



INSTITUTE OF AERONAUTICAL ENGINEERING **(Autonomous)**

Dundigal, Hyderabad - 500 043

Multimedia and Rich Internet Applications

Course Code: A80547

IV B. Tech II semester (JNTUH-R15)

Prepared by:

Ms. Y Harika Devi

Assistant Professor

Multimedia and Rich Internet Applications

UNIT-I

Multimedia

- A PC vendor: a PC that has sound capability, a DVD-ROM drive, and perhaps the superiority of multimedia-enabled microprocessors that understand additional multimedia instructions.
- A consumer entertainment vendor: interactive cable TV with hundreds of digital channels available, or a cable TV-like service delivered

over a high-speed Internet connection.

- A Computer Science (CS) student: applications that use multiple modalities, including text, images, drawings (graphics), animation, video, sound including speech, and interactivity.
- Multimedia and Computer Science:Graphics, HCI, visualization, computer vision, data compression, graph theory, networking, database systems.

Components of Multimedia

Multimedia involves multiple modalities of text, audio, images, drawings, animation, and video. Examples of how these modalities are put to use:

1. Video teleconferencing.
2. Distributed lectures for higher education.
3. Tele-medicine.
4. Co-operative work environments.
5. Searching in (very) large video and image databases for target visual objects.
6. Augmented" reality: placing real-appearing computer graphics and video objects into scenes.
7. Including audio cues for where video-conference participants are located.
8. Building searchable features into new video, and enabling very high to very low-bit-rate use of new, scalable multimedia products.

Multimedia and Hypermedia

History of Multimedia:

1. Newspaper: perhaps the first mass communication medium, uses text, graphics, and images.
2. Motion pictures: conceived of in 1830's in order to observe motion too rapid for perception by the human eye.
3. Wireless radio transmission: Guglielmo Marconi, at Pontecchio, Italy, in 1895
4. Television: the new medium for the 20th century, established video as a commonly available medium and has since changed the world of mass communications.
5. The connection between computers and ideas about multimedia covers what is actually only a short period:
1945 - Vannevar Bush wrote a landmark article describing what amounts to a hypermedia system called Memex.

Hypermedia and Multimedia

- ***Hypermedia***: not constrained to be text-based, can include other media, e.g., graphics, images, and especially the continuous media | sound and video.
- **The World Wide Web (WWW)** | the best example of a hypermedia application.
- **Multimedia** means that computer information can be represented through audio, graphics, images, video, and animation in addition to traditional media.



Fig 1.1: Hypertext is nonlinear

Examples of typical present multimedia applications include:

- Digital video editing and production systems.
- Electronic newspapers/magazines.
- World Wide Web.
- On-line reference works: e.g. encyclopedias, games, etc.
- Home shopping.
- Interactive TV.
- Multimedia courseware.
- Video conferencing.
- Video-on-demand.
- Interactive movies.

World Wide Web

- The W3C has listed the following goals for the WWW:

1. Universal access of webresources (by everyone everywhere).
2. Electiveness of navigating available information.
3. Responsible use of posted material.

- History of the WWW

1960s-Charles Goldfarb et al. developed the Generalized Markup Language (**GML**) for **IBM**.

HTML (Hypertext Markup Language)

- **HTML: a language for publishing Hypermedia on the World Wide Web defined using SGML:**
 1. HTML uses ASCII, it is portable to all different (possibly binary incompatible) computer hardware.
 2. The current version of HTML is version 4.01.
 3. The next generation of HTML is XHTML | a reformulation of HTML using XML.

HTML uses tags to describe document elements:

- `<token params>` defining a starting point,
- `</token>` | the ending point of the element.
- Some elements have no ending tags.

A very simple HTML page is as follows:

```
<HTML> <HEAD>
```

```
<TITLE>
```

A sample web page.

```
</TITLE>
```

```
<META NAME = "Author" CONTENT = "Cranky Professor">
```

```
</HEAD> <BODY>
```

```
<P>
```

We can put any text we like here, since this is a paragraph element.

```
</P>
```

```
</BODY> </HTML>
```

Naturally, HTML has more complex structures and can be mixed in with other standards.

XML (Extensible Markup Language)

- *XML: a markup language for the WWW in which there is modularity of data, structure and view so that user or application can be able to define the tags (structure).*
- *Example of using XML to retrieve stock information from a database according to a user query:*
 1. First use a global Document Type Definition (**DTD**) **that** is already defined.
 2. The server side script will abide by the DTD rules to generate an XML document according to the query using data from your database.
 3. Finally send user the *XML Style Sheet (XSL)* depending on the type of device used to display the information.

- The current XML version is XML 1.0, approved by the W3C in Feb. 1998.
- *XML syntax looks like HTML syntax, although it is much more strict:*

All tags are in lower case, and a tag that has only inline data has to terminate itself, i.e., <token params />.

Uses name spaces so that multiple DTDs declaring different elements but with similar tag names can have their elements distinguished.

- DTDs can be imported from URIs as well.

- The following XML related specifications are also standardized:
 - XML Protocol:** used to exchange XML information between processes.
 - XML Schema:** a more structured and powerful language for defining XML data types (tags).
 - XSL:** basically CSS for XML.

SMIL (Synchronized Multimedia Integration Language)

- **Purpose of SMIL:** it is also desirable to be able to publish multimedia presentations using a markup language.
- *A multimedia markup language needs to enable scheduling and synchronization of different multimedia elements, and define their interactivity with the user.*
- *The W3C established a Working Group in 1997 to come up with specifications for a multimedia synchronization language -SMIL 2.0 was accepted in August 2001.*
- *SMIL 2.0 is specified in XML using a modularization approach similar to the one used in xhtml:*
 - 1. All SMIL elements are divided into modules -sets of XML elements, attributes and values that define one conceptual functionality.

Overview of Multimedia Software Tools

- *The categories of software tools briefly examined here are:*

1. Music Sequencing and Notation
2. Digital Audio
3. Graphics and Image Editing
4. Video Editing
5. Animation
6. Multimedia Authoring

Music Sequencing and Notation

Cakewalk: now called Pro Audio.

- The term sequencer comes from older devices that stored sequences of notes ("events", in MIDI).

Digital Audio

- Digital Audio tools deal with accessing and editing the actual sampled sounds that make up audio:
- Cool Edit: A very powerful and popular digital audio toolkit; emulates a professional audio studio-multi track productions and sound le editing including digital signal processing effects.
- Sound Forge: a sophisticated PC-based program for editing audio WAV les.
- Pro Tools: a high-end integrated audio production and editing environment -MIDI creation and manipulation; powerful audio mixing, recording, and editing software.

Graphics and Image Editing

Adobe Illustrator: A powerful publishing tool from Adobe.

Uses vector graphics; graphics can be exported to Web.

- *Adobe Photoshop: The standard in a graphics, image processing and manipulation tool.*

- Allows layers of images, graphics, and text that can be separately manipulated for maximum flexibility.

- *Macromedia Freehand: a text and web graphics editing*

- Tool that supports many bitmap formats such as GIF, PNG, and JPEG.

Video Editing

- *Adobe Premiere: an intuitive, simple video editing tool for nonlinear editing, i.e., putting video clips into any order:*
 - Video and audio are arranged in tracks".
 - A large library of built-in transitions, filters and motions for clips) *effective multimedia productions with little efforts.*
- *Adobe After Effects: a powerful video editing tool that enables users to add and change existing movies. Can add many effects: lighting, shadows, motion blurring; layers.*
- *Final Cut Pro: a video editing tool by Apple; Macintosh only.*

Animation

- Multimedia APIs:
 - Java3D: API used by Java to construct and render 3D graphics, similar to the way in which the Java Media Framework is used for handling media les.
 - 1. Provides a basic set of object primitives (cube, splines,etc.) for building scenes.
 - 2. It is an abstraction layer built on top of OpenGL or DirectX (the user can select which).
 - DirectX : Windows API that supports video, images, audio and 3-D animation
 - OpenGL: the highly portable, most popular 3-D API.

1-Bit Images

- Images consist of pixels (picture elements in digital images).
- A 1-bit image (also called binary image) consists of on and off bits only and thus is the simplest type of image.
- Each pixel is stored as a single bit (0 or 1)
- It is also sometimes called a *1-bit monochrome (called Lena image by scientists)* image since it contains no color. See Figure in next slide.
- Monochrome

8-Bit Gray-Level Images

- 8-bit image is one for which each pixel has a *gray value* between 0 and 255.
- Each pixel is represented by a single byte.
- The entire image can be thought of as a two-dimensional array of pixel values referred to as a *bitmap*.
- *Image resolution* refers to the number of pixels in a digital image (higher resolution always yields better quality but increases size)

24-Bit Color Images

- In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB.
- Since each value is in the range 0–255, this format supports $256 \times 256 \times 256$, or a total of 16,777,216, possible combined colors; which increases storage size.
- A 640×480 24-bit color image would require 921.6 KB of storage. (without any compression applied)
- Compression is used to decrease the image size by simply grouping pixels effectively.

Higher Bit-Depth Images

- In some fields such as medicine (security cameras, satellite imaging) more accurate images are required to see the patient's liver, for example.
- To get such images, special cameras that view more than just 3 colors (RGB) are used.
- Such images are called *multispectral* (more than three colors) or *hyper spectral* (224 colors for satellite imaging).

8-Bit Color Images

- Color quantizing example: reducing the number of colors required to represent a digital image makes it possible to reduce its file size.
- 8-bit color image (so-called 256 colors). Files use the concept of a *lookup table (LUT)* to store color information.
- For example:
 - if exactly 23 pixels have RGB values (45, 200, 91)
 - then store the value 23 in a three-dimensional array, at the element indexed by the index values [45, 200, 91].
- This data structure is called a *color histogram*.

