Department of Economic Informatics and Cybernetics
Bucharest University of Economic Studies

Multimedia

Liviu-Adrian Cotfas, PhD.

liviu.cotfas@ase.ro

Few words about me…

https://ro.linkedin.com/in/cotfasliviu

Evaluation

- Final test – 60%

- Seminar – 40%
  - project (mandatory)

Recommended Reading / Watching

- Slides, Examples, Books:
  - http://liviucotfas.ase.ro

Further Reading / Watching

- Courses on **Microsoft Virtual Academy** - mva.microsoft.com
  - Free

- Courses on **PluralSight** - www.pluralsight.com
  - Free trial
  - Free access (limited period) through Microsoft DreamSpark

Definition, Characteristics, Concepts

Definition

- **Multimedia** is:
  - any combination of **text**, **graphics**, **video**, **audio**, and **animation**
  - in a distributable format that consumers can interact with using a digital device.

- **Multimedia** can be thought of as a super-medium of sorts, because it represents the blending together of previously distinct, and noncombinable, forms of human expression

Triggering factors

- The development of multimedia has been made possible by the **Digital Revolution**, through:
  - analog-digital conversion;
  - data compression.

- The **Digital Revolution** represents the change from mechanical and analogue electronic technology to digital electronics which began anywhere from the late 1950s to the late 1970s with the adoption and proliferation of digital computers and digital record keeping that continues to the present day.

Data Compression

- There are two major categories of compression algorithms:
  - Lossless compression
  - Lossy compression
- Compression algorithms used in multimedia are usually **asymmetrical** – the compression time is grater than the decompression time.

Lossless compression

- substitutes a more efficient encoding to reduce the file size while preserving all of the original data. When the file is decompressed it will be identical to the original.
- Run Length Encoding - RLE
  - one of the simpler strategies to achieve lossless compression
  - can be used to compress bitmapped image files (ex: *pcx format). Bitmapped images can easily become very large because each pixel is represented with a series of bits that provide information about its color. RLE generates a code to "flag" the beginning of a line of pixels of the same color. That color information is then recorded just once for each pixel. In effect, RLE tells the computer to repeat a color for a given number of adjacent pixels rather than repeating the same information for each pixel over and over. The RLE compressed file will be smaller, but it will retain all the original image data—it is "lossless."

Run Length Encoding - RLE

Lossy Compression

- the number of bits in the original file is reduced and some data is lost. Lossy compression is not an option for files consisting of text and numbers, so-called *alphanumeric* information. Losing a single letter or number could easily alter the meaning of the data.

- often possible to maintain high-quality images or sounds with less data than was originally present (especially useful for multimedia)

- exploits the limits in human perception

Lossy Compression

- Examples:
  - JPEG – images;
  - MPEG – sound and video.

- MP3 compression:
  - analyzes the sound file and discards data that is not critical for high-quality playback;
  - removes frequencies above the range of human hearing. It may also evaluate two sounds playing at the same time and eliminate the softer sound. These types of data can be eliminated without significant impact on quality;
  - can reduce by a factor of 10 the amount of data required to represent digital audio recordings yet still sound like the original uncompressed audio to most listeners.

Multimedia Applications

- Characteristics:
  - Multimedia systems must be computer controlled.
  - Multimedia systems are integrated.
  - The information they handle must be represented digitally.
  - The interface to the final presentation of media is usually interactive.

Approaches for building multimedia applications

- Multimedia authoring – high-level
  - involves the assembly and bringing together of Multimedia with possibly high level graphical interface design and high level scripting
  - well-known multimedia authoring software: Animate / Flash (Adobe), Director (Adobe)

- Multimedia programming – low-level
  - assembly, construction and control of Multimedia that involves programming languages such as C, C# and Java and specialized libraries.

Classification of multimedia applications

- Serval classification criteria based on:
  - domain: economy, education, advertising, medicine, industrial, entertainment, navigation and information systems, communications;
  - interactivity: interactive, static;
  - location: local, remote (video-streaming, distant learning).

Hardware / Software Requirements

Hardware Requirements

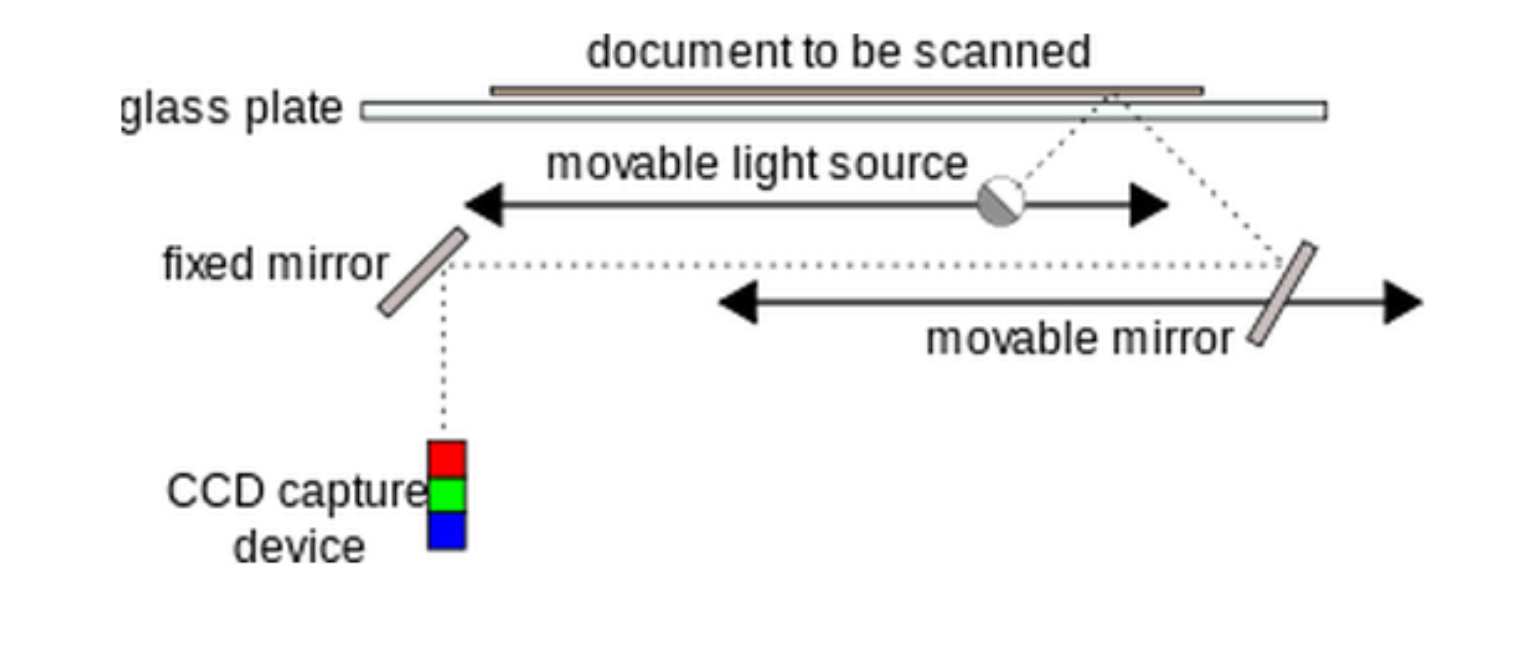
- Input and processing hardware for:
  - Images;
  - Sound;
  - Video.

Input Devices - Images

- Scanners
  - capture text or images using a light-sensing device.
  - types of scanners: flatbed, sheet fed, and handheld
  - operation: a light passes over the text or image, and the light reflects back to a **CCD** (charge-coupled device). A CCD is an electronic device that captures images as a set of analog voltages. The analog readings are then converted to a digital code by another device called an **ADC** (analog-to-digital converter) and transferred through the interface connection (usually USB) to RAM.
  - Optical character recognition (OCR) is the process of converting printed text to a digital file that can be edited in a word processor.

Input Devices - Images

- scanner

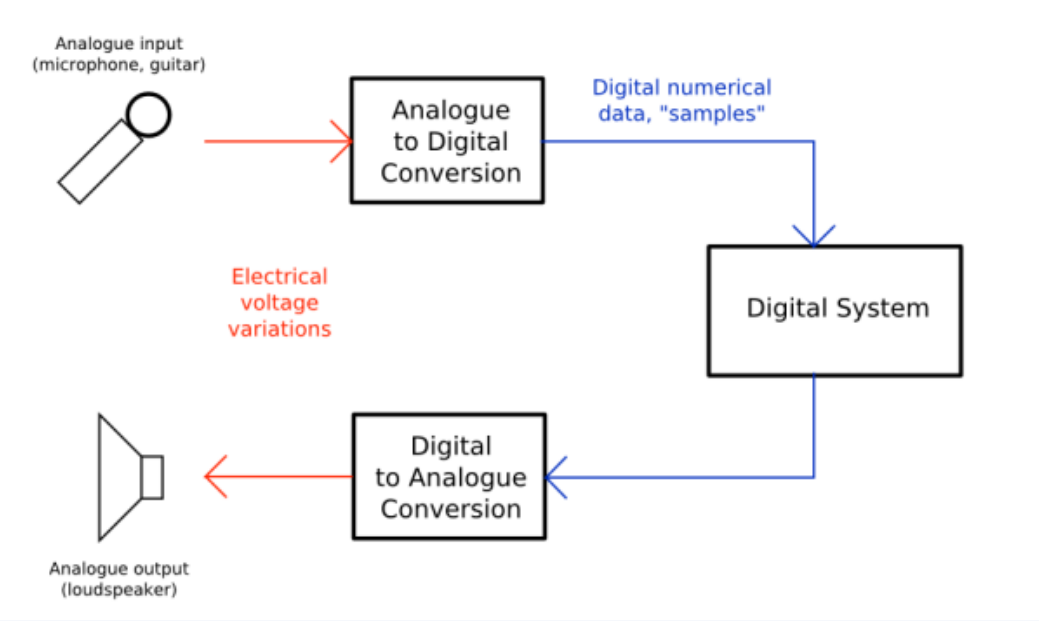


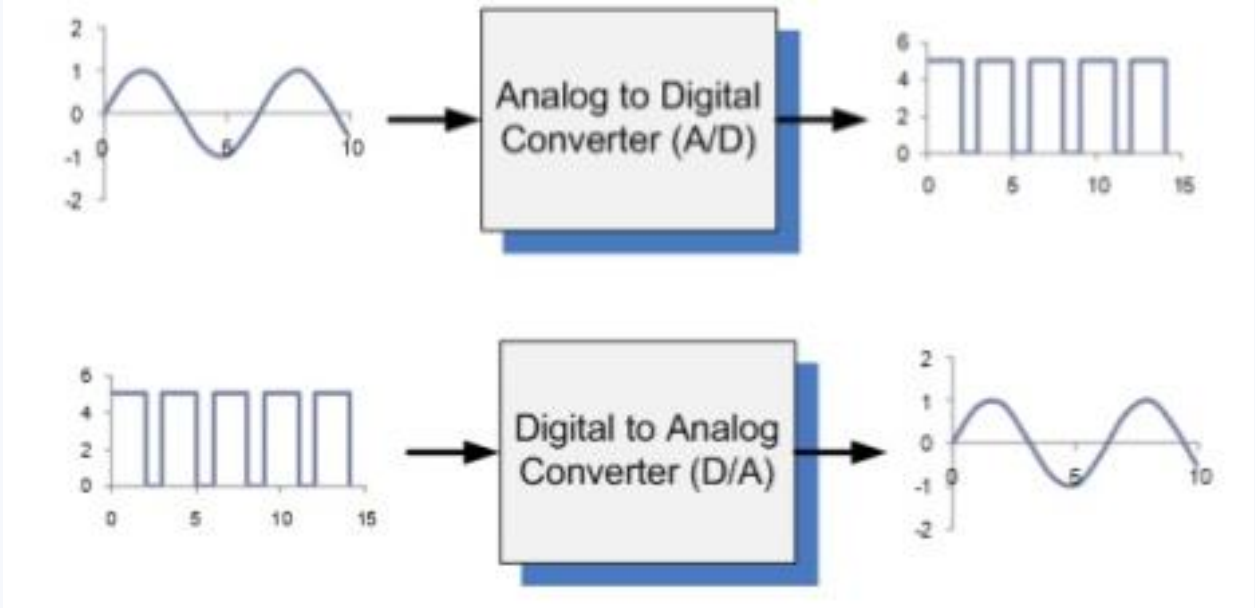
Input Devices - Images

- Digital cameras
  - When the camera shutter is opened to capture an image, light passes through the camera lens. The image is focused onto a CCD, which generates an analog signal.
  - The analog signal is converted to digital form by an ADC and then sent to a digital signal processor (DSP) chip that adjusts the quality of the image and stores it in the camera's built-in memory or on a memory card.

Input Devices - Sound

- Sound card
  - converts the analog signal from the microphone to a digital representation;
  - converts the digital representation to a analogue one that can be played back by the speakers.





Input Devices - Video

- Video card
- Digital video camera

Software Requirements

- Device drivers - computer program that operates or controls a particular type of device that is attached to a computer. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used.

- OS Utility Multimedia Applications – music player, video player, image viewer, basic image editor (ex: Windows Media Player)

- Application Software

Application Software

- An application is software that performs a specific task. These programs combine with the operating system to make computers productive tools.

- There are two major types of software for multimedia development:
  - **Media-specific** applications are used to create and edit the individual media elements (text, graphics, sound, video, animation) that make up a multimedia product.

  - **Authoring** applications contain software tools to integrate media components and provide a user interface.

Graphics

- generate 2-D or 3-D paint and draw images.

- Categories:
  - Paint programs;
  - Draw programs;
  - 3-D imaging programs.

Paint programs

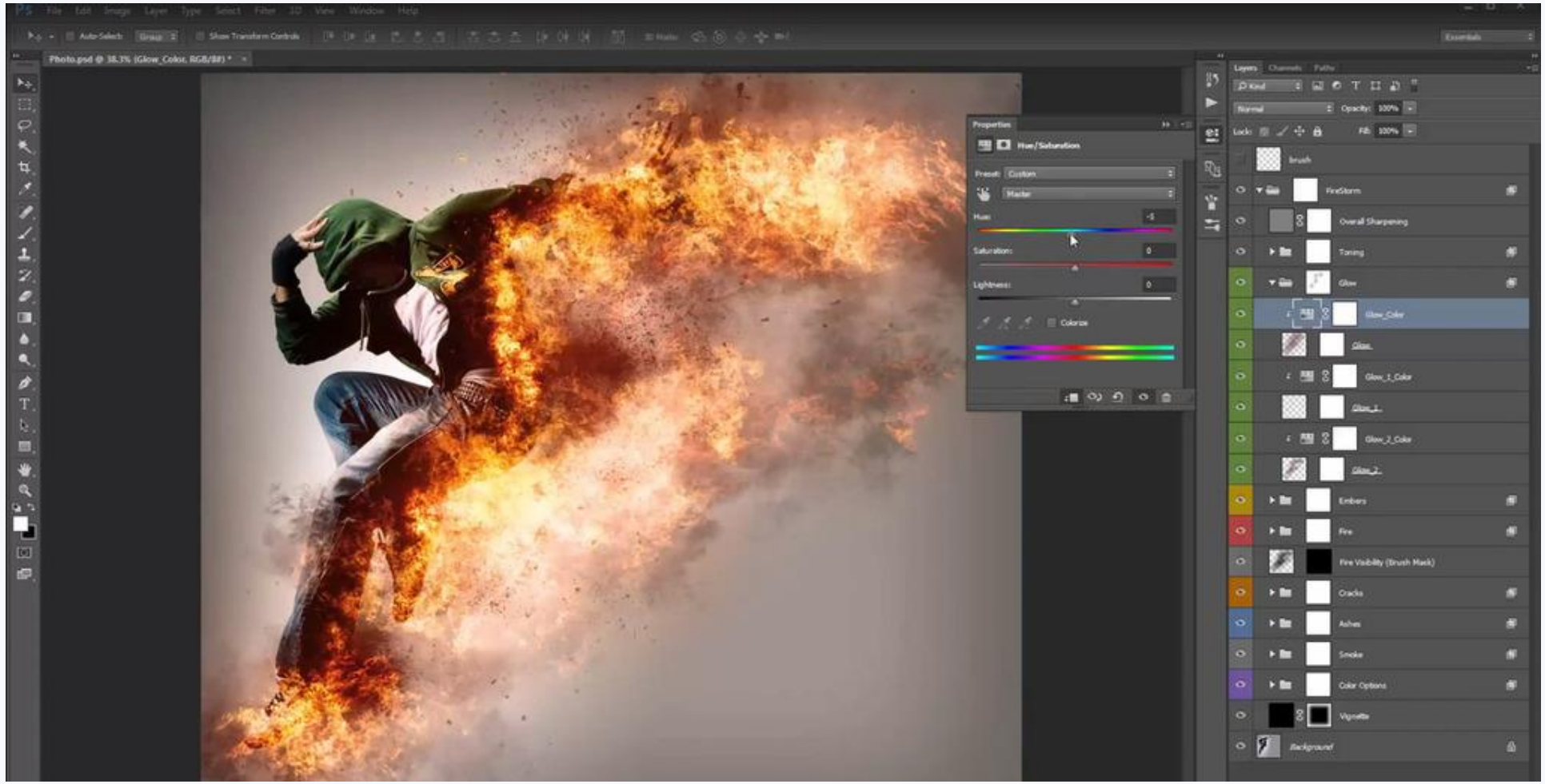
- contain tool sets to create graphics objects as well as editing tools for digital photos or scanned images

- offer a wide array of features such as filters (blur, emboss, pixelate), image adjustment settings (scale, brightness, rotate), and special effects (drop shadow, gradient overlay).

