Spécifications pour les fichiers .xml instrumenpoche

(version du 11 septembre 2005)

Table des matières

Introduction

Cet article explique la façon dont sont enregistrées les figures Instrumenpoche. Il s'agit de fichiers XML.
Pour une introduction au format XML, voir Wikipedia , Comment ça marche ? ou SelfHTML.

indique la façon dont Instrumenpoche enregistre la figure.

explique comment on peut obtenir des fonctionnalités supplémentaires en modifiant le fichier .xml avec un éditeur de texte.

Architecture

Chaque figure Instrumenpoche est écrite dans un fichier .xml. L'élément racine est une balise `<INSTRUMENPOCHE></INSTRUMENPOCHE>`. Ses enfants sont des éléments `<action/>` avec divers attributs définis ci-dessous. Chaque noeud `<action/>` désigne une action a effectuer et doit comporter tous les paramètres nécessaires sous forme d'attributs prédéterminés. Les noeuds sont lus et exécutés dans l'ordre de leur écriture dans le fichier.

Attributs communs

Deux attributs sont communs à tous les noeuds `<action/>` :

- **L'attribut *`objet`* indique l'objet sur lequel porte l'action.**

Exemple : `<action objet="compas"/>` désigne une action portant sur le compas virtuel.

Voici la liste des valeurs que peut prendre l'attribut `objet` :

- `angle_droit` pour un codage d'angle droit
- `crayon`
- `equerre`
- `image`
- `compas`
- `longueur` pour un codage de longueur
- `point`
- `rapporteur`
- `regle`
- `repere`
- `requerre` pour la règle-équerre
- `scene` pour tout ce qui concerne la zone de dessin dans son ensemble
- `texte`
- `trait`

- **L'attribut *`mouvement`* indique l'action à effectuer.**

Exemple :`<action mouvement="montrer"/>` désigne une action faisant apparaître un instrument virtuel sur la scène.

NB : Ces attributs n'apparaissent jamais seuls. Ainsi, pour faire apparaitre le compas sur la scène, il faut coder : `<action mouvement="montrer" objet="compas"/>`. Peu importe l'ordre dans lequel sont codés les attributs dans une balise.

- XML **L'attribut *`tempo`* permet de faire une pause** automatique dans la lecture de l'animation, juste après l'action indiquée par le noeud `<action />` dans lequel est codé cet attribut. Sa valeur est le temps que dure la pause, en dixièmes de seconde.

Exemple : `<action mouvement="montrer" objet="compas" tempo="20"/>` fait apparaître le compas sur la scène. S'ensuit une pause, l'action suivante aura lieu deux secondes plus tard.

Valeurs de mouvement communes à tous les instruments virtuels

Les valeurs suivantes sont utilisées par les instruments virtuels, c'est-à-dire les valeurs de objet égales à "compas", "crayon", "equerre", "requerre" et "rapporteur".

- mouvement="montrer" : Cette valeur permet de faire apparaître un instrument de la scène.
- mouvement="masquer" permet de faire disparaître un instrument de la scène.
  Exemple : <action mouvement="montrer" objet="crayon"/> fait apparaître le crayon virtuel.
- mouvement="translation" indique un glissement de l'instrument depuis sa position actuelle. Il est valable également pour les zones de texte (objet="texte") et les points (objet="point"). Pour un tel mouvement, il est nécessaire de donner exactement deux paramètres : abscisse et ordonnee, qui indiquent respectivement l'abscisse et l'ordonnée du point d'arrivée de l'objet.
  NB : Les coordonnées sont repérées par rapport au coin supérieur gauche de la scène. Les ordonnées positives sont vers le bas. L'unité est le pixel.
  Exemple : <action mouvement="translation" objet="compas" abscisse="500" ordonnee="300" /> envoie le compas au point de coordonnées (500 ;300).
- mouvement="rotation" fait pivoter l'instrument depuis sa position actuelle. Pour un tel mouvement, il est nécessaire de donner exactement deux paramètres : angle et sens, qui indiquent d'une part l'angle final de la rotation (en degrés) et d'autre part la vitesse et le sens de cette rotation. sens est positif pour une rotation dans le sens trigonométrique et négatif pour une rotation dans le sens des aiguilles d'une montre. La valeur absolue de sens indique la vitesse angulaire, exprimée en degrés par dixièmes de seconde.
  Exemple : <action mouvement="rotation" objet="rapporteur" sens="3" angle="50" /> fait pivoter le rapporteur jusqu'à ce qu'il fasse un angle de 50 degrés par rapport à l'horizontale.