Notice that the difference between Fig. a, the 24-bit image, and Fig. b, the 8-bit image, is reasonably small.



Fig. a, the 24-bit image



Fig. b, the 8-bit image

Another example for difference between Fig. c, the 24-bit image, and Fig. d the 8-bit image, is reasonably small.



Fig. c, the 24-bit image



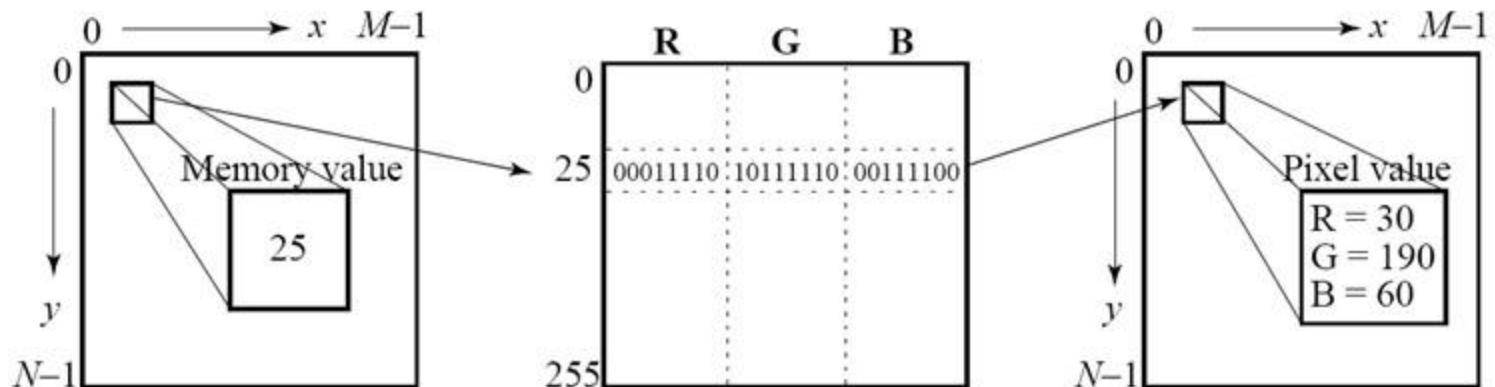
Fig. d, the 8-bit image

8-Bit Color Images

- Note the great savings in space for 8-bit images over 24-bit ones:
- 640×480 8-bit color image requires only 300 KB of storage.
- compared to 921.6 KB for a color image (again, without any compression applied).

Color Lookup Tables

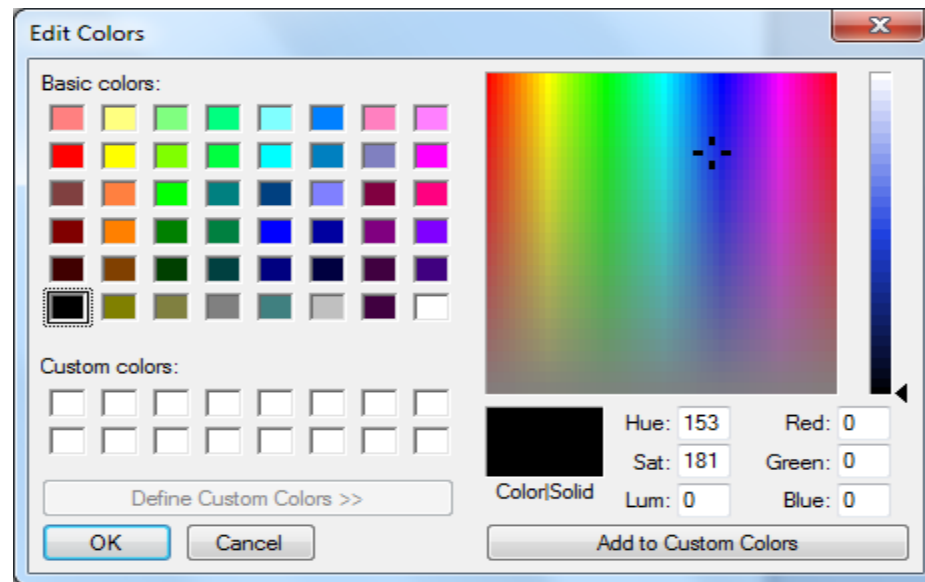
- The LUT is often called a *palette*.
- The idea is to store only the index, or code value, for each pixel.



- A **Color-picker** consists of an array of fairly large blocks of color (or a semi-continuous range of colors) such that a mouse-click will select the color indicated.

- In reality, a color-picker displays the palette colors associated with index values from 0 to 255.

- The below figure displays the concept of a color-picker: if the user selects the color block with index value 2, then the color meant is cyan, with RGB values (0, 255, 255).



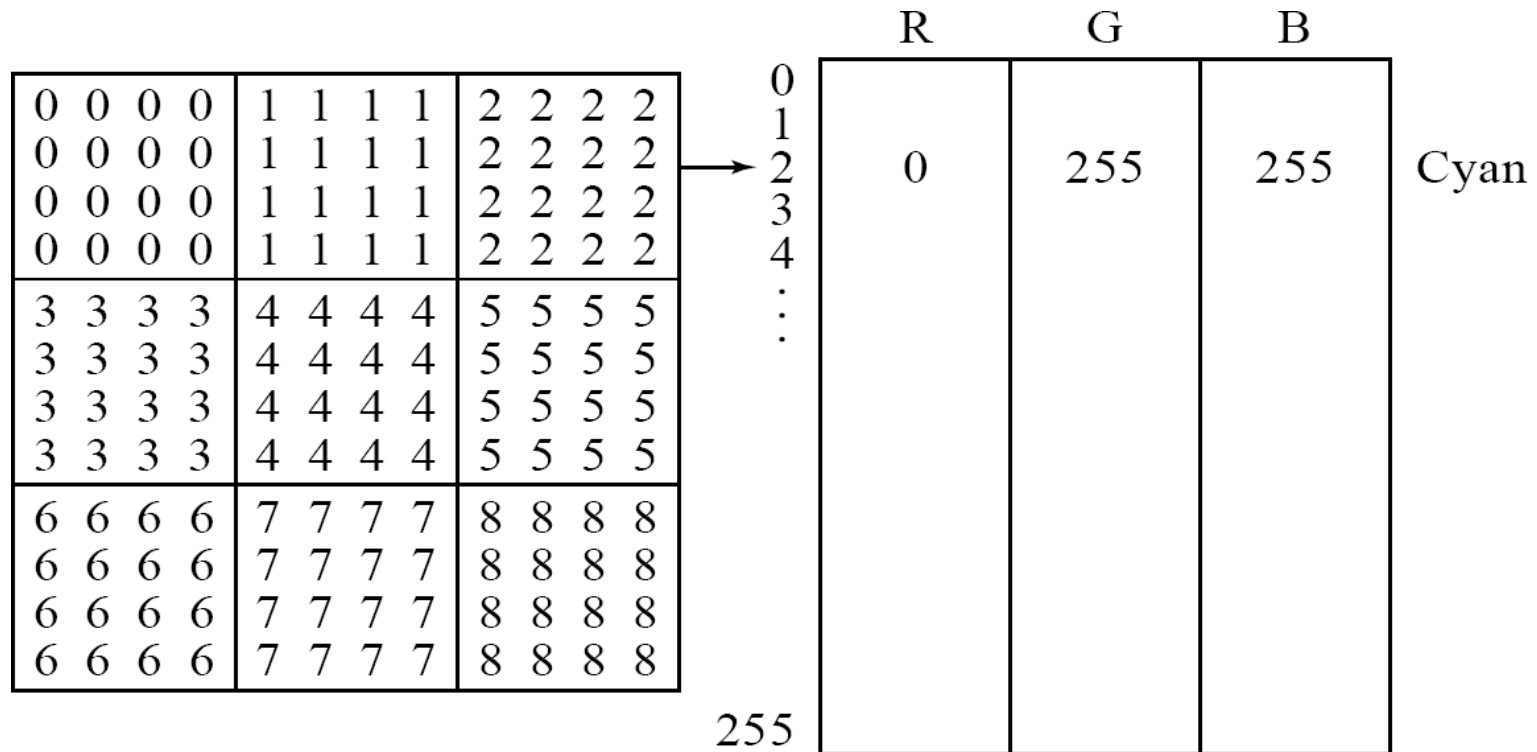


Fig. Color-picker for 8-bit color: each block of the color-picker corresponds to one row of the color LUT

GIF

- GIF standard supports interlacing — successive display of pixels in widely-spaced rows by a 4-pass display process.
- Interlacing allows a quick sketch to appear when a web browser displays the image, followed by more detailed fill-ins.
- The JPEG standard (below) has a similar display mode, denoted *progressive mode*.
- GIF has two formats GIF87 (standard) and GIF89 supports simple animation.

- **Color Map** is set up in a very simple fashion as in Fig. 3.14. However, the actual length of the table equals $2^{(pixel+1)}$ as given in the Screen Descriptor.

Bits								Byte #	
7	6	5	4	3	2	1	0		
Red intensity								1	Red value for color index 0
Green intensity								2	Green value for color index 0
Blue intensity								3	Blue value for color index 0
Red intensity								4	Red value for color index 1
Green intensity								5	Green value for color index 1
Blue intensity								6	Blue value for color index 1
⋮									(continues for remaining colors)

Fig: GIF color map.

