

OpenRaster

A general multilayered raster image file format

Cyrille Berger, Boudewijn Rempt

1. Goals

The main goal of OpenRaster is to allow exchange of image documents between layered raster image editors (Cinepaint, Gimp, Krita, PhotoShop¹ etc.). The document format is modelled on the OASIS OpenDocument specification. This means that it should be possible to embed OpenRaster documents in other OpenDocument documents thus enabling interoperability with word processing applications like Abiword, KWord, OOWrite and others, desktop publishing applications like Scribus, and vector graphics applications like Inkscape and Karbon14.

Currently, no standard for the interchange of complex layered raster images exist. The most used format is Photoshop, but since version 7, the Photoshop file format is closed. Tiff could conceivably be extended to include most of the requirements outlined in this document, but that would complicate an already complicated standard and would still not give us compatibility with OpenDocument.

OpenRaster will be extensible and open. Which means that no private parts (like in the tiff file format) are allowed, but if anyone needs to extend some part of the specification for a specific use he will have to publish the details of his extension. At the same time, it is not necessary to implement the whole specification.

Since OpenDocument is a format specification, there are no compliance requirements. However, this may change in the future, and implementations will only be useful when a certain identifiable baseline has been met. Applications can implement a subset of the specification and need not guarantee round-trip safety. However, applications should warn if they are about to degrade an image.

This document is the initial draft for the solicitation of input for an RFC for the definition of a schema for inclusion in the OpenDocument standard.

The authors would like to implement the resulting specification in version 2.0 of Krita, which should be released in the second quarter of 2007.

2. Preliminaries

2.1. Approach

There are two ways to approach the problem of saving complex images: saving a description of the result, or saving a description of the steps needed to arrive at the result. The former is taken by xcf, psd and Krita's old file format. The latter is shown in <http://pippin.gimp.org/gegl/gegl-demo/effect-layers-and-clones.html>. Any image format that makes adjustment layers possible is at least half way towards a programming language that describes graphics operations.

Given the intention to produce an implementable specification, half-way should be far enough.

2.2. Precision

Floats versus integers... Coordinates are proposed to be in integers. Ranges for values like

¹ Strictly alphabetic order to prevent any stepping on toes.

opacity are proposed to be floating numbers between 0 and 1.

3. Requirements

3.1. Storage

- OpenRaster should be an archive that supports inclusion of multiple file, and it should be an archive format which already exists and is well supported on unix (like zip or tar, and unlike rar), it's better if the archive format allow to select the compression level for each subfile. In order to maximize compatibility with OpenDocument, the zip file format is proposed. See Chapter 17 of “Open Document Format for Office Applications (OpenDocument) v1.0” for a discussions of the properties of zip files in this context.
- In order to facilitate partial reads, it is proposed to store data for the various elements in separate files inside the zip file. Binary elements, like pixel data, will not be stored in xml, but in a well-defined defined binary format.

3.2. Contents

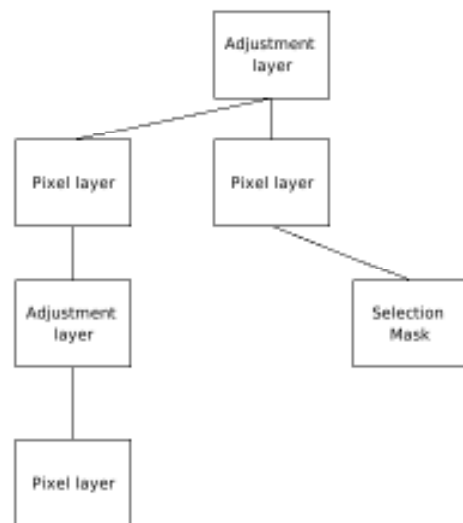
An OpenRaster image file can contain the following types of data:

