INTERNATIONAL STANDARD

ISO/ASTM 52915

First edition
2013-06-01

Standard specification for additive manufacturing file format (AMF) Version 1.1

Spécification normalisée pour le format de fichier pour la fabrication additive (AMF) Version 1.1

Reference number
ISO/ASTM 52915:2013(E)

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2. www.iso.org/directives.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Neither ISO nor ASTM International shall be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received. www.iso.org/patents.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

ISO/ASTM 52915 was prepared by ASTM International (as ASTM F2915) and was adopted, under a special "fast-track procedure", by Technical Committee ISO/TC 261, *Additive manufacturing,* in parallel with its approval by the ISO member bodies. This has been done under a Partner Standards Development Organization (PSDO) Cooperation Agreement between ISO/TC 261, *Additive manufacturing,* and ASTM International Committee F42, *Additive Manufacturing Technologies.* ASTM F2915 was developed by ASTM Subcommittee F42.04, *Design.*

This first edition of ISO/ASTM 52915 cancels and replaces ASTM F2915-12.

Standard Specification for Additive Manufacturing File Format (AMF) Version 1.1[1]

This standard is issued under the fixed designation ISO/ASTM 52915; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision.

1. Scope

1.1 This specification describes a framework for an interchange format to address the current and future needs of additive manufacturing technology. For the last three decades, the STL file format has been the industry standard for transferring information between design programs and additive manufacturing equipment. An STL file contains information only about a surface mesh and has no provisions for representing color, texture, material, substructure, and other properties of the fabricated target object. As additive manufacturing technology is quickly evolving from producing primarily single-material, homogenous shapes to producing multimaterial geometries in full color with functionally graded materials and microstructures, there is a growing need for a standard interchange file format that can support these features.

1.2 The additive manufacturing file (AMF) may be prepared, displayed, and transmitted on paper or electronically, provided the information required by this specification is included. When prepared in a structured electronic format, strict adherence to an extensible markup language (XML)(**1**)[2] schema is required to support standards-compliant interoperability. The adjunct to this specification contains a W3C XML schema and Annex A1 contains an implementation guide for such representation.

1.3 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.*

1.4 *This standard also does not purport to address any copyright and intellectual property concerns, if any, associated with its use. It is the responsibility of the user of this standard to meet any intellectual property regulations on the use of information encoded in this file format.*

2. Terminology

2.1 *Definitions of Terms Specific to This Standard:*

2.1.1 This section provides definitions of terms specific to this standard—these terms also include the common terms seen in many documents related to extensible markup language (XML) and additive manufacturing. See also Annex A1 for definitions of additional terms specific to this specification.

2.1.2 *attribute, n*—characteristic of data, representing one or more aspects, descriptors, or elements of the data.

2.1.2.1 *Discussion*—In object-oriented systems, attributes are characteristics of objects. In XML, attributes are characteristics of elements.

2.1.3 *comments, n*—all text comments associated with any data within the additive manufacturing file (AMF) not containing core relevant, technical, or administrative data and not containing pointers to references external to the AMF.

2.1.4 *domain-specific applications, n*—additional, optional sets of AMF data elements specific to such areas as novel additive manufacturing processes, enterprise workflow, and supply chain management.

2.1.4.1 *Discussion*—Data sets for optional AMF domain-specific applications will be developed and balloted separately from this specification.

2.1.5 *extensible markup language, XML, n*—standard from the WorldWideWeb Consortium (W3C) that provides for tagging of information content within documents offering a means for representation of content in a format that is both human and machine readable.

2.1.5.1 *Discussion*—Through the use of customizable style sheets and schemas, information can be represented in a uniform way, allowing for interchange of both content (data) and format (metadata).

2.1.6 *STL (file format), n*—file format native to the stereolithography computer-aided drafting (CAD) software that is supported by many software packages; it is widely used for rapid prototyping and computer-aided manufacturing.