- Each image in the file has its own Image Descriptor, defined as in Figure

Bits						Byte #	
7	6	5	4	3	2		
0	0	1	0	1	1	0	0
Image left						1	Image separator character (comma)
						2	Start of image in pixels from the left side of the screen (LSB first)
Image top						3	
						4	Start of image in pixels from the top of the screen (LSB first)
Image width						5	
						6	Width of the image in pixels (LSB first)
Image height						7	
						8	Height of the image in pixels (LSB first)
Image height						9	
						10	
m	i	0	0	0	0	pixel	
						m = 0	Use global color map, ignore 'pixel'
						m = 1	Local color map follows, use 'pixel'
						i = 0	Image formatted in Sequential order
						i = 1	Image formatted in Interlaced order
						pixel + 1	# bits per pixel for this image

Figure: GIF image descriptor.



Figure: JPEG image with low quality specified by user.

PNG

- PNG format: standing for Portable Network Graphics — meant to supersede the GIF standard, and extends it in important ways.
- Special features of PNG files include:
 - 1.Support for up to 48 bits of color information — a large increase.
 - 2.Files may contain gamma-correction information for correct display of color images, as well as alpha-channel information for such uses as control of transparency.
 - 3.The display progressively displays pixels in a 2-dimensional fashion by showing a few pixels at a time over seven passes through each 8 X 8 block of an image.

TIFF

- TIFF: stands for Tagged Image File Format.
- The support for attachment of additional information (referred to as “tags”) provides a great deal of flexibility.
 - 1.The most important tag is a format signifier: what type of compression etc. is in use in the stored image.
 - 2.TIFF can store many different types of image: 1-bit, grayscale, 8-bit color, 24-bit RGB, etc.
 - 3.TIFF was originally a lossless format but now a new JPEG tag allows one to opt for JPEG compression.

PS and PDF

- PostScript is an important language for typesetting, and many high-end printers have a PostScript interpreter built into them.
- PostScript is a vector-based, rather than pixel based, picture language: page elements are essentially defined in terms of vectors.
- PostScript includes vector/structured graphics as well as text
- Several popular graphics programs, such as Adobe Illustrator, use PostScript.
- Note, however, that the PostScript page description language does not provide compression; in fact, PostScript files are just stored as ASCII.

Color Science

- **Light and Spectra**
 - Light is an electromagnetic wave. Its color is characterized by the wavelength content of the light.
- (a) Laser light consists of a single wavelength: e.g., a ruby laser produces a bright, scarlet-red beam.
- (b) Most light sources produce contributions over many wavelengths.
- (c) However, humans cannot detect all light, just contributions that fall in the "visible wavelengths".
- (d) Short wavelengths produce a blue sensation, long wavelengths produce a red one.

Color Science

- **Human Vision**
- *The eye works like a camera, with the lens focusing an image onto the retina (upside-down and left-right reversed).*
- *The retina consists of an array of rods and three kinds of cones. See images (rods_cones, rods_cones1).*
- *The rods come into play when light levels are low and produce a image in shades of gray ("all cats are gray at night!").*
- *For higher light levels, the cones each produce a signal. Because of their differing pigments, the three kinds of cones are most sensitive to red (R), green (G), and blue (B) light.*
- *It seems likely that the brain makes use of differences R-G, G-B, and B-R, as well as combining all of R, G, and B into a high-light-level achromatic channel.*

Color Science

- **Spectral Sensitivity of the Eye**
- The eye is most sensitive to light in the middle of the visible Spectrum.
- *The Blue receptor sensitivity is not shown to scale because it is much smaller than the curves for Red or Green – Blue is a late addition, in evolution.*
- Figure shows the overall sensitivity as a dashed line – this important curve is called the luminous-efficiency function.
 - It is usually denoted $V(\lambda)$ and is formed as the sum of the response curves for Red, Green, and Blue.

Color Science

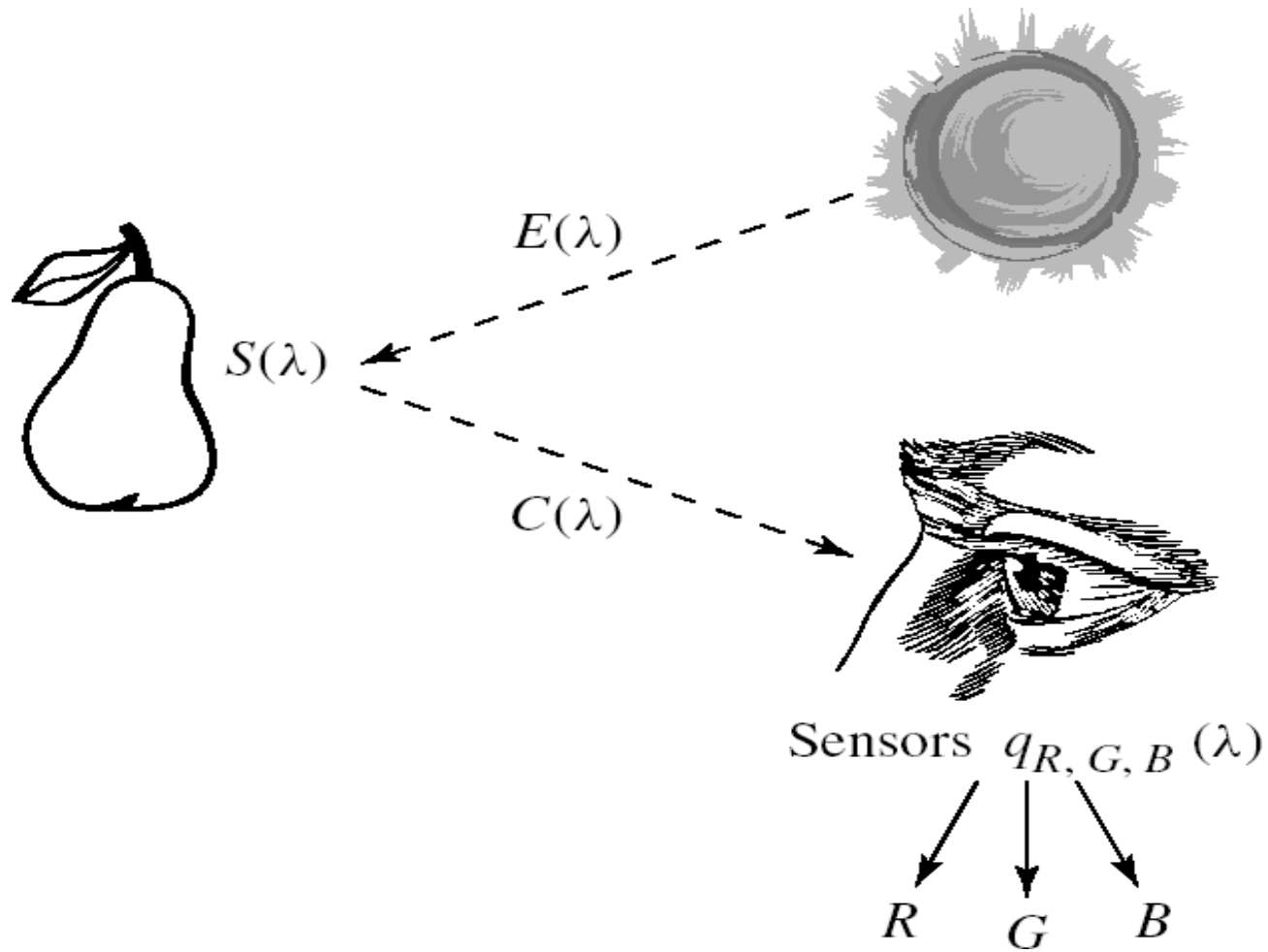


Fig. : Image formation model.

Color Science

- **Camera Systems**
- Camera systems are made in a similar fashion; a good camera has three signals produced at each pixel location (corresponding to a retinal position).
- Analog signals are converted to digital, truncated to integers, and stored. If the precision used is 8-bit, then the maximum value for any of R ; G ; B is 255, and the minimum is 0.

UNIT II

- An analog signal $f(t)$ samples a time-varying image. So-called *progressive* scanning traces through a complete picture (a frame) row-wise for each time interval.
- A high-resolution computer monitor typically uses a time interval of $1/72$ s.
- In TV and in some monitors and multimedia standards, another system, *interlaced* scanning, is used.
- Here, the odd-numbered lines are traced first, then the even-numbered lines.
- This results in “odd” and “even” *fields*—two fields make up one frame.

Interlacing

- In fact, the odd lines (starting from 1) end up at the middle of a line at the end of the odd field, and the even scan starts at a half-way point.
- First the solid lines are traced P to Q , then R to S , and so on, ending at T
- Then the even field starts at U and ends at V .
- The scan lines are not horizontal because a small voltage is applied, moving the electron beam down over time.

Interlacing

Interlacing was invented because, when standards were being defined, it was difficult to transmit the amount of information in a full frame quickly enough to avoid flicker, the double number of fields presented to the eye reduces the eye perceived flicker.

- The jump from Q to R and so on in Figure is called the *horizontal retrace*, during which the electronic beam in the CRT is blanked.
- The jump from T to U or V to P is called the *vertical retrace*.

