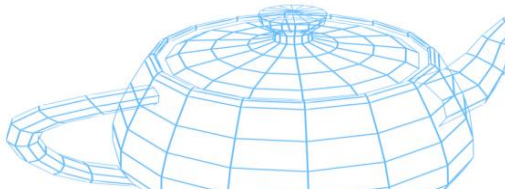
SUPSI

Computer Graphics

2D File Formats

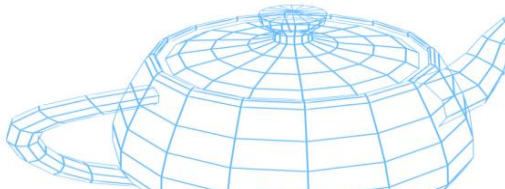
Achille Peternier, adjunct professor



Raster VS vector graphics

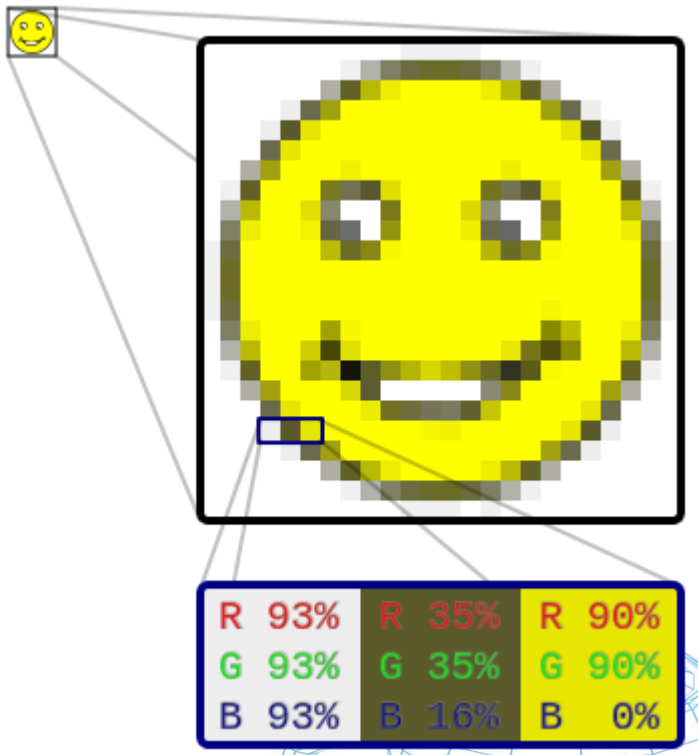
- There are two main ways for defining and storing images:
  - Using **raster** graphics (mosaic/framebuffer-like approach).
  - Using **vector** graphics (OpenGL primitive-like approach).

- Hybrid approaches also exist and combine both techniques into one same format:
  - Raster or vector graphics are used according to the needs:
    - E.g.: digital camera images = raster graphics, text paragraphs = scalable vector fonts.



Raster graphics

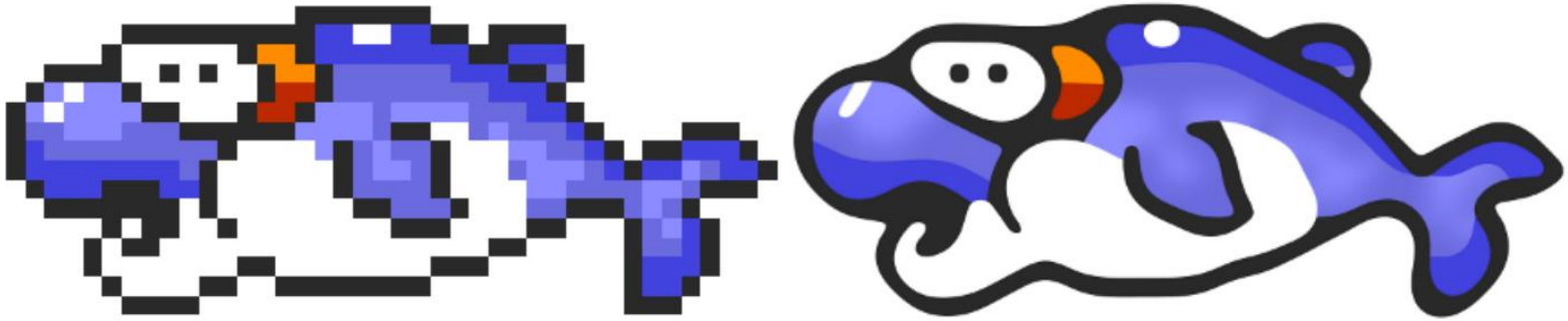
- Raster graphics are composed of a series of pixels, like the OpenGL framebuffer or a mosaic:
  - A raster is a grid of *xy* coordinates on a display space.
  - Also referred to as **bitmap** since the information is directly mapped into the display grid.



Raster graphics

- Typical raster images are acquired through camera sensors/scanners, or edited pixel by pixel by artists.

- Images have a fixed resolution:
  - Magnification/minification introduce aliasing and other artifacts:
    - Use filters to improve quality (e.g., linear filtering, as used in texture mapping).
    - Use specific filters for scaling pixel-art content:

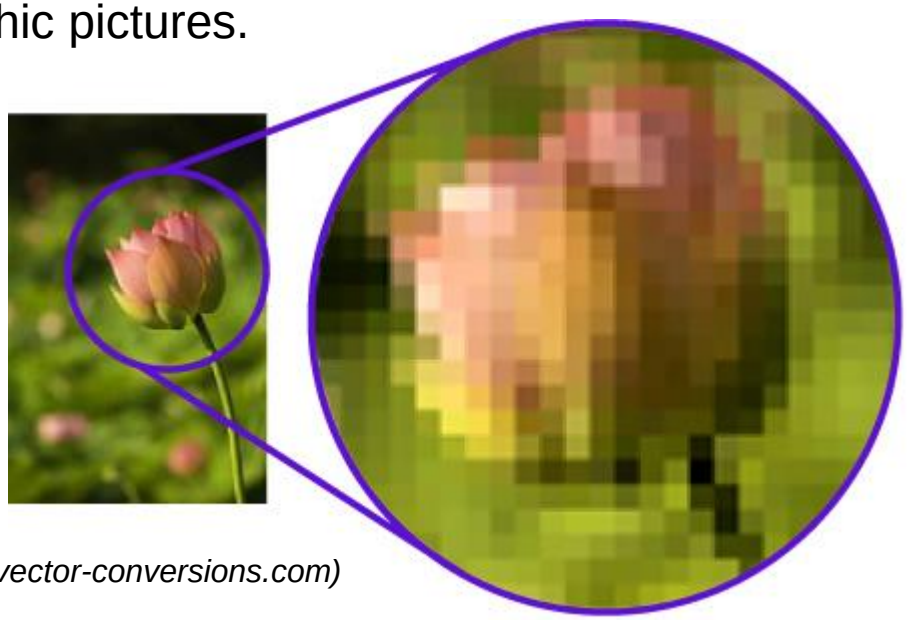


(Microsoft Research)

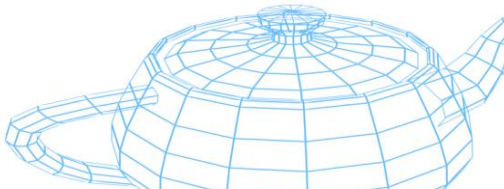
Raster graphics

- Raster data is often compressed to reduce size:
  - Uncompressed raster data corresponds to a bitmap of size $width \times height \times colorDepth$ (where *colorDepth* is the number of bytes per pixel, usually RGB with 1 byte per channel).

