INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 M10097
MPEG03
October 2003

Title: A Simpler, Efficient Binary Encoding for MPEG-4 Scenes

Source: JC Dufourd, Cyril Concolato, Jean Le Feuvre (ENST)

Status: for consideration at MPEG meeting in Brisbane

Abstract

This contribution describes a proposal to change the scene encoding to make decoders smaller and faster, while keeping a high level of compression efficiency, very similar to the current BIFS encoding compression efficiency.
The Trondheim contribution m9790 described a low-complexity version of BIFS by changing the object set and keeping the current BIFS encoding. The results showed good progress on the complexity part, but failed to reach the compression goals. In order to reach our compression goals, we had to drop the current binary encoding and go even further in the direction of the simplification.

Current Problems

Remark: No ISG study was ever made on the complexity of the BIFS decoding. Apart from the comparison of PMF with the Samsung interpolator coding, no comparative study was ever made about having or not any of the BIFS encoding features.

Quantization

The quantization has a complex and expensive setup (in terms of bandwidth), inappropriate in many circumstances unless you use global quantization (which is not in the used profiles) since you cannot use it on replace commands, and for smaller devices with no floats, it is very difficult to implement the quantization decoding with suitable efficiency while keeping enough precision (because of the floating point division).

Proposed Solution

Simplify the quantization:

- no min/max which forces the use of floats in decoding, just a number of bits
- allow the decoder to be implemented completely in fixed point by signalling the resolution for coordinates, scales and angles.

Array compression

Arithmetic coding is even worse than quantization in terms of decoding complexity, and PMF is not efficient on small arrays. Using a delta scheme with the signalling of the number of bits per component, we get better results than PMF on the kind of data that is relevant to 2D applications. The decoding complexity is ridiculously small compared to PMF.
See table 1.

Proposed Solution

Use a simple scheme which gains 35% to 40% on the current encoding, and is much simpler than PMF:

- encode the number of bits for the first X and Y

- encode the first X and Y
- encode the number of bits for remaining deltaX and separately for deltaY
- encode the deltas

List Encoding vs Vector Encoding

The use of list encoding is extremely stupid from the decoder implementation point of view. At the beginning of the decoding of the array, if the list encoding is used, then you cannot allocate one array of the right size. You have to either allocate too much, or do multiple allocations and copies. With vector encoding, you know the length of the array from the start.

Proposed Solution

Do not allow list encoding, and remove the flag.

Contextual Node Type Encoding, Escape Codes and Multiple Field Indexes

In BIFSLC, for each proto instance, 7 bits are wasted because of the v1 escape code. Since there are only proto instances and two nodes in BIFSLC, those 7 bits are a pure loss.
In this respect, we have made a test with Osmo4, by compiling 3 different versions of the BIFS subsystem:

- BIFSLC: 32.7 kbytes
- BIFSv2 (no 3D): 94.2 kbytes
- BIFSv6 (incl. 3D): 253.9 kbytes

This was computed by difference between the application with MP4 support, no scene graph, no BIFS, and the configuration described. This indicates that the node tables are unacceptably big.

Proposed Solution

By getting rid of the contextual node encoding and the multiple field indices and having only SFWorldNode with room for future extensions, we gain more than a factor of 2 on the size of the BIFS decoder, because the huge node tables disappear.

New Proposals

These proposals are simple and well-known techniques. The fact that they are not used in BIFS is difficult to explain.

Color Palette

In order to optimize the color information transmission, we introduce a simple color palette scheme. We create the following updates:

- reset color palette
- add color list to palette: a list of colors is added to the current palette
- remove color index list from palette: a list of indices of colors to be removed from the palette is given

When a color palette is defined, SFColor is modified: it contains only an index into the color palette. The number of bits of the index is log2(paletteSize).

Optimized Flexible Encoding

We propose to declare a “node palette” at the beginning of the scene. This node palette is a list of node indices signalling which nodes are going to be present in the stream. Then the node tags are indices in that node palette table instead of the bigger node indices from the list containing all nodes. This improves the coding efficiency. It can also improve the resource usage in the player: if the terminal sees that no sensors will be used, then it can disable the allocation of sensor contexts in the scene tree.
Contrary to the color palette, we think the node palette should not change over time because player and decoder optimisation could not be statically made.

