This document proposes some solutions to fulfill some of the requirements expressed in the Call for Proposals issued by the MPEG-4 Systems group. The proposed solutions mainly address the 2D graphics part of the requirements.

The solutions which are described in this document are all backward compatible and do not invalidate existing bit streams. They are based on the definition of new nodes or the extension of existing nodes with the addition of new fields. In the latter case, care has been taken to add field to nodes where there was space left in the node coding table. This way of extension is very efficient in terms of compression. It is also less confusing because it groups several levels of complexity for the same functionality in the same node. Of course, if this solution is to be chosen by the group, it will require to clarify the existing profiles by saying that if any field is added to a node in future version of the standard these fields will not be supported in these profiles. Moreover, special care need to be taken that extension fields do not modify the behavior of existing fields.

**Requirement #1:**

It shall be possible to allow the use of a viewport in conjunction with Layer2D, with all appropriate options to deal with possible differences of aspect ratio between the viewport and the enclosing Layer2D.

**Proposal**

To fulfill this requirement, we propose the creation of the Viewport node (previously presented as Viewpoint2D in N7865) of type SFViewportNode. This node adds 3 new aspects to the standard: first the ability for the content to adapt itself to the size of the terminal it is displayed on; second, the bindability of this node allows for navigating in a 2D scene as with the Viewpoint node in a 3D scene; and finally, it allows for viewing part of the scene with some constraints on the aspect ratio.
9.4.2.X Viewport

9.4.2.X.1 Node interface

Viewport {
  eventIn  SFBool  set_bind
  exposedField  SFVec2F  position  0 0
  exposedField  SFVec2F  size  -1 -1
  exposedField  SFFloat  orientation  0
  field  MFInt32  alignment  [0 0]
  field  SFInt32  fit  0
  field  SFString  description  ""
  eventOut  SFTime  bindTime
  eventOut  SFBool  isBound
}

NOTE - For the binary encoding of this node see Annex H.

9.4.2.X.2 Functionality and semantics

A Viewport node is placed in the viewport field of a Layer2D or CompositeTexture2D node. It defines a new viewport and implicitly establishes a new local coordinate system. The bounds of the new viewport are defined by the size and position field. The new local coordinate system’s origin is at the center of the parent node in the parent’s local coordinate system. The orientation field specifies the rotation which is applied to the viewport in the parent node’s local coordinate system with respect to the X-axis.

Viewport nodes are bindable nodes (see 9.2.2.14) and thus there exists a Viewport node stack which follows the same rules than other bindable nodes (e.g. Background2D).

The description field specifies a textual description of the Viewpoint2D node.

The alignment and fit fields specify how the viewing area is mapped to the rendering area of the parent node (i.e. Layer2D, CompositeTexture2D, or the 2D top-node).

If the fit field is set to 0, the viewing area is scaled to fit the rendering area without preserving the aspect ratio.
If the fit field is set to 1, the viewing area is scaled preserving the aspect ratio to fit entirely inside the rendering area. The scaling operation is performed possibly after rotation as specified by the orientation field.
If the fit field is set the 2, the viewing area is scaled preserving the aspect ratio to cover entirely the rendering area. The scaling operation is performed possibly after rotation as specified by the orientation field.

The alignment field is an MFInt32 field that contains two values. The first value specifies alignment along the X-axis and the second value specifies alignment along the Y-axis. The first value belongs to the following set of SFInt32: -1, 0, 1. The second value belongs to the following set of SFInt32: -1, 0, 1. An empty alignment field is equivalent to the default value. When the fit field is set to 0, the alignment field is ignored. The meaning of the different values of the fit and alignment fields is described in the following figure.
Figure 1: description of alignment and fit fields

**Requirement #2**
It shall be possible to allow the use a 2x3 matrix to specify a 2D transformation.

**Proposal**
To fulfill this requirement, we propose the creation of the following new node.

9.4.2.X TransformMatrix2D

9.4.2.X.1 Node interface

```plaintext
TransformMatrix2D { 
  evenIn MFNode addChildren 
  evenIn MFNode removeChildren 
  exposedField MFNode children 
  exposedField MFFloat matrix [ 1 0 0 0 1 0 ]
}
```

NOTE - For the binary encoding of this node see Annex H.
9.4.2.X.2 Functionality and semantics

The TransformMatrix2D node is a grouping node that defines a coordinate system for its children that is relative to the coordinate systems of its ancestors. See ISO/IEC 14772-1:1998 for a description of coordinate systems and transformations and for a description of the children, addChildren, and removeChildren fields and eventIns.