- An document contents description enumerating the properties pertinent to the document and the hierarchy of layers.
- Metadata as defined in section 2.2 of the OpenDocument specification.
- Image specific metadata, such as icc profiles, exif data, colorspace definitions. These are saved as uuencoded data inside the document or as separate binary data.
- Pixel layers, a combination of binary pixel data and an XML document describing the properties of the layer, such as composition modes.
- Group layers, that allow nesting layers to an arbitrary depth.
- Adjustment layers that filter the underlying or associated layer(s).
- mask layers that extend the properties of associated layers. These can be selection layers, background/substrate layers, mask layers.
- Paths: we could possibly use embedded OpenDocument graphics documents for these.
- list of resources (brushes, gradient and pattern) used by the image, some of those resources might be included in the file, only brushes respecting an open specification like those described by Create are allowed, for instance if the resource is used with a path. This point is important for applications that want to restrict the resources that are shown in their panels.
- Embedded documents following the relevant OpenDocument specifications.

3.3. Layer structure

Layers can be grouped. It should be possible for layers of any type to occur more than once in the hierarchy². Adjustment layers can be a group layer and plain group layer can be viewed as an adjustment layer with a null filter; and any layer type can be grouped. An adjustment layer can group two (or more) hierarchies and use both hierarchies as input for the result of the group. Mask layers can be associated as “secondary” groups with layers and group layers.

In the example, and adjustment layer uses the selected part of a pixel layer as input for a function



² See <http://pippin.gimp.org/gegl/gegl-demo/dropshadow.html> for uses of such “cloning” operations.

that works on the left-hand layer stack and produces a projection.

3.4. Layers

All layers have a number of common properties. These properties are given per layer in the layer description document. This document can be embedded in the main document description, or in a separate embedded file.

- name
- versioning / collaborative work
- layer states (e.g. locked/unlocked, visible/invisible)
- position / size
- effects (like shadow)
- composition (type of blending)
- other metadata

It should be possible to associate mask layers with different but well-defined meanings with any layer.

3.4.1. Pixel layers

Pixel layers contain pixel information.

There are two possibilities: save the pixel information in a standard file format like png and jpg inside the archive, or use a (possibly run-length-encoded) tiled/scanline based pixel representation defined as follows:

The pixel information is saved in a separate file. The format of the pixel information is dependent on the colorspace. Pixel information can be saved as tiles (of any size) or scanlines, with the channels packed or separately.

Saving as scanlines is technically the same as saving in tiles, only with a height of 1 and width equal to the layer width. Conforming applications must be able to read any configuration of tile sizes.

There must be at least one complete representation at 100% zoom level of the pixel data. Optionally, representations of the layer at various zoom levels (image pyramids) may be saved.

Pixel layers in the same document need not all have the same colorspace and profile: colorspace and icc profile is a per-layer property.

Colorspaces are an extensible set, initially defined by enumeration and identified by identifier strings:

1. 8 bit rgb
2. 8 bit rgba
3. 16 bit rgb
4. 16 bit rgba
5. etc ad infinitum

3.4.2. Group layers

All layers can be grouped, including group layers. The layers constituting a group layer are composited together; the result is composited with the other layers on the level of the group layer. Adjustment layers can be group layers, too.

3.4.3. Adjustment layers

Adjustment layers apply an effect, for instance a particular filter, to a layer or a stack of layers.

Adjustment layers can be associated with a selection mask. The particular implementation of the filter is application specific. The filter configuration options are given in a separate xml document or in a subdocument in the main xml document.

Note: this may be the hardest bit to get interoperable. We may want to define a common, core set of filters and have them work similar everywhere.

Note: whether the difference between layer styles and adjustment layers is worth making is not obvious to the present author.

Adjustment layers have no size or location, although the associated selection mask has.

Adjustment layers can be group layers (or group layers can have an adjustment operation associated with them) where the layers in the group can serve as input for the adjustment operation. together

3.4.4. Mask layer

Mask layers are 8 or 16 bit or float precision raster layers stored separately but associated with a single particular layer.

