Codage MPEG-4 de dessins animés

Jean-Claude Moissinac Cyril Concolato Jean-Claude Dufourd

Ecole Nationale Supérieure des Télécommunications

46 rue Barrault 75013 Paris
cyril.concolato@enst.fr, dufourd@enst.fr, moissinac@enst.fr

Résumé

MPEG évoque immédiatement la vidéo. Pourtant MPEG-4 permet de coder diverses données multimédia : graphismes 2D et 3D, sons naturels et synthétiques…
Nous nous sommes intéressé au codage du dessin animé avec MPEG-4. Bien sûr, une séquence de dessin animé peut être codée en tant que séquence vidéo ; cependant, cela présente des inconvénients. Nous avons donc étudié le codage du dessin animé en tant que suite de graphismes 2D.
Nous avons été amené à étudier les conditions d'utilisations de diverses techniques d'optimisation du codage MPEG-4 de graphismes 2D. Après une brève présentation du dessin animé du point de vue technique, nous présentons une série de techniques d'optimisation et leur efficacité. Ces techniques peuvent trouver des applications dans d'autres domaines.

Mots Clef

MPEG-4, graphique 2D, animation, dessin animé, codage, optimisation

1 Introduction

Nous avons travaillé sur l'utilisation de MPEG-4 pour coder du dessin animé. Les tests et comparaisons ont été effectués sur des dessins animés de type assez classique : des dessins accumulés sur un décor. Les dessins sont composés de zones de couleur uniforme, généralement bordées d'un trait de couleur différente.

Du fait des discontinuités fortes présentes dans les images de dessin animé, nous avons choisi de nous intéresser dans MPEG-4 à la représentation vectorielle des dessins plutôt qu'à la représentation vidéo des séquences d'images. En effet, cette dernière repose de façon importante sur une hypothèse de continuité des signaux représentés (pour la DCT et les images prédites), vérifiée pour les images naturelles, mais pas pour le dessin animé.

Nous utilisons donc MPEG-4 BIFS version 2 [3]. Nous montrons ici comment représenter des dessins animés à l'aide des descriptions de scène BIFS. Nous montrons ensuite diverses techniques qui peuvent être utilisées pour rendre ces descriptions plus compactes.

2 Le dessin animé

Nous avons choisi de travailler sur le codage de dessins animés de facture assez classique. Nous décrivons ici les hypothèses sur leurs caractéristiques techniques significatives pour le codage.

Les dessins animés de type classique (Tex Avery, Disney) sont généralement composés d'un décor sur lequel évoluent des personnages. La méthode traditionnelle de composition de ces images est de dessiner le décor d'un côté, puis les personnages sur des feuilles de celluloïde transparent. Les celluloïdes vont être empilés sur le décor afin d'être filmés.

Les dessins des personnages sont généralement composés de zones de couleur uniforme, souvent bordées d'un trait de couleur différente. Les dessins sont créés séparément et leur assemblage est défini par la feuille de prise de vue.

Pour chaque instant, la feuille de prise de vue décrit la façon de composer l'image : désignation du décor et numéro de chaque feuille de celluloïde à empiler sur le décor. Une indication synthétique permet de montrer qu'un élément de l'instant précédent est reconduit.

D'un point de vue technique, cela sera représenté par :

- une liste d'actions à entreprendre à chaque instant : ajout d'un nouvel élément à afficher, suppression d'un élément qui était affiché dans l'image précédente, transformation d'un élément,
- une série d'éléments, définis soit une fois pour toute, soit au moment où ils vont servir.

Les éléments sont des groupes de polygones remplis d'une couleur uniforme et bordés d'une autre couleur ou de polylignes de couleur uniforme. L'essentiel de la description de ces éléments est donc constitué de séries de paire de coordonnées (points 2D).

3 Les techniques de codage utilisées

MPEG-4 présente des caractéristiques qui paraissent bien adaptées au dessin animé, principalement la possibilité de définir chaque image d'une séquence par la manière dont on la compose à partir d'éléments –vidéo, graphismes 2D, scène 3D [1].