- Raster formats are mainly used for storing unstructured image data like photographic pictures.



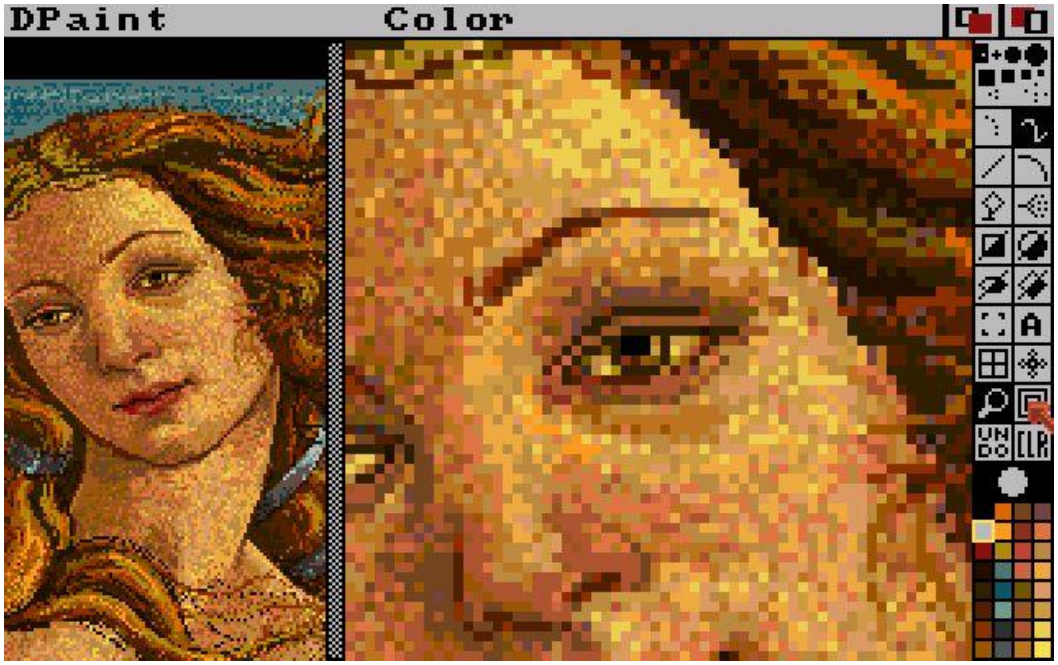
(http://vector-conversions.com)

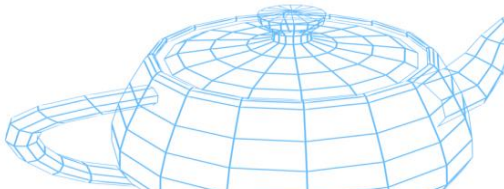


Raster graphics

- Raster graphics image editors are Photoshop, GIMP, MS Painter, Corel Painter, Deluxe Paint, etc.

- Examples of raster-based formats are BMP, JPEG, PNG, TGA, etc.





Palette

- To reduce the number of bytes required for storing each pixel, a (limited) palette of colors can be used.

- The palette works like a look-up table:

Image (no palette)

0 255 0	0 0 255	0 255 0	0 255 0	0 255 0	0 255 0

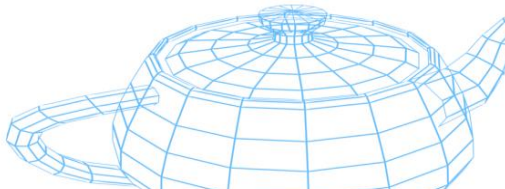
18 bytes

Image (using palette)

2	1	3	3	3	3

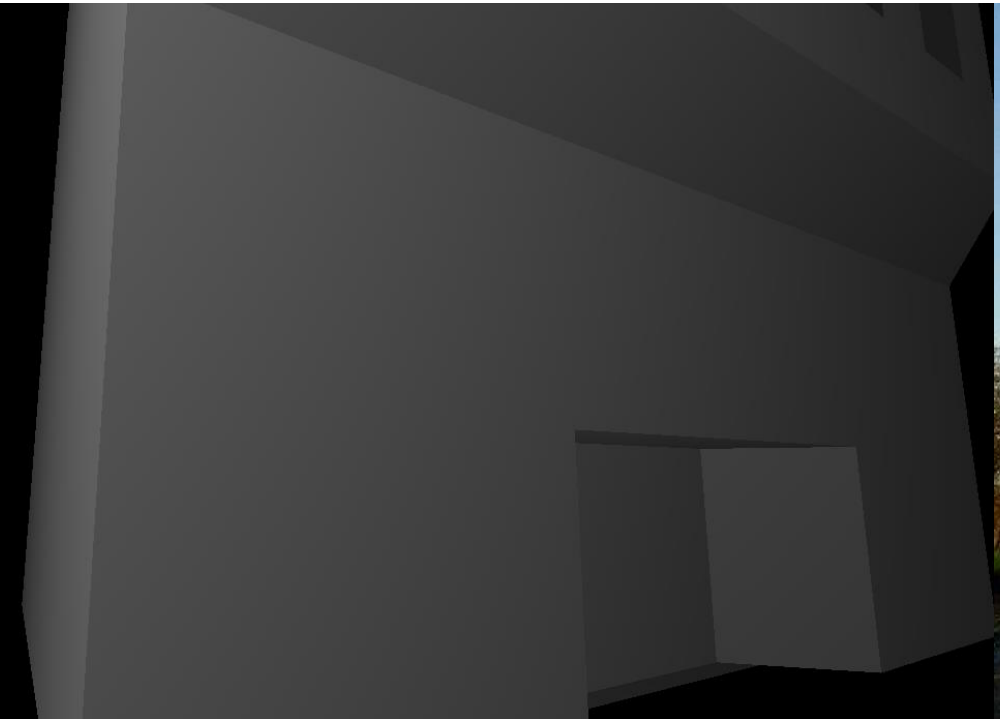
6 bytes

Palette ID	Color
1	0 0 255
2	0 255 0
3	255 0 0



Color banding

- Problem due to the lack of precision in the level of gradients used to shade a color (e.g., when 8-16 bit color depth images are generated).



Dithering

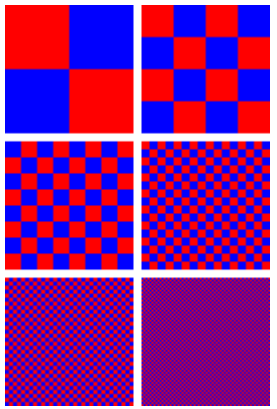
- Dithering is a form of controlled distributed noise to reduce artifacts introduced by the lack of resolution:
  - For example, when a lower color depth is used for converting a grey scale image to black and white, or for simulating colors that are not available in the palette.