Instrumenpoche enregistre toujours des rotations avec une valeur de sens égale à 5 ou -5.

On peut accélérer la rotation en modifiant la valeur de sens. Par exemple, sens="360" fait pivoter l'objet immédiatement, sans animation.

Liste des objets d'InstrumenPoche et codage correspondant

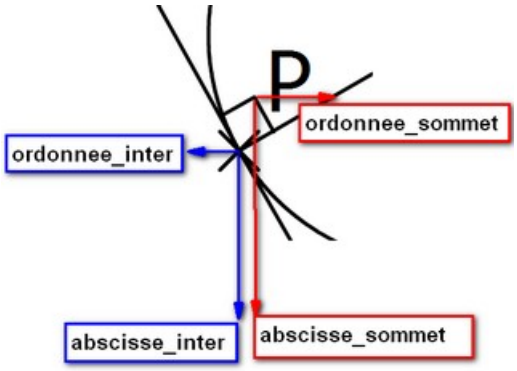
La suite de ce document présente les valeurs de `objet` par ordre alphabétique et, pour chaque `objet`, les différentes valeurs pouvant être prises par l'attribut `mouvement`.

L'objet `angle_droit`

Cet objet n'admet que la valeur mouvement="tracer", avec nécessairement les attributs suivants :

- `couleur` pour la couleur du tracé
- `epaisseur` pour l'épaisseur du trait (en pixels).
- `abscisse_inter` et `ordonnee_inter` pour les coordonnées du point d'intersection des segments dessinant le codage.
- `abscisse_sommet` et `ordonnee_sommet` pour les coordonnées du sommet de l'angle droit à coder.

On peut considérer que le codage d'un angle droit est un demi-carré. `abscisse_inter` et `ordonnee_inter` désignent les coordonnées d'un sommet du carré (le point d'intersection des deux droites perpendiculaires que l'on veut coder), `abscisse_sommet` et `ordonnee_sommet` les coordonnées du sommet du carré opposé à ce point.
Voir le schéma :



L'objet `crayon`

Outre les attributs `mouvement="montrer"`, `mouvement="masquer"`, `mouvement="rotation"` et `mouvement="translation"` décrits plus haut, cet objet admet également : `mouvement="tracer"` qui permet de tracer un segment depuis la position actuelle du crayon, avec les attributs suivants :

- `abscisse` et `ordonnee` qui indiquent les coordonnées d'arrivée du crayon, et donc l'extrémité du segment tracé.
- `couleur` est un nombre entier indiquant la couleur du tracé.
- `epaisseur` est un nombre indiquant l'épaisseur du trait, en pixels.
- `style` peut prendre deux valeurs :
  → `style="segment"` pour un trait simple
  → `style="vecteur"` pour un trait terminé par une flèche
  Si `style` est omis, le lecteur d'animations trace un trait simple.
- `pointille` peut prendre deux valeurs :
  → `pointille="non"` pour un trait continu
  → `pointille="tiret"` pour un trait en pointillés : ---
- Par défaut, la valeur de `pointille` est considérée comme étant `pointille="non"`.

L'objet `equerre`

Outre les attributs communs à tous les instruments, décrits plus haut, cet objet admet également : `mouvement="zoom"` qui permet de modifier la taille de l'équerre virtuelle.
Dans ce cas, il faut l'attribut `echelle` qui prend comme valeur un nombre proportionnel à la taille de l'équerre. `echelle="100"` code la taille normale de l'équerre.

L'objet `image`

Cet objet admet une seule valeur de `mouvement` : `mouvement="chargement"`. Son seul autre attribut est `url`, qui prend pour valeur le nom de l'image à charger en arrière-plan ou une url où le logiciel de lecture peut la trouver. Seules les images au format .jpg ou .swf peuvent être chargées

XML L'attribut facultatif `echelle` permet de modifier la taille de l'image, la valeur `echelle="100"` correspondant à la taille d'origine.
Exemple : `<action echelle="50" url="monimage.jpg" objet="image"/>` permet de charger l'image "monimage.jpg" avec une réduction de 50%.