- provide special control over individual image elements is possible using layers and mask options. Text tools are used to generate graphics text with distinctive patterns, shapes, and 3-D effects.

Paint programs

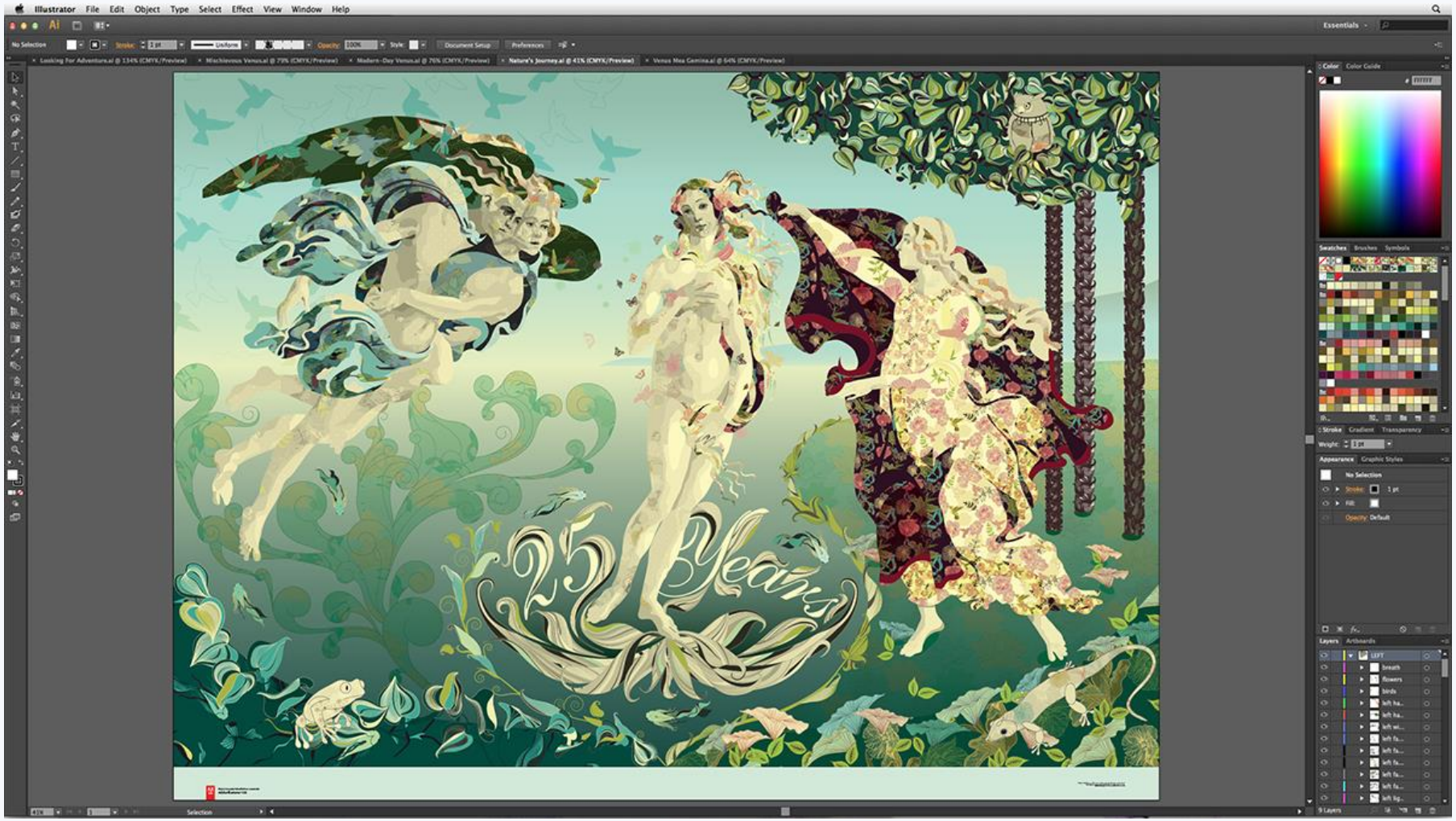
- Examples: **Photoshop** (Adobe), Gimp (open source)



Draw programs

- contain a distinctive set of tools for creating basic shapes such as ovals, rectangles, Bezier curves, and polygons generated from mathematical formulas
- the shapes can be grouped, filled, and scaled to produce complex drawn images.
- such programs can be used to create unique logos, designs, and graphics objects that can easily be resized for specific multimedia projects.
- Examples: Illustrator (Adobe), Draw (Corel), Inkscape (open source)

Draw programs



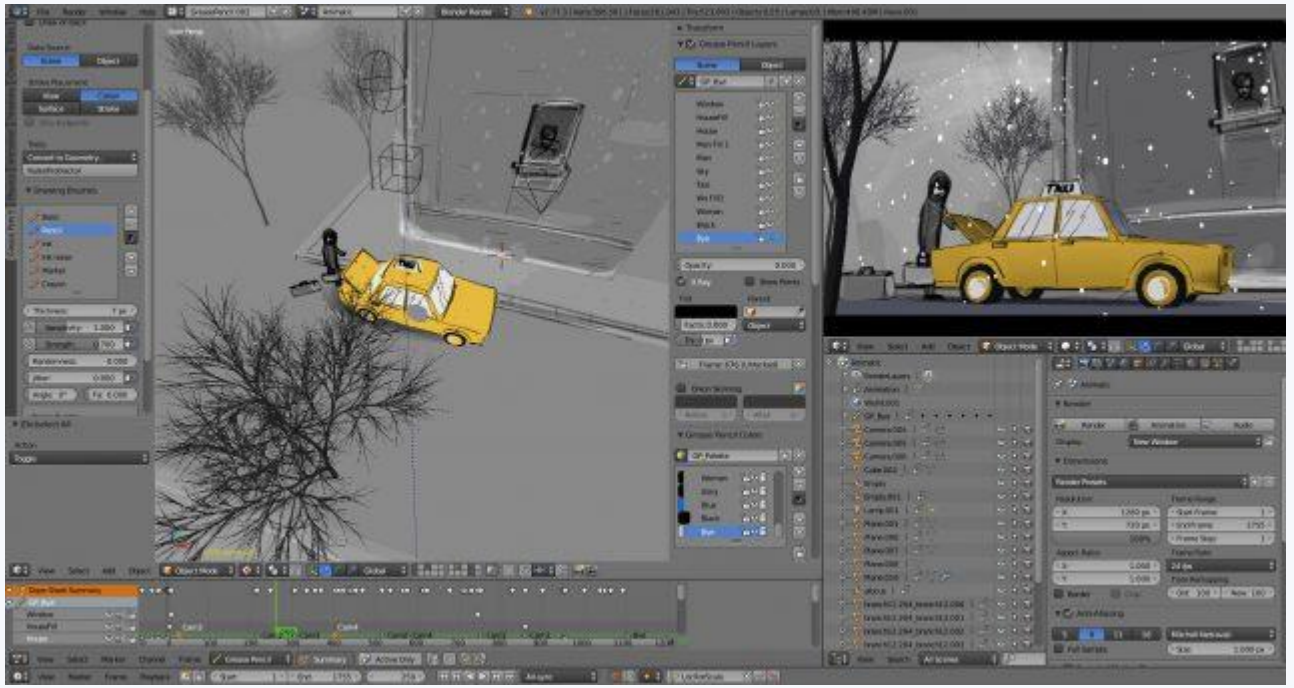
3-D imaging

- used to model 3-D objects, define surfaces, compose scenes, and render a completed image;

- In ***modeling***, the graphic artist creates the shape of an object; in s***urface definition***, color and texture are applied; in ***scene composition***, objects are arranged, lighting is specified, and backgrounds and special effects are added. The final stage of 3-D graphics is ***rendering***. *Rendering* creates a 3-D image from the specified scene. Rendering is both processor intensive and time consuming because the software must calculate how the image should appear based on the object's position, surface materials, lighting, and specific render options.

- Examples: Blender (open source), Bryce

3-D imaging



Sound

- There are two major types of sound applications for multimedia development:
  - *sampled;*
  - *synthesized*.
- Examples: Audition (Adobe)

Sampled

- sampled sounds are digital representations of analog sound sources captured from microphones or other devices. Software settings control the sound format of the sound recording.

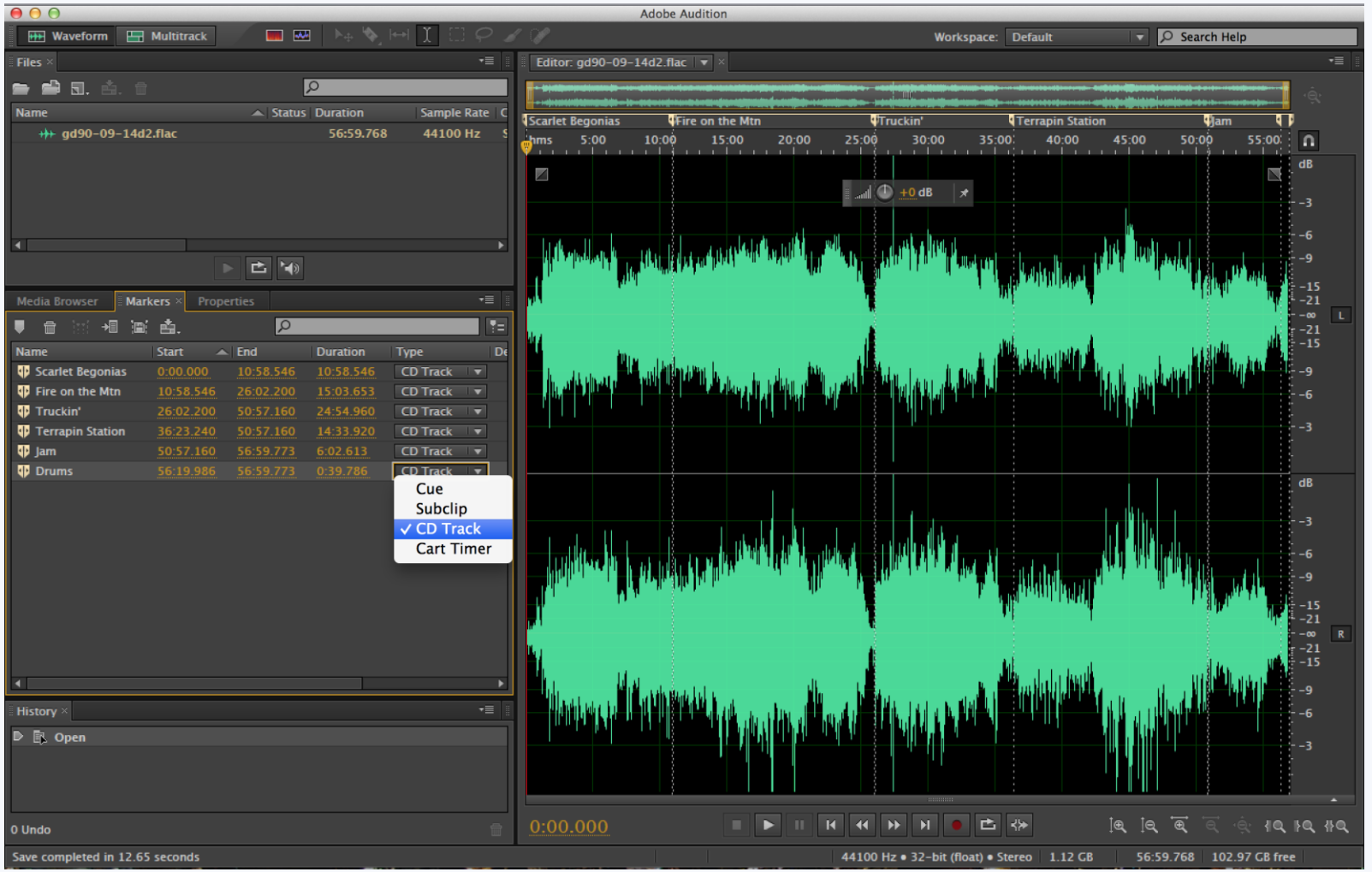
- Sampled sounds can be edited in a wide variety of ways such as trimming to delete dead space, splicing to combine sound segments, setting fade-in and fade-out (enveloping), adjusting volume, and adding special effects such as echoes or sound reversals

Synthesized

- Synthesized sound applications use digital commands to generate sounds. These commands can be captured from a MIDI instrument such as an electronic keyboard or created with a sequencer program

- The musical file is then saved and played back on a computer's synthesizer, an electronic device to generate sound. MIDI applications are a good source of original music for multimedia applications.

Sound


Adobe Audition
Waveform
Multitrack
Workspace: Default
Search Help
Files
Name
Status
Duration
Sample Rate
gd90-09-14d2.flac
56:59.768
44100 Hz
Editor: gd90-09-14d2.flac
Scarlet Begonias
Fire on the Mtn
Truckin'
Terrapin Station
Jam
+0 dB
Media Browser
Markers
Properties
Name
Start
End
Duration
Type
Scarlet Begonias 0:00.000 10:58.546 10:58.546 CD Track
Fire on the Mtn 10:58.546 26:02.200 15:03.653 CD Track
Truckin' 26:02.200 50:57.160 24:54.960 CD Track
Terrapin Station 36:23.240 50:57.160 14:33.920 CD Track
Jam 50:57.160 56:59.773 6:02.613 CD Track
Drums 56:19.986 56:59.773 0:39.786 CD Track
Cue
Subclip
CD Track
Cart Timer
History
Open
0 Undo
0:00.000
Save completed in 12.65 seconds
44100 Hz • 32-bit (float) • Stereo
1.12 GB
56:59.768
102.97 GB free

Video

- an environment to combine source material called *clips*, synchronize clips to a sound track, add special effects, and save the work as a digital video.

- A video project starts by assembling film clips in a project window. The clips can be still images, animations, sounds, or digital video files.

- Video applications provide tools to move and insert clips on a timeline, trim the clip, and define transitions between tracks. Sound tracks, title fields, and special effects such as superimposing, transparency, and lens flare add to the video composition. Video editing applications also define playback size and frame rate. When the video project is complete, the application provides settings to save it in specific file formats and compression schemes.

Video

- Examples: Premiere (Adobe)

Animation

- used to create and edit animated sequences. **Animation** is the technique of using a series of rapidly displayed still images to produce the appearance of motion.

- Objects are drawn or imported into the software where they are manipulated in a series of still frames. Frames are played back in sequence to create an illusion of motion.

- Each frame represents a single instance of the animated sequence. Typical animation tools control the path of an object, object shape, and color changes over the frame sequence. Objects are placed on a timeline where effects can be applied to fade in, morph, rotate, spin, flip, or change pace. Multiple objects can be layered to interact with each other to create more complex animations.

Animation

- Examples: Director (Adobe), Flash (Adobe), Animate (Adobe)

Authoring Software

- consists of programs specifically designed to facilitate the creation of multimedia products.

- they are used to assemble media elements, synchronize content, design the user interface, and provide user interactivity.

Authoring Software

- Categories based on the metaphor they use to organize media element:
  - card-based metaphor;
  - timeline;
  - flow diagram.

Card-based metaphor

- elements are arranged much as they might be on index cards or the pages of a book.
- such applications are easy to use and are ideal for products such as information kiosks, lectures, and tutorials that do not require precise synchronization of individual media elements.
- Examples: **PowerPoint**, ToolBook

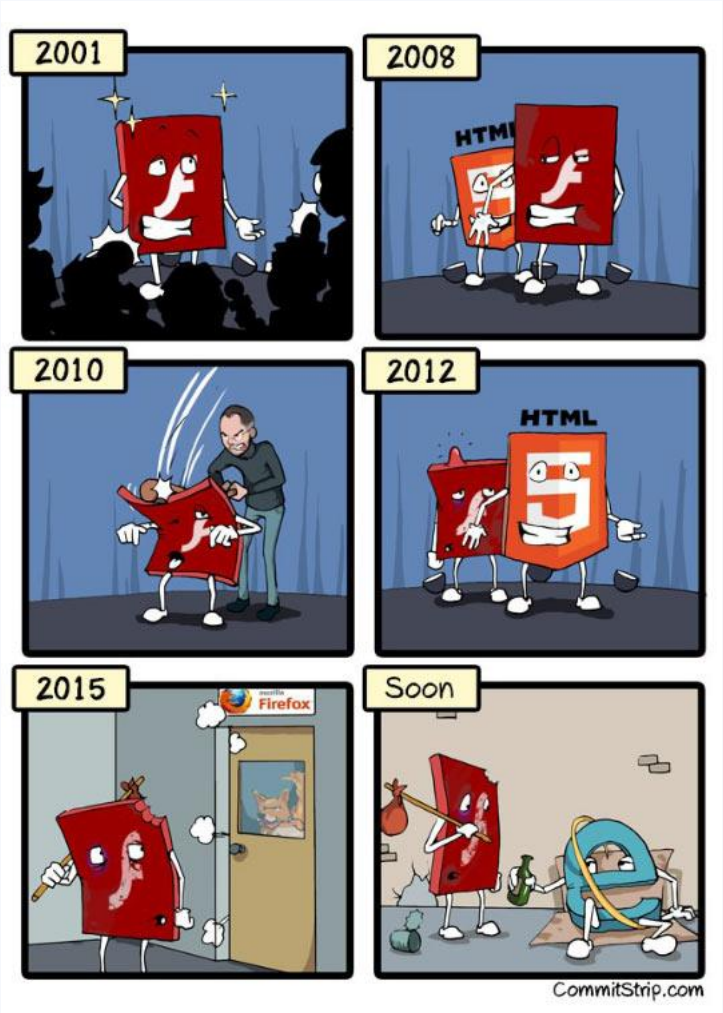
Timeline

- use a *timeline* of separate frames much like a motion picture film;
- such applications provide the precise control needed for advanced animations;
- Examples: Director (Adobe), Flash (Adobe), Animate(Adobe).