The matrix field defines a geometric 2D transformation based on the following transformation matrix, where \( \text{matrix}_i \) is the \( i \)-th SFFloat in the matrix field:

\[
\begin{bmatrix}
\text{matrix}_1 & \text{matrix}_2 & \text{matrix}_3 \\
0 & \text{matrix}_4 & \text{matrix}_5 \\
0 & 0 & 1
\end{bmatrix}
\]

Given a 2-dimensional point \( P \) and TransformMatrix2D node, \( P \) is transformed into point \( P' \) in its parent's coordinate system by the transformation whose matrix is \( T \).

\[
P' = T \times P
\]

The behaviour of TransformMatrix2D with respect to the Sound2D node is the same as the behaviour of the Transform2D node.

The addChildren and removeChildren eventIns are used to add or remove child nodes from the children field of the node as for a Transform2D node.

**Requirement #3**

It shall be possible to draw circular, parabolic and elliptical arcs.

**Proposal**

The following types of drawing are allowed in SVG:
- The "moveto" commands
- The "closepath" command
- The "lineto" commands
- The curve commands
- The cubic Bezier curve commands
- The quadratic Bezier curve commands
- The elliptical arc curve commands

Each type of command (except the closepath one) is duplicated to allow relative and absolute positioning of coordinates.

BIFS currently allows to perform some of the SVG commands: moveTo, lineTo and Bézier curves (since quadratic Bézier curves can be expressed in term of cubic Bézier curve), with absolute positioning. We propose to add to BIFS the ability to perform all the other SVG commands.

The node which seems the most appropriate for this functionality is Curve2D. Fortunately, it holds only three fields which gives the opportunity to introduce a new field for the purpose of drawing circular, parabolic and elliptical arcs.

The principle of this proposal is to add a new field to the Curve2D node. This field is called enhancedType. When not empty, it supersedes the type field, i.e. the coordinate
pairs of the point field of the Coordinate2D node are consumed following the values of the enhancedType field.

Care has been taken that:
- a reserved value is left for future extensions of the type field (parabolic arcs, predictive coding),
- and the added types are consistent with the previous ones. For example, elliptical arcs could be drawn without the knowledge of focal points but with angle and sweeping information but then it would not be consistent with the use of the Coordinate2D node.

The Curve2D node would become as follows.

9.4.2.X Curve2D

9.4.2.X.1 Node interface

```yaml
Curve2D {
    exposedField SFNode point NULL
    exposedField SFInt32 fineness 0.5
    exposedField MFInt32 type []
    exposedField MFInt32 enhancedType []
}
```

9.4.2.X.2 Functionality and semantics

[...]

The permitted values of `enhancedType` are:
- 0 = MoveTo: Same as the value 0 for the `type` field. Moreover, the coordinate pair consumed from the `point` list also defines the starting point of the new subpath \( P_0 \). MoveTo shall not occur neither as the first element in `enhancedType` field.

- 1 = LineTo: Same as the value 1 for the `enhancedType` field.

- 2 = CurveTo: Same as the value 2 for the `enhancedType` field.

- 3 = NextCurveTo: Same as the value 3 for the `enhancedType` field.

- 4 = IncreasingArcTo: Three coordinate pairs in the `point` list are consumed, defining \( F_1, F_2 \) and \( N \). \( F_1 \) and \( F_2 \) are the focal points of the ellipse to which \( P \) and \( N \) belong. On this ellipse, \( P \) and \( N \) define two arcs. Considering the polar parametric representation of the ellipse \( r(Nx\cos(\theta),Ny\sin(\theta)) \) and assuming that \( F_1 \) is the focal point with the negative coordinate on the x-axis, the drawn arc is the one that corresponds to an increase of \( \theta \) when sweeping the arc from \( P \) to \( N \).

- 5 = DecreasingArcTo: Same as IncreasingArcTo except that the drawn arc is the one that corresponds to a decrease \( \theta \) when sweeping the arc from \( P \) to \( N \).

- 6 = ClosePath: No coordinate pair is consumed from the `point` list. Close the current subpath by drawing a straight line from \( P \) to the current subpath's initial point \( P_0 \). If a ClosePath is followed immediately by any other command than a MoveTo or RelativeMoveTo, then the next subpath starts at the same initial point as the current subpath, i.e. \( P_0 \). Note: The difference between closing the subpath and explicitly drawing a line between \( P \) and \( P_0 \) is that
in the first case the line in $P_0$ will be closed with the current value of line-join while in the second case the line will be closed using the current value of line-cap.

