Architecture interne de FID_ABS (v 0.3.5)

Conventions de notation

fid_class Objet / abstraction / external dans pure data

[class_name] Paramètre d'un objet

Structures de données dans **pool**

<sid>	Dossier
coords	Clé
x y	Valeur
float	Type de donnée

Structures de stockage des données dans fid_abs

L'architecture du projet FID ABS repose essentiellement sur l'objet **pool**, développé dans l'environnement multiplateforme flext par Thomas Grill.
Cet objet permet d'organiser des informations (*float*, *symbol*, *list*) de manière hiérarchique, à l'aide de dossiers.

La pierre angulaire du système est la structure **pool fid_abs**, dans laquelle sont stockées – pendant leur durée de vie – les informations sur chaque fiducial : position, angle, vitesse, accélération, numéro de fiducial, ainsi que d'autres informations, qui peuvent aussi être définies par l'utilisateur en fonction de ses besoins.
Chaque fiducial se voit affecter un identifiant unique (*sid*), qui est comme un clé permettant d'accéder aux informations qui lui sont associées.

Outre le nom de structure **pool fid_abs**, d'autres espaces de stockage sont réservés :

- Chaque objet **fid_preprocess** contient deux structures utilisées en interne, **pool $0-preprocess** et **pool $0-vault**. La première assure le mapping entre les numéros d'instance fournis par ReacTivision ou autre et les identifiants uniques utilisés par le système (sid). La seconde permet de temporiser la destruction d'une instance de sid, ce qui permet par exemple de gérer les occultations passagères d'un fiducial, ou encore de créer un effet de persistance de la représentation à l'écran d'un fiducial, même après que celui-ci ait été retiré de la surface tangible.
- Les objets **fid_class *[class_name]*** possèdent une structure de stockage publique, **pool *[class_name]***, qui permet d'obtenir rapidement la liste de toutes les instances associées à une classe.

Espaces de noms réservés

value fid_abs.current_sid	permet de connaître quel est la prochaine valeur de sid libre

- Structure **pool fid_abs**

<sid> **x** *float*
y *float*
alpha *float*
Vx *float*
Vy *float*
Valpha *float*
accel *float*
raccel *float*
coords *float float*
vitesse *float float*
alpha *float*
fid *float*
life_time *float*

etc....

- structure **pool fid_class *[class_name]***

pool class_name
sid1 sid1
sid2 sid2
sid3 sid3

Gestion des liens

Les liens (link) permettent de définir des relations / interactions entre objets/instances différentes.

Remarque
Jusqu'à la version 0.2.5.3, les liens entre objets divers n'étaient gérés que pour les classes dynamiques. A présent (v >= 0.3.5), n'importe quelle instance (sid) peut supporter des liens.

Les informations de liaison sont stockées dans la structure de données **pool fid_abs**, pour chacun des deux <sid> concernés, dans un sous-dossier <link>.
Si un sid est supprimé, on explore son dossier <link> et on supprime les lignes correspondantes pour chacun des sid rencontrés.

<sid0><link>	<sid1>	**tag_A**	→
	<sid2>	**tag_B**	↔
<sid1><link>	<sid0>	**tag_A**	←
<sid2><link>	<sid0>	**tag_B**	↔

Les flèches renseignent sur la nature du lien :

directionnel	→
directionnel	←
symétrique	↔

EN PROJET :
On peut spécifier aussi un type de lien spécial, <=, =>, <=>, qui définit des relations parent/enfant
Dans ce cas, si un sid parent est supprimé, tous les sid enfants le seront également (uniquement s'ils ont été générés par des classes dynamiques)

Les tags sont définis par l'utilisateur, ils permettent de créer différentes 'classes' de liens.

Un objet « lecteur de liaison » (**fid_get_link**) renvoie, pour un sid spécifié, tous les sid associés par lien, avec la possibilité de filtrer par le type ou le nom de la liaison.

Les objets **fid_link** et **fid_unlink** permettent de créer et de supprimer des liens.
Ces deux objets sont utilisables comme modules indépendants, mais sont aussi invoqués par **fid_preprocess** et **fid_dynamic_class** lors de la création/ destruction d'instances.