L'objet `compas`

Outre les attributs communs à tous les instruments décrits plus haut, cet objet admet également :

- `mouvement="tracer"` trace un arc de cercle dont le centre est placé à l'extrémité de la pointe du compas. Il nécessite les attributs suivants :
  - → `couleur` est un nombre entier indiquant la couleur du tracé.
  - → `epaisseur` est un nombre indiquant l'épaisseur du trait, en pixels.
  - → `debut` donne l'angle de départ de l'arc de cercle par rapport à l'horizontale.
  - → `sens` donne le sens de rotation et la vitesse, sur le modèle de `sens` pour `mouvement="rotation"` décrit ci-dessus.
  - → `fin` donne l'angle d'arrivée de l'arc de cercle par rapport à l'horizontale.
- `mouvement="lever"` permet de mettre le compas en position "verticale" (sa position de tracé).
- `mouvement="coucher"` permet de mettre le compas en position "horizontale" (la position qui permet de le faire glisser, pivoter, de modifier son écartement sans tracer).
- `mouvement="ecarter"` permet de modifier l'écartement du compas. Il faut alors préciser `ecart`, qui prend pour valeur l'écartement final du compas, en pixels (1cm virtuel correspond à 30 pixels).

L'objet `longueur`

Cet objet admet les valeurs de `mouvement` suivantes :

- `mouvement="creer"` est nécessaire pour déclarer un codage de longueur, avant toute autre action sur ce codage. Une telle action nécessite les attributs :
  - → `id` qui doit être un identifiant unique pour chaque codage. C'est par cet identifiant que le logiciel de lecture reconnaitra le codage.
  - → `couleur` indique la couleur du codage.
  - → `epaisseur` indique l'épaisseur du trait du codage.
  - → `forme` indique la forme approximative du trait du codage. Les valeurs possibles sont : `forme="/"`, `forme="\"`, `forme="//"`, `forme="\\"`, `forme="///"`, `forme="\\\"`.

- `mouvement="translation"` sur le même modèle que les instruments de géométrie.

L'objet `point`

Cet objet admet les valeurs de `mouvement` suivantes :

- `mouvement="creer"` est nécessaire pour déclarer un point, avant toute autre action sur ce point. Une telle action nécessite les attributs :
  - → `id` qui doitêtre un identifiant unique pour chaque point. C'est par cet identifiant que le logiciel de lecture reconnaitra le point.
  - → `couleur` indique la couleur de la croix représentant le point.
- `mouvement="translation"` sur le même modèle que les instruments de géométrie
- `mouvement="nommer"` qui permet de donner un nom attaché au point. Il faut alors coder les attributs `nom` (qui prend comme valeur le nom donné au point), `couleur` (qui prend pour valeur un nombre donnant la couleur du texte qui indique le nom du point) et `id`, identifiant unique du point considéré.
  Exemple : `<action id = "12" couleur = "0" nom = "A'" mouvement="nommer" />` écrit A' en noir à côté de la croix symbolisant le point dont l'identifiant est égal à "12" (ce point doit d'abord être déclaré avec une action contenant `mouvement` = "creer").

L'objet `rapporteur`

Outre les attributs `mouvement="montrer"`, `mouvement="rotation"` et `mouvement="translation"` décrits plus haut, cet objet admet également :

- `mouvement="zoom"` permet de faire grandir ou rapetissir le rapprorteur virtuel. Dans ce cas, il faut l'attribut `echelle` qui prend comme valeur un nombre proportionnel à la taille du rapporteur. `echelle="100"` code la taille normale du rapporteur.
- `mouvement="circul"` permet de passer du rapporteur semi-circulaire au rapporteur circulaire et inversement. Par défaut, le rapporteur est semi-circulaire.
- `mouvement="double"` permet de passer de la graduation simple à la graduation à double sens et inversement. Par défaut, la graduation est simple.

L'objet `regle`

Cet objet admet les attributs communs à tous les instruments décrits plus haut.

L'objet `repere`

Cet objet n'est pas encore implémenté.

L'objet `requerre`

Cet objet admet les attributs communs à tous les instruments décrits plus haut.
En outre, il admet le `mouvement="glisser"` qui permet de faire glisser l'équerre le long de la règle. Dans ce cas, il faut coder `abscisse` qui indique la position de l'équerre par rapport à la règle.

L'objet `scene`

Cet objet admet deux valeurs pour l'attribut `mouvement` :

- `mouvement="translation"` dont les attributs associés sont `var1` et `var2` qui admettent comme valeurs respectives le décalage de la scène par rapport à sa position initiale, en abscisse et en ordonnée.
  Exemple : `<action var1="-50" var2="60" mouvement="translation" objet="scene"/>` décale la scène de 50 pixels vers la gauche et de 60 pixels vers le bas
- `mouvement="zoom"` pour modifier la taille de la zone de dessin. Il faut alors coder `var1`, qui indique l'agrandissement. `var1="100"` donne la taille normale dela scène.

