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

ISO/IEC JTC1/SC29/WG11
MPEG96/M1261
Chicago, USA

Title: **The Architectural Framework of MSDL-S**

Source: **Yihan Fang and Alexandros Eleftheriadis(Columbia University)**

Group: **MPEG-4 Integration/MSDL**
Date: **September 26, 1996**

In this document we describe the integration of MSDL-S into the overall MSDL architecture using specific code examples from MSDL Version 1.1. We are assuming an API-based interface to the syntax decoder, where parsing is performed by explicitly calling a get() function that is automatically defined for each parsable class. Similarly, a function put() would be automatically generated to put the content of the bitstream syntax into a specific bitstream.

The following example of objects with encoder-controlled behaviors describes a rotating cube that can be modified to read the axis of rotation from the bit stream. On each frame, the *(x,y,z)* axis of rotation is read from the bit stream. Each component is read as an 8-bit integer in {-127,...,128}, and is then normalized to lie in (-1,1]:

The classes are declared in an MSDL file, as shown in the following. There are two classes `Compositor` and `RotatingCube`. Each class declaration consists of type declarations which include syntax declarations (if there is information to be parsed from the bitstream) as well as method declarations.

```
class Compositor is Object{
  ...
  int(8) x;
  int(8) y;
  int(8) z;

  // method declaration
  void pushTransform();
  // rotate angle deg around (x,y,z)
  void rotate(int angle, int x, int y, int z);
  void render(Cube cube);
};

class RotatingCube is AVObject {
  void render(Compositor c);
};
```

The class methods are implemented in a separate file. Here we show an example of the implementation of the above classes in C++ (Java could have been used just as well):

```
void Compositor::pushTransform()
{
  ...
}

void Compositor::rotate(int angle, int x, int y, int z)
{
  ...
}

void Compositor::render(Cube cube)
{
  ...
}

void RotatingCube::render(Compositor c)
{
  ...
  float x = c.x/128.0;
  float y = c.y/128.0;
  float z = c.z/128.0;
  c.pushTransform();
  c.rotate(angle,x,y,z);
  c.render(cube);      // render cube
  c.popTransform();
}
```

When calling the automatically generated `get` function of the object `c` of the parsable class `Compositor`, the three 8-bit integers `x, y, z` are parsed from the bitstream and they are used to provide information when an object of `RotatingCube`—`rc` is rendered.

```
main()
{
  InputStream is;
  Compositor c;
  c.get(is);
  RotatingCube rc;
  rc.render(c);
}
```

The above example shows how the bitstream syntax of an object is separated from the decoding and/or object rendering tools using MSDL-S representation. Clearly, this would easily allow for different decoding implementations of the relevant algorithms for a given syntax specification.

An alternative approach for the parsing process is to parse the bitstream while constructing an object and put the content into the bitstream when deconstructing an object (when encoder code is generated). The benefit of this approach is that it is totally transparent to the programmer; parsing is conceptually performed right before any constructor is called. Experimentation is needed to identify the most suitable approach.

References

[1] MSDL Specification. Version 1.1, ISO/IEC JTC1/SC29/WG11 N1246, Florence, Italy, March 1996.