MPEG-4 présente une grande richesse de techniques qui peuvent permettre de représenter des séquences de dessins animés : BIFS-Command, BIFS-ANIM, MPEG-4 vidéo, avec de nombreuses possibilités pour chacune de ces structures.

Nous avons choisi de nous intéresser d'abord à la représentation vectorielle des animations. En effet, l'animation se prête assez mal aux représentations de type vidéo (flux compressé d'images bitmap) : les discontinuités spatiales et temporelles y sont fortes. Obtenir une bonne qualité nécessite donc des relativement hauts débits. Une représentation vectorielle a l'avantage de permettre un rendu dépendant des capacités du terminal.

Nous avons donc travaillé avec BIFS V2 [2]. BIFS permet de définir une scène par un arbre de construction de l'image. Une grande variété de nœuds est définie par BIFS. Les nœuds qui nous concernent le plus contiennent soit des descriptions de formes géométriques, soit des références à ces formes, soit des transformations géométriques sur ces formes, soit des paramètres d'affichage (couleur, épaisseur de trait…).

3.1 Représentation de l'animation

Nous proposons une façon de représenter la feuille de prise de vue avec MPEG-4. On peut énoncer le principe de la façon suivante :

- on dispose d'un certain nombre de couches qui seront empilées pour composer l'image,
- à chaque nouvelle image, pour chaque couche, on peut enlever un élément d'une couche, transformer certains paramètres d'un élément (ex : son échelle) et mettre un nouvel élément dans une couche.

Le principe de représentation repose sur la définition initiale d'un nœud servant à représenter les couches et qui ne contient rien au départ. Dans la suite du document, les exemples de codage sont donnés au format MPEG-4 XMT [1].
Le principe est présenté de façon simplifiée dans la structure ci-dessous :

```
<Group DEF="root">
    <children>
        <Switch whichChoice="-1">
        <choice>
            ...
            <GroupDEF="Formes">
            <children>
                <Shape DEF="FormeVide" />
                </children></Group>
                ...
        </choice>
        </Switch>
        ...
        <children>
        <OrderedGroup><children>
        <Transform2D DEF="Couche0">
                <children><Shape/>
                </children>
        </Transform2D>
        <Transform2D DEF=" Couche1">
                <children><Shape/>
                </children>
        </Transform2D>
        ...
        </Transform2D>
        </children>
        </OrderedGroup>
        </children>
        </Transform2D>
    </children>
</Group>
```

Le switch permet de définir une branche conditionnelle de l'arbre avec une condition jamais remplie (-1). Les éléments définis dans ce sous-arbre ne seront jamais vus directement, mais seulement s'ils sont référencés ailleurs dans l'arbre. Les éléments seront ajoutés ou enlevés de la branche "**Formes**".
Des branches visibles ("**Couche***i*"), dont l'ordre d'affichage est défini (OrderedGroup), sont créées, vide au départ. Des éléments sont ajoutés ou enlevés dans ces « couches » en donnant leur référence dans le groupe "**Formes**" et en donnant la référence "**Couche***i*" de la couche modifiée.
Les éléments ajoutés présentent des structures répétitives dont nous proposons ci-après une représentation.

3.2 Prototypes de structures

Nous avons vu que les formes élémentaires qui servent de base à la construction des images sont des polygones de couleur uniforme ou des polylignes.

Avec MPEG-4, chaque polygone nécessite de définir une structure Shape précisant les paramètres d'affichage, puis la géométrie de la forme. La structure ainsi définie est répétée de nombreuses fois.

Figure 1 : exemple de représentation d'une forme

```
<Shape>
   <appearance><Appearance
   USE="ap1"/></appearance>
   <geometry>
      <IndexedFaceSet2D colorPerVertex="false">
         <coord><Coordinate2D point="-103.0 -290.0
         -101.0 -284.0... ..." /></coord>
      </IndexedLineSet2D>
   </geometry>
</Shape>
```

Il est donc possible de gagner sur le coût de cette structure en utilisant un prototype de structure à l'aide de PROTO (voir figure 2).