2.1.6.1 *Discussion*—STL files describe only the surface geometry of a three-dimensional object as a tessellation of triangles without any representation of color, texture, or other common CAD model attributes. The STL format specifies both the American Standard Code for Information Interchange (ASCII) and binary representations.

3. Key Considerations

3.1 There is a natural a tradeoff between the generality of a file format and its usefulness for a specific purpose. Thus,

[1] This specification is under the jurisdiction of ASTM Committee F42 on Additive Manufacturing Technologies and is the direct responsibility of Subcommittee F42.04 on Design, and is also under the jurisdiction of ISO/TC 261.
Current edition approved March 26, 2013. Published May 2013. Originally published as ASTM F2915-11. Last previous edition ASTM F2915-12.

[2] The boldface numbers in parentheses refer to the list of references at the end of this standard.

features designed to meet the needs of one community may hinder the usefulness of a file format for other uses. To be successful across the field of additive manufacturing, this file format is designed to address the following concerns:

3.1.1 *Technology Independence*—The file format shall describe an object in a general way such that any machine can build it to the best of its ability. It is resolution and layer-thickness independent and does not contain information specific to any one manufacturing process or technique. This does not negate the inclusion of properties that only certain advanced machines support (for example, color, multiple materials, and so forth), but these are defined in such a way as to avoid exclusivity.

3.1.2 *Simplicity*—The AMF file format is easy to implement and understand. The format can be read and debugged in a simple ASCII text viewer to encourage understanding and adoption. No identical information is stored in multiple places.

3.1.3 *Scalability*—The file format scales well with increase in part complexity and size and with the improving resolution and accuracy of manufacturing equipment. This includes being able to handle large arrays of identical objects, complex repeated internal features (for example, meshes), smooth curved surfaces with fine printing resolution, and multiple components arranged in an optimal packing for printing.

3.1.4 *Performance*—The file format should enable reasonable duration (interactive time) for read-and-write operations and reasonable file sizes for a typical large object. Detailed performance data are provided in Appendix X1.

3.1.5 *Backwards Compatibility*—Any existing STL file can be converted directly into a valid AMF file without any loss of information and without requiring any additional information. AMF files are also easily converted back to STL for use on legacy systems, although advanced features will be lost. This format maintains the triangle-mesh geometry representation to take advantage of existing optimized slicing algorithm and code infrastructure already in existence.

3.1.6 *Future Compatibility*—To remain useful in a rapidly changing industry, this file format is easily extensible while remaining compatible with earlier versions and technologies. This allows new features to be added as advances in technology warrant, while still working flawlessly for simple homogenous geometries on the oldest hardware.

4. Structure of This Specification

4.1 Information specified throughout this specification is stored in XML format. XML is an ASCII text file comprising a list of elements and attributes. Using this widely accepted data format opens the door to a rich host of tools for creating, viewing, manipulating, parsing, and storing AMF files. XML is human readable, which makes debugging errors in the file possible. XML can be compressed or encrypted or both if desired in a post-processing step using highly optimized standardized routines.

4.2 Another significant advantage of XML is its inherent flexibility. Missing or additional parameters do not present a problem for a parser as long as the document conforms to the XML standard. Practically, this allows new features to be added without needing to update old versions of the parser, such as in legacy software.

4.3 *Precision*—This file format is agnostic as to the precision of the representation of numeric values. It is the responsibility of the generating program to write as many or as few digits as are necessary for proper representation of the target object. However, a parsing program should read and process real numbers in double precision (64 bit).

4.4 *Future Amendments and Additions*— Additional XML elements can be added provisionally to any AMF file for any purpose but will not be considered part of this specification. An unofficial AMF element can be ignored by any reader and does not need to be stored or reproduced on output. An element becomes official only when it is formally accepted into this specification.