Original

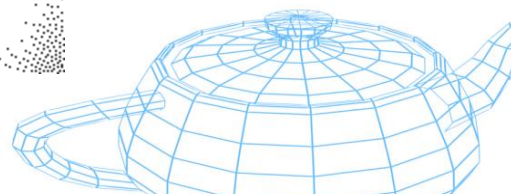
Threshold

Halftones

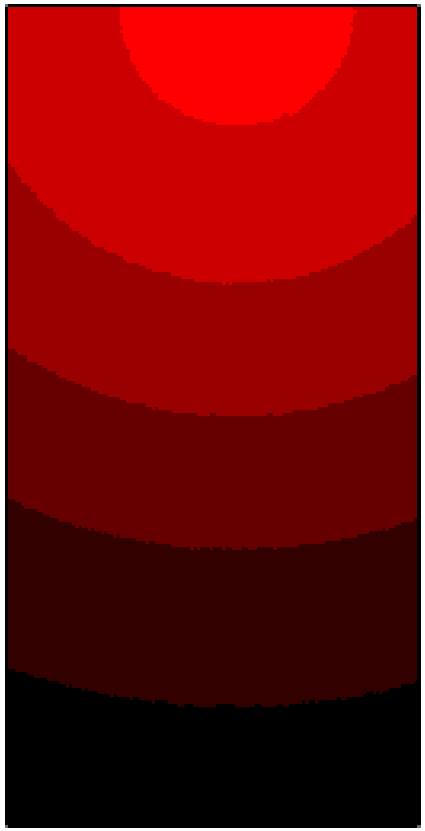
Sierra



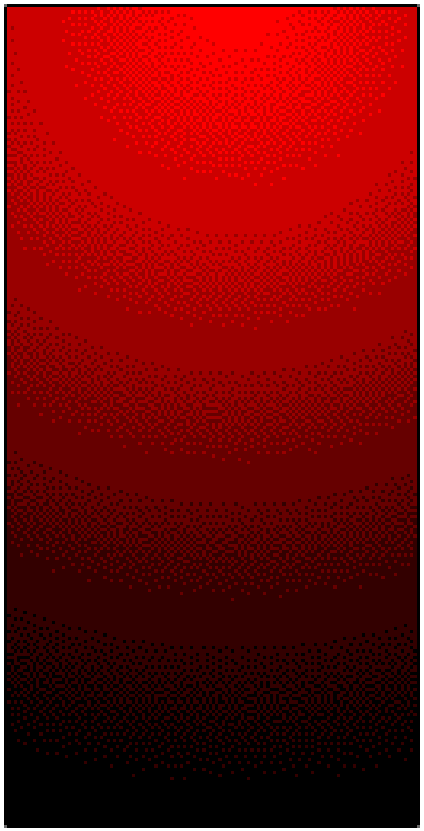
red and
blue pattern



Dithering

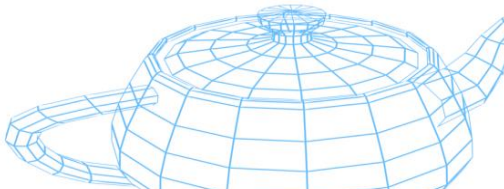


low res



low res + dithering

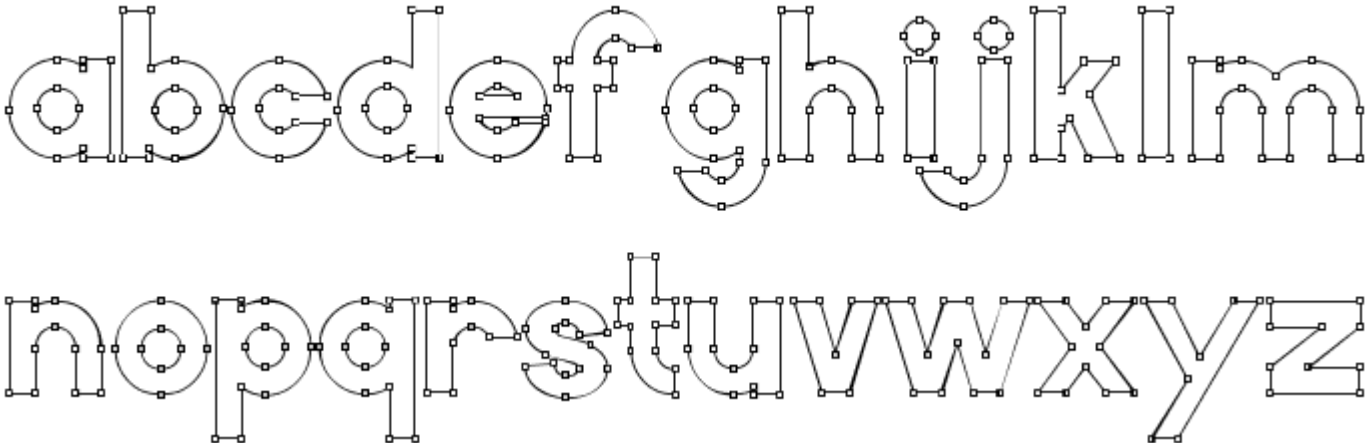
high res

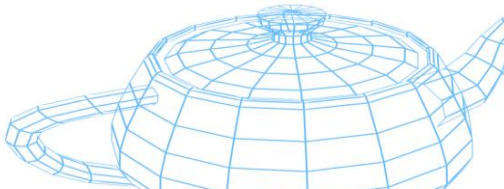


Vector graphics

- Represented by geometric primitives like points, lines, curves, etc.

- Conceptually like OpenGL: you pass a series of primitives that are evaluated, and their final resolution depends on the screen size and the various matrices involved:
  - Images are dynamically generated according to the current resolution and other parameters.
  - Perfect match between screen (150-300 DPI) and printer (>1200 DPI) resolution.

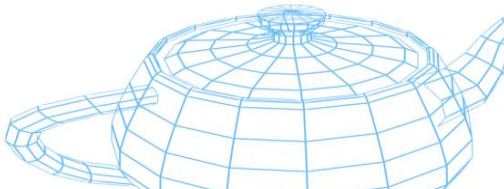




Vector graphics

- Vector graphics is particularly suitable for storing and defining structured data like blueprints, technical drawings, glyphs, clip art, logos, maps, etc.

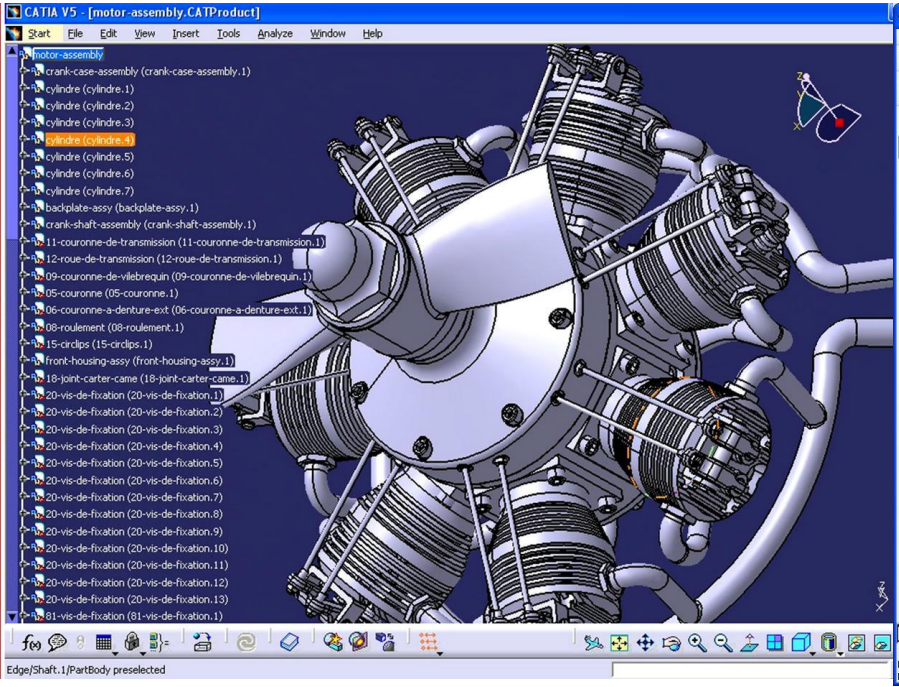
- Good information/storage ratio:
  - No need for compression, although it is always possible.
  - Bigger/smaller images have the same size.