NTSC Video

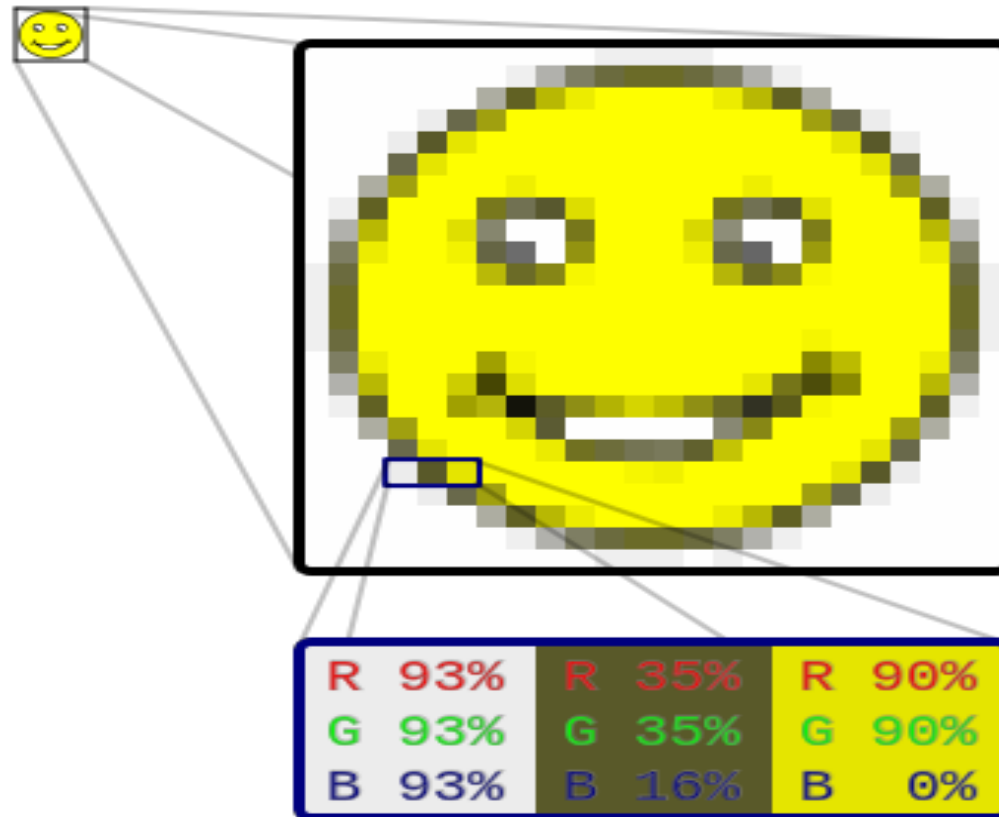
- NTSC stands for (National Television System Committee of the U.S.A)
- The NTSC TV standard is mostly used in North America and Japan.
- It uses a familiar 4:3 *aspect ratio* (i.e., the ratio of picture width to height) and 525 (interlaced) scan lines per frame at 30 fps.
- Figure shows the effect of “vertical retrace and sync” and “horizontal retrace and sync” on the NTSC video raster.

What is Raster Graphics?

- A raster graphics image is a dot matrix data structure representing a generally rectangular grid of pixels, or points of color, viewable via a monitor, paper, or other display medium. (=Bitmap)
- A *raster* is technically characterized by the width and height of the image in pixels and by the number of bits per pixel (a color depth, which determines the number of colors it can represent)
- Most computer images are stored in raster graphics formats.
- Raster graphics are resolution dependent, meaning they cannot scale up to an arbitrary resolution without loss of apparent quality. This property contrasts with the capabilities of vector graphics , which easily scale up to the quality of the device rendering them.

What is Raster Graphics

- The smiley face in the top left corner is a raster image. 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.



NTSC Video

- Figure 5.4 shows the effect of “vertical retrace and sync” and “horizontal retrace and sync” on the NTSC video raster.
- Blanking information is placed into 20 lines reserved for control information at the beginning of each field.
- Hence, the number of *active video lines* per frame is only 485.
- Similarly, almost 1/6 of the raster at the left side is blanked for horizontal retrace and sync.
- The non blanking pixels are called *active pixels*.
- Image data is not encoded in the blanking regions, but other information can be placed there, such as V-chip information, stereo audio channel data, and subtitles in many languages.

NTSC Video

- NTSC video is an analog signal with no fixed horizontal resolution.
- Therefore, we must decide how many times to sample the signal for display.
- Each sample corresponds to one pixel output.
- A *pixel clock* divides each horizontal line of video into samples.
- The higher the frequency of the pixel clock, the more samples per line.
- Different video formats provide different numbers of samples per line.

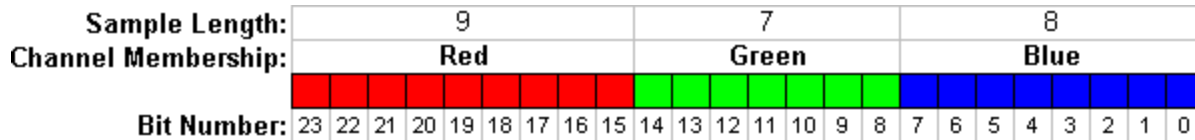
NTSC Video

Table : Samples per line for various analog video formats

Format	Samples per line
VHS	240
S-VHS	400-425
Betamax	500
Standard 8m	300
Hi-8 mm	425

Sampling

- A sample is an intersection of channel and a pixel
- The diagram below depicts a 24-bit pixel, consisting of 3 samples for Red (channel) , Green (channel) , and Blue (channel) .
- In this particular diagram, the Red sample occupies 9 bits, the Green sample occupies 7 bits and the Blue sample occupies 8 bits, totaling 24 bits per pixel



Vertical Trace

- Alternatively referred to as a vertical blanking interval or the vertical sync signal, vertical retrace is used to describe the action performed within the computer monitor that turns the monitor beam off when moving it from the lower-right corner of a monitor to the upper-left of the monitor.
- This action takes place each time the beam has completed tracing the entire screen to create an image.

PAL Video

- PAL (Phase Alternating Line) is a TV standard originally invented by German scientists.
- This important standard is widely used in Western Europe, China, India, and many other parts of the world.
- Because it has higher resolution than NTSC, the visual quality of its pictures is generally better.

Digital Video

- The advantages of digital representation for video:
 - Storing video on digital devices or in memory, ready to be processed (noise removal, cut and paste, and so on) and integrated into various multimedia applications.
 - Direct access, which makes nonlinear video editing simple.
 - Repeated recording without degradation of image quality.
 - Ease of encryption and better tolerance to channel noise.

CCIR and ITU-R Standards for Digital Video

- The CCIR is the Consultative Committee for International Radio.
- One of the most important standards it has produced is CCIR-601 for component digital video.
- This standard has since become standard ITU-R Rec. 601, an international standard for professional video applications.
- It is adopted by several digital video formats, including the popular DV video.

High-Definition TV

The discovery that viewers seated near the screen enjoyed a level of participation (sensation of immersion) not experienced with conventional movies.

- Apparently the exposure to a greater field of view, especially the involvement of peripheral vision, contributes to the sense of “being there.”
- The main thrust of High-Definition TV (HDTV) is not to increase the “definition” in each unit area, but rather to increase the visual field, especially its width.
- First-generation HDTV was based on an analog technology developed by Sony and NHK in Japan in the late 1970s.

High-Definition TV

- Multiple sub-Nyquist Sampling Encoding (MUSE) was an improved NHK HDTV with hybrid analog/digital technologies that was put in use in the 1990s.
- It has 1,125 scan lines, interlaced (60 fields per second), and a 16:9 aspect ratio. (compare with NTSC 4:3 aspect ratio)
- In 1987, the FCC decided that HDTV standards must be compatible with the existing NTSC standard and must be confined to the existing Very High Frequency (VHF) and Ultra High Frequency (UHF) bands.

Ultra High Definition TV (UHDTV)

- UHDTV is a new development—a new generation of HDTV!
- The standards announced in 2012
- The aspect ratio is 16:9.
- The supported frame rate has been gradually increased to 120 fps.

Video Display Interfaces

- Interfaces for video signal transmission from some output devices (e.g., set-top box, video player, video card, and etc.) to a video display (e.g., TV, monitor, projector, etc.).
- There have been a wide range of video display interfaces, supporting video signals of different formats (analog or digital, interlaced or progressive), different frame rates, and different resolutions
 - analog interfaces, including Component Video, Composite Video, and S-Video,
 - and then digital interfaces, including DVI, HDMI, and Display Port.

Analog Display Interfaces

- Analog video signals are often transmitted in one of three different interfaces:
 - Component video,
 - Composite video, and
 - S-video.
- Figure: shows the typical connectors for them



Fig.: Connectors for typical analog display interfaces. From left to right: Component video, Composite video, S-video, and VGA

Analog Display Interfaces

- **Composite Video**
- When connecting to TVs or VCRs, composite video uses only one wire (and hence one connector, such as a BNC connector at each end of a coaxial cable or an RCA plug at each end of an ordinary wire), and video color signals are mixed, not sent separately.
- The audio signal is another addition to this one signal.

Analog Display Interfaces

- **S-Video**
- As a compromise, S-video (separated video, or super-video, e.g., in S-VHS) uses two wires: one for luminance and another for a composite chrominance signal.
- The reason for placing luminance into its own part of the signal is that black-and white information is most important for visual perception.
- As noted in the previous chapter, humans are able to differentiate spatial resolution in the grayscale (“black and-white”) part much better than for the color part of RGB images.
- Therefore, color information transmitted can be much less accurate than intensity information.
- We can see only fairly large blobs of color, so it makes sense to send less color detail.

Analog Display Interfaces

- **Video Graphics Array (VGA)**
- The Video Graphics Array (VGA) is a video display interface that was first introduced by IBM in 1987, along with its PS/2 personal computers. It has since been widely used in the computer industry with many variations, which are collectively referred to as VGA.
- The initial VGA resolution was 640×480 pixels.
- The VGA video signals are based on analog component RGBHV (red, green, blue, horizontal sync, vertical sync).

Digital Display Interfaces

- Given the rise of digital video processing and the monitors that directly accept digital video signals, there is a great demand toward video display interfaces that transmit digital video signals.
- Such interfaces emerged in 1980s (e.g., Color Graphics Adapter (CGA))
- Today, the most widely used digital video interfaces include Digital Visual Interface (DVI), High-Definition Multimedia Interface (HDMI), and Display Port, as shown in Fig.