5. General Structure

5.1 The AMF file begins with the XML declaration line specifying the XML version and encoding, for example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

5.2 Blank lines and standard XML comments can be interspersed in the file and will be ignored by any interpreter, for example:

```
<!-- ignore this comment -->
```

5.3 The remainder of the file is enclosed between an opening `</amf>` element and a closing `</amf>` element. These elements are necessary to denote the file type, as well as to fulfill the requirement that all XML files have a single-root element. The version of the AMF standard as well as all standard XML namespace declarations can be used, such as the `lang` attribute designed to identify the human language used. The unit system can also be specified (mm, inch, ft, meters, or micrometers). In absence of a unit specification, millimeters are assumed.

```
<amf unit="millimeter" version="1.0" xml:lang="en">
```

5.4 Within the AMF brackets, there are five top level elements:

5.4.1 `<object>` —The object element defines a volume or volumes of material, each of which are associated with a material identification (ID) for printing. At least one object element shall be present in the file. Additional objects are optional.

5.4.2 `<material>` —The optional material element defines one or more materials for printing with an associated material ID. If no material element is included, a single default material is assumed.

5.4.3 `<texture>` —The optional texture element defines one or more images or textures for color or texture mapping each with an associated texture ID.

5.4.4 `<constellation>` —The optional constellation element hierarchically combines objects and other constellations into a relative pattern for printing. If no constellation elements are specified, each object element will be imported with no relative position data. The parsing program can determine the relative positioning of the objects if more than one object is specified in the file.

5.4.5 `<metadata>` —The optional metadata element specifies additional information about the object(s) and elements contained in the file.

5.5 Only a single `object` element is required for a fully functional AMF file.

6. Geometry Specification

6.1 The top level `<object>` element specifies a unique `id` and contains two child elements: `<vertices>` and `<volume>`. The `<object>` element can optionally specify a material.

6.2 The required `<vertices>` element lists all vertices that are used in this object. Each vertex is implicitly assigned a number in the order in which it was declared starting at zero. The required child element `<coordinates>` gives the position of the point in three-dimensional (3D) space using the `<x>`, `<y>` , and `<z>` elements.

6.3 After the vertex information, at least one `<volume>` element shall be included. Each volume encapsulates a closed volume of the object. Multiple volumes can be specified in a single object. Volumes may share vertices at interfaces but may not have any overlapping volume.

6.4 Within each volume, the child element `<triangle>` shall be used to define triangles that tessellate the surface of the volume. Each `<triangle>` element will list three vertices from the set of indices of the previously defined vertices. The indices of the three vertices of the triangles are specified using the `<v1>`, `<v2>`, and `<v3>` elements. The order of the vertices shall be according to the right-hand rule such that vertices are listed in counter-clockwise order as viewed from the outside. Each triangle is implicitly assigned a number in the order in which it was declared starting at zero (see Fig. 1).

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <object id="0">
    <mesh>
      <vertices>
        <vertex>
          <coordinates>
            <x>0</x>
            <y>1.32</y>
            <z>3.715</z>
          </coordinates>
        </vertex>
        <vertex>
          <coordinates>
            <x>0</x>
            <y>1.269</y>
            <z>2.45354</z>
          </coordinates>
        </vertex>
        ...
      </vertices>

      <volume>
        <triangle>
          <v1>0</v1>
          <v2>1</v2>
          <v3>3</v3>
        </triangle>
        <triangle>
          <v1>1</v1>
          <v2>0</v2>
          <v3>4</v3>
        </triangle>
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

FIG. 1 Basic AMF File Containing Only a List of Vertices and Triangles—This Structure Is Compatible with the STL Standard

6.5 *Smooth Geometry:*

6.5.1 By default, all triangles are assumed to be flat and all triangle edges are assumed to be straight lines connecting their two vertices. However, curved triangles and curved edges can optionally be specified to reduce the number of mesh elements required to describe a curved surface.

