Contemporains, Tourcoing).

2) Prérequis

PdTl est entièrement écrite en Pure Data (elle est constituée uniquement d'*abstractions*). Elle nécessite l'installation de **Pd** ou de **Pd-extended**.

Les bibliothèques qu'elle utilise sont intégrées dans Pd-extended : **zexy, hcs, iemlib, ggee, moonlib** (cette dernière n'est toutefois pas déclarée en standard, et demande soit une modification des préférences de Pd-extended, soit l'ajout d'un [import moonlib] dans le *patch*). Pour Pd il faudra les installer manuellement (se reporter à la documentation de Pd).

PdTl utilise également la bibliothèque **AutoPreset**, il est donc nécessaire d'ajouter le chemin de cette bibliothèque, soit dans les préférences générales de Pure Data, soit *via* l'ajout d'un : [declare -path path/to/AutoPreset] dans le *patch*.

3) Éditeur

Les fonctions de l'éditeur peuvent être comprises selon les trois niveaux d'organisation de la timeline : la ***Song*** représente l'ensemble de la composition. Elle est composées de pistes nommées ***Tracks***, comprenant elles-mêmes des événements (***Events***).

a) Song

- Le contrôle *scroll* : il fait glisser la fenêtre d'observation de la composition dans le temps. Un double-clic permet de sauter à une position arbitraire.
- Le contrôle *zoom* : il change le facteur d'échelle de la fenêtre d'observation.
- Le contrôle *index* : il déplace le curseur (qui représente la tête de lecture). Les événements sont émis lors de ce déplacement. Un double-clic permet de faire sauter l'index à une position arbitraire.
- Le bouton *reset* : un double-clic sur ce bouton efface tous les événements de la composition.
- Le bouton *delete* : efface le ou les événements sélectionnés.
- L'option *follow* : automatise le contrôle *scroll* de façon à garder l'index dans la fenêtre d'observation.

b) Track

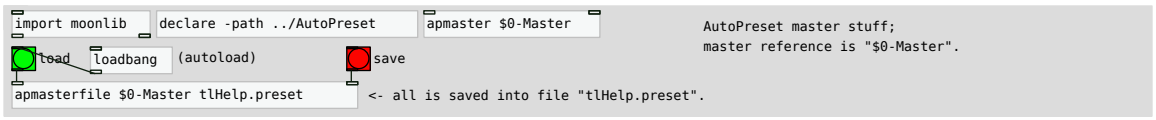
- Le bouton vert + (à gauche) : ajoute un événement à la piste, à la position de l'index.
- Le bouton jaune (à gauche) : ouvre la fenêtre d'édition des propriétés de la piste, permettant de configurer la couleur et le nom de la piste.
- L'option *mute* (à droite) : désactive l'émission d'événement pour cette piste.

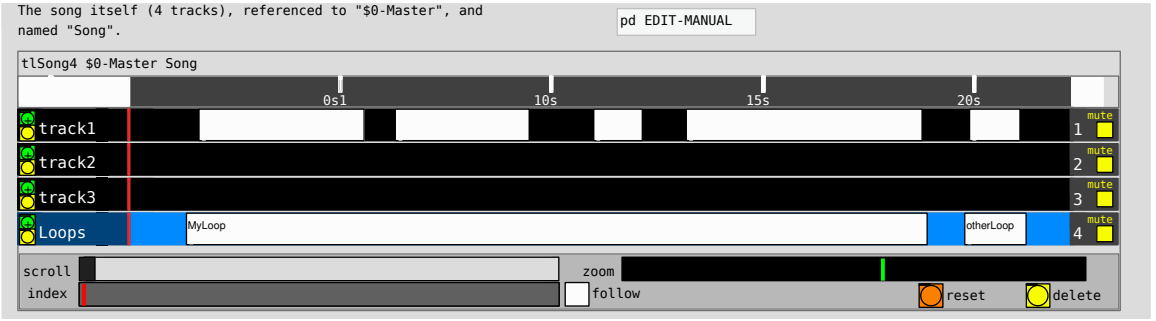
c) Event

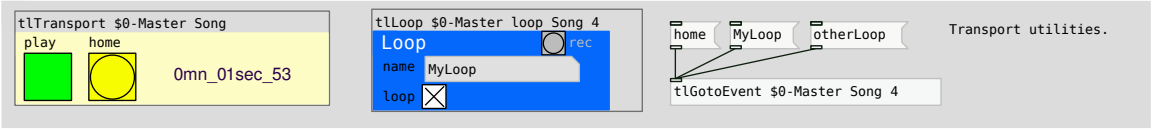
- Un clic sur un événement le sélectionne. Si *Shift* est appuyé l'événement est ajouté à la sélection.
- Pour déplacer un événement, il faut le tirer en l'attrapant par son milieu.
- Pour changer la durée d'un événement, il faut le tirer en l'attrapant par l'une de ses extrémités.