Authoring Software

Adobe Flash

The Evolution Of Flash



Flow diagram

- use *icons* arranged on flow lines to quickly develop a wide range of multimedia products including advanced tutorials, product demonstrations, and simulations. Icons can represent both content (images, text, animations, video) and a wide range of interactions (play, stop, go to, calculate, etc.).

- example: Authorware

Web Multimedia

Web Multimedia

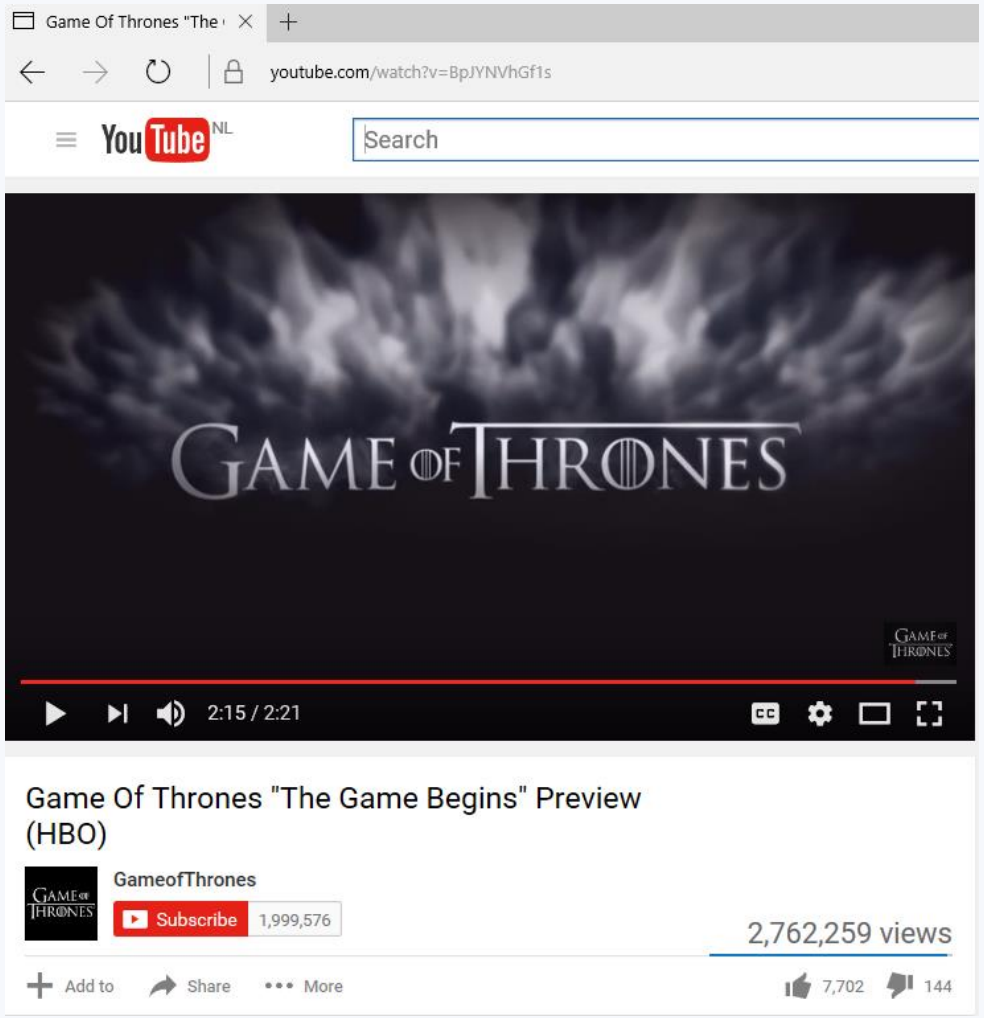
- HTML5 provides a greatly improved support for web multimedia, by including elements such as audio, video, canvas

- supported multimedia elements:
  - text
  - images
  - animations
  - raster graphics
  - vector graphics
  - 3D graphics
  - audio
  - video

HTML5 Games

http://www.cuttherope.net/basic_standalone/game.html

HTML5 Videos



Images

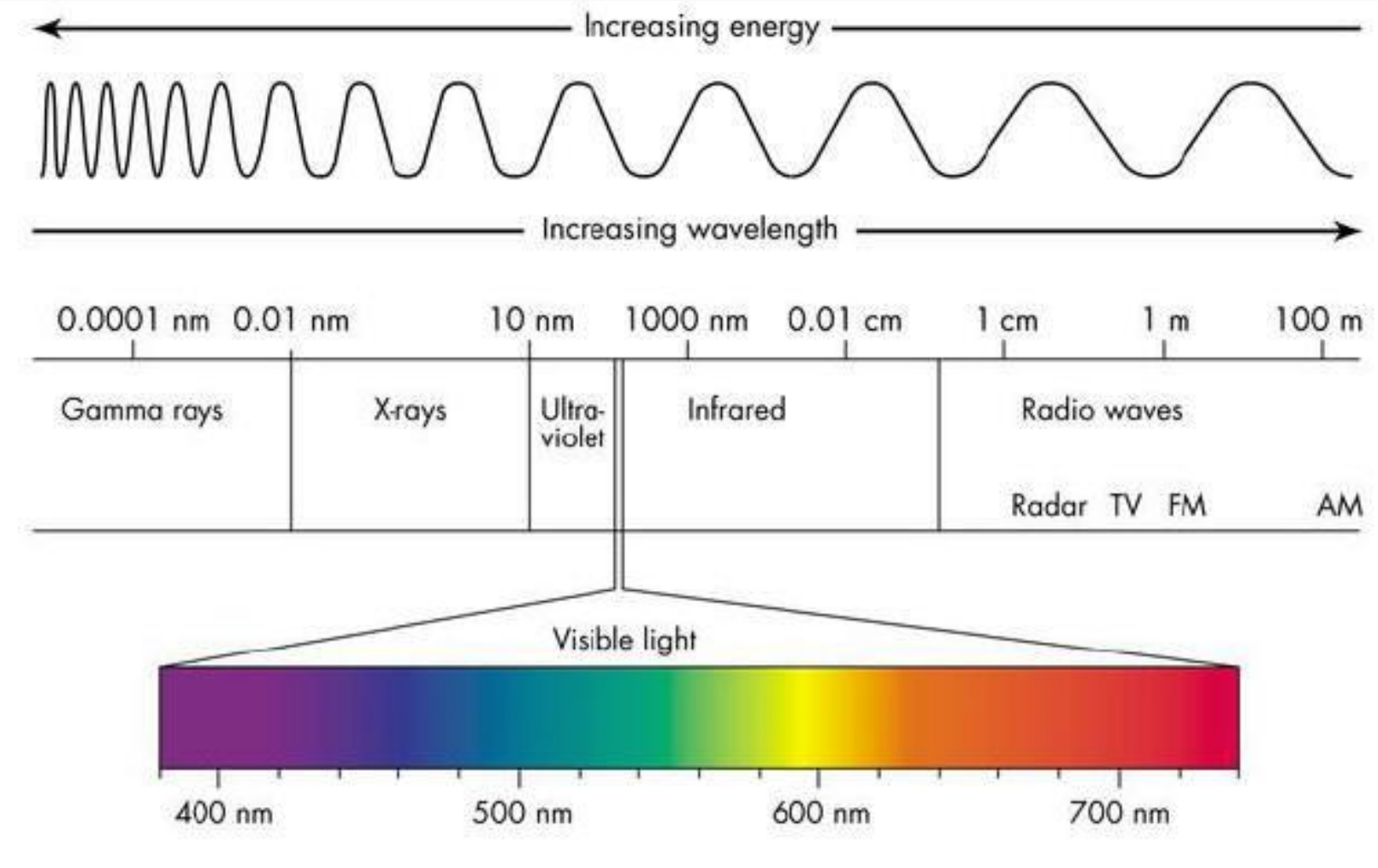
Digital Images

- An **image** is a two- or three-dimensional representation of a person, animal, object, or scene in the natural world [1].

- A **digital image** is a numeric representation of a two-dimensional image.

[1] Costello, V.: Multimedia Foundations: Core Concepts for Digital Design. Focal Press, New York (2016).

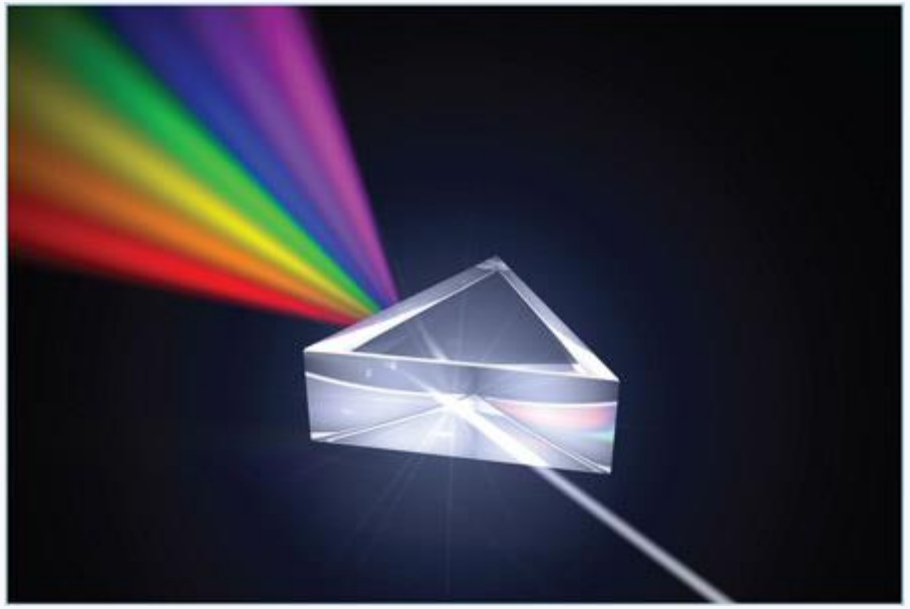
Visible light



Color Space

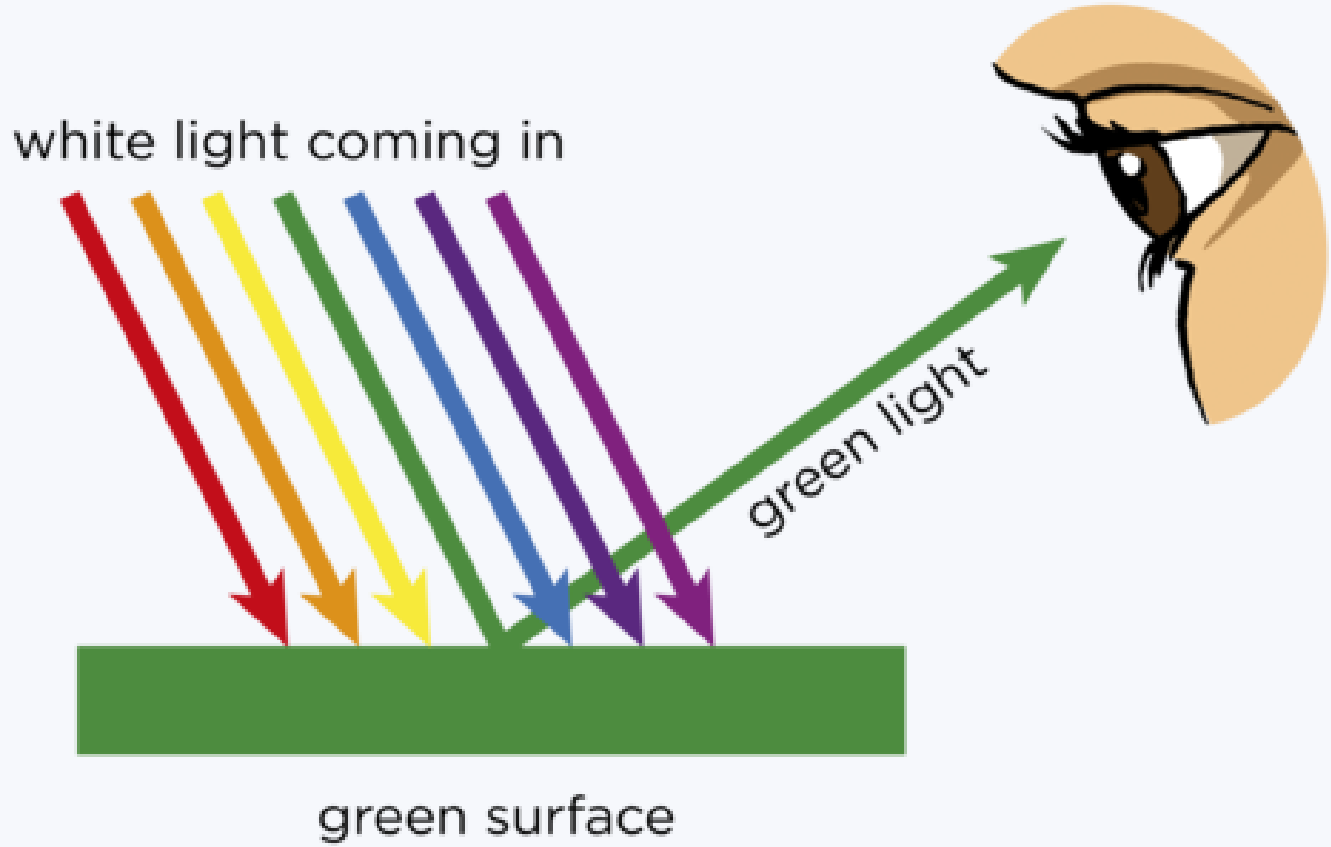
- We refer to natural sunlight as white light because it appears to the human eye to be colorless.

- it's possible to separate white light into a dazzling display of various colors using a prism. As light travels through a prism, it's refracted (bent), causing the beam of white light to break apart into its component color wavelengths



The primary and secondary colors of white light become visible when refracted by a glass prism.

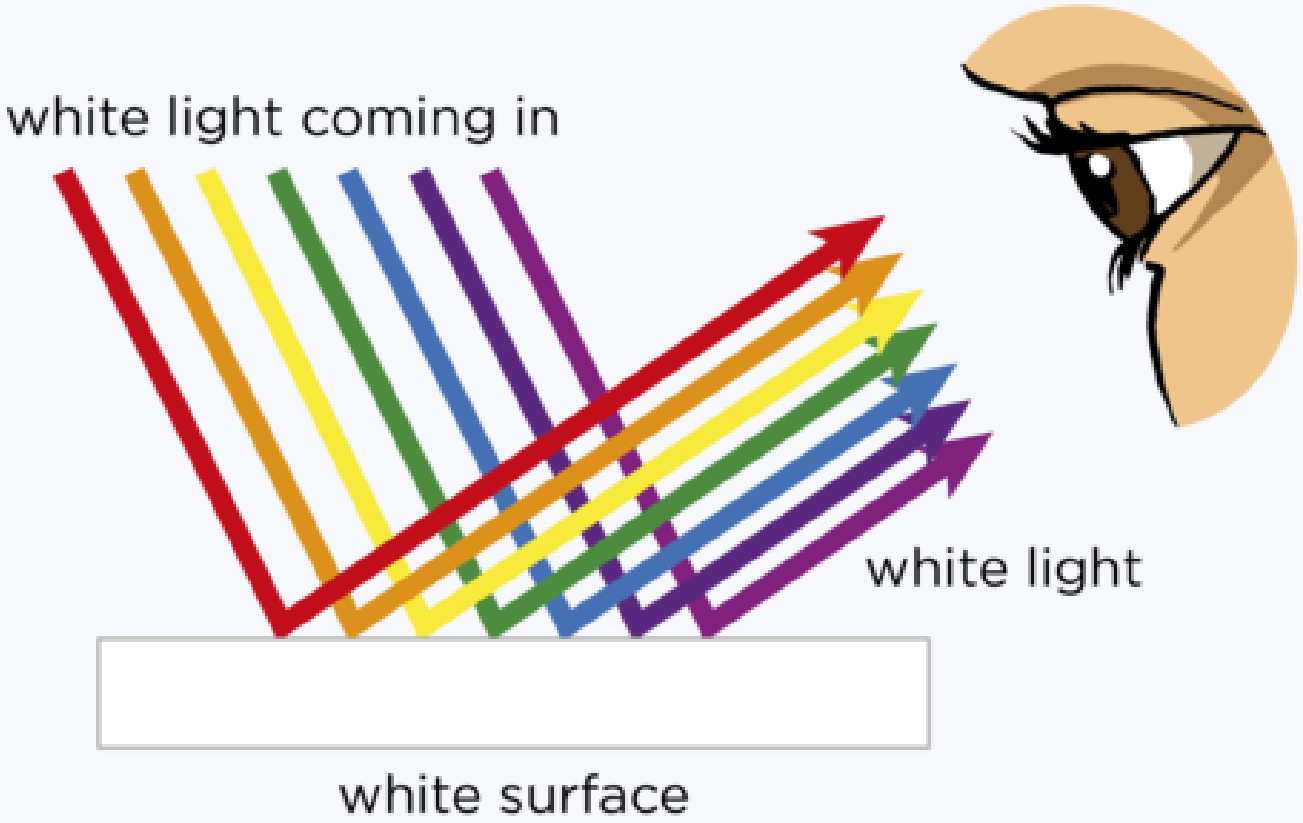
Color Space



The green leaf absorbs all the colors except green which it reflects back into our eyes.