6.5.2 During read, a curved triangle patch shall be recursively subdivided into four triangles by the parsing program to generate a temporary set of flat triangles at any desired resolution for manufacturing or display. The depth of recursion shall be determined by the parsing program, but a minimal level of four is recommended (that is, convert a single curved triangle into 256 flat triangles).

6.5.3 During write, the encoding software shall determine automatically the minimum number of curved triangles required to specify the target geometry to the desired tolerance, assuming that the parser will perform at least four levels of subdivision for any curved triangle.

6.5.4 To specify curvature, a vertex can optionally contain a child element `<normal>` to specify desired surface normal at the location of the vertex. The normal should be unit length and pointing outwards. If this normal is specified, all triangle edges meeting at that vertex should be curved so that they are perpendicular to that normal and in the plane defined by the normal and the original straight edge. If a vertex is referenced by two volumes, the normal is considered separately for each volume, and its direction should be interpreted as consistent with the volume in consideration (pointing outwards). Vertices that have an ambiguous normal because they are common to multiple surfaces, should not specify a normal.

6.5.5 When the curvature of a surface at a vertex is undefined (for example, at a cusp, corner, or edge), an `<edge>` element can be used to specify the curvature of a single nonlinear edge joining two vertices. The curvature is specified using the tangent direction vectors at the beginning and end of that edge. The `<edge>` element will take precedence in case of a conflict with the curvature implied by a `<normal>` element.

6.5.6 Normals shall not be specified for vertices referenced only by planar triangles. Edge tangents shall not be specified for linear edges in flat triangles.

6.5.7 When interpreting normal and tangents, Hermite interpolation will be used. See Annex A3 for formulae for carrying out this interpolation.

6.5.8 The geometry shall not be used to describe support structure. Only the final target structure shall be described.

6.6 *Restrictions on Geometry*—All geometry shall comply with the following restrictions:

6.6.1 Every triangle shall have exactly three different vertices.

6.6.2 Triangles may not intersect or overlap except at their common edges or common vertices.

6.6.3 Volumes shall enclose a closed space with nonzero volume.

6.6.4 Volumes may not overlap.

6.6.5 Every vertex shall be referenced by at least three triangles.

6.6.6 Every pair of vertices shall be referenced by zero or two triangles per volume.

6.6.7 No two vertices can have identical coordinates.

6.6.8 The outward direction of triangles that share an edge in the same volume must be consistent.

7. Material Specification

7.1 Materials are introduced using the `<material>` element. Any number of materials may be defined using the `<material>` element. Each material is assigned a unique `id`. Geometric volumes are associated with materials by specifying a `materialid` within the `<volume>` element. Any number of materials may be defined. The `materialid` "0" is reserved for no material (void) (see Fig. 2).

7.2 Material attributes are contained within each `<material>`. The element `<color>` is used to specify the red/green/blue/alpha (RGBA) appearance of the material (see Section 8 on color). Additional material properties can be specified using the `<metadata>` element, such as the material name for operational purposes or elastic properties for equipment that can control such properties. See Annex A1 for more information (see Fig. 3).

7.3 *Mixed and Graded Materials and Substructures:*

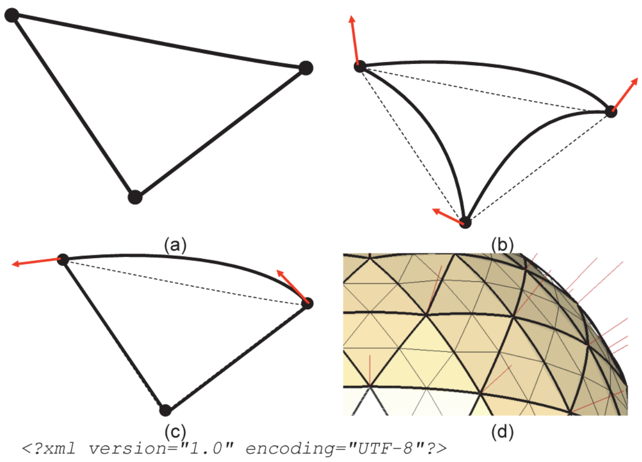
7.3.1 New materials can be defined as compositions of other materials. The element `<composite>` is used to specify the proportions of the composition as a constant or a formula dependent of the *x*, *y*, and *z* coordinates. A constant mixing proportion will lead to a homogenous material. A coordinate-dependent composition can lead to a graded material. More complex coordinate-dependent proportions can lead to nonlinear material gradients as well as periodic and nonperiodic substructure. The proportion formula can also refer to a texture map using the `tex(textureid,x,y,z)` function (see Annex A1).