L'objet `texte`

Cet objet admet les valeurs de `mouvement` suivantes :

- `mouvement="creer"` est nécessaire pour déclarer une zone de texte, avant toute autre action sur cette zone. Une telle action nécessite les attributs :
  - → `id` qui doit être un identifiant unique pour chaque zone. C'est par cet identifiant que le logiciel de lecture reconnaitra la zone de texte.
  - → `couleur` indique la couleur du texte.
- `mouvement="translation"` sur le même modèle que les instruments de géométrie.
- `mouvement="ecrire"` qui permet de changer le texte contenu dans la zone. Il faut alors coder `texte` qui prend pour valeur le nouveau texte. On peut aussi coder `couleur` et `taille` (taille de la police de caractère).
- `mouvement="dimension"` qui permet de modifier la taille de la zone de texte. Il faut alors coder `largeur` et `hauteur` qui prennent pour valeurs les nouvelles dimensions de la zone, exprimées en pixels.

XML On peut mettre le texte en forme grâce à des balises HTML. Les balises prises en charge sont : `<B>` (gras), `<I>` (italique), `<U>` (souligné), `<FONT>` (police de cacatères, prend en charge les attributs `SIZE`, `COLOR`, et `FACE`), `<P>` (paragraphe), `<BR>` (retour à la ligne), `<A>` (lien hypertexte) et `<LI>`(liste, un seul niveau, sans qu'une balise `<UL>` soit obligatoire). Les valeurs d'attributs doivent être entre guillemets. Les caractètes < et > doivent être remplacés respectivement par `&lt;` et `&gt;` pour créer des balises.

L'objet `trait`

Cet objet admet les attributs `abscisse1`, `ordonnee1`, `abscisse2` et `ordonnee2` qui indiquent respectivement les coordonnées de l'origine et de l'extrémité du trait.
Pour le reste, son fonctionnement et ses attributs sont les mêmes que `objet="crayon" mouvement="tracer"`.

ANNEXE 1 : les couleurs dans Instrumenpoche

Ajout du 31 juillet 2005

- Une couleur peut être codée sous forme de nombre hexadécimal Rouge-Vert-Bleu sous la forme `couleur = "0xRRVVBB"` ou avec l'écriture décimale de ce nombre.
  Exemple : Le noir se code `couleur = "0x000000"` ou `couleur = "0"`, le rouge `couleur = "0xFF0000"`.

- D'autres noms, francisés, sont également disponibles : `noir`, `blanc`, `rouge`, `vert` et `bleu`.

- Depuis la version 1.0.5 du lecteur d'animations, on peut aussi donner le nom HTML d'une couleur comme indiqué dans le tableau situé ci-dessous.

aliceblue	darkred	lemonchiffon	
antiquewhite	darksalmon	lightblue	
aqua	darkseagreen	lightcoral	
aquamarine	darkslateblue	lightcyan	
azure	darkslategray	lightgoldenrodyellow	
beige	darkturquoise	lightgreen	
bisque	darkviolet	lightgrey	
black	deeppink	lightpink	
blanchedalmond	deepskyblue	lightsalmon	
blue	dimgray	lightseagreen	
blueviolet	dodgerblue	lightskyblue	
brown	firebrick	lightslategray	
burlywood	floralwhite	lightsteelblue	
cadetblue	forestgreen	lightyellow	
chartreuse	fuchsia	lime	
chocolate	gainsboro	limegreen	
coral	ghostwhite	linen	
cornflowerblue	gold	magenta	
cornsilk	goldenrod	maroon	
crimson	gray	mediumaquamarine	
cyan	green	mediumblue	
darkblue	greenyellow	mediumorchid	
darkcyan	honeydew	mediumpurple	
darkgoldenrod	hotpink	mediumseagreen	
darkgray	indianred	mediumslateblue	
darkgreen	indigo	mediumspringgreen	
darkkhaki	ivory	mediumturquoise	
darkmagenta	khaki	mediumvioletred	
darkolivegreen	lavender	midnightblue	
darkorange	lavenderblush	mintcream	
darkorchid	lawngreen		

mistyrose		peru		slateblue	
moccasin		pink		slategray	
navajowhite		plum		snow	
navy		powderblue		springgreen	
oldlace		purple		steelblue	
olive		red		tan	
olivedrab		rosybrown		teal	
orange		royalblue		thistle	
orangered		saddlebrown		tomato	
orchid		salmon		turquoise	
palegoldenrod		sandybrown		violet	
palegreen		seagreen		wheat	
paleturquoise		seashell		white	
palevioletred		sienna		whitesmoke	
papayawhip		silver		yellow	
peachpuff		skyblue		yellowgreen	