[1] http://www.mstworkbooks.co.za/natural-sciences/gr8/gr8-ec-04.html.

Color Space

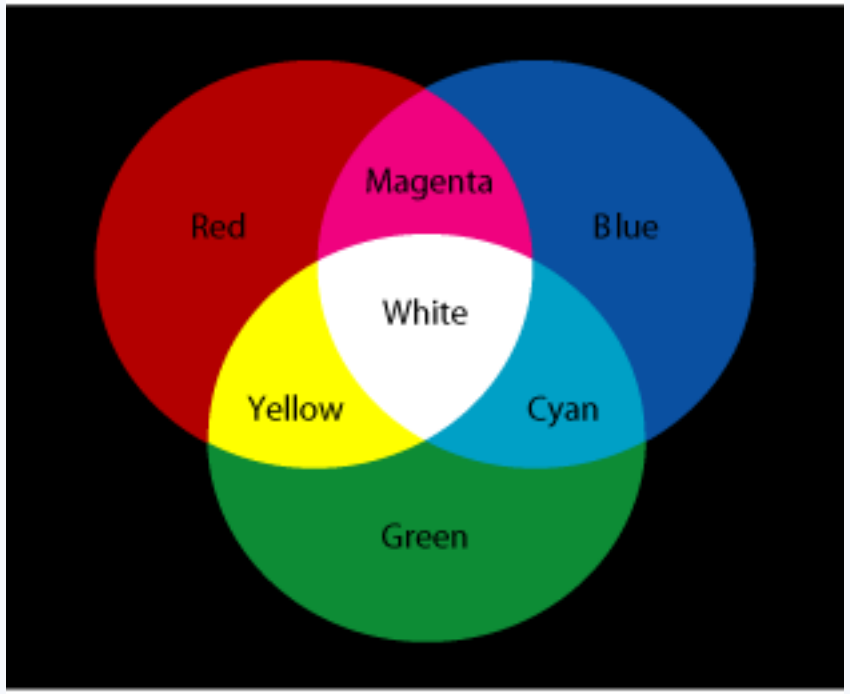


[1] http://www.mstworkbooks.co.za/natural-sciences/gr8/gr8-ec-04.html.

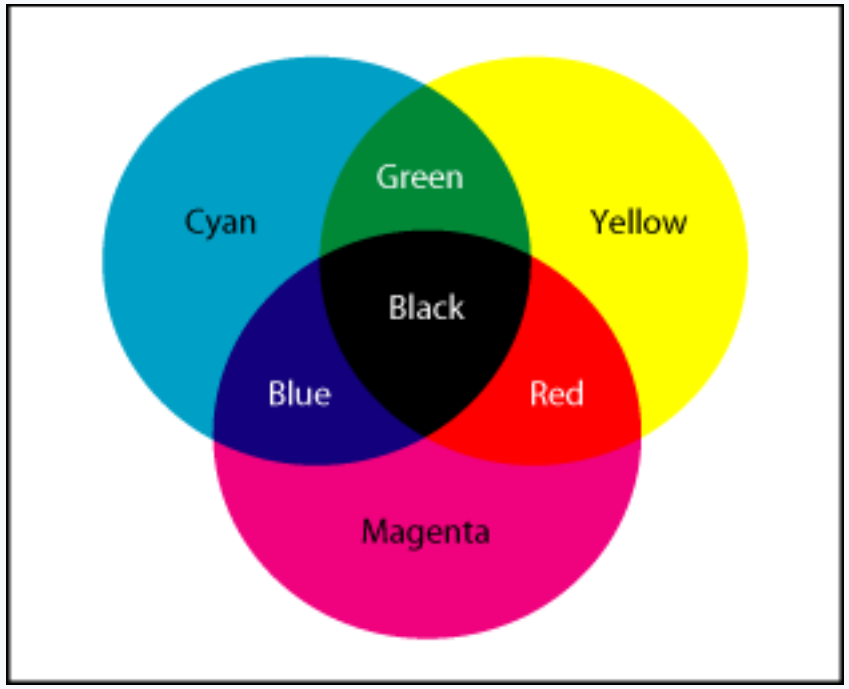
Color Space

- Although we can get black paint as a pigment, black is not a color of light. Black is the result of the complete absorption of light.

Color Systems



Additive Model (RGB)



Subtractive Model (CMYK)

Additive Model

- **Additive color** is color created by mixing a number of different light colors.
- Shades of red, green, and blue are the most common primary colors used in additive color system.
- Additive color mixing begins with **black** and ends with white; as more color is added, the result is lighter and tends to **white**.
- The combination of **two** of the standard **three** additive primary colors in equal proportions produces an additive secondary color - **cyan**, **magenta** or **yellow**.

RGB Color Model (or Mode)

- the primary colors of light are **red**, **green**, and **blue** (**RGB**).

- by adjusting the intensity of each, you can produce **all the colors** in the visible light spectrum. You get **white** if you add all the colors equally and **black** by removing all color entirely from the mix.

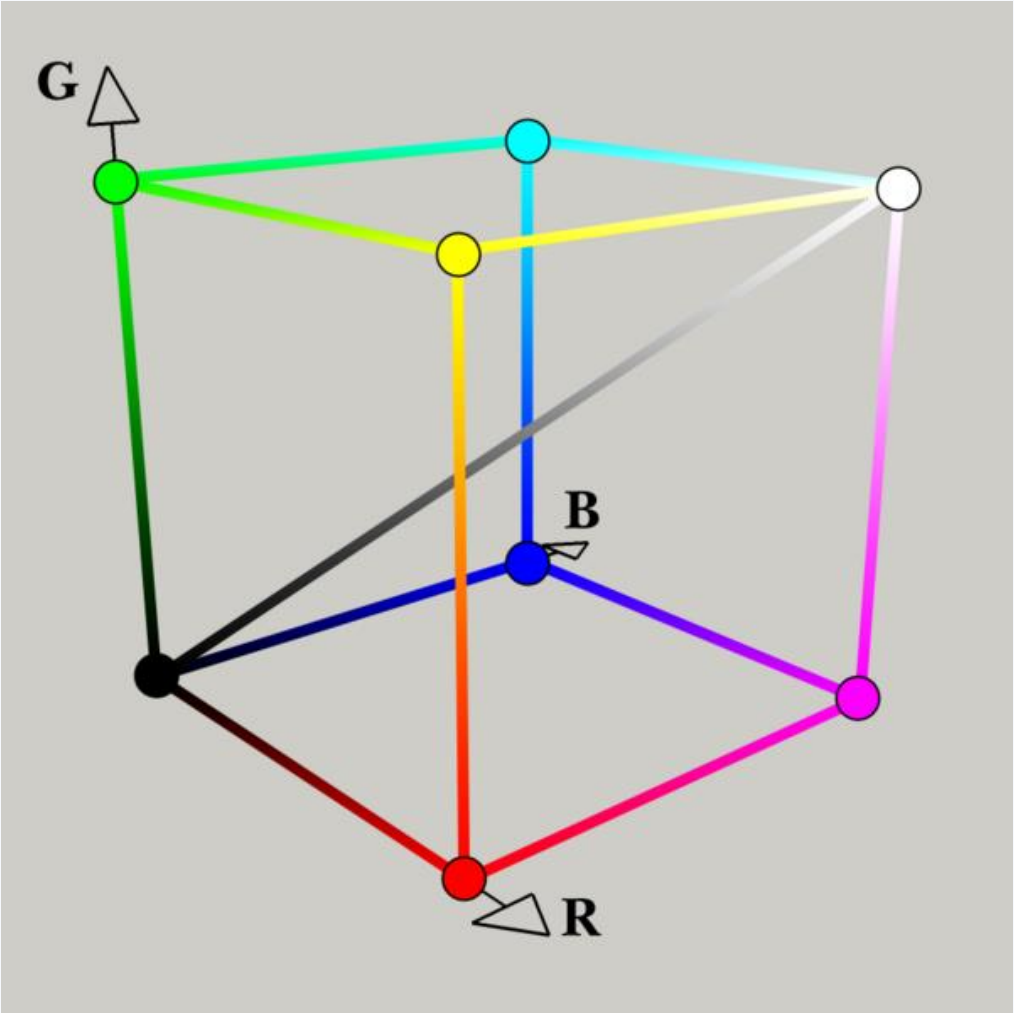
- If you were to look at a monitor such as an LCD (liquid crystal display) under a microscope, you'd see that each pixel really displays only the three primary colors of light.

RGB Color Model (or Mode)

- In additive color mixing, red and green make yellow. If you fill a graphic with intense yellow in Photoshop, the pixels really display stripes of intense red and green, with no blue.

- The individual points of color are tiny, so our brains add the colors together into yellow.

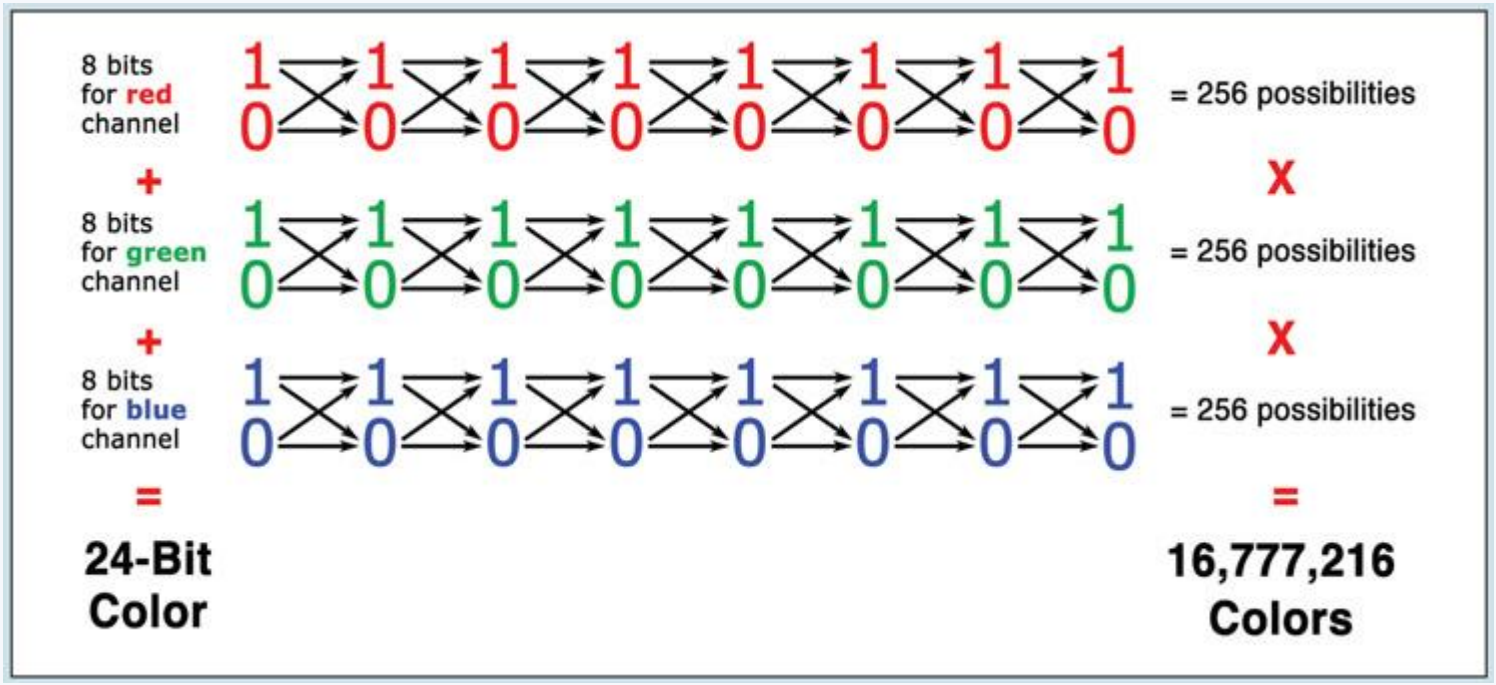
RGB Color Model (or Mode)



Adobe Photoshop – RGB Color

Costello, V.: Multimedia Foundations: Core Concepts for Digital Design. Focal Press, New York (2016).

RGB Color Model (or Mode)



The possible color combinations for any pixel in an eight-bit graphic (or a 24-bit display).

RGB Color Model (or Mode)

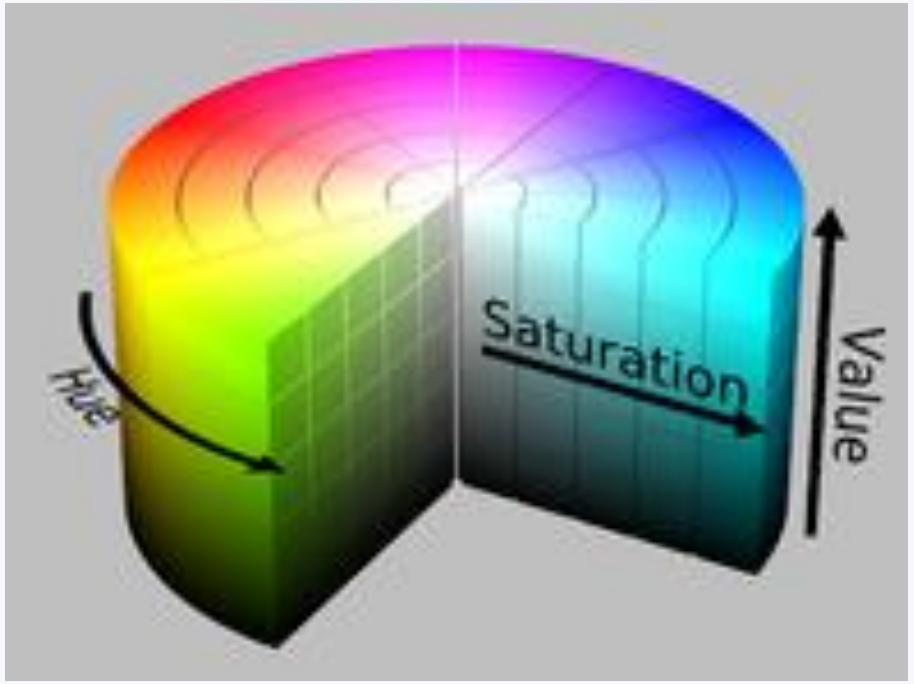
- 256 possibilities for each color channel.

- 256 possibilities for red × 256 for green × 256 for blue - **16.8 million** possible combinations / colors.

HSB - Hue, Saturation and Brightness Color Model

- Also known as **HSV** (Hue, Saturation and Value):
- Together with **HSL** (Hue, Saturation and Lightness) are the most common cylindrical-coordinate representations of points in an RGB color model.
- The two representations rearrange the geometry of RGB in an attempt to be more intuitive and perceptually relevant than the cartesian (cube) representation.