Annex: Table 1

	Points only				All ES (BIFS or STZ)			All file			SWF size		BifsLC /SWF	BIFS all ES	BIFS /SWF
Sequence	BIFSLC	STZ	nbPoints	%	BIFSLC total	STZ total	%	MP4	STZP	%			%		
toutou	5226	3405	1742	65%	6632	4231	64%	7289	4300	59%	7115	60%	102%	5886	83%
pistache_	8646	6193	2882	72%	27825	21735	78%	32154	24864	77%	31879	78%	101%	30677	96%
frog1	45933	31498	15311	69%	54487	37282	68%	55504	37651	68%	ecs				
kangaroo	52128	29555	17376	57%	68038	39702	58%	70975	41671	59%	79687	52%	89%	56930	71%
gen002	131178	89662	43726	68%	163898	111427	68%	165827	112556	68%	ecs				
jack_marcel	580965	387435	193655	67%	760881	512794	67%	770778	520563	68%	ecs				
butchcassidy	303900	148922	101300	49%	597428	359926	60%	615593	374585	61%	871868	43%	71%	461203	53%
minitoon	34434	22984	11478	67%	40942	27145	66%	42019	27564	66%	32013	86%	131%	21514	67%
tarzan	132951	83950	44317	63%	153514	97833	64%	155815	99272	64%	235638	42%	66%	87789	37%
ballon	97959	65632	32653	67%	109479	73605	67%	110664	74114	67%	132200	56%	84%	54377	41%
tuture	13215	8681	4405	66%	55427	38653	70%	58076	40382	70%	64393	63%	90%	49065	76%
chat	55431	38769	18477	70%	64065	45142	70%	66090	46351	70%	138956	33%	48%	25058	18%
dancer	20361	13191	6787	65%	23147	14999	65%	24548	15688	64%	37816	41%	65%	12340	33%
mission impossible	31842	19065	10614	60%	70796	50243	71%	73949	52392	71%	78091	67%	95%	71411	91%
birdrats	184140	101662	61380	55%	230961	131663	57%	232494	132462	57%	175685	75%	132%	201098	114%
cats	129753	68887	43251	53%	177197	100011	56%	178550	100660	56%	102131	99%	175%	173169	170%
ptiluc	213528	136982	71176	64%	247037	165964	67%	249230	167313	67%	245760	68%	101%	165071	67%
petrole	127194	72692	42398	57%	152222	91293	60%	155063	93182	60%	154226	60%	101%	135822	88%
jm	597831	348679	199277	58%	704977	430449	61%	712966	436628	61%	627873	70%	114%	628941	100%
vecto	143478	89764	47826	63%	163939	107135	65%	166612	108884	65%	181159	60%	92%	129564	72%
waterfall	96102	55852	32034	58%	105053	63072	60%	106454	63761	60%	173410	37%	61%	151816	88%
palmtree f2	20844	16556	6948	79%	31128	19340	62%	32217	19769	61%	23241	85%	139%	30181	130%
cow	17700	13023	5900	74%	21537	15649	73%	22578	16038	71%	21265	75%	106%	21205	100%
vegetable	10788	7677	3596	71%	12809	8871	69%	13790	9210	67%	12171	76%	113%	9394	77%
				64%			**65%**	**4119235**	**2619860**	**65%**		**63%**	**99%**		**80%**
	BIFSLC with 12bits								64%						

The sequences in table 1 have various origins:

- ecs: they were designed with the MediaPEGS cartoon design tool, and optimized for mobile viewing
- suushi (last 4): they were designed in Flash for mobile viewing
- bibo (toutou, tuture, pistache): they were designed in Flash for mobile viewing
- pegs (birdrats, cats, ptiluc, petrole, jm, vecto): they were designed in Flash for Internet viewing
- minitoon: was designed in Flash for mobile viewing
- tarzan, ballon: were designed for Internet and adapted to mobile
- butchcassidy, chat, dancer: designed in Flash for Internet viewing

STZ is a proprietary format implementing the BIFSLC object set and using all recommandations of this document for the binary encoding.

BIFSLC encoding uses a quantization on 12 bits, same as STZ. On the encoding of coordinates only, BIFSLC is 50% bigger than STZ. The same is true when the whole ES is compared, or the whole file. Yet BIFSLC is not bad itself, since it achieves a size equivalent to SWF.

The BIFS column contains a version of the content with either curves or filtered polygons (whichever is smaller), v2 nodes, protos, optimized quantization and predictive MFField on points and indexes.