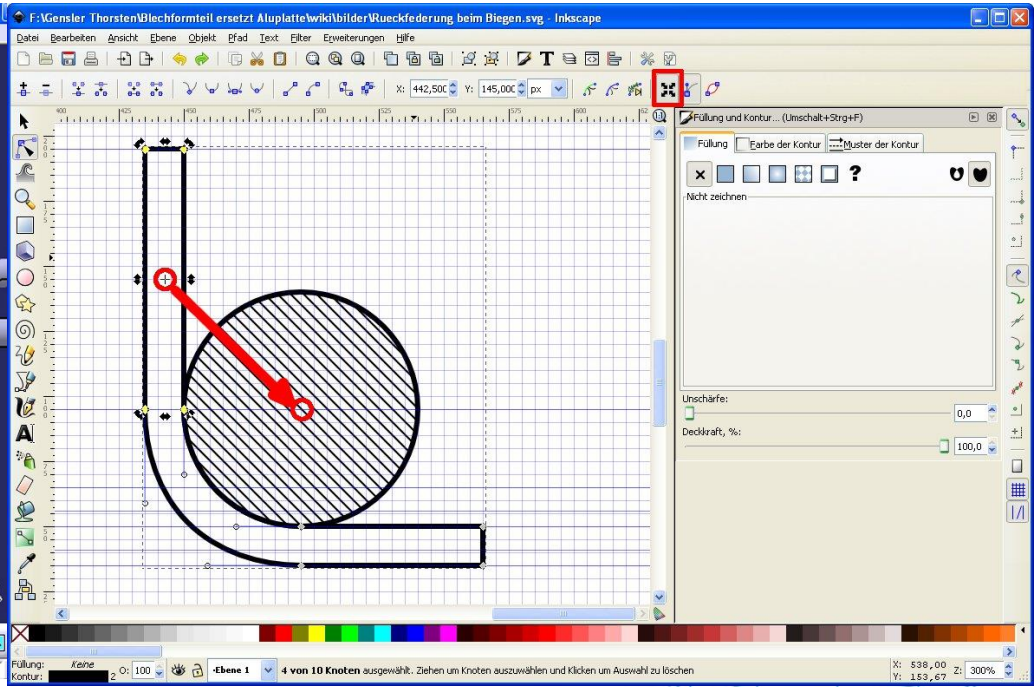


Vector graphics

- Vector graphics editors are CAD applications (Catia, AutoCAD, SolidWorks), Inkscape, Corel DRAW, Adobe Illustrator, Latex, etc.

- Examples of vector graphics file formats are SVG, EPS, CGM, DXF, STEP, etc.

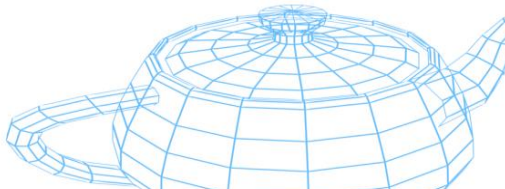




Hybrid approaches

- Several formats use the most suitable technique depending on the case:
  - E.g., using raster data for storing photos and vector data for storing fonts.

- Examples of hybrid file formats include PDF, PS, etc.

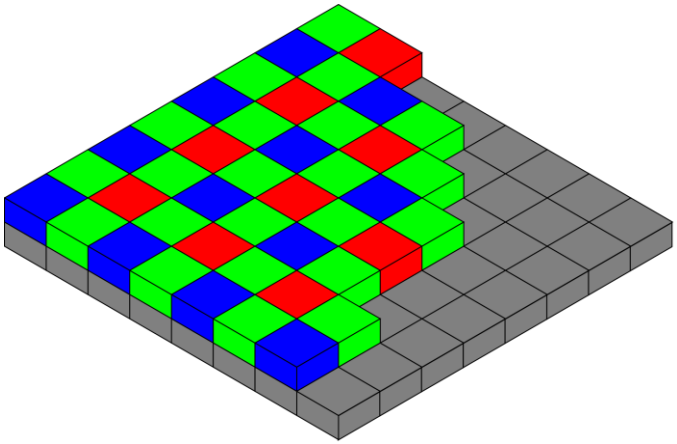


Pure raw data

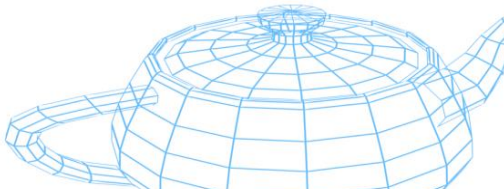
- It's an unformatted piece of raw graphics memory dumped to file, e.g., framebuffer memory dumped to disk (e.g., **`glReadPixels()`**), camera sensor intensity values, etc.

- No portability, limited usage (only as native format of the application that generated the file).

- Very difficult to encode/decode without some documentation or reverse engineering.

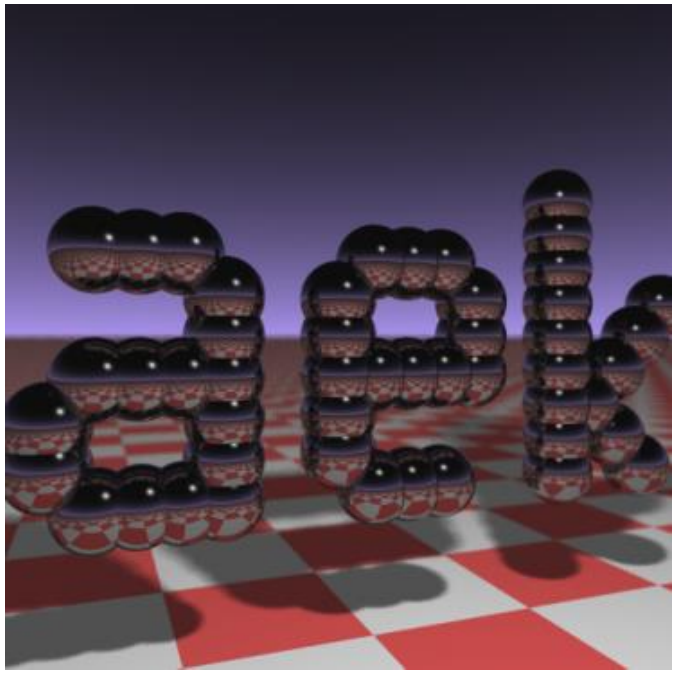


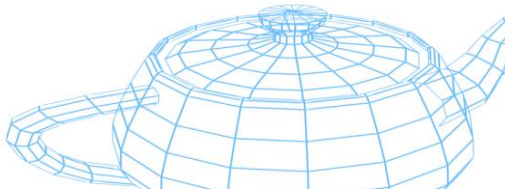
Bayer arrangement on a camera image sensor



PNM

- **P**ortable a**N**y **M**ap file formats:
  - Include PBM, PGM, and PPM.
  - Part of the *Netpbm* open package.
- Introduced by Jef Poskanzer in the ‘80s.
- Extremely simple ASCII and binary formats:
  - We already used it in the ray-tracing demo.