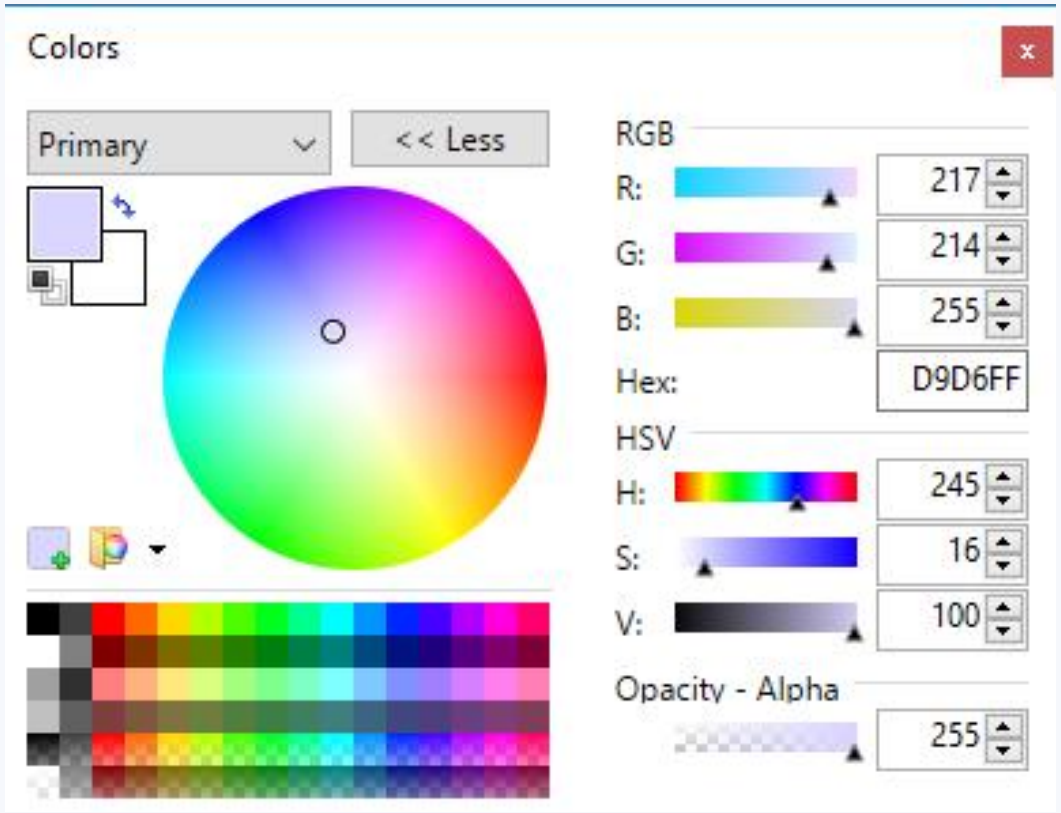
HSB - Hue, Saturation and Brightness Color Model



- Red - 0^0
- Green - 120^0
- Blue - 240^0

HSB - Hue, Saturation and Brightness Color Model

- HSL and HSV are used primarly in color pickers, in image editing software, and less commonly in image analysis and computer vision.



Color Picker in Paint.NET:
http://www.getpaint.net/index.html

Subtractive Color System

- When we mix colors using paint, or through the printing process, we are using the **subtractive** color method [1].

- A **subtractive color** model explains the mixing of a limited set of dyes, inks, paint pigments or natural colorants to create a wider range of colors, each the result of partially or completely subtracting (that is, absorbing) some wavelengths of light and not others [1]

- Subtractive color mixing means that one begins with white and ends with black; as one adds color, the result gets darker and tends to black [2].

[1] https://en.wikipedia.org/wiki/Subtractive_color
[2] http://www.worqx.com/color/color_systems.htm

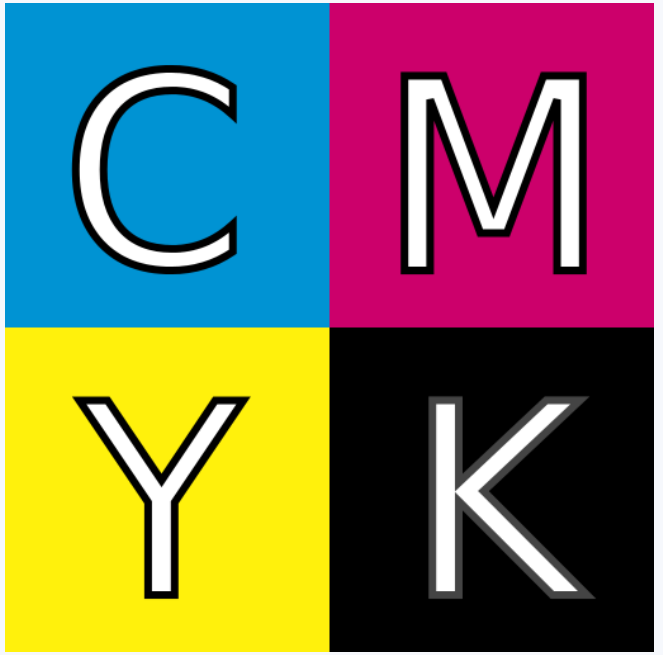
Subtractive Color System

- The color that a surface displays depends on which parts of the visible spectrum are not absorbed and therefore remain visible [1].

[1] https://en.wikipedia.org/wiki/Subtractive_color

CMYK Color Model (or Mode)

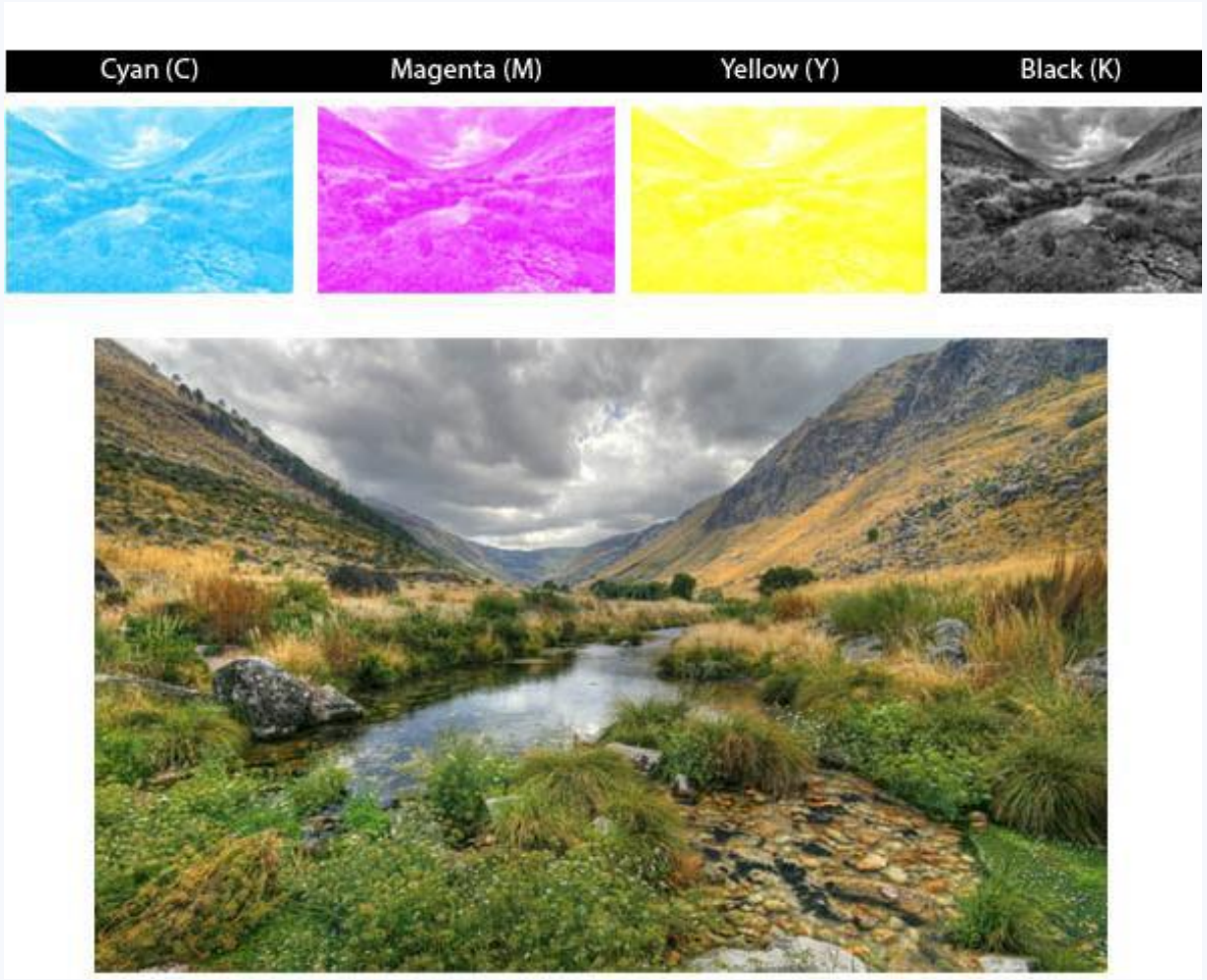
- The **CMYK** color model (process color, four color) is a subtractive color model, used in color printing, and is also used to describe the printing process itself [1].
- CMYK refers to the four inks used in some color printing: cyan, magenta, yellow and key (black) [1].
- Though it varies by print house, press operator, press manufacturer, and press run, ink is typically applied in the order of the abbreviation [1].



CMY Color Model (or Mode)

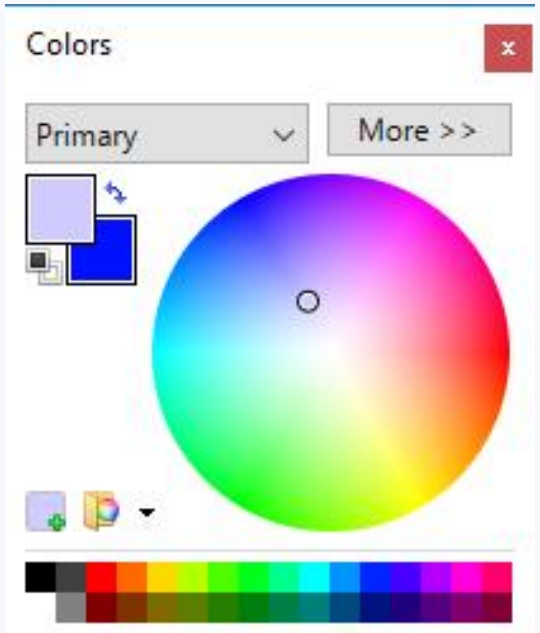
https://commons.wikimedia.org/wiki/File:NIEdot367.jpg

CMYK Color Model (or Mode)



Color Wheel

- A **color wheel** (also referred to as a color circle) is a visual representation of colors arranged according to their **chromatic relationship**. Begin a color wheel by positioning **primary** hues equidistant from one another, then create a bridge between primaries using **secondary** and **tertiary colors** [1].



Color Picker in Paint.NET:
http://www.getpaint.net/index.html

Color Wheel

- **Primary Colors:** Colors at their basic essence; those colors that cannot be created by mixing others [1].
- **Secondary Colors:** Those colors achieved by a mixture of two primaries [1].
- **Tertiary Colors:** Those colors achieved by a mixture of primary and secondary hues [1].

Color Wheel

- **Complementary Colors:** Those colors located opposite each other on a color wheel [1].
- **Analogous Colors:** Those colors located close together on a color wheel [1].

Active & Passive Colors

- The color wheel can be divided into ranges that are visually:
  - **Active colors** - will appear to advance when placed against passive hues.
  - **Passive colors** - appear to recede when positioned against active hues.

- Advancing hues are most often thought to have less visual weight than the receding hues.

- Most often warm, saturated, light value hues are "active" and visually advance.

Active & Passive Colors

- Cool, low saturated, dark value hues are "passive" and visually recede.
- Tints or hues with a low saturation appear lighter than shades or highly saturated colors.
- Some colors remain visually neutral or indifferent.

Complementary Colors

- When fully saturated complements are brought together, interesting effects are noticeable. This may be a desirable illusion, or a problem if creating visuals that are to be read.

- Notice the illusion of highlighted edges and raised text. This may occur when opposing colors are brought together.

2-D Computer Graphics

- Computers create either 2-D (width and height) or 3-D images (width, height, and depth).

- There are two main types of 2-D computer graphics:

  - **bitmapped** approach - particularly well suited for images with fine detail, such as paintings or photographs. Also called **raster** images.

  - **vector** drawing - used for graphic designs ranging from simple drawings and logos to sophisticated artistic creations.

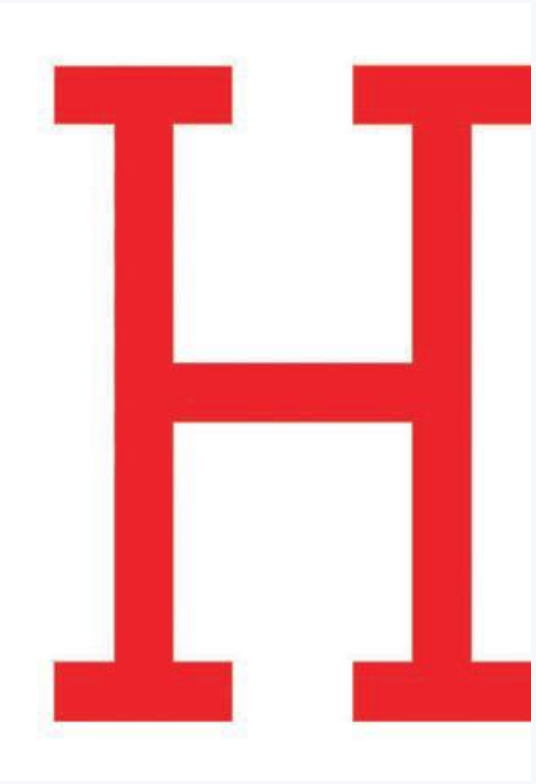
Raster Graphics

Raster Graphics

- a **raster graphics** image is formed by dividing the area of an image into a rectangular matrix of rows and columns comprised of pixels [1].

- A **pixel**, short for picture element, is a square (or occasionally rectangular) area of light representing a single point in a raster image [2].

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	1	1	1	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0
0	1	1	1	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0



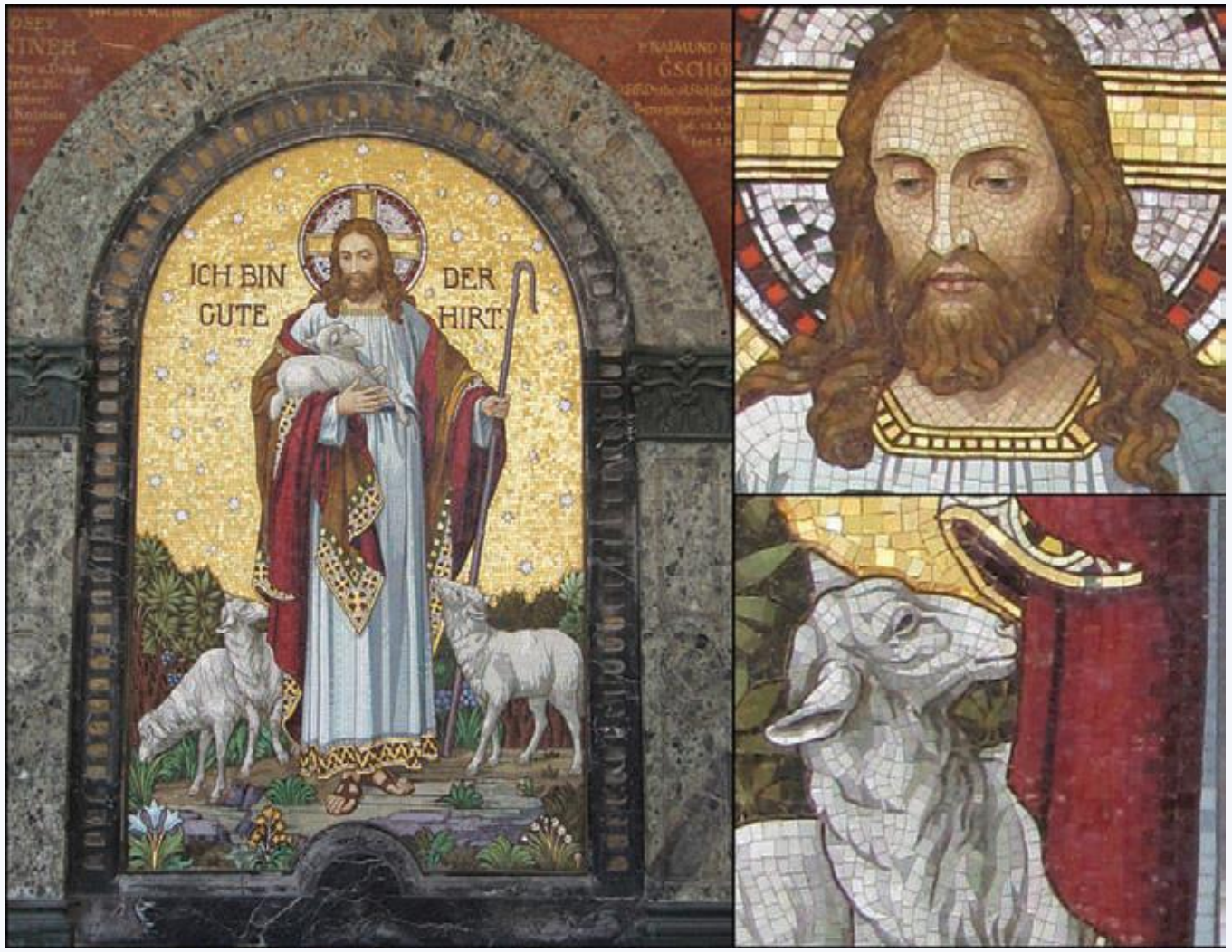
Raster Graphics

- Every pixel in a raster image is exactly the same size and contains a single color value that's typically stored as a 24-bit string of binary data.

- The total number of pixels in a raster image is fixed. In order to make a raster image physically larger, more pixels have to be added to the raster matrix.