- 7 = Reserved

**Requirement #4**

It shall be possible to:
- render text on a baseline that follows any path/curve.
- easily combine individual text objects with different decorations to build a continuous text

**Proposal**

We propose the definition a new SFGGeometryNode node called TextSpan. The new node definition would allow to apply transformation over the positioning of the text contained in several Text nodes. This would make possible to position some text along a path or a curve and also to put several text together. It would behave differently with regards to filling and stroking than the Text node therefore allowing fill, stroking and texture as for other SFGGeometryNode nodes.

The new node to be defined is called TextSpan. It uses the notion of index that was used for the IndexedFaceSet or IndexedLineSet nodes. For this new node, the indices are used to reference characters or glyphs.

9.4.2.X TextSpan

9.4.2.X.1 Node interface

```plaintext
TextSpan {
  eventIn  MFNode  addChildren
  eventIn  MFNode  removeChildren
  exposedField MFInt32 children []
  exposedField MFInt32 adjust 0
  exposedField MFVec2F bbox -1, -1
  exposedField MFNode fontStyle []
  exposedField MFFloat fontStyleIndex []
  exposedField MFFloat orientation []
  exposedField MFFloat orientationIndex []
  exposedField SFNode translation []
  exposedField MFInt32 translationIndex []
  exposedField SFNode color []
  exposedField MFInt32 colorIndex []
  exposedField SFNode path []
  exposedField SFFloat startOffset 0
}
```

**NOTE** - For the binary encoding of this node see Annex H.

9.4.2.X.2 Functionality and semantics

The `addChildren` and `removeChildren` are identical to the `addChildren` and `removeChildren` eventin of the Transform2D node.

The `children` field shall only contain `Text` or `TextSpan` nodes.
The `adjust` field specifies if the text content from the `children` field shall be adjusted to fit in a bounding box as defined by the `bbox` field. Three values are allowed: 0 means no adjustment, 1 means only spaces shall be stretched and 2 means both spaces and glyphs are adjusted.

The `fontStyles` field shall list the `FontStyle` nodes to be used to render the text content of the `children` field. When `fontStyleIndex` is empty, the `FontStyle` node to be used is the one specified by each `Text` node (either explicitly or default). Otherwise, the `fontStyleIndex` fields determines the `FontStyle` of each character contained in all the `Text` or `TextSpan` nodes. If there are more `FontStyle` nodes than characters, the remaining nodes are ignored. If there are less nodes than characters, the remaining characters use the last `FontStyle` node.

The `orientation` field shall list the rotations to be applied to each character of the text content of the `children` field. When `orientation` is empty, no rotation is applied for this node. An other `TextSpan` node can possibly specify the `orientation` field for its content. Otherwise, the `orientation` fields determines the rotation of each character contained in all the `Text` or `TextSpan` nodes. If there are more angles specified in the `orientation` field than characters, the remaining angles are ignored. If there are less angles than characters, the remaining characters use the last angle.

The `translation` field shall list the translation to be applied to each character of the text content of the `children` field. When `translation` is empty, no translation is applied for this node. An other `TextSpan` node can possibly specify the `translation` field for its content. Otherwise, the `translation` fields determines the translation of each character contained in all the `Text` or `TextSpan` nodes. If there are more `SFVec2F` specified in the `translation` field than characters, the remaining `SFVec2F` are ignored. If there are less `SFVec2F` than characters, the remaining characters use the last `SFVec2F`.

The `color` and `colorIndex` field act as for the `IndexedLineSet2D` node except that the vertices are replace by the characters contained in the `children` field, that there is no color interpolation and that the top most `TextSpan` node has precedence to set the color of the characters.

The `path` field contains an `SFGGeometry` node along which the text is to be rendered. The `startOffset` field determines the offset to be used along the shape before starting to render the text.

**Requirement #5**

It shall be possible to:
- define the shape of each character in a font using a set of `Shape` nodes

**Proposal**

The following node allows for defining user font data such as glyphs, font data (i.e. mapping of characters to glyphs, size of the glyphs, position of the glyphs, ascent, descent ...).
9.4.2.X FontData

9.4.2.X.1 Node interface

FontData {
  Field SFString fontName = ""
  Field MFNode glyphs []
  Field MFString glyphsRefs []
  Field SFNode missingGlyphs NULL
  Field SFInt32 weight 0
  Field SFInt32 cap-height 0
  Field SFInt32 x-height 0
  Field SFInt32 ascent 0
  Field SFInt32 units-per-em 0
  Field SFString style "NORMAL"
  Field MFInt32 baseline-data []
}

NOTE - For the binary encoding of this node see Annex H.