La définition d'un PROTO permet de stocker une structure répétitive une seule fois en lui attribuant des parties variables. La structure est ensuite réutilisée à l'aide d'une référence et de la définition de ses parties variables. Dans le cas d'une forme, les parties variables sont la référence à une apparence (couleur, épaisseur de trait…) et la table des points de son contour.

Figure 2: prototype définissant la structure de codage d'une forme

```
<ProtoDeclare name="PForm" protoID="0">
   <field name="appearance" type="Node"
   vrml97Hint="field" />
   <field name="point" type="Vector2Array"
   vrml97Hint="exposedField" vector2ArrayValue="" />
   <Shape>
     <geometry>
       <IndexedFaceSet2D colorPerVertex="false">
         <coord>
           <Coordinate2D>
             <IS><connect nodeField="point"
             protoField="point" /></IS>
           </Coordinate2D>
         </coord>
       </IndexedFaceSet2D>
     </geometry>
     <IS><connect nodeField="appearance"
     protoField="appearance" /></IS>
   </Shape>
</ProtoDeclare>
```

Ce prototype de structure peut être utilisé répétitivement tel que présenté à la figure 3.

Figure 3: utilisation d'un prototype de structure de forme

```
<ProtoInstance name="PForm">
   <fieldValue
   name="appearance"><node><Appearance
   USE="app1"/></node>
   </fieldValue>
   <fieldValue name="point" vector2ArrayValue="-
   103.0 -290.0 -101.0 -284.0" />
</ProtoInstance>
```

Ce type de codage permet de gagner sur la structure de la séquence, mais les gains principaux sont à attendre du codage des grandes quantités de points qui décrivent les dessins.

3.3 UseEfficientCoding

Ce drapeau permet d'indiquer qu'on utilise un codage des réels (floats) optimisé en taille. La représentation de base, sans useEfficientCoding, est IEEE 754 [4], dans sa version exploitant 32 bits par float. Ce codage présente les avantages suivants :

- codage normalisé : un seul codage est possible pour une valeur donnée,
- même nombre de bits quelque soit la valeur,
- exploitation directe par des coprocesseurs.

Le codage proposé par la norme MPEG-4 permet de coder la plupart des valeurs sur moins de 32 bits, en gardant la même précision. Il présente par contre l'inconvénient de nécessiter du traitement au codage et au décodage.

A titre d'exemple, notre fichier aa001b comporte 276000 floats. Codé de façon brute, il nécessite 765 ko avec useEfficientCoding désactivé et 446 ko avec useEfficientCoding activé, soit un gain d'environ 40% et de 9 bits par float en moyenne grâce à cette seule optimisation.

3.4 Quantification

Les coordonnées des points MPEG-4 sont, de base, représentées en flottants sur 32 bits (codage IEEE). Chaque point 2D nécessite donc 64 bits dans cette représentation.

Cependant, on peut attribuer une quantification aux points 2D. Une structure de quantification (QP), applicable aux points 2D d'un sous-arbre, défini une borne minimale, une borne maximale et le nombre de bits affectés à chaque coordonnée. Cette structure occupe entre 100 et 200 bits. D'après nos tests, sur une scène typique –points entre (0,0) et (768,576)-, une structure QP globale est toujours efficace et permet de ramener la représentation de chaque point de 64 bits à moins de 28 bits.

3.5 Utilisation d'index de points

Il est possible de ne pas représenter un contour par la liste des points qui le composent, mais par une liste d'index sur une table de points préalablement définie.

La réutilisation des points d'un tableau s'avère problèmatique. La plupart des points sont réutilisés dans des portions de contours que partagent deux formes voisines. Pour être réutilisés, ces points doivent être dans une même table et cette table doit contenir tous les points du premier contour ET tous les points du second contour. De proche en proche, on se rend compte que pratiquement tous les points d'une couche de l'animation devraient être contenus dans une même table.

Sur nos séquences de test, les points sont en moyenne utilisés R fois avec 1.5<R<2. Pour chaque point réutilisé, on gagne la différence entre la représentation du point quantifié et la taille de l'index; pour chaque point non réutilisé, on perd la taille de l'index. Nous devons compléter notre évaluation de cette technique.