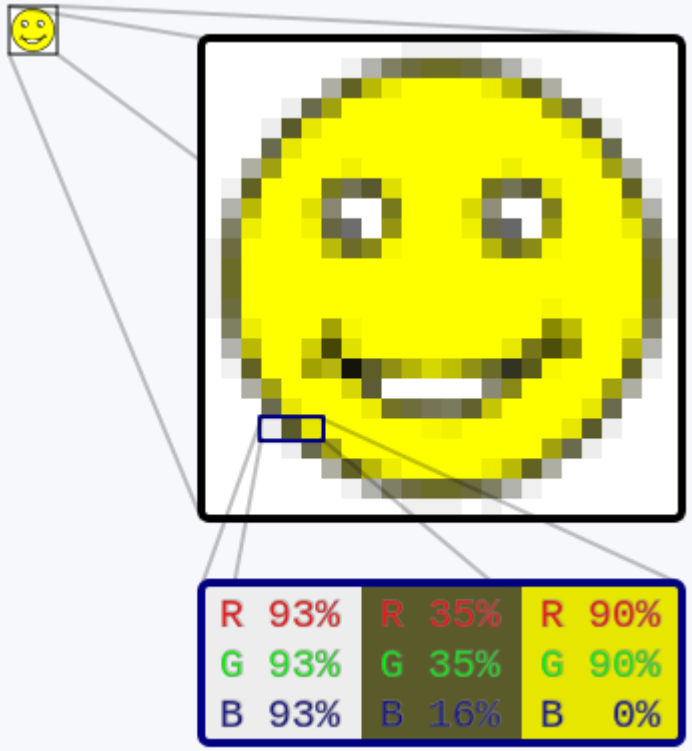
- Likewise, pixels need to be discarded when making a raster image smaller. The width and height of a raster image is determined by how many pixels each row and column contains.

Mosaic



From a distance, the individual tiles forming the image are barely perceptible to the naked eye. Up close, however, we can see that many small pieces of visual information went into forming this 19th-century mosaic of Christ, the good shepherd. This technique of using tiny bits of colored material to form a composite visual impression dates back to about 3,000 bc.

Raster Graphics



When enlarged, individual pixels appear as squares. Zooming in further, they can be analyzed, with their colors constructed by adding the values for red, green and blue.

Formats

- BMP (Microsoft Windows Bitmap) - *.bmp
  - also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap
  - either uncompressed or compressed using RLE (lossless compression format)
  - monochrome or color (various color depths are supported)

Characteristics

- Resolution
  - describes the image quality of a raster image and directly relates to the size and quantity of the pixels the image contains.
- Pixel Dimensions
  - describe the size of a raster image, expressed as the number of pixels along the x-axis (width) by the number of pixels along the y-axis (height).
  - ex: 800 px * 600 px
- Pixel Count
  - Pixel count is the total number of pixels in a raster matrix. To determine the pixel count, multiply the horizontal and vertical pixel dimensions.
  - ex: a 30 * 18 pixel image has a pixel count of 540 pixels.

Characteristics

- Pixel Density
  - We express the pixel density or resolution of a raster image in pixels per inch (ppi)—this is pixels per linear inch (across or down), not square inch.
  - Most television and computer monitors have a display resolution of either 72 or 96 ppi.
  - The resolution determines the maximum size of an image you print. In order to produce a high-quality print of any size, digital photographs need a pixel density of at least 300 ppi—that's 90,000 pixels in a square inch!

Characteristics

- Color-Depth (Color Resolution)
  - measure of the number of different colors that can be represented by an individual pixel. The number of bits assigned to each pixel, or bit depth, determines color resolution.
  - 8-bits (256 colors), 24 bits (16.8 million colors)
  - A drawing using 16 distinct colors, for example, would only require 4 bits per pixel. Using a code with greater bit depth produces a larger file with no increase in quality.

Advantages and Disadvantages

- Advantages:
  - great when creating rich and detailed images. Every pixel in a raster image can be a different colour therefore complex images with any kind of colour changes and variations can be created.

Advantages and Disadvantages

- Disadvantages:
  - the files are often **quite large** because they contain all the information for every single pixel of the image.
  - cannot be scaled up in size very well. If enlarged, a raster image will look grainy and distorted because raster images are created with a finite number of pixels. When you increase the size of a raster image, the image increases in size however, because there are no longer enough pixels to fill in this larger space, gaps are created between the pixels in the image. The photo editing software will try to fill these gaps the best they can however, the resulting image is often blurry.
  - no information is provided regarding the shapes that make up the image.

Formats

- while working with an image file, it should be saved it in a format that supports layers (*.PSD, *TIFF), so it can be changed it in the future or resaved in a new format for another purpose.

- when you prepare a raster image to incorporate into a multimedia project, you'll need a **flattened**, **compressed** image. For lossy formats there is always a tradeoff between image quality and the file size.

Formats

- GIF (Graphics Interchange Format) - *.gif
  - maximum 256 colors and transparency (transparent pixels);
  - lossless compression format.
  - commonly used for logos and other images with lines and solid blocks of color.
  - supports interlacing, so every odd line of pixels loads, then every even line loads, making graphics seem to appear faster (users see the full-sized, half-loaded graphic before the rest of the pixels appear).
  - supports animation.

Formats

A rotating globe in GIF format [1].

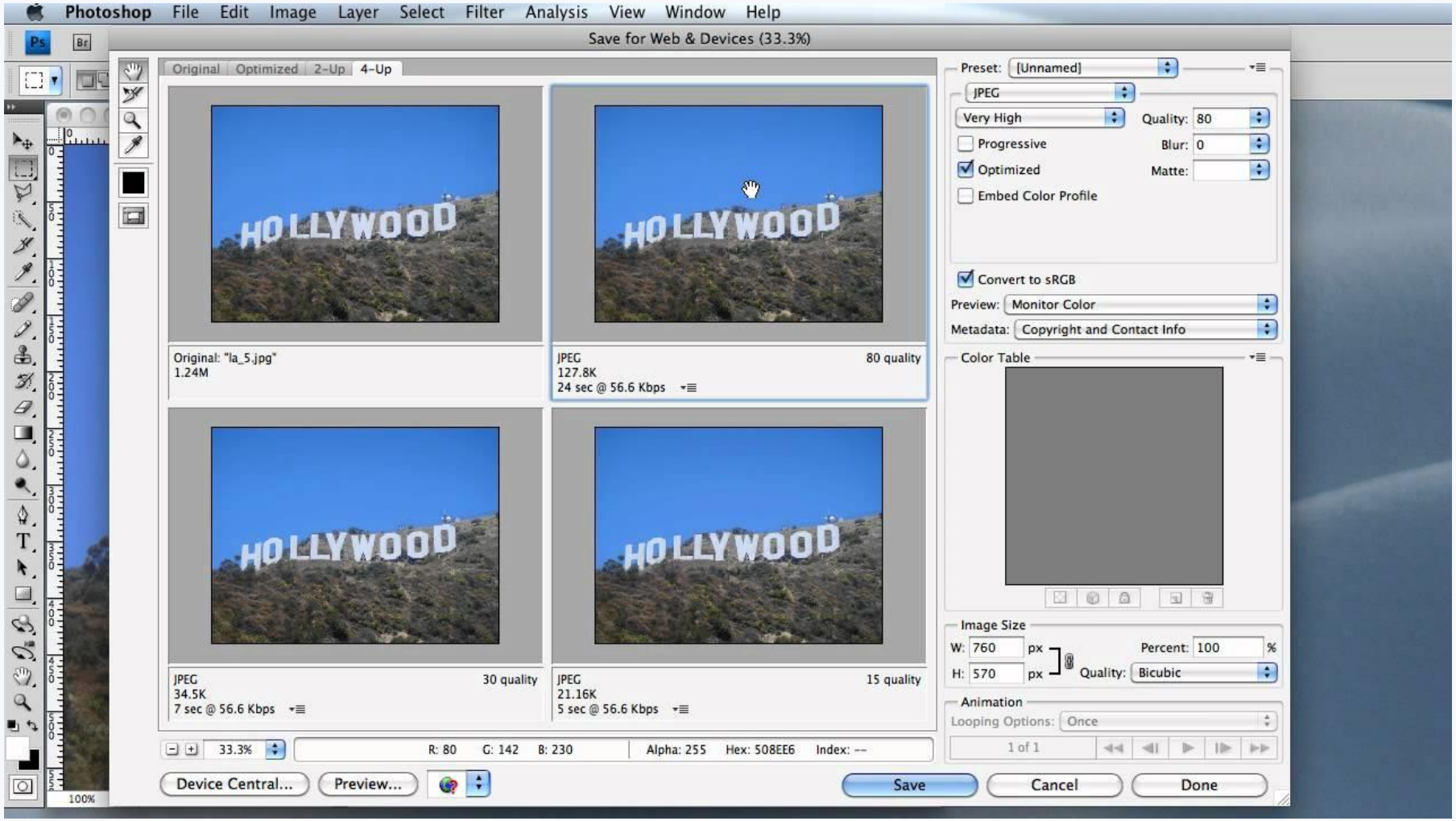
Formats

- JPEG (Joint Photographic Experts Group) - *.jpeg
  - offers 16.8 million colors
  - does not support transparency (has no transparent pixels).
  - lossy compression format and is used most often for photographs.
  - does not support interlacing.

Formats

JPEG - a photo of a cat with the compression rate decreasing, and hence quality increasing, from left to right [1].

Formats

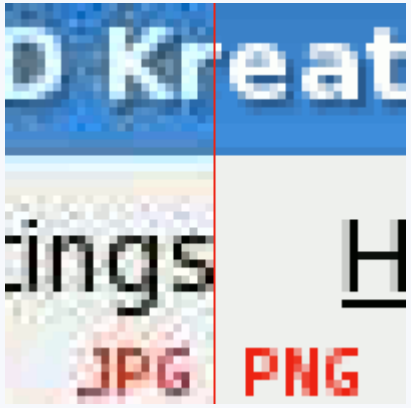


Photoshop – Save for Web menu

Formats

- **PNG** (Portable Network Graphics) - ***.png**
  - the most widely used lossless image compression format on the Internet [1]
  - created as an improved, non-patented replacement for Graphics Interchange Format (GIF) [1]
  - offers 16.8 million colors and transparency, but you can choose to use fewer colors to save file space (PNG 8, or PNG with 8-bit color) [2]
  - **Lossless compression** format [2]
  - common for a wide range of images, including ***favicons*** (the small web page icons in browser tabs) [2]
  - PNG files can be very small, but for photographs with many colors, they may be larger than comparable JPEGs [2]
  - This format supports interlacing.[2]

Formats



Composite image comparing **lossy compression** in JPEG with **lossless compression** in PNG: the JPEG artifacts are easily visible in the background, where the PNG image has solid color [1].

Formats

- Other formats:
  - ICO - Icon Resource File;
  - DIB - Device Independent Bitmap;
  - PCX - PC PaintBrush File Format;
  - TIFF - Tag Image File Format.

Huge images

- Dubai: http://gigapan.com/gigapans/48492
- Size: 44.88 Gigapixels

Compression

- Compression algorithms:
  - Run-length encoding (RLE)
  - Lempel–Ziv–Welch (LZW)
  - Huffman
  - JPEG

Compression - Run Length Encoding - RLE

- one of the simpler strategies to achieve **lossless** compression
- can be used to compress bitmapped image files (ex: *pcx format). Bitmapped images can easily become very large because each pixel is represented with a series of bits that provide information about its color.
- RLE generates a code to "flag" the beginning of a line of pixels of the same color. That color information is then recorded just once for each pixel. In effect, RLE tells the computer to repeat a color for a given number of adjacent pixels rather than repeating the same information for each pixel over and over. The RLE compressed file will be smaller, but it will retain all the original image data—it is "lossless."

Compression - Run Length Encoding - RLE

Compression - Lempel–Ziv–Welch (LZW)

- **lossless** data compression algorithm
- used in the GIF image format

- **Encoding**
  - Initialize the dictionary to contain all strings of length one.
  - Find the longest string W in the dictionary that matches the current input.
  - Emit the dictionary index for W to output and remove W from the input.
  - Add W followed by the next symbol in the input to the dictionary.
  - Go to Step 2.

HTML5 Images

- represents an image in the document.
- declaring an image element:

```
<img alt="..." src="...">
```

- API: https://developer.mozilla.org/en-US/docs/Web/API/HTMLImageElement
- DOM interface – **HTMLImageElement**:

HTML5 Images

- Properties:
  - width, height: the rendered width / height of the image in CSS pixels;
  - naturalWidth, naturalHeight: unsigned long representing the intrinsic width / height of the image in CSS pixels;
  - src: DOMString that reflects the src HTML attribute, containing the full URL of the image including base URI.

- Events:
  - load: fired when a resource and its dependent resources have finished loading.

HTML5 Images

```
var image = $("<img>")
            .load(function () { $("body").append($(this)); })
            .attr("src", "media/Penguins.jpg");
```

HTML5 Canvas Element

- Can be used to draw graphs, make photo compositions or even perform animations.

- Declaring a canvas element:

```
<canvas id= "test" width= "250" height="150">
An alternative text describing what your canvas displays.
</canvas>
```

- API: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas

- DOM interface: HTMLCanvasElement

HTML5 Canvas Element - Methods

- getContext(contextType, contextAttributes); - returns a drawing context on the canvas, or null if the context identifier is not supported;

```
var canvas = document.getElementById('test');  // sau var canvas = $('test')[0];
var w = canvas.width, h = canvas.height;
var ctx = canvas.getContext('2d');
```

  - contextType: DOMString containing the context identifier defining the drawing context associated to the canvas.

HTML5 Canvas Element - Methods

- possible values for contextType:
  - "2d", leading to the creation of a **CanvasRenderingContext2D** object representing a two-dimensional rendering context.
  - "webgl" which will create a **WebGLRenderingContext** object representing a three-dimensional rendering context. This context is only available on browsers that implement WebGL version 1 (OpenGL ES 2.0).
  - "bitmaprenderer" which will create a **ImageBitmapRenderingContext** which only provides functionality to replace the content of the canvas with a given ImageBitmap.

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- used for drawing rectangles, text, images and other objects onto the canvas element. It provides the 2D rendering context for the drawing surface of a <canvas> element.

- API: developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

- Drawing rectangles:
  - clearRect() - sets all pixels in the rectangle defined by starting point (x, y) and size (width, height) to transparent black, erasing any previously drawn content.
  - fillRect() - draws a filled rectangle at (x, y) position whose size is determined by width and height.
  - strokeRect() - paints a rectangle which has a starting point at (x, y) and has a w width and an h height onto the canvas, using the current stroke style.

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Drawing text:
  - fillText() - draws (fills) a given text at the given (x,y) position;
  - strokeText() - draws (strokes) a given text at the given (x, y) position;
  - measureText() - returns a TextMetrics object.