Fig.: Connectors of different digital display interfaces. From left to right: DVI, HDMI, Display Port

Digital Display Interfaces

- **Digital Visual Interface (DVI)**
- Digital Visual Interface (DVI) was developed by the *Digital Display Working Group* (DDWG) for transferring digital video signals, particularly from a computer's video card to a monitor.
- It carries uncompressed digital video and can be configured to support multiple modes, including DVI-D (digital only), DVI-A (analog only), or DVI-I (digital and analog).
- The support for analog connections makes DVI backward compatible with VGA (though an adapter is needed between the two interfaces).

Digital Display Interfaces

- **Display Port**
- Display Port is a digital display interface. It is the first display interface that uses packetized data transmission, like the Internet or Ethernet
- Display Port can achieve a higher resolution with fewer pins than the previous technologies.
- The use of data packets also allows Display Port to be extensible, i.e., new features can be added over time without significant changes to the physical interface itself.
- Display Port can be used to transmit audio and video simultaneously, or either of them.
- Compared with HDMI, Display Port has slightly more bandwidth, which also accommodates multiple streams of audio and video to separate devices.

D Video and TV

- the rapid progress in the research and development of 3D technology and the success of the 2009 film Avatar have pushed 3D video to its peak.
- The main advantage of the 3D video is that it enables the experience of immersion be there, and really Be there!
- Increasingly, it is in movie theaters, broadcast TV (e.g., sporting events), personal computers, and various handheld devices.

Monocular Cues

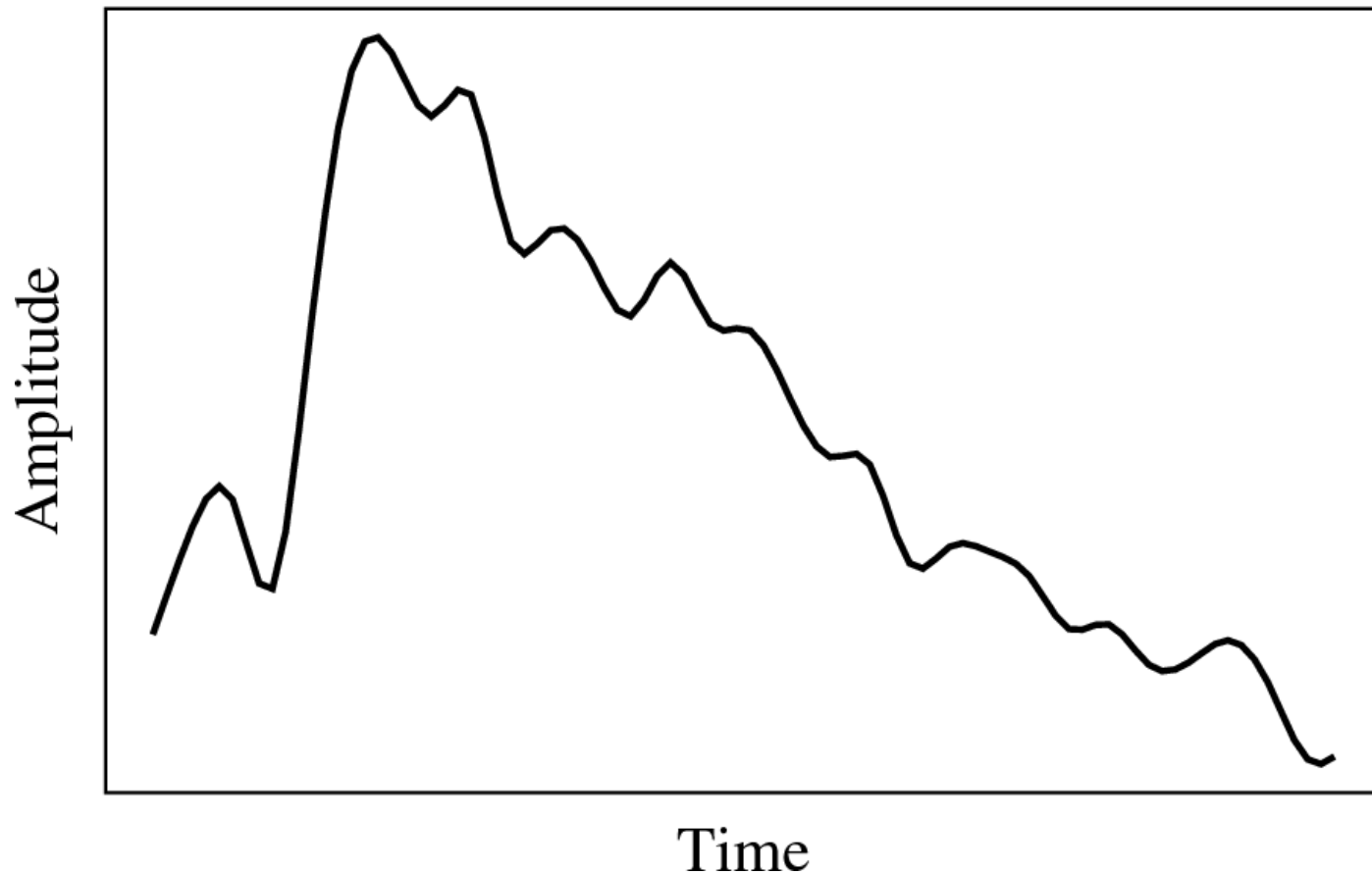
- The monocular cues that do not necessarily involve both eyes include:
 - **Shading**—depth perception by shading and highlights
 - **Perspective** scaling—converging parallel lines with distance and at infinity
 - **Relative** size—distant objects appear smaller compared to known same-size objects not in distance
 - **Texture** gradient—the appearance of textures change when they recede عجارتی in distance
 - **Blur** gradient—objects appear sharper at the distance where the eyes are focused, whereas nearer and farther objects are gradually blurred
 - **Haze**—due to light scattering by the atmosphere, objects at distance have lower contrast and lower color saturation
 - **Occlusion** a far object occluded by nearer object(s)
 - **Motion** parallax—induced by object movement and head movement, such that nearer objects appear to move faster.

Binocular Cues

- The human vision system utilizes effective binocular vision, i.e., stereo vision or stereopsis (Greek word "stereos" which means firm or solid).
- Our left and right eyes are separated by a small distance, on average approximately 2.5 inches, or 65mm, which is known as the *interocular distance*.
- As a result, the left and right eyes have slightly different views, i.e., images of objects are shifted horizontally.
- The amount of the shift, or disparity, is dependent on the object's distance from the eyes, i.e., its depth, thus providing the binocular cue for the 3D percept.
- The horizontal shift is also known as *horizontal parallax*.
- *The fusion* of the left and right images into single vision occurs in the brain, producing the 3D percept.
- Current 3D video and TV systems are almost all based on stereopsis because it is believed to be the most effective cue.

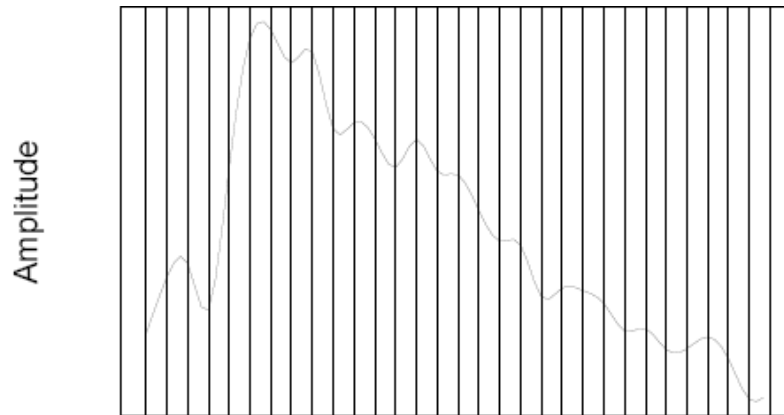
3D Camera Models

- **Simple Stereo Camera Model**
- We can design a simple (artificial) stereo camera system in which the left and right cameras are identical (same lens, same focal length, etc.); the cameras' optical axes are in parallel, pointing at the Z-direction, the scene depth
- **Toed-in Stereo Camera Model**
- Human eyes can be emulated by so-called Toed-in Stereo Cameras, in which the camera axes are usually converging **يبراق** and not in parallel.
- One of the complications of this model is that objects at the same depth (i.e., the same Z) *in the scene no longer yield the same disparity* In other words, the “disparity planes” are now curved.
- Objects on both sides of the view appear farther away than the objects in the middle, even when they have the same depth Z .



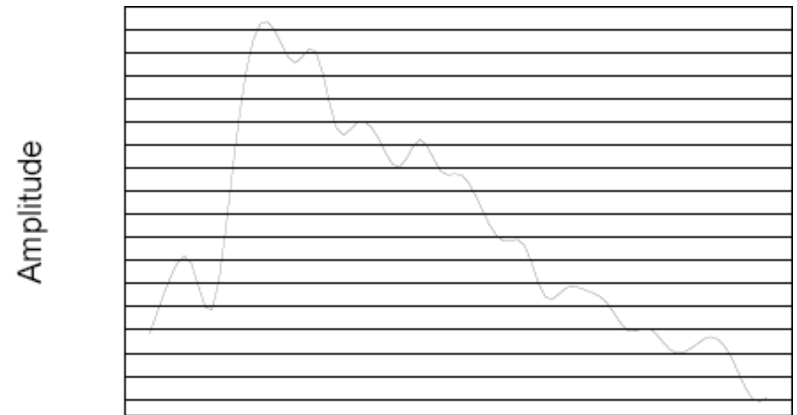
- An analog signal: continuous measurement of pressure wave.
- Sound is 1-dimensional (amplitude values depend on a 1D variable, time) as opposed to images which are 2D (x,y)

- The graph in Fig. 6.1 has to be made digital in both time and amplitude. To digitize, the signal must be **sampled** in each dimension: in time, and in amplitude.
 - (a) Sampling means measuring the quantity we are interested in, usually at evenly-spaced intervals.
 - (b) The first kind of sampling, using measurements only at evenly spaced time intervals, is simply called, sampling. The rate at which it is performed is called the *sampling frequency*.
 - (c) For audio, typical sampling rates are from 8 kHz (8,000 samples per second) to 48 kHz. This range is determined by the Nyquist theorem, discussed later.
 - (d) Sound is a continuous signal (measurement of pressure). Sampling in the amplitude or voltage dimension is called **quantization**. We quantize so that we can represent the signal as a discrete set of values.