7.3.2 Any number of materials can be specified. Any negative material proportion value will be interpreted as a zero proportion. Material proportions shall be normalized to determine actual ratios.

7.3.3 Although the `<composite>` element could theoretically be used to describe the complete geometry of an object as a single function or texture, such use is discouraged (but see Appendix X2). The intended use of the `<composite>` element is for the description of cellular mesostructures.

7.4 *Porous Materials*—Reference to `materialid` "0" (void) can be used to specify porous structures. The proportion of void can be either 0 or 1 only. Any fractional value will be interpreted as 1 (that is, any fractional void will be assumed fully void).

7.5 *Stochastic Materials*—Reference to the `rand(x,y,z)` function can be used to specify pseudo-random materials. For example, a composite material could combine two base materials in random proportions in which the exact proportion can depend on the coordinates in various ways. The `rand(x,y,z)` function produces a random scalar in the range [0,1) that is persistent across function calls (see Annex A4).



```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <object id="0">
    <mesh>
      <vertices>
        <vertex>
          <coordinates >
            ...
          </coordinates >
          <normal>
            <nx>0</nx>
            <ny>0.707</ny>
            <nz>0.707</nz>
          </normal>
        </vertex>
        ...
        <edge>
          <v1>0</v1>
          <dx1>0.577</dx1>
          <dy1>0.577</dy1>
          <dz1>0.577</dz1>
          <v2>1</v2>
          <dx2>0.707</dx2>
          <dy2>0</dy2>
          <dz2>0.707</dz2>
        </edge>
        ...
      </vertices>

      <volume materialid="0">
        <triangle>
          ...
        </triangle>
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

(e)

FIG. 2 (a) Default (Flat) Triangle Patch, (b) Triangle Curved Using Vertex Normals, (c) Triangle Curved Using Edge Tangents, (d) Subdivision of a Curved Triangle Patch into Four Curved Subpatches, and (e) AMF File Containing Curved Geometry

8. Color Specification

8.1 Colors are introduced using the `<color>` element by specifying the RGBA (transparency) values in a specified color space. By default, the color space shall be sRGB (**2**) but alternative profiles could be specified using the metadata tag in the root `<amf>` element (see Annex A1). The `<color>` element can be inserted at the material level to associate a color with a material, the object level to color an entire object, the

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <material id="1">
    <metadata type="Name">StiffMaterial</metadata>
  </material>
  <material id="2">
    <metadata type="Name">FlexibleMaterial</metadata>
  </material>
  <material id="3">
    <metadata type="Name">MediumMaterial</metadata>
    <composite materialid="1">0.4</composite>
    <composite materialid="2">0.6</composite>
  </material>
  <material id="4">
    <metadata type="Name">VerticallyGraded</metadata>
    <composite materialid="1">z</composite>
    <composite materialid="2">10-z</composite>
  </material>
  <material id="5">
    <metadata type="Name">Checkerboard</metadata >
    <composite materialid="1">
          floor(mod(x+y+z,1))+0.5) </composite>
    <composite materialid="2">
          1-floor(mod(x+y+z,1))+0.5) </composite>
  </material>

  <object id="0">
    <mesh>
      <vertices>
        ...
      </vertices>
      <volume materialid="1">
        ...
      </volume>
      <volume materialid="2">
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

NOTE 1—An AMF file containing five materials. Material 3 is a 40/60 % homogenous mixture of the first two materials. Material 4 is a vertically graded material and Material 5 has a periodic checkerboard substructure.

FIG. 3 Homogenous and Composite Materials