Exemple : Les différentes façons de coder le noir sont donc :

- `couleur = "noir"`
- `couleur = "black"`
- `couleur = "0"`
- `couleur = "0x000000"`

ANNEXE 2 : Caractères spéciaux

On ne doit pas écrire de caractères accentués dans les fichiers xml.
Cela concerne les **caractères accentués**, suivant le principe utilisé dans LaTeX. Pour cela, il faut faire précéder la lettre accentuée de :

- « \` » pour un accent grave (donc « \`A » donne « À »).
- « \’ » pour un accent aigu (donc « \’e » donne « é »).
- « \^ » pour un accent circonflexe (donc « \^O » donne « Ô »).
- « \¨ » pour un tréma (donc « \¨E » donne Ë).

On peut obtenir un « ç » en tapant « \c{c} » (ou « \c{C} » pour une majuscule).
On peut obtenir « œ » et « æ » en tapant « \oe » et « \ae ».

Certains **caractères mathématiques** sont également accessibles :

- « \div » pour le sybole de la division « ÷ ».
- « \times » pour le symbole de la multiplication « × ».
- « \euro » pour le symbole « € ».

On peut obtenir un chiffre en exposant en le faisant précéder de « ^ » et en indice en le faisant précéder de « _ ».
Exemple : « a^2+u_n » donne « a^2+u_n ».

Liste exhaustive des caractères spéciaux d’Instrumenpoche

Majuscule	Minuscule	Autres carctères
"\'E" donne "É"; "\`E" donne "È"; "\^E" donne "Ê"; "\¨E" donne "Ë"; "\`A" donne "À"; "\^A" donne "Â"; "\¨A" donne "Ä"; "\^I" donne "Î"; "\¨I" donne "Ï"; "\`U" donne "Ù"; "\^U" donne "Û"; "\¨U" donne "Ü"; "\^O" donne "Ô"; "\¨O" donne "Ö"; "\c{C}" donne "Ç";	"\'e" donne "é"; "\`e" donne "è"; "\^e" donne "ê"; "\¨e" donne "ë"; "\`a" donne "à"; "\^a" donne "â"; "\¨a" donne "ä"; "\^\i" donne "î"; "\¨\i" donne "ï"; "\`u" donne "ù"; "\^u" donne "û"; "\¨u" donne "ü"; "\^o" donne "ô"; "\¨o" donne "ö"; "\c{c}" donne "ç";	"\times" donne "×"; "^0" donne puissance 0; "^1" donne puissance 1; "^2" donne "²"; "^3" etc... jusqu'à 9 "\times" donne "×"; "_0" donne "indice 0" "_1" donne "indice 1" "_2" donne "indice 2" etc jusqu'à 9 "\div" donne "÷"; "\euro" donne "€"; "\ae" donne "æ"; "\oe" donne "œ";

NB. Certains caractères spéciaux peuvent parfois être remplacés par un rectangle blanc. Il semble que cela soit dû à un bug de Flash. Dans ce cas, la seule solution est de ne pas les utiliser.

ANNEXE 3 : L'attribut `cible`

Ajout du 11 septembre 2005

Chaque point créé doit posséder un attribut *id* permettant de l'identifier. On peut se servir des points déjà créés pour mouvoir les instruments, en remplaçant certains attributs par *cible*, qui fait référence à l'atribut *id* d'un point.
Cela permet de programmer une construction basée sur des points. Si l'on modifie la position des points, toute la construction s'en trouve changé, comme pour un logiciel de géométrie dynamique.
Supposons que l'on ait créé un point tel que *id="12"*. On peut alors placer n'importe quel instrument sur ce point, avec *mouvement="translation"*, en supprimant `abscisse` et `ordonnee` et en ajoutant `cible="12"`. De même pour une rotation en remplaçant `angle`, un ecartement de compas en remplaçant `ecart` par *cible="12"*.