- Drawing paths:
  - check documentation

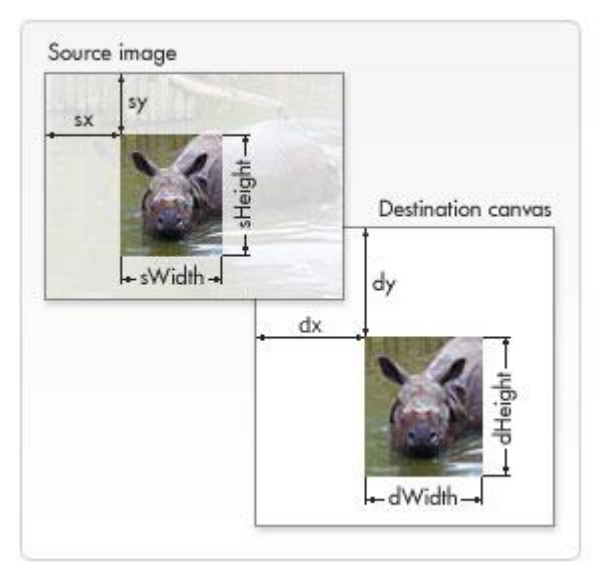
- Line styles:
  - check documentation

HTML5 Canvas Element - CanvasRenderingContext2D Interface

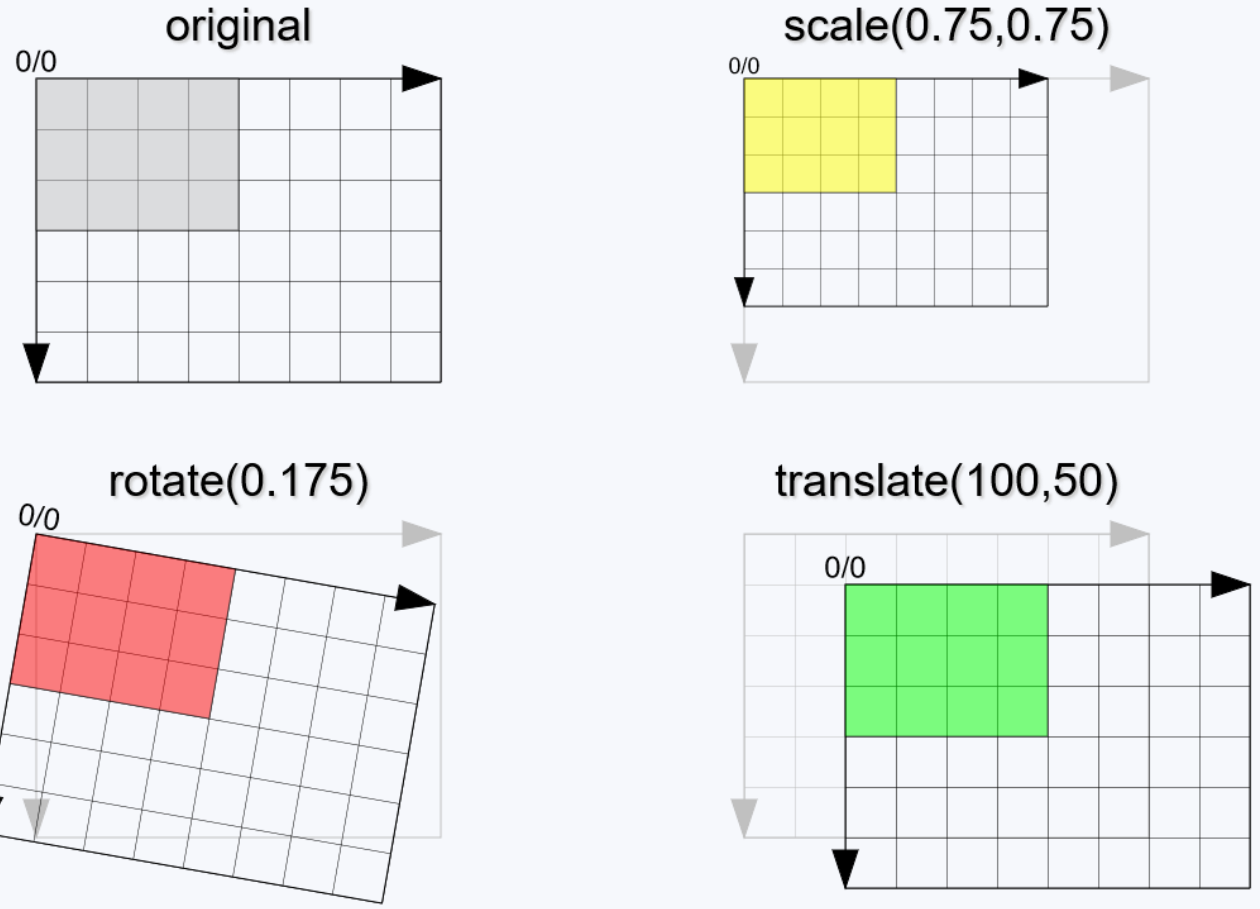
- Text styles:
  - check documentation
- Fill and stroke styles
  - check documentation
- Gradients and patterns
  - check documentation
- Shadows
  - check documentation

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Drawing images:
  - Without scaling
    - void ctx.drawImage(image, dx, dy);
  - With scaling
    - void ctx.drawImage(image, dx, dy, dWidth, dHeight);
    - void ctx.drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight);
  - API: Mozilla Developer Network

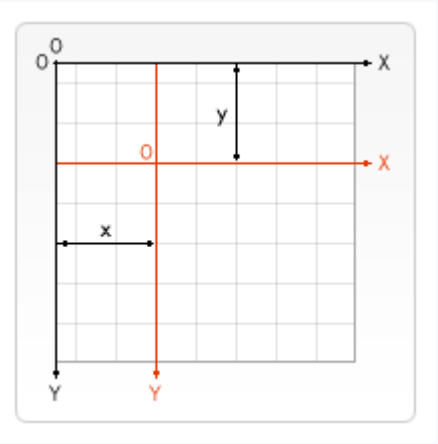


HTML5 Canvas Element - Transformations



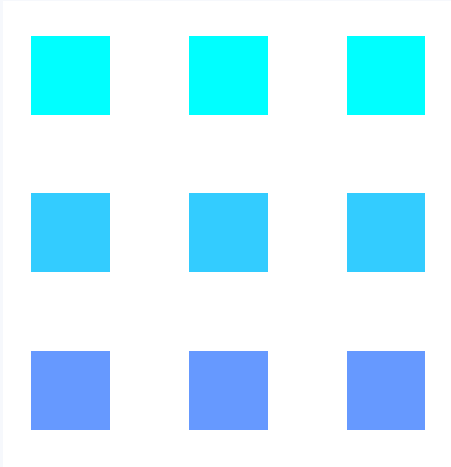
HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Translating
  - translate(x, y)
    - changes the origin.
    - x indicates the horizontal distance to move, and y indicates how far to move the grid vertically.
  - API: Mozilla Developer Network



HTML5 Canvas Element - CanvasRenderingContext2D Interface

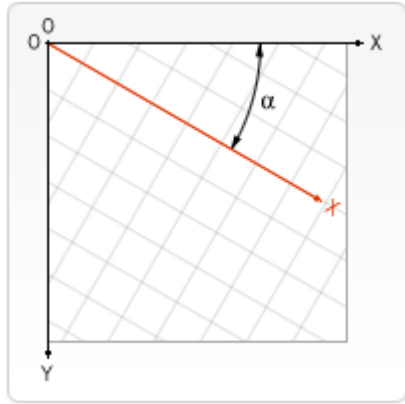
```
function draw() {
  var ctx = document.getElementById('canvas').getContext('2d');
  for (var i=0;i<3;i++) {
    for (var j=0;j<3;j++) {
      ctx.save();
      ctx.fillStyle = 'rgb('+(51*i)+','+(255-51*i)+',255)';
      ctx.translate(10+j*50,10+i*50);
      ctx.fillRect(0,0,25,25);
      ctx.restore();
    }
  }
}
```



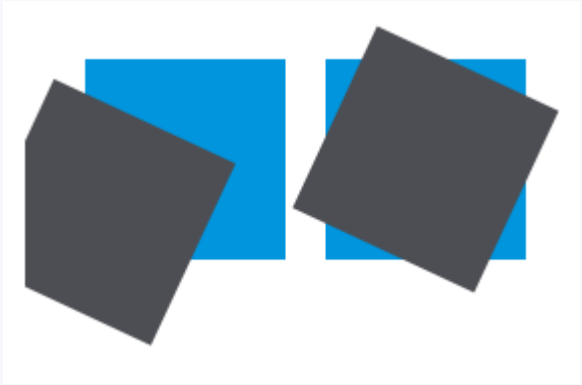
- Without the translate() method, all of the rectangles would be drawn at the same position (0,0). Using the translate() method we don't have to manually adjust the coordinates in the fillRect() function.
- JSFiddle: https://jsfiddle.net/liviucotfas/9tfdegn7/

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Rotating
  - rotate(angle)
    - Rotates the canvas clockwise around the current origin by the angle number of radians.
    - The rotation center point is always the canvas origin, unless it has been changed using the translate() method
    - Note: Angles are in radians, not degrees. Convert using: radians = (Math.PI/180)*degrees.
    - API: Mozilla Developer Network



HTML5 Canvas Element - CanvasRenderingContext2D Interface



- rectangle 1: rotated based on the canvas origin
- rectangle 2: rotated from the center of the rectangle itself with the help of **translate()** method.
- JSFiddle: https://jsfiddle.net/liviucotfas/2yf241q1/

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Scaling
  - scale(x, y)
    - Scales the canvas units by x horizontally and by y vertically.
    - Both parameters are real numbers. Values that are smaller than 1.0 reduce the unit size and values above 1.0 increase the unit size. Values of 1.0 leave the units the same size.
    - Using negative numbers you can do axis mirroring
    - API: Mozilla Developer Network

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- rectangle: scaled
- text: mirrored.
- JSFiddle: https://jsfiddle.net/liviucotfas/Lyycq7gv/

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Combining Transforms
  - the browser is using a transformation matrix

```
context.scale(-.5, 1);
context.rotate(45 * Math.PI / 180);
context.translate(40, 10);
```

  - the translate, rotate, and scale methods end up affecting the values stored by this matrix

$$\begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- Transforms
  - allow modifications directly to the transformation matrix
  - transform(m11, m12, m21, m22, dx, dy)
    - multiplies the current transformation matrix with the matrix described by its arguments.
    - API: Mozilla Developer Network

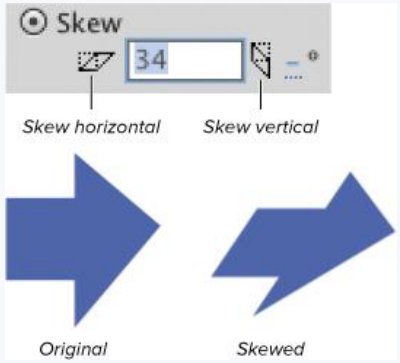
$$\begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- m11 - Horizontal scaling.
- m12 - Horizontal skewing.
- m21 - Vertical skewing.
- m22 - Vertical scaling.
- dx - Horizontal moving.
- dy - Vertical moving.

transform(m11, m12, m21, m22, dx, dy)

$$\begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix}$$



HTML5 Canvas Element - CanvasRenderingContext2D Interface

- ex: coordinates transformation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{21} & d_x \\ m_{12} & m_{22} & d_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- resetTransform()
  - resets the current transform to the identity matrix. This is the same as calling: ctx.setTransform(1, 0, 0, 1, 0, 0);

HTML5 Canvas Element - CanvasRenderingContext2D Interface

- transform(m11, m12, m21, m22, dx, dy)
  - resets the current transform to the identity matrix, and then invokes the transform() method with the same arguments.
  - basically undoes the current transformation, then sets the specified transform, all in one step.

HTML5 Canvas Element - ImageData Interface

- The ImageData interface represents the underlying pixel data of an area of a <canvas> element.

- API: Mozilla Developer Network

HTML5 Canvas Element - ImageData Interface

- It is created using the creator methods on the CanvasRenderingContext2D object associated with a canvas:
  - CanvasRenderingContext2D.**createImageData()**: creates a new, blank ImageData object with the specified dimensions. All of the pixels in the new object are transparent black.
  - CanvasRenderingContext2D.**getImageData(sx, sy, sw, sh)**: returns an ImageData object representing the underlying pixel data for the area of the canvas denoted by the rectangle which starts at (sx, sy) and has an sw width and sh height

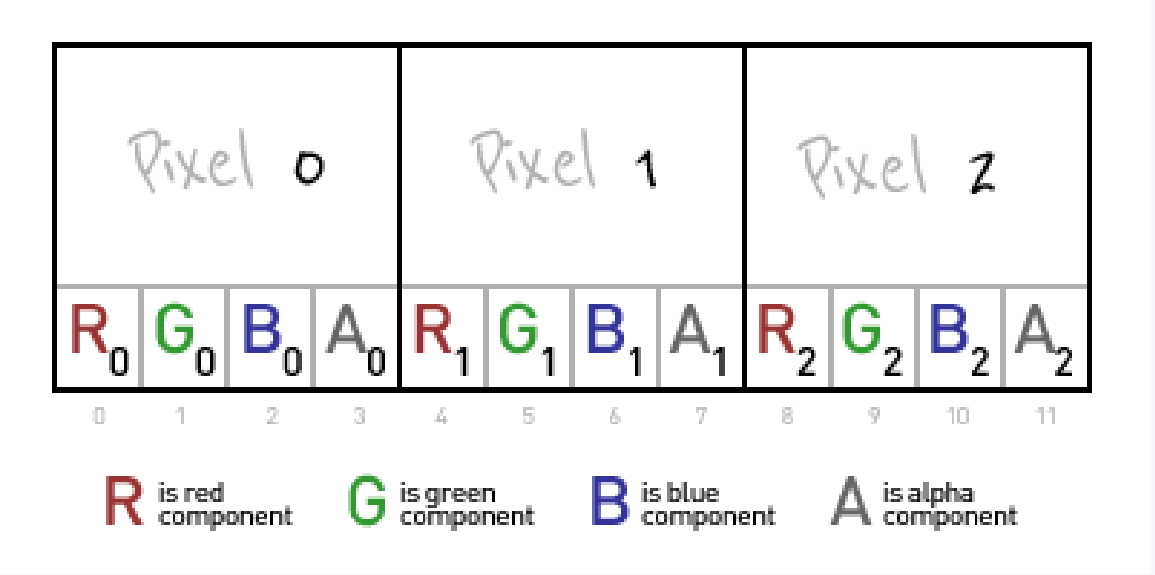
HTML5 Canvas Element - ImageData Interface

- It can also be used to set a part of the canvas by using:
  - void ctx.putImageData(imagedata, dx, dy): paints data from the given ImageData object onto the bitmap at the given coordinates.

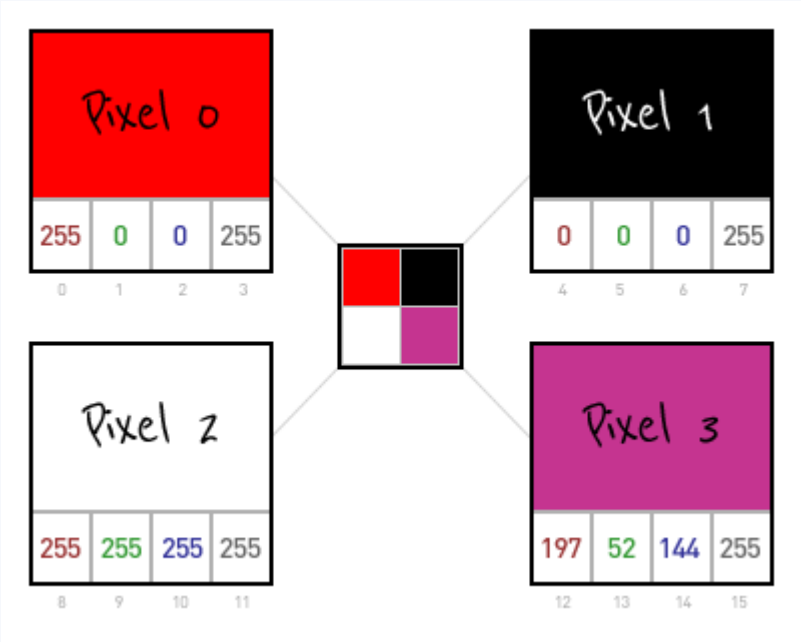
- JSFiddle: https://jsfiddle.net/liviucotfas/64fzcaea/

HTML5 Canvas Element - ImageData Interface

- Properties
  - ImageData.data - a Uint8ClampedArray representing a one-dimensional array containing the data in the RGBA order, with integer values between 0 and 255 (included).



HTML5 Canvas Element - ImageData Interface



[255,0,0,255, 0,0,0,255, 255,255,255,255, 203,53,148,255]

HTML5 Canvas Element - ImageData Interface

- ImageData.height - an unsigned long representing the actual height, in pixels, of the ImageData.

- ImageData.width - an unsigned long representing the actual width, in pixels, of the ImageData.

HTML5 Canvas Element - ImageData Interface

- iterating over all the pixels in a canvas

```
var imageData = context.getImageData(0, 0, canvas.width, canvas.height);

for (var y = 0; y < canvas.height; y++) {
   for (var x = 0; x < canvas.width; x++) {
      var i = (y * canvas.width * 4) + x * 4;

      var rosu = imageData.data[i]; // [0..255]
      var verde = imageData.data[i+1]; // [0..255]
      var albastru = imageData.data[i+2]; // [0..255]
      var transparenta = imageData.data[i+3]; // [0..255]
   }
}
```

HTML5 Canvas Element

- Further reading:
  - http://www.williammalone.com/articles/html5-canvas-javascript-paint-bucket-tool/
  - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Pixel_manipulation_with_canvas

Effects

- Color Filter
  - Red filter: $r' = r; g = 0; b = 0$
- Negative
  - $r' = 255 - r; g' = 255 - g; b' = 255 - b;$
- Greyscale
  - $r' = g' = b' = (r + g + b) / 3$
  - $r' = g' = b' = 0.299 * r + 0.587 * g + 0.114 * b$

Effects

- Brightness
  - r' = r + valoare; if (r > 255) r = 255 else if (r < 0) r = 0;

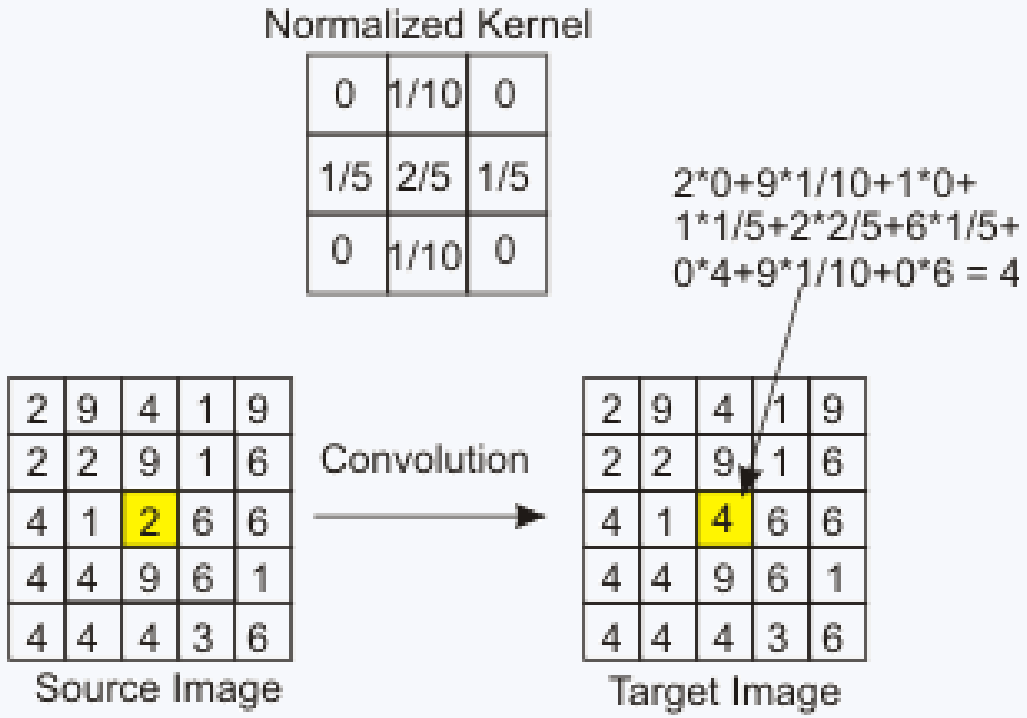
- Threshold
  - v = (0.2126*r + 0.7152*g + 0.0722*b >= threshold) ? 255 : 0; r'= g' = b' = v

Effects

- Further reading / examples:
  - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Pixel_manipulation_with_canvas
  - https://www.html5rocks.com/en/tutorials/canvas/imagefilters/

Effects - Convolution filters

- Calculates the new value for a pixel based on the values of nearby pixels in the original image.