```
<?xml version="1.0"?>
<amf unit="millimeter">

  <material id="1">
    <metadata type="Name">StiffMaterial</metadata>
      <color>
        <r>0</r>
        <g>z</g>
        <b>1-z</b>
      </color>
  </material>

  <texture id="1" width="10" height="26" type="grayscale">
    TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCB
    vbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieS
    B0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvb
    SBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBh
    ...
  </texture>


  <object id="0">
    <mesh>
      <vertices>
        ...
      </vertices>
      <volume materialid="1">
        <color>
          <r>0.9</r>
          <g>0.9</g>
          <b>0.2</b>
          <a>0.8</a>
        </color>
        ...
        <triangle>
          <v1>0</v1>
          <v2>1</v2>
          <v3>3</v3>
          <texmap rtexid="1" gtexid="2" btexid="3">
            <utex1>0.1</utex1>
            <utex2>0.21</utex2>
            <utex3>0.15</utex3>
            <vtex1>0.65</vtex1>
            <vtex2>0.72</vtex2>
            <vtex3>0.91</vtex3>
          </texmap>
        </triangle>
      </volume>
    </mesh>
  </object>
</amf>
```

NOTE 1—Absolute color can be associated with a material, a volume, or a vertex. A vertex can also be associated with a coordinate in a color texture file.

FIG. 4 Color Specification

volume level to color an entire volume, a triangle level to color a triangle, or a vertex level to associate a color with a particular vertex (see Fig. 4).

8.2 Object color overrides material color specification, a volume color overrides an object color, vertex colors override volume colors, and triangle coloring overrides a vertex color.

8.3 *Graded Colors and Texture Mapping:*

8.3.1 A color can also be specified by referring to a formula that can use a variety of functions including a texture map.

8.3.2 When referring to a formula, the `<color>` element can specify a color that depends on the coordinates such as a graded color or a spotted color. Any mathematical expression that combined the functions described in Annex A2 can be used. For example, use of the `rand` function can allow for pseudo-random color patterns. The `tex` function can allow the color to depend on a texture map or image. To specify a full-color graphic, typically three textures will be needed, one for each color channel. To create a monochrome graphic, typically only one texture is sufficient.

8.3.3 When the vertices of a single triangle have different colors, the interior color of the triangle will linearly interpolate between those colors, unless a triangle color has been explicitly specified (because a triangle color takes precedence over a vertex color). If all three vertices of a triangle contain a mapping to the same texture id for any channel (*r*, *g*, *b*, or *a*), the color of this channel of the triangle will be specified by the texture map, overriding the triangle color.

8.4 *Transparency*—The transparency channel `<a>` determines alpha compositing for combining the specified foreground color with a background color to create the appearance of partial transparency. A value of zero specifies zero transparency, that is, only the foreground color is used. A value of one specifies full transparency, that is, only the background color is used. Intermediate values are used for a linear combination. Negative values are rounded to zero and values greater than one are truncated to one. The background color of a triangle is the vertex color. The background color of a vertex is the volume color, then object, and material in decreasing precedence.

9. Texture Specification

9.1 The `<texture>` element can be used to associate a `textureid` with a particular texture data. The texture map size will be specified and both two-dimensional (2D) and 3D maps are supported. The data will be an encoded string of bytes in Base64 encoding as grayscale values. Grayscale will be encoded as a string of individual bytes, one per pixel, specifying the grayscale level in the 0-255 range. The ordering of data will start with the top left corner and proceeding left to right then top to bottom. A 3D texture will specify the first layer initially and repeat for all subsequent depths into the screen according to the right-hand rule. The data will be truncated or appended with zero values as needed to meet the specified texture size.

9.2 In order to map a texture onto a triangle, the `<texmap>` element may be used to define *u*, *v*, and (optionally) *w* coordinates for each vertex of this triangle. If the texture's "tiled" property is true, any *u,v,w* mappings outside of the [0,1] range will be determined according to the coordinate modulo 1. If the texture's tiled property is false, and mappings that fall outside of the [0,1] range will return transparent. Textures shall be linearly interpolated for each triangle. A triangle shall include only a single `<texmap>` element. Overlapping textures shall be combined into a single texture before being mapped onto a mesh.

10. Print Constellations

10.1 Multiple objects can be arranged together using the `<constellation>` element (see Fig. 5). A constellation can specify the position and orientation of objects to increase packing efficiency and describe large arrays of identical objects. The `<instance>` element specifies the displacement and rotation an existing object needs to undergo into its position in the constellation. The displacement and rotation are always defined relatively to the original position and orientation in which the object was originally defined. Rotation angles are specified in degrees and are applied first to rotation about the *x* axis, then the *y* axis, and then the *z* axis.

10.2 A constellation can refer to another constellation, with multiple levels of hierarchy. However, recursive or cyclic definitions of constellations are not allowed.