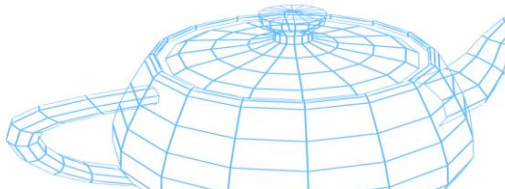




PNM

- 3 ASCII and 3 binary formats defined:
  - Bitmap (.pbm): black and white content
    - 1 bit per pixel.
  - Graymap (.pgm): single channel used for gray shades
    - 8 bits per pixel.
  - Pixmap (.ppm): three channels used (RGB values)
    - 24 bits per pixel.

Header	Type	Content
P1	Portable bitmap	ASCII
P2	Portable graymap	ASCII
P3	Portable pixmap	ASCII
P4	Portable bitmap	Binary
P5	Portable graymap	Binary
P6	Portable pixmap	Binary

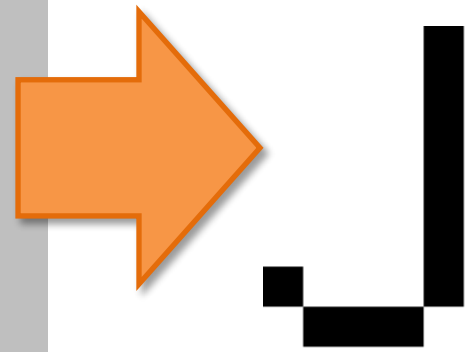


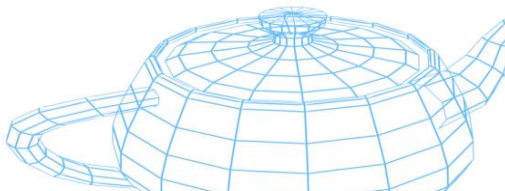
PBM

- ASCII .pbm file example:

```
P1
# This is an example bitmap of the letter "J"
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

width, height



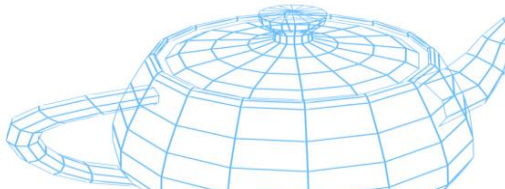


PGM

- ASCII .pgm file example:

```
P2
# Shows the word "FEEP"
24 7
9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 2 2 2 2 0 0 4 4 4 4 0 0 7 7 7 7 0 0 9 9 9 9 0
0 2 0 0 0 0 0 4 0 0 0 0 0 7 0 0 0 0 0 9 0 0 9 0
0 2 2 2 0 0 0 4 4 4 0 0 0 7 7 7 0 0 0 9 9 9 9 0
0 2 0 0 0 0 0 4 0 0 0 0 0 7 0 0 0 0 0 9 0 0 0 0
0 2 0 0 0 0 0 4 4 4 4 0 0 7 7 7 7 0 0 9 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

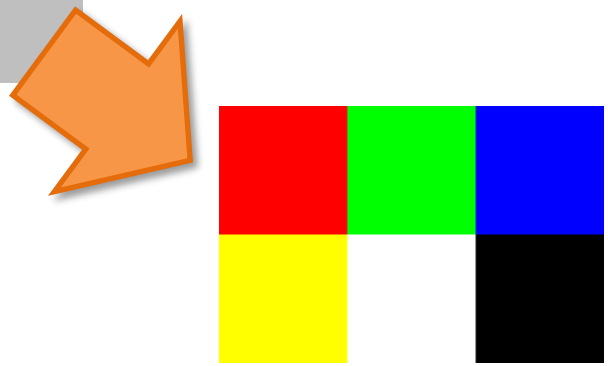
max range

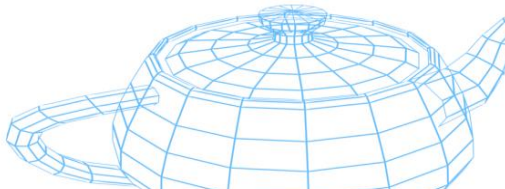


PPM

- ASCII .ppm file example:

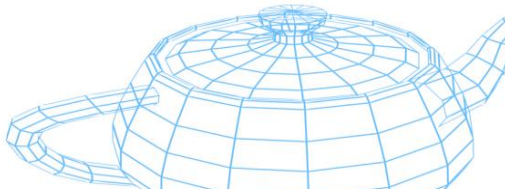
```
P3
3 2
255
255   0     0    0     255   0     0   0   255
255   255   0    255   255   255   0   0   0
```





TGA

- Introduced by Truevision Inc. (now part of Avid Technology).
- **T**ruevision [**A**dvanced **R**aster] **G**raphics **A**dapter (TGA or T[AR]GA).
- Originally, it was the native file format for the first graphics adapter for PC capable of high- and true-color graphics (16 and 24 bits per pixel).
- TGA supports 8, 15, 16, 24, and 32 bits per pixel (32 bit = 24 bit for RGB + 8 bit for alpha):
  - The 32 bit format is useful for its alpha channel to store transparent textures.



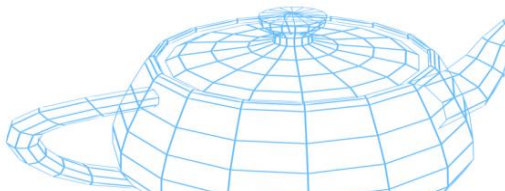
TGA

- Easy format to use.

- Each file is basically made of raw data (unless it is compressed) preceded by a compact header and an optional color map (palette) definition:
  - Header (18 bytes).
  - Image ID data (optional additional field for storing information such as image serial number, date, etc.).
  - Color map data (optional palette).
  - Image data (raw or compressed).

- Current version is 2.0 (1989):
  - Version 1.0 back-compatible.

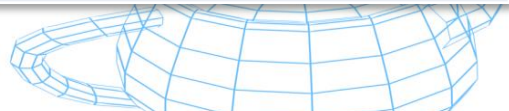
- Data is stored in little-endian.



TGA header

Field	Size	Description
ID length	BYTE	Length of the image ID field (zero if not used)
Color map type	BYTE	1 if included, 0 otherwise
Data type	BYTE	Code specifying the type of data used: 0 = no image data 1 = uncompressed color-mapped image 2 = uncompressed true-color image 3 = uncompressed black-and-white (grayscale) image 9 = run-length encoded color-mapped image 10 = run-length encoded true-color image 11 = run-length encoded black-and-white (grayscale) image
Color map origin	WORD	Color map starting offset
Color map length	WORD	Number of elements in the color map
Color map depth	BYTE	Number of bits used for each element

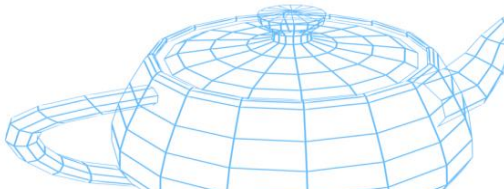
...



TGA header

...

Field	Size	Description
Origin X	WORD	Lower-left absolute X origin coordinate
Origin Y	WORD	Lower-left absolute Y origin coordinate
Width	WORD	Image width in pixels
Height	WORD	Image height in pixels
Color depth	BYTE	Number of bits per pixel
Image descriptor	BYTE	bits 3-0 give the alpha channel depth, bits 5-4 give direction



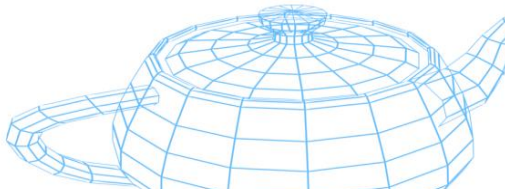
TGA optional sections

(only if *ID length* > 0)

Field	Size	Description
Image ID content	*ID length* bytes	Additional information such as image serial number, date, etc.