Effects - Convolution filters

- Further reading / examples:
  - https://www.html5rocks.com/en/tutorials/canvas/imagefilters/

Vector Graphics

Vector Graphics

- In **bitmapped graphics**, the computer is given a detailed description of an image that it then matches, pixel by pixel.

- In **vector-drawn graphics**, the computer is given a **set of commands** that it executes to draw the image. Pictures contain *paths* made up of points, lines, curves and shapes.

- A **vector** is a line with a particular length, curvature, and direction. **Vector graphics** are composed of lines that are mathematically defined to form shapes such as rectangles, circles, triangles, and other polygons.

Vector Graphics

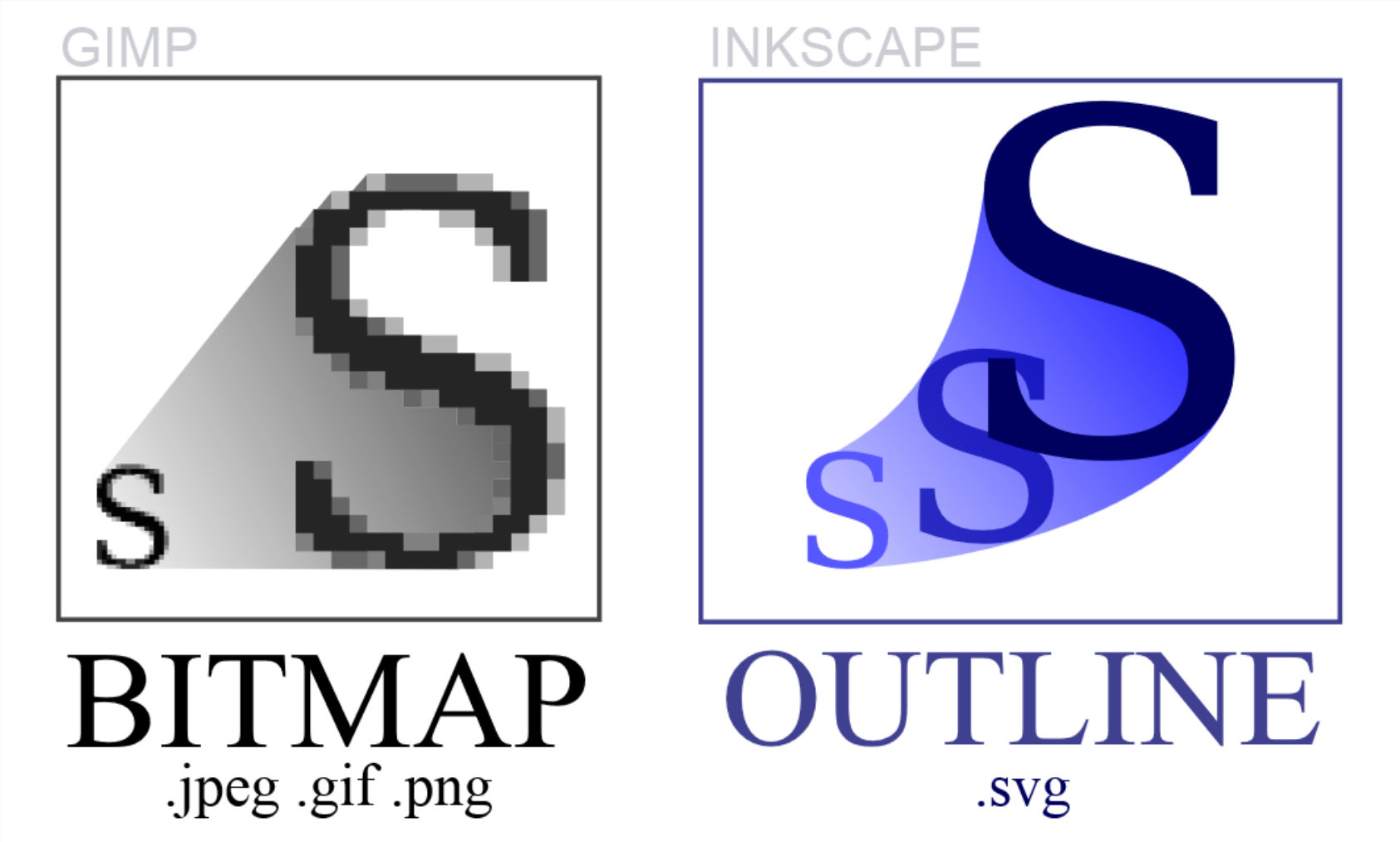
- Each vector path forms the outline of a geometric region containing color information.

Vector Graphics

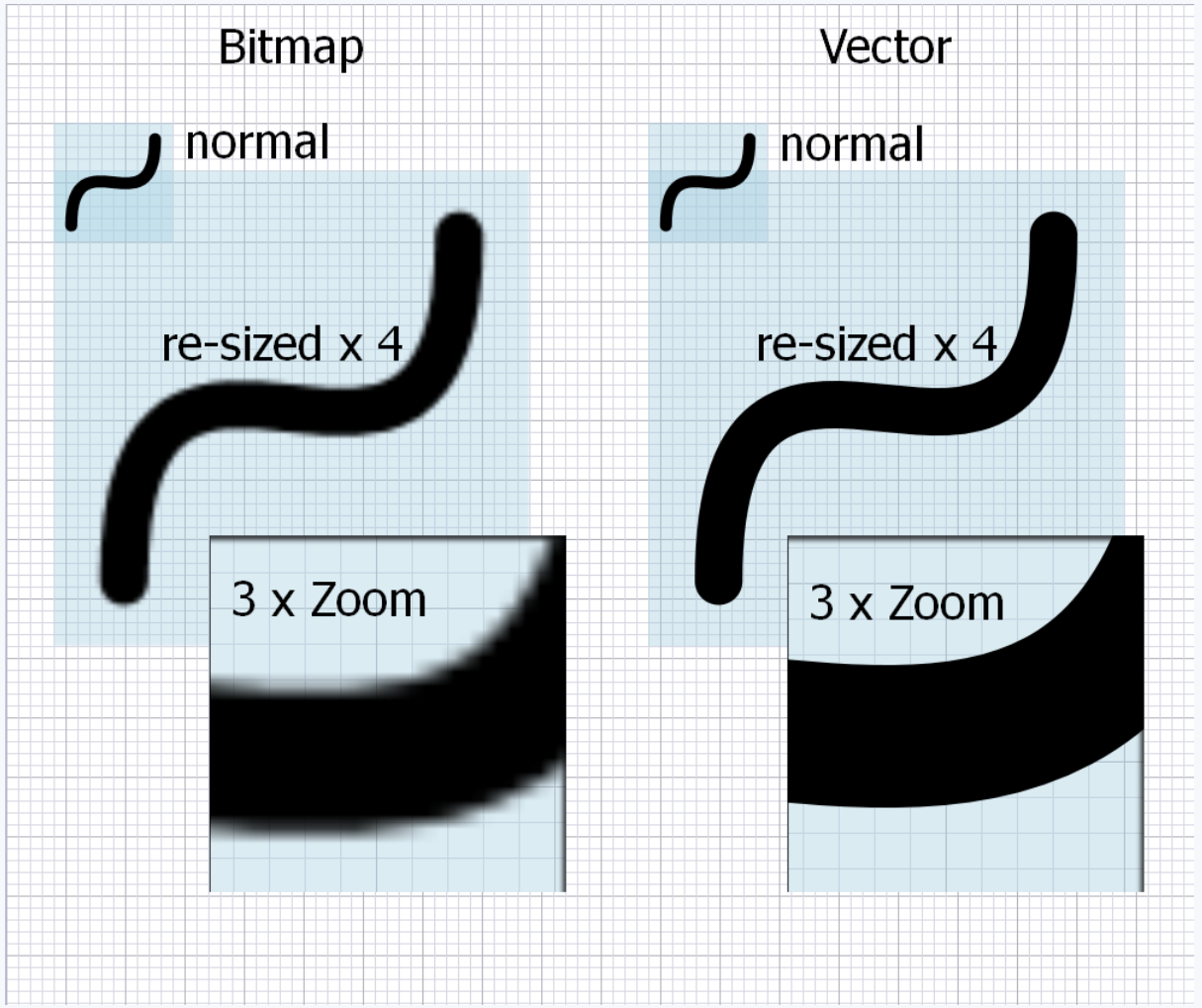
Vector Graphics

- Because paths can be mathematically resized, vector graphics can be scaled up or down without losing any picture clarity.

Scaling / Zoom



Scaling / Zoom



Animation

- You could also use vector graphics to create an animation, such as with Flash. Instead of drawing every separate frame of your project—with 24 frames appearing each second—you could create two different graphics for a segment and let your animation software mathematically interpolate the positions of the components in the in-between frames (a technique known as tweening).

Advantages and Disadvantages

- Advantages:
  - For relatively simple images, the list of drawing commands takes up much less file space than a bitmapped version of the same graphic.
  - A draw program might use a command similar to “RECT 300, 300, RED” to create a red square with sides of 300 pixels. The file for this image contains 15 bytes that encode the alphanumeric information in the command.
  - The same image could also be created with a paint program as a bitmapped graphic. Using 8-bit color resolution (one byte per pixel), this file would require 90,000 bytes (300 × 300). The much smaller files sizes of drawn images can be a significant advantage for multimedia developers.

Advantages and Disadvantages

- Another advantage of vector-drawn graphics is smooth **scaling**. Vector images are enlarged by changing the parameters of their component shapes. The new image can then be accurately redrawn at the larger size without the distortions typical of enlarged bitmapped graphics.

- Images that are needed in several sizes, such as a company logo, are often best handled as vector-drawn graphics.

Advantages and Disadvantages

- Disadvantages:
  - The principal disadvantage of vector-drawn graphics is lack of control over the individual pixels of the image. As a result, draw programs cannot match the capabilities of bitmapped applications for display and editing of photo-realistic images.

Formats

- SVG (Scalable Vector Graphics)
  - XML-based vector image format for two-dimensional graphics
  - provides supports for interactivity and animation

- DXF (Drawing Exchange Format)
  - is a CAD data file format developed by Autodesk for enabling data interoperability between AutoCAD and other programs.

Formats

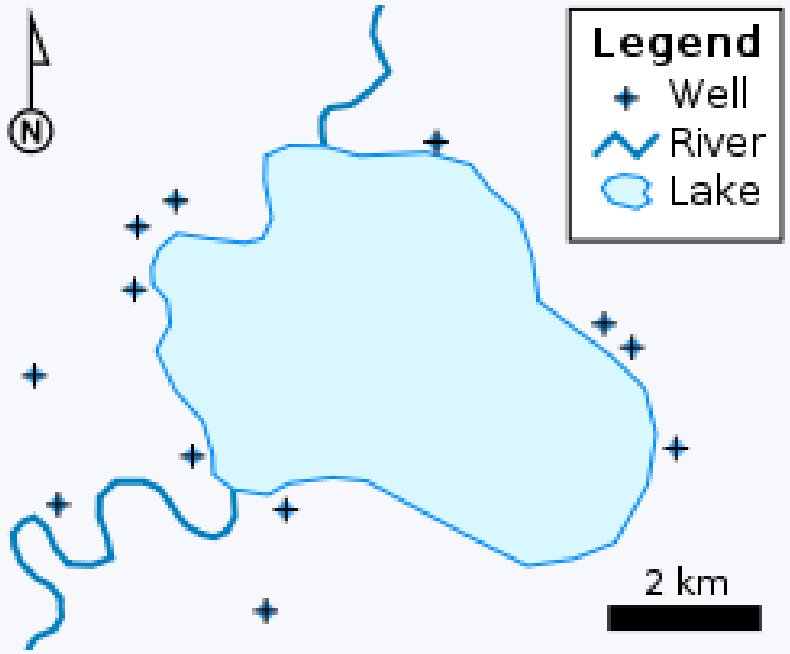
- EPS (Encapsulated Post Script)
  - Created by Adobe Systems for representing vector graphics
  - Uses a computer language called Post Script

- SHP (Shapefile)
  - eveloped and regulated by Esri as a (mostly) open specification for data interoperability among Esri and other GIS software products
  - popular geospatial vector data format for geographic information system (GIS) software

Formats

Simple vector map that can be stored as Shape file:

- points (wells),
- polylines (rivers), and
- polygons (lake).

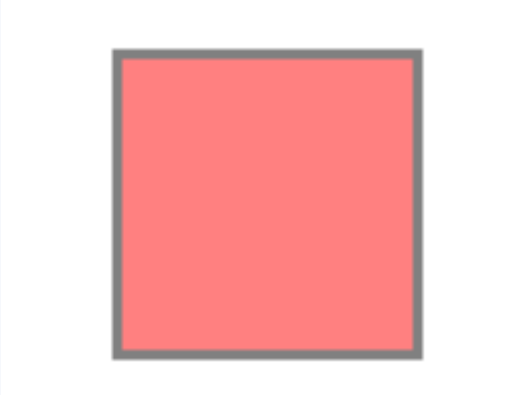


SVG – Scalable Vector Graphics

- XML-based vector image format for two-dimensional graphics
- provides supports for interactivity and animation

```
<!DOCTYPE html>
<html>
<body>
    <svg width="400" height="180">
        <rect x="50" y="20" width="150" height="150"
style="fill:red;stroke:black;stroke-width:5;opacity:0.5">
    </svg>
</body>
```

SVG – Scalable Vector Graphics



SVG – Scalable Vector Graphics

- Creating SVG Images
- SVG images can be created with any text editor, but it is often more convenient to create SVG images with a drawing program, like Inkscape.

SVG – Scalable Vector Graphics

- Line

<line x1="start-x" y1="start-y" x2="end-x" y2="end-y">

- Example:
  - http://www.w3schools.com/graphics/svg_line.asp

```
<svg height="210" width="500">
        <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
</svg>
```

SVG – Scalable Vector Graphics

- Rectangle

<rect x="start-x" y="start-y" width="width" height="height" rx="radius-x" ry="radius-y"/>

- *Example*
  - *http://www.w3schools.com/graphics/svg_rect.asp*

```
<svg width="400" height="110">
 <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
</svg>
```

SVG – Scalable Vector Graphics

- Circle

<circle cx="center-x" cy="center-y" r="radius"/>

- *Example*
  - *http://www.w3schools.com/graphics/svg_circle.asp*

```
<svg height="100" width="100">
 <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```

SVG – Scalable Vector Graphics

- Elipse

<ellipse cx="center-x" cy="center-y" rx="radius-x" ry="radius-y"/>

- Polygon

<polygon points="x1,y1 x2,y2 ..."/>

- Polyline

<polyline points="x1,y1 x2,y2 ..."/>

- Text

<text x="start-x" y="start-y">conținut</text>

SVG – Scalable Vector Graphics

- Grouping elements

<g id="id_grup">... <!– elemente --> </g>

- Defining groups

<defs>... <!– definire grupuri --> </defs>

- Reusing groups

<use xlink:href="#id_grup" x="30" y="14"/>

SVG – Scalable Vector Graphics

- Interacting with SVG using CSS

- Link: *https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/SVG_and_CSS*

SVG – Scalable Vector Graphics

- Interacting with SVG using JavaScript / jQuery
  - similar to the approach used for HTML elements;

- Particularities
  - when creating an element we need to use the SVG namespace

```
document.createElementNS("http://www.w3.org/2000/svg", „TAG_SVG")
```

SVG – Scalable Vector Graphics

- Example

```
$(document.createElementNS("http://www.w3.org/2000/svg", "rect"))
        .attr({x:160, y:160, width:12, height:12})
        .appendTo($("#desen"));
```