Time

(a)



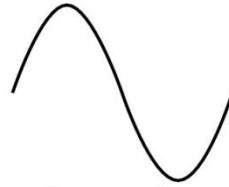
Amplitude

Time

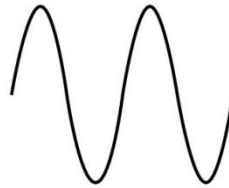
(b)

•Fig. :Sampling and Quantization. (a): Sampling the analog signal in the time dimension. (b): Quantization is sampling the analog signal in the amplitude dimension.

Fundamental
frequency



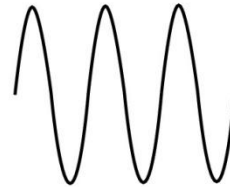
+ 0.5 \times
2 \times fundamental



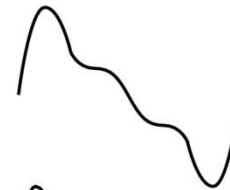
=



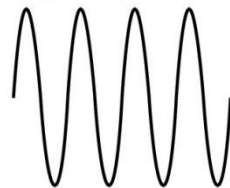
+ 0.33 \times
3 \times fundamental



=



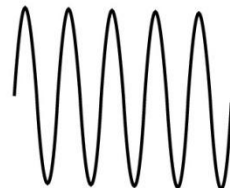
+ 0.25 \times
4 \times fundamental



=



+ 0.5 \times
5 \times fundamental



=



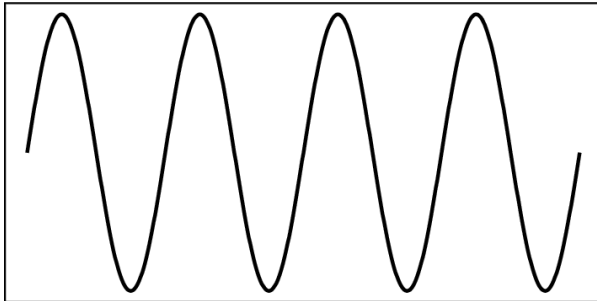
Signals can be decomposed into a weighted sum of sinusoids:

- Building up a complex signal by superposing sinusoids

- Whereas **frequency** is an absolute measure, **pitch** is generally relative — a perceptual subjective quality of sound.
 - (a) Pitch and frequency are linked by setting the note A above middle C to exactly 440 Hz.
 - (b) An **octave** above that note takes us to another A note.

An octave corresponds to *doubling the frequency*. Thus with the middle “A” on a piano (“A4” or “A440”) set to 440 Hz, the next “A” up is at 880 Hz, or one octave above.
 - (c) **Harmonics**: any series of musical tones whose frequencies are integral multiples of the frequency of a fundamental tone.
 - (d) If we allow non-integer multiples of the base frequency, we allow non-“A” notes and have a more complex resulting sound.

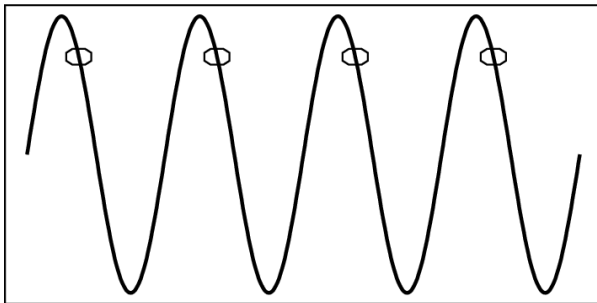
- The Nyquist theorem states how frequently we must sample in time to be able to recover the original sound.
 - (a) Fig. (a) shows a single sinusoid: it is a single, pure, frequency (only electronic instruments can create such sounds).
 - (b) If sampling rate just equals the actual frequency, Fig. (b) shows that a false signal is detected: it is simply a constant, with zero frequency.
 - (c) Now if sample at 1.5 times the actual frequency, Fig. (c) shows that we obtain an incorrect (alias) frequency that is lower than the correct one — it is half the correct one (the wavelength, from peak to peak, is double that of the actual signal).
 - (d) Thus for correct sampling we must use a sampling rate equal to at least twice the maximum frequency content in the signal. This rate is called the Nyquist rate.



(a)

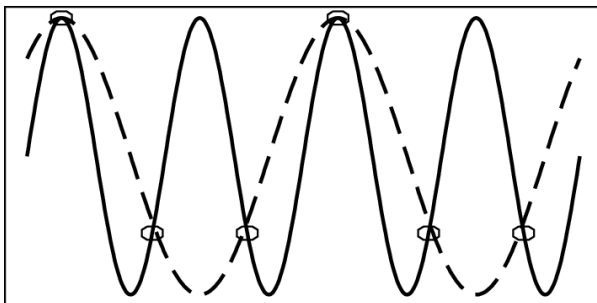
Fig. Aliasing.

(a): A single frequency.



(b)

(b): Sampling at exactly the frequency produces a constant.



(c)

(c): Sampling at 1.5 times per cycle produces an *alias* perceived frequency.

- Nyquist Theorem: If a signal is band-limited, i.e., there is a lower limit f_1 and an upper limit f_2 of frequency components in the signal, then the sampling rate should be at least $2(f_2 - f_1)$.
- Nyquist frequency: half of the Nyquist rate.
 - Since it would be impossible to recover frequencies higher than Nyquist frequency in any event, most systems have an anti aliasing filter that restricts the frequency content in the input to the sampler to a range at or below Nyquist frequency.

Aliasing

- The relationship among the Sampling Frequency,
- True Frequency, and the Alias Frequency is as follows:

$$f_{\text{alias}} = f_{\text{sampling}} - f_{\text{true}}, \text{ for } f_{\text{true}} < f_{\text{sampling}} < 2 \times f_{\text{true}}$$

- If true freq is 5.5 kHz and sampling freq is 8 kHz.
- Then what is the alias freq?

Signal to Noise Ratio (SNR)

- The ratio of the power of the correct signal and the noise is called the *signal to noise ratio* (SNR) — a measure of the quality of the signal.
- The SNR is usually measured in decibels (dB), where 1 dB is a tenth of a bel. The SNR value, in units of dB, is defined in terms of base-10 logarithms of squared amplitudes.

- The usual levels of sound we hear around us are described in terms of decibels, as a ratio to the quietest sound we are capable of hearing. Table 6.1 shows approximate levels for these sounds.

- Table 1: Magnitude levels of common sounds, in decibels

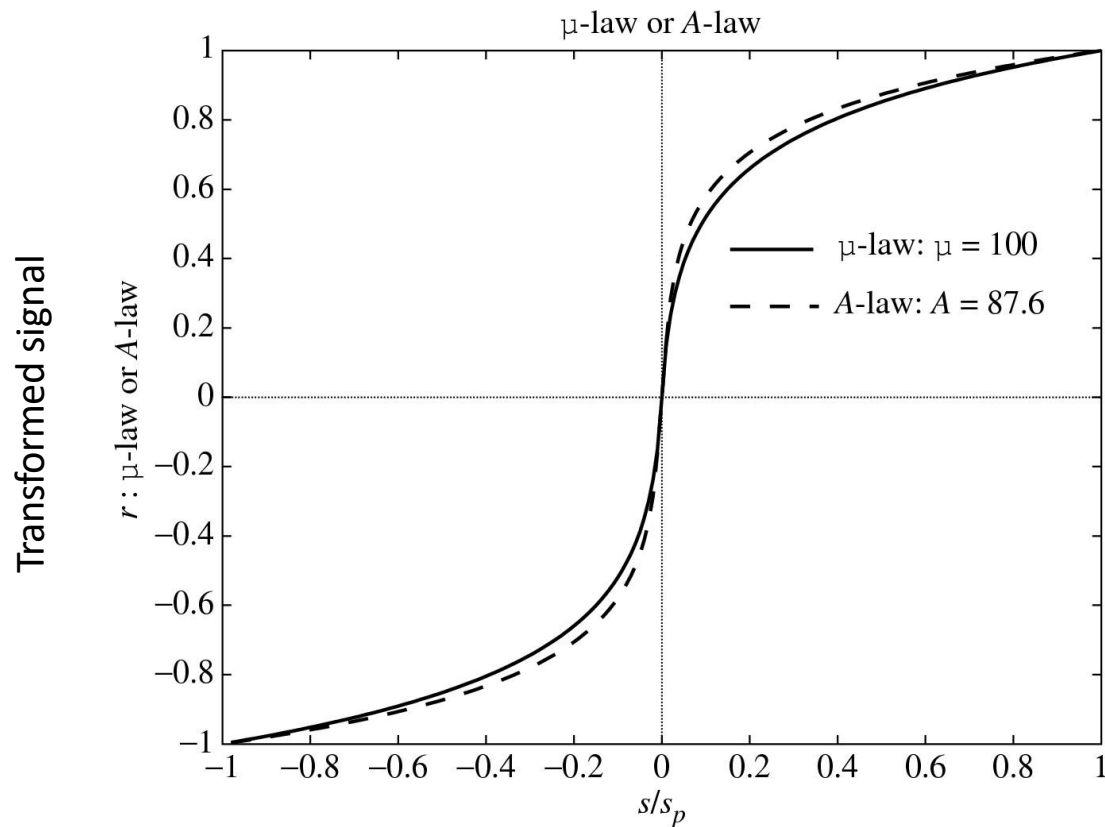
Threshold of hearing	0
Rustle of leaves	10
Very quiet room	20
Average room	40
Conversation	60
Busy street	70
Loud radio	80
Train through station	90
Riveter	100
Threshold of discomfort	120
Threshold of pain	140
Damage to ear drum •	160

Merits of dB

- The decibel's logarithmic nature means that a very large range of ratios can be represented by a convenient number. This allows one to clearly visualize huge changes of some quantity.
- The mathematical properties of logarithms mean that the overall decibel gain of a multi-component system (such as consecutive amplifiers) can be calculated simply by summing the decibel gains of the individual components, rather than needing to multiply amplification factors. Essentially this is because $\log(A \times B \times C \times \dots) = \log(A) + \log(B) + \log(C) + \dots$
- The human perception of sound is such that a doubling of actual intensity causes perceived intensity to always increase by the same amount, irrespective of the original level. The decibel's logarithmic scale, in which a doubling of power or intensity always causes an increase of approximately 3 dB, corresponds to this perception.