(only if *Color map type* = 1, repeated *color map length* times)

Field	Size	Description
Color map element	*Color map depth* bits	Color description for each element of the color map



TGA image data

(if uncompressed [*data type = 1, 2, 3*])

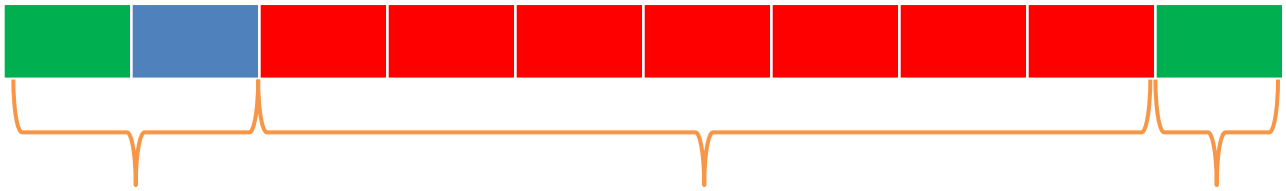
Field	Size	Description
Image data	*width * height * color depth* bytes	Uncompressed data is stored *as is*, like a memory dump. If a color map is used, each entry in the image data must be replaced by the color map element it is referring to.

(if compressed [*data type = 9, 10, 11*])

Field	Size	Description
Image data	must be processed	Data is stored using a simple compression algorithm called run-length encoding: if the first byte is 0, the next byte tells how many different pixels are specified starting from the next-next byte. If the first byte is 128, the second byte tells how many identical pixels will follow looking like the pixel specified starting from the next-next byte.

Run length encoding

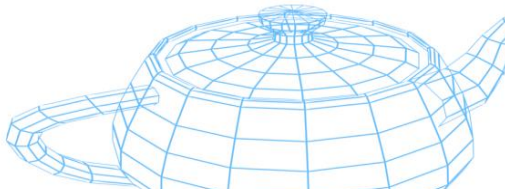
- Simple compression algorithm.
- If an image row contains N identical pixels in sequence, just store the pixel color and the number of times it is repeated.



Chunk ID: 0
Length: 2
Pixel: 0 255 0
Pixel: 0 0 255

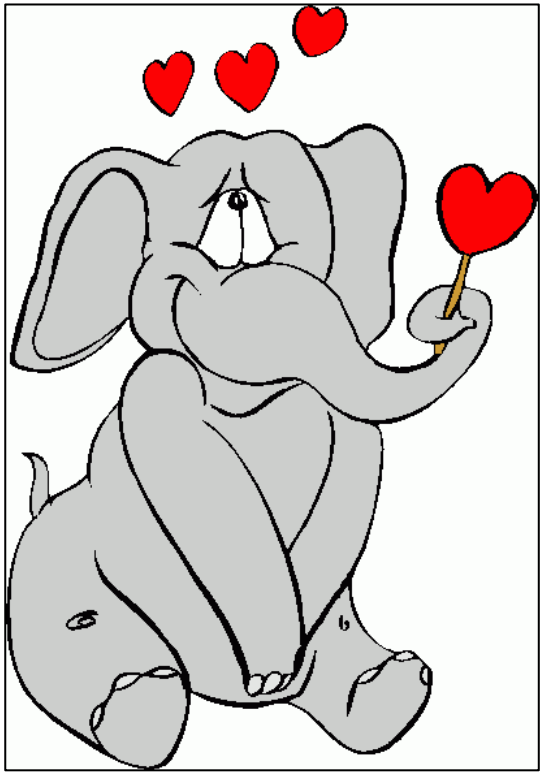
Chunk ID: 128
Length: 7
Pixel: 255 0 0

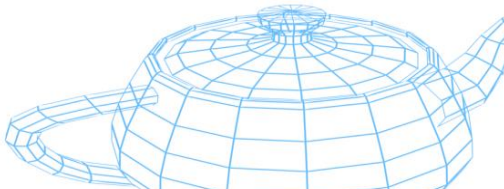
Chunk ID: 0
Length: 1
Pixel: 0 255 0



Run length encoding

- Works well for cartoon-like, manually painted images (e.g., pixel art).
- Very inefficient for real images (photos, scans) and for pictures using many gradients.





JPEG

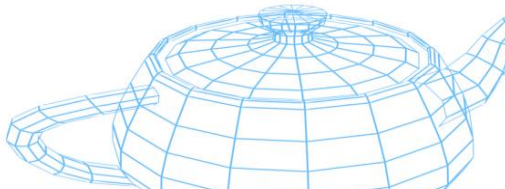
- **J**oint **P**hotographic **E**xpert **G**roup.

- Lossy compression format:
  - Some data is modified or lost to improve compression:

decompressed image != original image

    - As it happens with MP3, MPEG, OGG, …

- Uses a series of complex algorithms to compress the bitmap, since the human eye is less sensitive to specific light components (e.g., high frequencies).



JPEG

- You should avoid/limit the usage of JPEG as the file format for your textures:
  - Lossy compression is optimized for static images seen from a specific point of view.
  - Do not use JPEG for frequently modified files but only for the final output.
  - Texture interaction with filtering, interpolation, lighting model, etc., can lead to unwanted results.

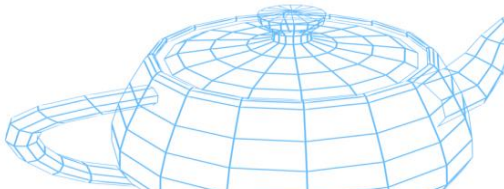
Quality = 100

Quality = 10

Quality = 1

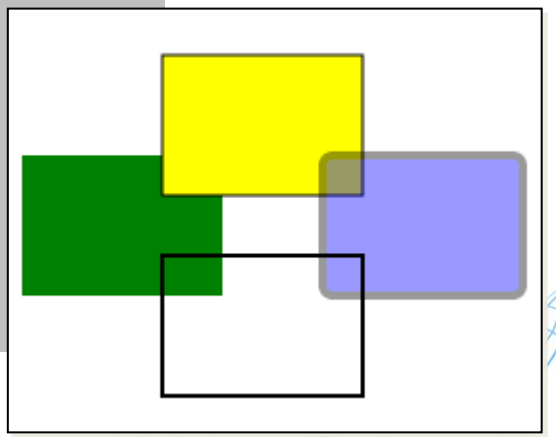
SVG file format

- **S**calable **V**ector **G**raphics.
- Open-standard introduced by the WWW Consortium in 1999.
- XML-based:
  - Can be edited as text files.
  - Can be searched/parsed for specific strings/values.
  - Can be compressed (as any other text file).
- Modern Web browsers can render them.