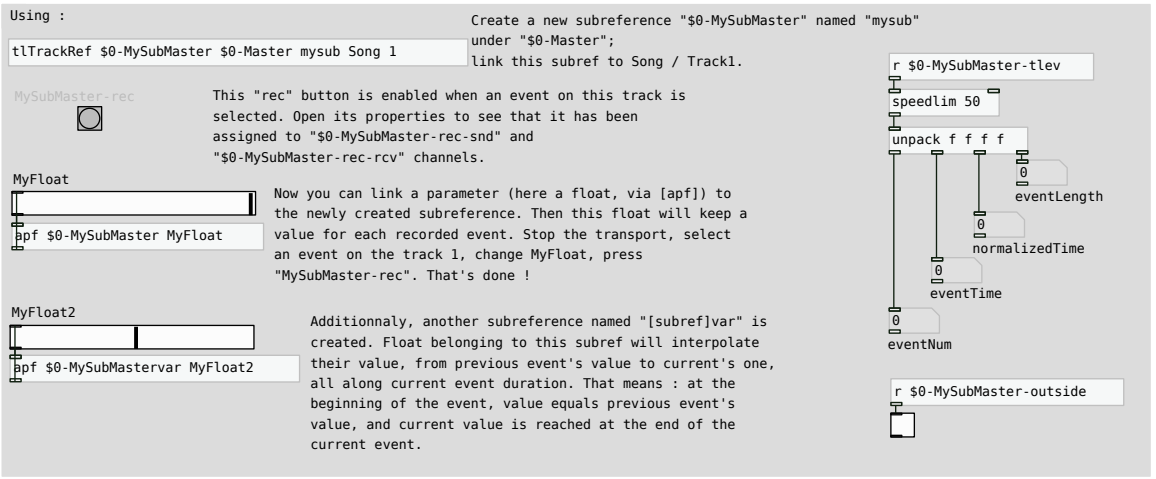
4) Utilisation

L'utilisation de **PdTl** est décrite dans le patch 0tlHelp.pd inclus dans la distribution de PdTl :










```
                    PdTl edition manual :


SONG :
     -scroll : translate the view of the song;
     double-click to jump to an arbitrary time.

     -index : move the time index (reading head) along the song;
     double-click to jump to an arbitrary time.

     -reset (double-click) : delete all events.

     -delete : delete selected events.

     -follow : auto-scroll to keep the index at the center of
     the screen.


TRACK :
     -green "+" button (left): add an event, starting at index
     time.
     -yellow button (left): edit track properties : name and
     color.
     -yellow mute toggle(right): stop this track to emit events.


EVENTS :
     -click on an event to select it;
     shift-click to add an event to selection.

     -grab an event by its middle to move it;
     works for the whole events selection.

     -grab an event by one of its end to resize it.
```

a) Mise en place générale

Après s'être assuré d'avoir bien déclaré toutes les bibliothèques et chemins nécessaires, on commence par définir un référentiel maître pour la sauvegarde des paramètres, avec l'objet [apmaster Reference] ; l'objet [apmasterfile Reference fichier] permet de simplifier les opérations de sauvegarde et de chargement du fichier de paramètres.

On ajoute ensuite l'objet représentant la timeline elle-même, en choisissant le nombre de pistes et le nom qu'elle porte dans le référentiel ; par exemple pour 4 pistes, on utilisera : [tlSong4 Reference nomSong] .

L'objet [tlTransport Reference nomSong] sert à lancer la lecture de la composition.

L'objet [tlLoop Reference nom nomSong numTrack] sert à lire en boucle la composition sur un segment de temps défini par l'événement d'une piste (nom est ici l'identifiant de l'objet dans le référentiel maître ; il sera utilisé dans le fichier de sauvegarde. Tous les objets d'un référentiel doivent avoir un nom unique, donc nomSong et nom, par exemple, doivent être différents).

b) Exploitation des données

Pour exploiter les données d'événements fournies par la timeline, il est possible soit d'utiliser les données brutes (fournies par le bus Reference-song-nomSong-snd), soit d'utiliser l'objet :

[tlTrackRef sousReference Reference nom nomSong numTrack]

Cet objet crée un nouveau sous-référentiel sousReference, identifié nom dans Reference (attention ici encore à l'unicité de nom).

Tous les paramètres référencés dans ce sous-référentiel seront capables de mémoriser leur valeur pour chaque événement de la piste choisie. Les paramètres de type flottant référencés dans sousReferencevar (ajouter « var » à la fin de la sous-référence) verront leur valeur passer progressivement de la valeur de l'événement précédent à celle de l'événement courant pendant toute la durée de l'événement.

c) Créations de « blocs » fonctionnels

Il est par suite possible d'écrire des « blocs » fonctionnels spécifiques à une application donnée, sous la forme d'une *abstraction*, que l'on instanciera ensuite à l'une des pistes d'une *Song* donnée. L'objet [tlLoop] peut fournir un bon exemple en cela.

On peut imaginer d'écrire des blocs de lecture de fichier audio ou de fichier video (pour afficher dans Gem), des effets audio/vidéo, des contrôles de machines, etc...

5) Limitations

La charge imposée au CPU est assez élevée, du fait de l'absence de possibilité d'optimiser l'affichage graphique (le code se limitant au niveau *patch*).

Pour cette raison, le nombre total d'événements est actuellement limité à 50 par piste, ce qui est relativement faible mais suffisant pour des petits projets.

Un travail reste à faire pour pouvoir configurer facilement cette limite de 50 (actuellement cette limite est présente à plusieurs endroits du code, qui ne sont pas clairement répertoriés).

Des limitations sérieuses quant à un développement ultérieur sont imposées par celles de l'interface graphique de Pure Data, elle-même écrite en TclTk. Afficher et contrôler efficacement des courbes de contrôle, des formes d'ondes audio, des images clefs d'un fichier video, tout cela semble relativement inaccessible.