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/M1556
Maceió, Brazil

Title: **The Integration of MSDL-S with MSDL Architecture and VM**
Source: **Alexandros Eleftheriadis and Yihan Fang (Columbia University)**
Group: **MPEG-4 Integration/MSDL**
Date: **November 12, 1996**

In this document we describe the integration of MSDL-S into the overall MSDL architecture using specific code examples from MSDL Version 1.3 [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.

MSDL-S Code

```
class RotatingCube is AVObject{
  ...
  int(8) x;      // 3-bit integer representation
  int(8) y;
  int(8) z;

 Cube cube = new cube();

 // method declaration
 void render(Compositor c){

   ...
   float x = s.x/128.0;
   float y = s.y/128.0;
   float z = s.z/128.0;
   c.pushTransform();
   c.transform.rotate( angle,x,y,z);
   c.render(cube);          // render cube
   c.popTransform();
```

```
    }
  }
```

The above code can be run through a translator that will generate Java code as follows:

Java Code

```
class RotatingCube is AVObject{
  ...
   int x;
   int y;
   int z;

   void get();    // automatically generated
   void put();   // automatically generated

  Cube cube = new cube();

  // method declaration
  void render(Compositor c){

    ...
    float x = s.x/128.0;
    float y = s.y/128.0;
    float z = s.z/128.0;
    c.pushTransform();
    c.transform.rotate( angle,x,y,z);
    c.render(cube);          // render cube
    c.popTransform();
  }
};
```

When calling the automatically generated `get` function of the parsable class RotatingCube, 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() Code

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

Alternatively, the get() code can be embedded in the constructor of the object so that parsing is performed when the object is instantiated. It should be discussed if this or the direct get() approach are most appopriate for the needs of MSDL.

The above example clearly shows how the bitstream syntax of an object is separated from the decoding and/or object rendering tools using MSDL-S representation. In addition, it details how MSDL-S can be directly used in the current MSDL VM work to handle syntactic description.

Ultimately, the syntax will be described in a binary form and stored as part of the class declaration. This can be done by storing it in a static variable of the class as an array of compressed bytes. The syntax decoder then uses that binary information in order to perform parsing as appropriate.

References

[1] MSDL Specification. Version 1.3, ISO/IEC JTC1/SC29/WG11 N1401, Florence, Italy, March 1996.