SVG example

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="400" height="400" xmlns="http://www.w3.org/2000/svg">
  <title>Quads</title>
  <desc>Four quads with different attributes</desc>

  <rect x="50"  y="150" width="100" height="70"  fill="green"/>
  <rect x="120" y="100" width="100" height="70" fill="yellow"
  stroke="black" />
  <rect x="200" y="150" width="100" height="70" rx="5"
  fill="blue" opacity="0.4" stroke="black" stroke-width="4"/>
  <rect x="120" y="200" width="100" height="70" fill="none"
  stroke="black" stroke-width="2"/>
</svg>
```



SVG example

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
```

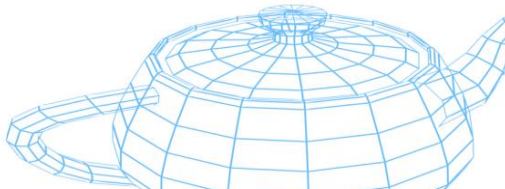
Specify the version and encoding used for this XML file

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
```

Specify that we are using an SVG file using the Document Type Definition declared in file svg10.dtd

```
<svg width="400" height="400" xmlns="http://www.w3.org/2000/svg">
```

Image width, height and namespace used



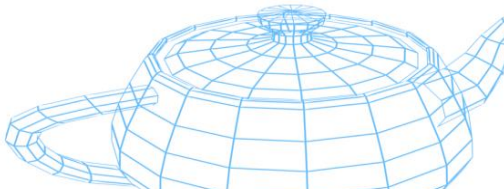
SVG example

```
<title>Quads</title>
<desc>Four quads with different attributes</desc>
```

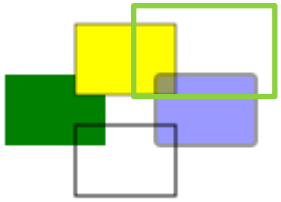
Additional details not used for rendering but useful for parsing the SVG

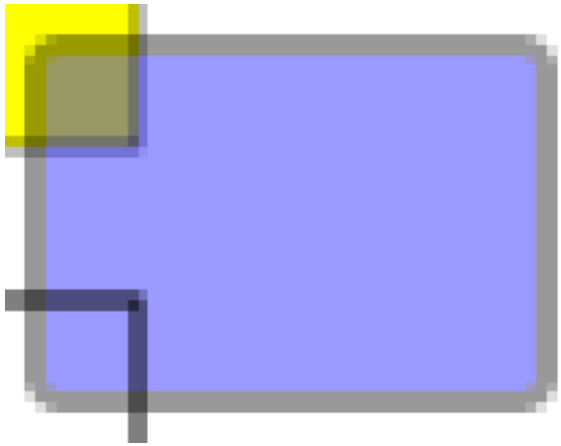
```
<rect x="50"  y="150" width="100" height="70"  fill="green"/>
<rect x="120" y="100" width="100" height="70" fill="yellow"
 stroke="black" />
<rect x="200" y="150" width="100" height="70" rx="5" fill="blue"
 opacity="0.4" stroke="black" stroke-width="4"/>
<rect x="120" y="200" width="100" height="70" fill="none"
 stroke="black" stroke-width="2"/>
```

Defines four quads with various positions and attributes (other primitives such as *ellipse*, *line*, *circle*, etc. also exist)

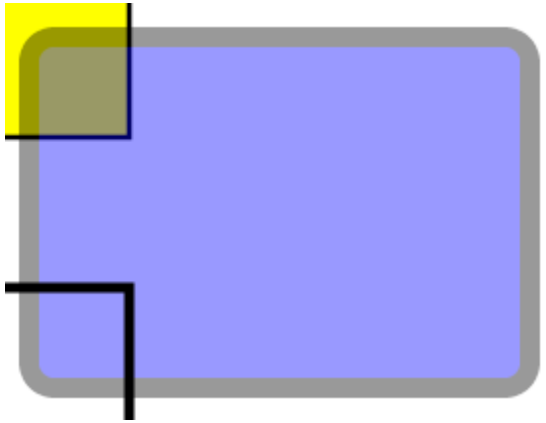


SVG example

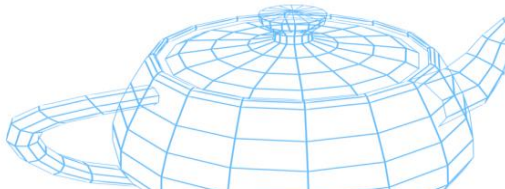




Raster



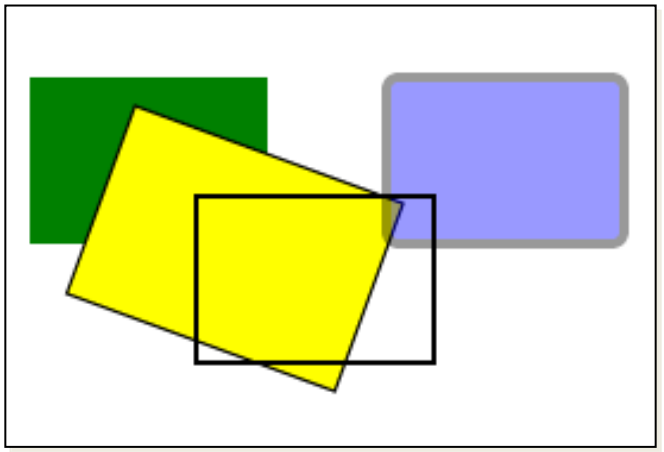
Vector

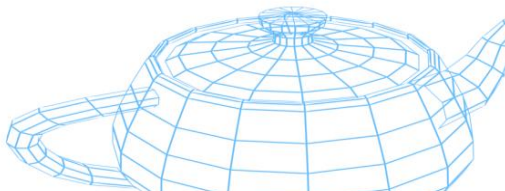


SVG

- SVG also supports transformations:

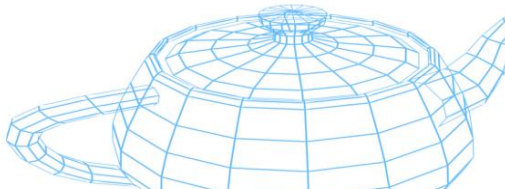
```
<rect x="120" y="100" width="100" height="70"
 fill="yellow" stroke="black" transform="scale(1.2) rotate(20)" />
```





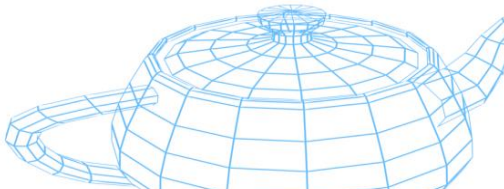
SVG

- Instead of creating SVG files manually, use an editor (e.g., Inkscape).
- SVG syntax and attributes are like the ones used in CSS files.



PDF

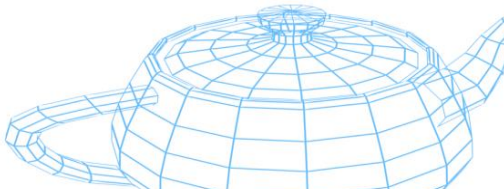
- **P**ortable **D**ocument **F**ormat.
- Introduced by Adobe Systems in 1993.
- Open, license-free standard since 2008.



PDF

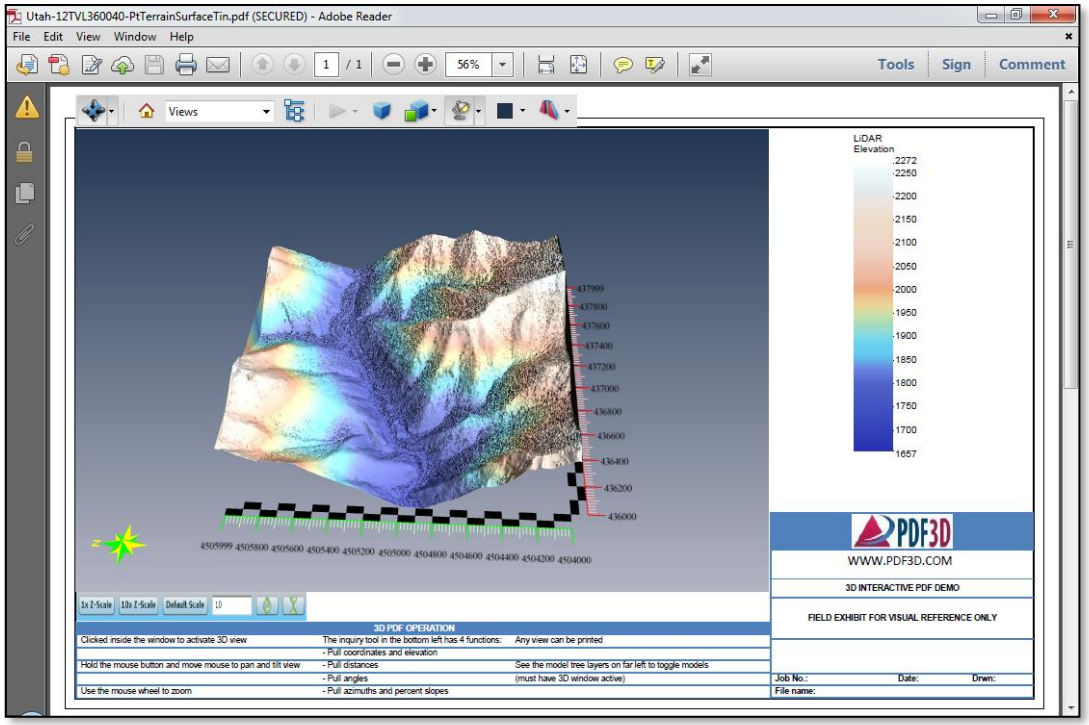
- A PDF file is a portable, hardware-independent container of objects used to define a document with a static layout:
  - Fonts, images, text, etc.

- Images can be both raster or vector graphics:
  - Vector graphics are defined in a PostScript-like way.
  - Raster graphics are stored using various internal filters that are like the compression techniques used in other file formats (e.g., RLE, JPEG2000, GZIP, …).



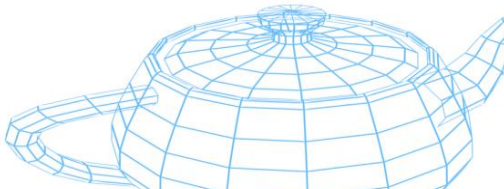
PDF

- PDF files can also contain 3D objects:
  - The PDF viewer embeds a 3D model viewer.



PDF

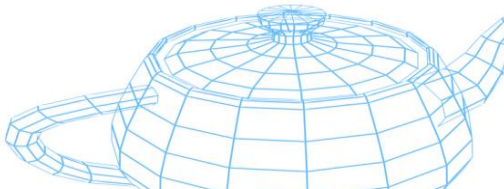
- PDF files are not often used in 3D graphics but are more suitable for generating portable, device-independent documents (e.g., using Microsoft Word, LibreOffice, Latex, etc.) and for printing.



Documentation

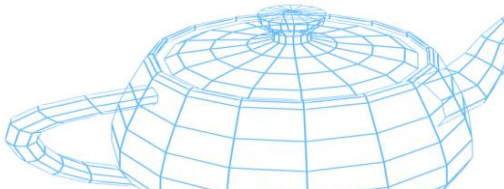
- James D. Murray, William van Ryper, *Encyclopedia of Graphics File Formats*, 2nd Edition, May 1996, ISBN: 1-56592-161-5

- http://www.fileformat.info/



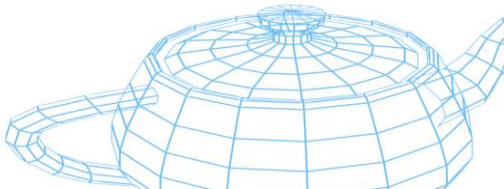
Libraries

- Encoding/decoding graphics file formats can be a time-consuming and complex task:
  - Use external libraries when possible (libJPEG, libPNG, etc.).
  - Higher-level libraries also exist (FreeImage, DevIL, SOIL, etc.):
    - Multiple file formats supported.
    - Additional functions provided (color depth conversion, resizing, etc.).



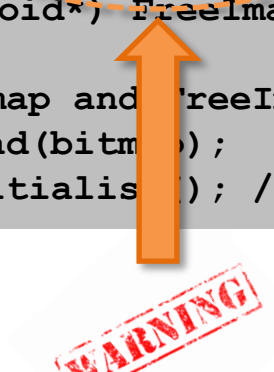
FreeImage

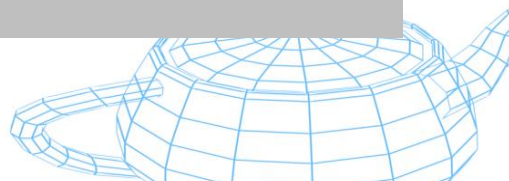
- Open-source library supporting popular graphics image formats like PNG, BMP, JPEG, TIFF, and more.
- Easy to use, fast, thread-safe, and cross-platform.
- Available at: http://freeimage.sourceforge.net/



FreeImage