3.6 Codage prédictif

Il est possible de coder une série de points à l'aide d'un codage prédictif ; par exemple, il est possible de coder le premier point de façon absolue (INTRA), et de coder la différence des points suivants par rapport à ce point. Nos travaux ont montré que ce codage amène un gain faible en complèment des méthodes précédentes, pour un coût de codage et décodage important.

4 Données et résultats

Nos données ont été produites avec le système PEGS[1] de production professionnelle de dessins animés. Ce système permet différents types de sorties, de la vidéo D1 aux fichiers Flash en passant par le film. Ce système permet de produire des animations à très haute résolution.
Pour nos exemples, les séquences sont réalisées pour un affichage en 768x576 pixels. Cela correspond à une qualité télévision (PAL) avec des pixels carrés. Nous avons défini un format de sortie intermédiaire –de type XML- dans lequel nous est fourni le plus d'informations possibles connues du système PEGS –changements d'échelle, effets spéciaux, notament sur les couleurs,…-, afin d'intégrer progressivement le maximum de possibilités dans notre production de MPEG-4.
Nos exemples comprenaient environ 500 nouveaux points par image sur des durées de une à six secondes.
La table qui suit (table 1) donne une idée de la complexité des scènes traitées.

Table 1: Exemples de séquences d'animation utilisées

Nom	Images	Points	Points/image
Vecto	40	17538	438
Aa001b	274	138059	504
Seq006	38	22345	588
Seq004av	16	7312	457
Compfxv	69	33384	484

La table qui suit (table 2) est représentative des résultats obtenus.

Table 2: Exemples de séquences d'animation codées en MPEG-4

Nom	Meilleure taille obtenue (Ko)	Images	Débit en kbps
Vecto	51,4	40	131
Aa001b	334	274	245
Seq006	76	38	400
Seq004av	16	16	213
Compfxv	110	69	163

[1] www.mediapegs.com

Table 3: Différents tests sur une même séquence

	Séquence compfxv QP=avec QuantizationParameters	Taille en Ko
1	sans optimisation	290
2	Avec useEfficientCoding	207
3	PROTO	281
4	UseEfficientCoding+QP+coordonnées entières	121
5	UseEfficientCoding+QP+ coordonnées entières +PROTO	112
6	UseEfficientCoding+QP+PROTO+ predictiveMFField+ coord. entières	110

Notons que l'utilisation de PROTO fournit un gain identique quelque soit les techniques de codage utilisées par ailleurs. En effet, nous utilisons les PROTOs pour le codage de la structure de la séquence, qui est bien sûr une constante structurelle de la séquence. Sur cette séquence, nous constatons un gain de 9 Ko identique entre le codage 1 et le 3 et entre le 4 et le 5 (Table 3).

5 Conclusion

Nous avons montré que MPEG-4 BIFS constitue un dispositif ouvert standardisé pour coder efficacement du dessin animé. Nos travaux ont permis de diviser par trois la taille obtenue par un codage naïf.

Les méthodes de codage utilisées jusqu'à présent préservent la structure des dessins animés (couches, formes, groupes de formes…). Nous souhaitons étudier les possibilités d'optimisation offertes par les cartes planaires [5]. Ces structures topologico-géométriques permettront par exemple d'éliminer des lignes cachées ou d'exploiter plus finement les réutilisations de points.

Nos futurs travaux porteront sur diverses optimisations supplémentaires, sur l'adaptation du codage aux conditions de distribution et sur l'optimisation de l'affichage de MPEG-4 BIFS.

Bibliographie

[1] *Overview of the MPEG-4 Standard*, ISO/IEC JTC1/SC29/WG11 N4030, March 2001, www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm
[2] *Coding of audio-visual objects – Part 2: Visual*, ISO/IEC 14496-2:2000
[3] *Coding of audio-visual objects – Part 1: Systems*, ISO/IEC 14496-1:2000
[4] *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE 754-1985
[5] *Incremental computation of planar maps* In Proc.of SIGGRAPH'89, Gangnet et al.