```
<?xml version="1.0"?>
<amf unit="millimeter">
  <object id="1">
        ...
  </object>
  <constellation id="2">
    <instance objectid="1">
      <deltay>5</deltay>
      <rz>90</rz>
    </instance>
    <instance objectid="1">
      <deltax>-10</deltay>
      <deltay>10</deltay>
      <rz>180</rz>
    </instance>
    ...
  </constellation>

</amf>
```

NOTE 1—A constellation can assemble multiple objects together.

FIG. 5 Print Constellations

10.3 When multiple objects and constellations are defined in a single file, only the top level objects and constellations are available for printing.

10.4 The orientation at which the objects will be printed will default to those specified in the constellation. The *z* axis is assumed to be the vertical axis, with the positive direction pointing upwards and zero referring to the printing surface. The *x* and *y* directions will correspond to the main build stage axes if a gantry positioning system is used, consistent with the right hand rule.

11. Metadata

11.1 The `<metadata>` element can optionally be used to specify additional information about the objects, geometries, and materials being defined (see Fig. 6). For example, this information can specify a name, textual description, authorship, copyright information, and special instructions. The `<metadata>` element can be included at the top level to specify attributes of the entire file or within objects, volumes, and materials to specify attributes local to that entity. Annex A1 lists reserved metadata types and their meaning.

12. Compression and Distribution

12.1 An AMF shall be stored either in plain text or be compressed. If compressed, the compression shall be in ZIP archive format (3) and can be done manually or at write time using any one of several open compression libraries, such as Ref (4) .

12.2 Both the compressed and uncompressed version of this file will have the AMF extension, and it is the responsibility of the parsing program to determine whether or not the file is compressed and if so, to perform decompression during read. Any files that do not begin with an `<?xml>` tag shall be interpreted as binary xml files.

12.3 Additional files may be included in the ZIP archive such as manifest files and electronic signatures. However, only the AMF file with the same name as the archive file will be parsed. Absence of a file with that name will constitute an error.

12.4 This specification does not specify any explicit mechanisms for ensuring data integrity, electronic signatures, and encryptions.

```
<?xml version="1.0"?>
<amf unit="millimeter">
  <metadata type="Description">Product 123</metadata>
  <metadata type="Author">John Smith</metadata>
  <metadata type="CAD">SolidX 2.2</metadata>
  <metadata type="Name">Part 1</metadata>
  <metadata type="Revision">1.3A</metadata>
        ...
  <object ObjectID="0">
     <metadata type="Name">Component 1</metadata>
        ...
  </object>
</amf>
```

NOTE 1—Additional information can be stored about the object using the metadata element.

FIG. 6 Metadata