```
#define FREEIMAGE_LIB     // Static lib only, put it in the project props
#include <FreeImage.h>

// Init FreeImage:
FreeImage_Initialise();   // Static lib only

// Load an image from file:
FIBITMAP *bitmap = FreeImage_Load(FreeImage_GetFileType("teapot.tga", 0),
                                  "teapot.tga");

// Load image into OpenGL:
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,
             FreeImage_GetWidth(bitmap), FreeImage_GetHeight(bitmap),
             0, GL_BGRA_EXT, GL_UNSIGNED_BYTE, // FreeImage uses BGR
             (void*) FreeImage_GetBits(bitmap));

// Release bitmap and FreeImage:
FreeImage_Unload(bitmap);
FreeImage_DeInitialise(); // Static lib only
```



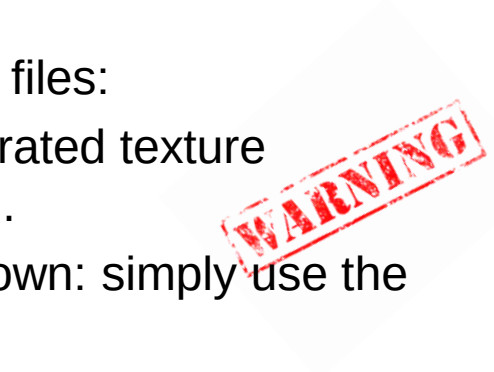


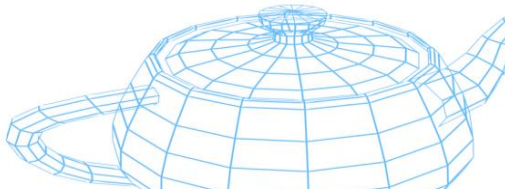
FreeImage

- On Windows:
  - Use the version provided on iCorsi (Windows, static lib only).

- On Ubuntu:
  - **`sudo apt install libfreeimage-dev`**

- Textures exported via the over3ds plugin are saved as DDS files:
  - This is a special format for better exploiting GPU-accelerated texture compression (more on that in the VR and ACG courses).
  - FreeImage supports them but they will appear upside-down: simply use the **`FreeImage_FlipVertical()`** method to fix that.





GIMP

- **G**NU **I**mage **M**anipulation **P**rogram.

- Free Photoshop-like software:
  - Multiplatform.
  - Supports several file formats (either natively or via additional plugins).

- Useful for converting/resizing images to be later used as textures.

- http://www.gimp.org/