Mask layers never occur on their own in the layer hierarchy, but are always dependent on another layer.

Mask layers can be used to implement selections (in most raster image editors, selections are effectively an 8 bit mask), masks or extended channel-like functionality like substrate, wetness or visibility. The function of a mask layer is indicated by an extensible set of identifiers, like:

1. selection
2. mask
3. wetness
4. heightfield
5. stickiness
6. gravity

3.4.5. Text layer

Text layers contain rich text (paragraphs, multiple fonts, bold, italic...) and can be converted to pixel layers, but not back. Text layers can be filtered by an adjustment layer. The contents of text layers should be compatible with a subset of the OASIS OpenDocument text specification, and it is debatable whether an embedded OASIS OpenDocument text would not suffice for this issue.

3.4.6. Path layer

Could be described using a subset of svg, or OpenDocument draw. This is an area where the current author is blissfully ignorant.

3.4.7. Embedded document layer

Any kind of OpenDocument document is allowed. The embedded content might be an external reference or a subfile, as per OASIS specifications ³.

3.5. Metadata

Note that this metadata is *not* the metadata as defined by OpenDocument, but data like exif data, icc profiles etc. What the Gimp calls “parasites” and Krita “annotations”.

- Metadata are stored as a triplet name, field type and value. The field type might be binary, in

³ I don't think it's a good idea to allow embedding of every type of document, from pdf to whatever.

such case, the best storing solution is likely to be encoding the data in base64 and embedded it in the xml file. Note, that binary fields must be described either in this specification or in an other documented specification

- Possibility to store metadata tags per layer
- All text data in Unicode (UTF-8 or UTF-16)

4. Compatibility

It is very unlikely that a complex image created in one application will look exactly the same in another application — even more than with text documents or html pages.

Color profiles are supposed to take care of display issues for single-layer images, but an OpenRaster document is much more complex, with multiple layers that can be combined with composite operations which might not use the same mathematics across different applications, and then there is the issue of adjustment layers which uses filters that may not be implemented in all places. Despite this, we think that it is important to strive for the best possible way of moving images between applications.

5. Example

Note that this example does not use the intended ODF XML syntax or conventions: it just cobbled together to show our intention in a more hands-on way. All these files are contained in a zip archive.

5.1. META-INF/manifest.xml

Contains a list of all component files

5.2. /meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-meta xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:xlink="http://www.w3.org/1999/xlink">
  <office:meta>
    <meta:generator>Application</meta:generator>
    <meta:initial-creator>Albert Nonymous</meta:initial-creator>
    <meta:editing-cycles>44</meta:editing-cycles>
    <meta:creation-date>2006-05-14T01:32:16</meta:creation-date>
    <dc:date>2006-06-10T22:17:14</dc:date>
    <dc:creator>Albert Nonymous</dc:creator>
    <meta:user-defined meta:name="company">valdyas.org</meta:user-defined>
  </office:meta>
</office:document-meta>
```

5.3. /content.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content>
  <IMAGE y-res="100" x-res="100"
    width="95" height="140"
    profile="sRGB built-in - (lcms internal)" colorspace="RGBA"
    name="built image">
    <LAYERS>
      <layer filtername="oilpaint" x="0" y="0" compositeop="normal" filterversion="1"
layertype="adjustmentlayer" locked="0" opacity="255" name="Oilpaint" filename="layer0"
visible="1" />
      <layer x="0" y="0" compositeop="normal" layertype="paintlayer" locked="0"
colorspace="RGBA" opacity="255" name="Layer 1" filename="layer1" visible="1" />
    </LAYERS>
  </IMAGE>
</office:document-content>
```

5.4. /preview.png

A png image of the complete rendered image in 8 (or 16) bit rgba.

5.5. **thumbnap.png**

An optional thumbnail png image of the complete rendered image.

5.6. **/image/metadata/icc**

The icc profile for the whole image

5.7. **/image/layers/layer1**

010101010101010010110010
--