Requirement #6
It shall be possible to:
- use line properties line-cap and line-join whose properties can be respectively butt, round, square or miter, round, bevel.
- texture the (wide) outline of shapes (including text)

Proposal
To satisfy theses requirements, we propose the definition of the following SFLLineProperties node.

9.4.2.X XLineProperties

9.4.2.X.1 Node interface

XLineProperties {
  exposedField SFColor lineColor 0, 0, 0
  exposedField SFInt32 line_style 0
  exposedField SFFloat width 1.0
  Field SFBool isScalable TRUE
  exposedField SFInt32 lineCap 0
  exposedField SFInt32 lineJoin 0
  exposedField SFInt32 miterlimit 4
  exposedField SFFloat transparency 0.0
  exposedField SFNode texture NULL
  exposedField SFNode textureTransform NULL
}

NOTE — For the binary encoding of this node see Annex H.

9.4.2.X.2 Functionality and semantics

The XLineProperties node specifies line parameters used in 2D and 3D rendering. The lineColor and the lineStyle fields are the same as for the LineProperties node. The
The **width** field determines the width, in the local coordinate system, of rendered lines. The width is subject to scaling only when the **isScalable** field is set.

The **lineCap** field shall contain the line cap style type to apply to lines. The allowed values are:

<table>
<thead>
<tr>
<th>lineCap</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>butt</td>
</tr>
<tr>
<td>1</td>
<td>round</td>
</tr>
<tr>
<td>2</td>
<td>square</td>
</tr>
</tbody>
</table>

The **lineJoin** field shall contain the line join style type to apply to lines. The allowed values are:

<table>
<thead>
<tr>
<th>lineJoin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>miter</td>
</tr>
<tr>
<td>1</td>
<td>round</td>
</tr>
<tr>
<td>2</td>
<td>bevel</td>
</tr>
</tbody>
</table>

The **miterlimit** field shall contain the limit on the ratio of the miter length to the line width as specified in the **width** field. The value of **miterlimit** must be a number greater than or equal to 1.

The **transparency** field specifies the transparency of the outline of a Shape when drawn. It supersedes the value of the transparency of a material node.

The **texture** and **textureTransform** fields are identical to those of an **Appearance** node except that texture is only applied to the outline of the shape using the same bounding box as for texturing the whole shape.

**Requirement #7**

It shall be possible to:
- use gradient and patterns as textures
Proposal
We propose to use the LinearGradient and RadialGradient nodes as defined in AFX [1] to perform gradient but we would like to define them as SFTextureNode instead of SFMaterialNode. As for Patterns, we propose to add the repeatS and repeatT fields to the compositeTexture2D and 3D nodes. Since, compositeTexture2D already has 7 fields we suggest to add one field holding both booleans, some sort of MFBool (repeatS\text{and}T).

Annex C: Registration form

<table>
<thead>
<tr>
<th>Company:</th>
<th>ENST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact name:</td>
<td>Jean-Claude Dufourd</td>
</tr>
<tr>
<td>Address:</td>
<td>46, Rue Barrault 75013 PARIS, FRANCE</td>
</tr>
<tr>
<td>Phone number:</td>
<td>+33145817807</td>
</tr>
<tr>
<td>Fax:</td>
<td>+33145804036</td>
</tr>
<tr>
<td>Email:</td>
<td><a href="mailto:dufourd@enst.fr">dufourd@enst.fr</a></td>
</tr>
</tbody>
</table>

Title of submission: Response to the CfP on Advanced Text and 2D Graphics

Abstract: This document proposes some solutions to fulfill some of the requirements expressed in the Call for Proposals issued by the MPEG-4 Systems group. The proposed solutions mainly address the 2D graphics part of the requirements.

...
References:

1. SVG 1.1, W3C, chapter 8, Paths: [http://www.w3.org/TR/SVG11/paths.html](http://www.w3.org/TR/SVG11/paths.html)
2. SVG 1.1, W3C, chapter 10, Text: [http://www.w3.org/TR/SVG11/text.html](http://www.w3.org/TR/SVG11/text.html)
5. XSL, W3C, chapter 7, Formatting Properties: [http://www.w3.org/TR/xsl/slice7.html#common-font-properties](http://www.w3.org/TR/xsl/slice7.html#common-font-properties)