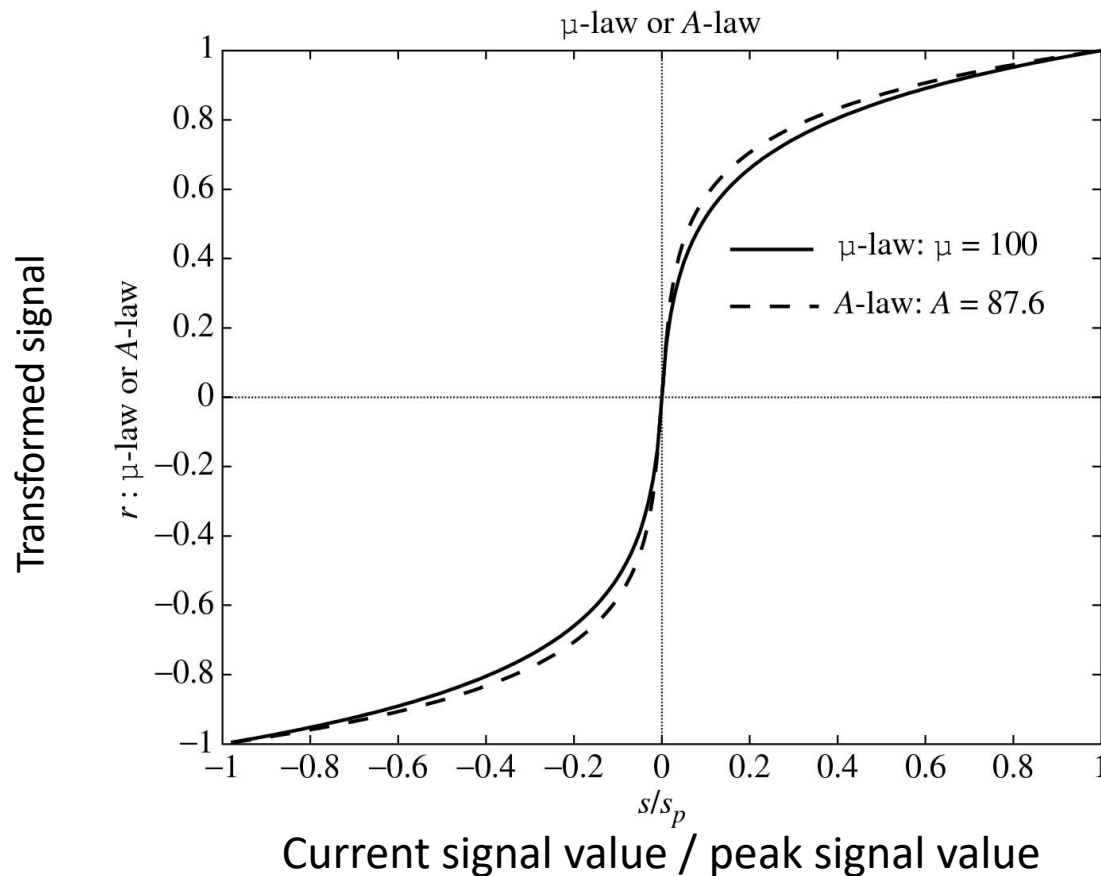
Signal to Quantization Noise Ratio (SQNR)

- Aside from any noise that may have been present in the original analog signal, there is also an additional error that results from quantization.
 - (a) If voltages are actually in 0 to 1 but we have only 8 bits in which to store values, then effectively we force all continuous values of voltage into only 256 different values.
 - (b) This introduces a round off error. It is not really “noise”. Nevertheless it is called quantization noise (or quantization error).



Current signal value / peak signal value

- Fig. : Nonlinear transform for audio signals
- The μ -law in audio is used to develop a non uniform quantization rule for sound: uniform quantization of r gives finer resolution in s at the quiet end (s/s_p near 0).



Values in s get mapped to values in r non-uniformly. “Perceptual coder” – allocates more bits to intervals for which a small change produces a large change in perception.

- Fig. : Nonlinear transform for audio signals
- The μ -law in audio is used to develop a non uniform quantization rule for sound: uniform quantization of r gives finer resolution in s at the quiet end (s/s_p near 0).

Audio Filtering

- Prior to sampling and AD conversion, the audio signal is also usually *filtered* to remove unwanted frequencies. The frequencies kept depend on the application:
 - (a) For speech, typically from 50Hz to 10kHz is retained, and other frequencies are blocked by the use of a band-pass filter that screens out lower and higher frequencies.
 - (b) An audio music signal will typically contain from about 20Hz up to 20kHz.
 - (c) At the DA converter end, high frequencies may reappear in the output — because of sampling and then quantization.
 - (d) So at the decoder side, a low pass filter is used after the DA circuit.

Audio Quality vs. Data Rate

- The uncompressed data rate increases as more bits are used for quantization. Stereo: double the bandwidth. to transmit a digital audio signal.
- Table : Data rate and bandwidth in sample audio applications

Quality	Sample Rate (Khz)	Bits per Sample	Mono / Stereo	Data Rate (uncompressed) (kB/sec)	Frequency Band (KHz)
Telephone	8	8	Mono	8	0.200-3.4
AM Radio	11.025	8	Mono	11.0	0.1-5.5
FM Radio	22.05	16	Stereo	88.2	0.02-11
CD	44.1	16	Stereo	176.4	0.005-20
DAT	48	16	Stereo	192.0	0.005-20
DVD Audio	192 (max)	24(max)	6 channels	1,200 (max)	0-96 (max)

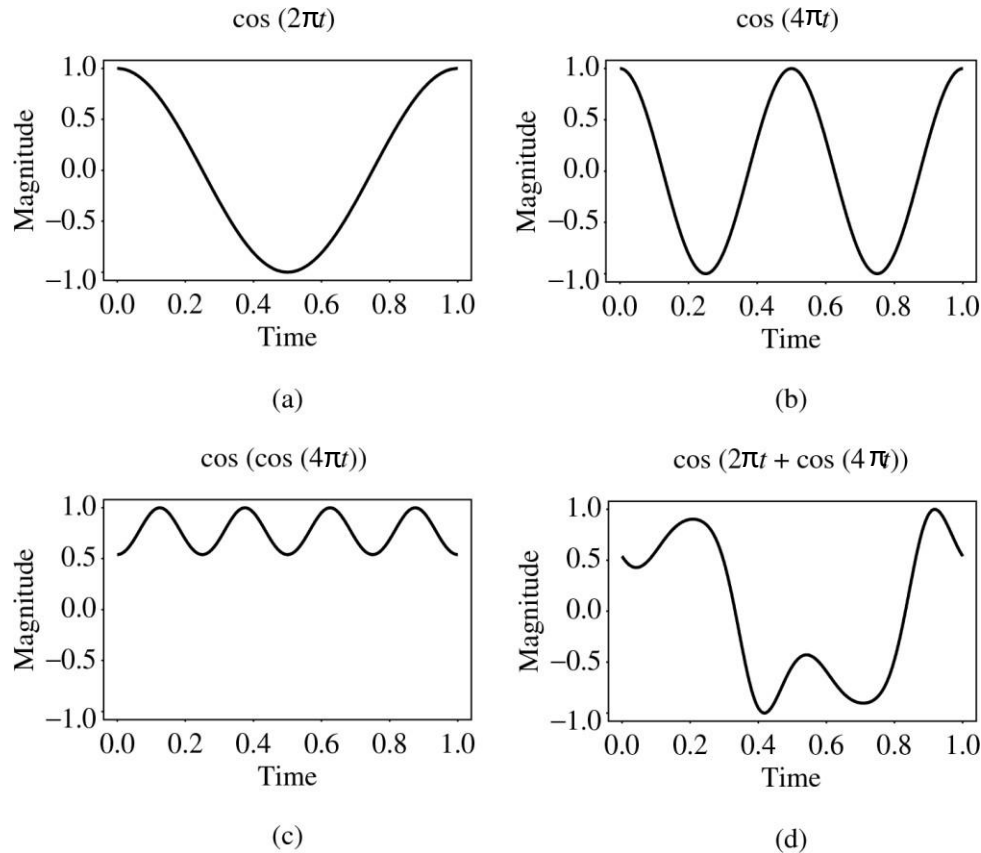


Fig.: Frequency Modulation. (a): A single frequency. (b): Twice the frequency. (c): Usually, FM is carried out using a sinusoid argument to a sinusoid. (d): A more complex form arises from a carrier frequency, $2\pi t$ and a modulating frequency $4\pi t$ cosine inside the sinusoid.

- Wave Table synthesis: A more accurate way of generating sounds from digital signals. Also known, simply, as sampling.
- In this technique, the actual digital samples of sounds from real instruments are stored. Since wave tables are stored in memory on the sound card, they can be manipulated by software so that sounds can be combined, edited, and enhanced.

MIDI: Musical Instrument Digital Interface

- **MIDI Overview**

- (a) MIDI is a protocol adopted by the electronic music industry in the early 80s for controlling devices, such as synthesizers and sound cards, that produce music and allowing them to communicate with each other.
- (b) MIDI is a scripting language — it codes “events” that stand for the production of sounds. E.g., a MIDI event might include values for the pitch of a single note, its duration, and its volume.

- (c) The MIDI standard is supported by most synthesizers, so sounds created on one synthesizer can be played and manipulated on another synthesizer and sound reasonably close.
- (d) Computers must have a special MIDI interface, but this is incorporated into most sound cards.
- (e) A MIDI file consists of a sequence of MIDI instructions (messages). So, would be quite small in comparison to a standard audio file.

MIDI Concepts

- MIDI channels are used to separate messages.
 - (a) There are 16 channels numbered from 0 to 15. The channel forms the last 4 bits (the least significant bits) of the message.
 - (b) Usually a channel is associated with a particular instrument: e.g., channel 1 is the piano, channel 10 is the drums, etc.
 - (c) Nevertheless, one can switch instruments midstream, if desired, and associate another instrument with any channel.

- **System messages**

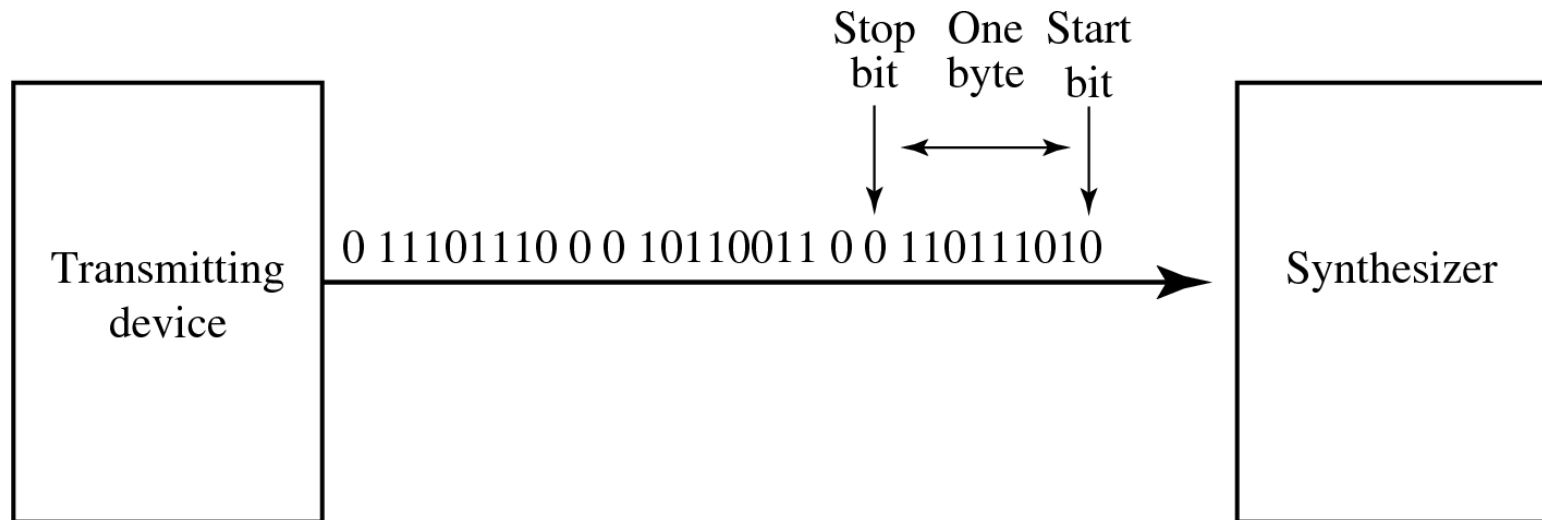
- (a) Several other types of messages, e.g. a general message for all instruments indicating a change in tuning or timing.

- The way a synthetic musical instrument responds to a MIDI message is usually by simply ignoring any play sound message that is not for its channel.

If several messages are for its channel (say play multiple notes on the piano), then the instrument responds, provided it is multi-voice, i.e., can play more than a single note at once (as opposed to violins).

- **General MIDI:** A standard mapping specifying what instruments will be associated with what channels.
 - (a) For most instruments, a typical message might be a Note On message (meaning, e.g., a keypress and release), consisting of what channel, what pitch, and what “velocity” (i.e., volume).
 - (b) For percussion instruments, however, the pitch data means which kind of drum.
 - (c) A Note On message consists of “status” byte — which channel, what pitch — followed by two data bytes. It is followed by a Note Off message, which also has a pitch (which note to turn off) and a velocity (often set to zero).

- The data in a MIDI status byte is between 128 and 255; each of the *data bytes* is between 0 and 127. Actual MIDI bytes are 10-bit, including a 0 start and 0 stop bit.



- A MIDI device often is capable of programmability, and also can change the envelope describing how the amplitude of a sound changes over time.
- Fig.: shows a model of the response of a digital instrument to a Note On message:

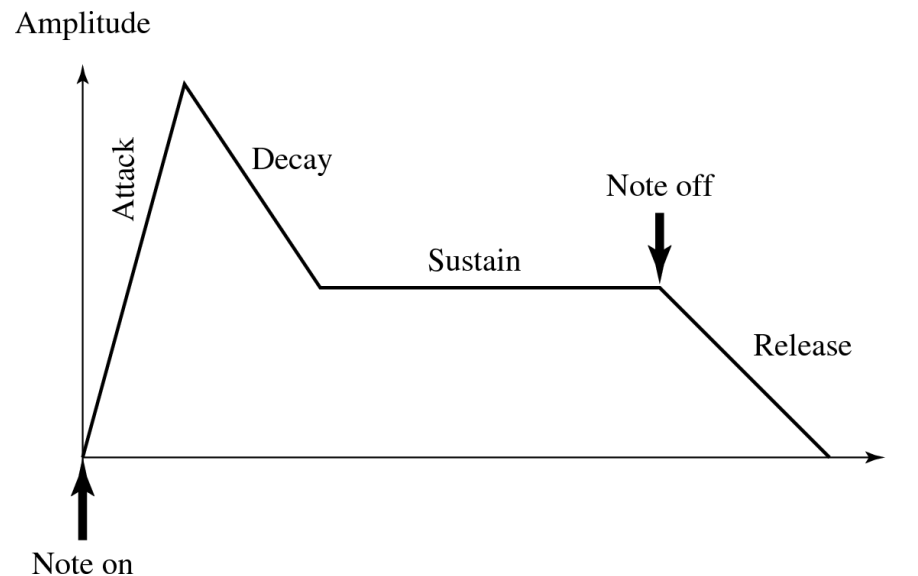
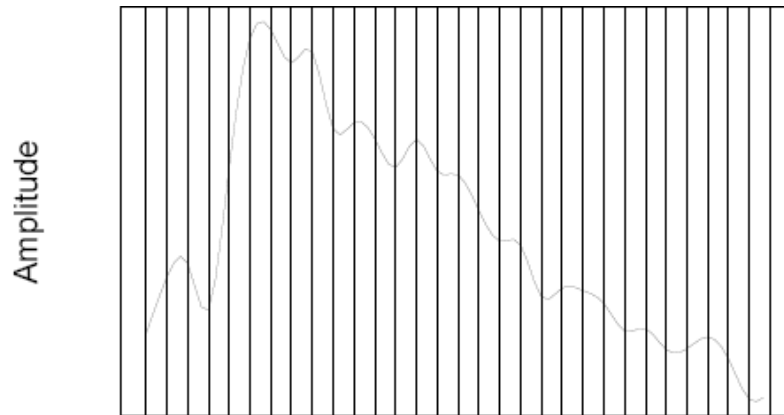


Fig. : Stages of amplitude versus time for a music note

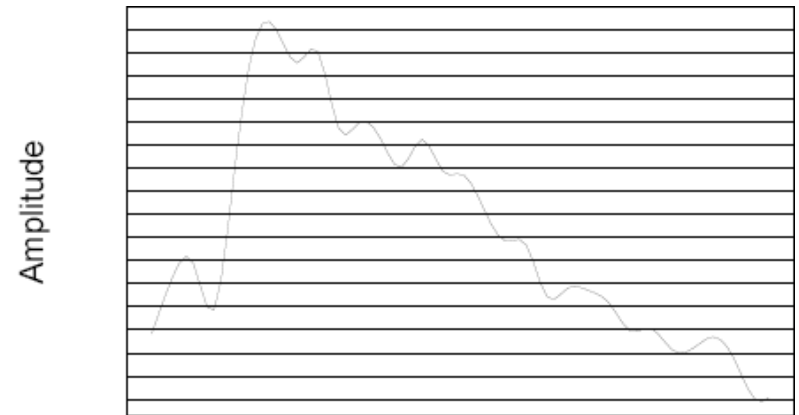
Quantization and Transmission of Audio

- **Coding of Audio:** Quantization and transformation of data are collectively known as **coding** of the data.
 - a) For audio, the μ -law technique for compounding audio signals is usually combined with an algorithm that exploits the temporal redundancy present in audio signals.
 - b) Encoding differences in signals between the present and a past time can reduce the size of signal values into a much smaller range.
 - c) The result of reducing the variance of values is that lossless compression methods produce a bit stream with shorter bit lengths for more likely values.



Time

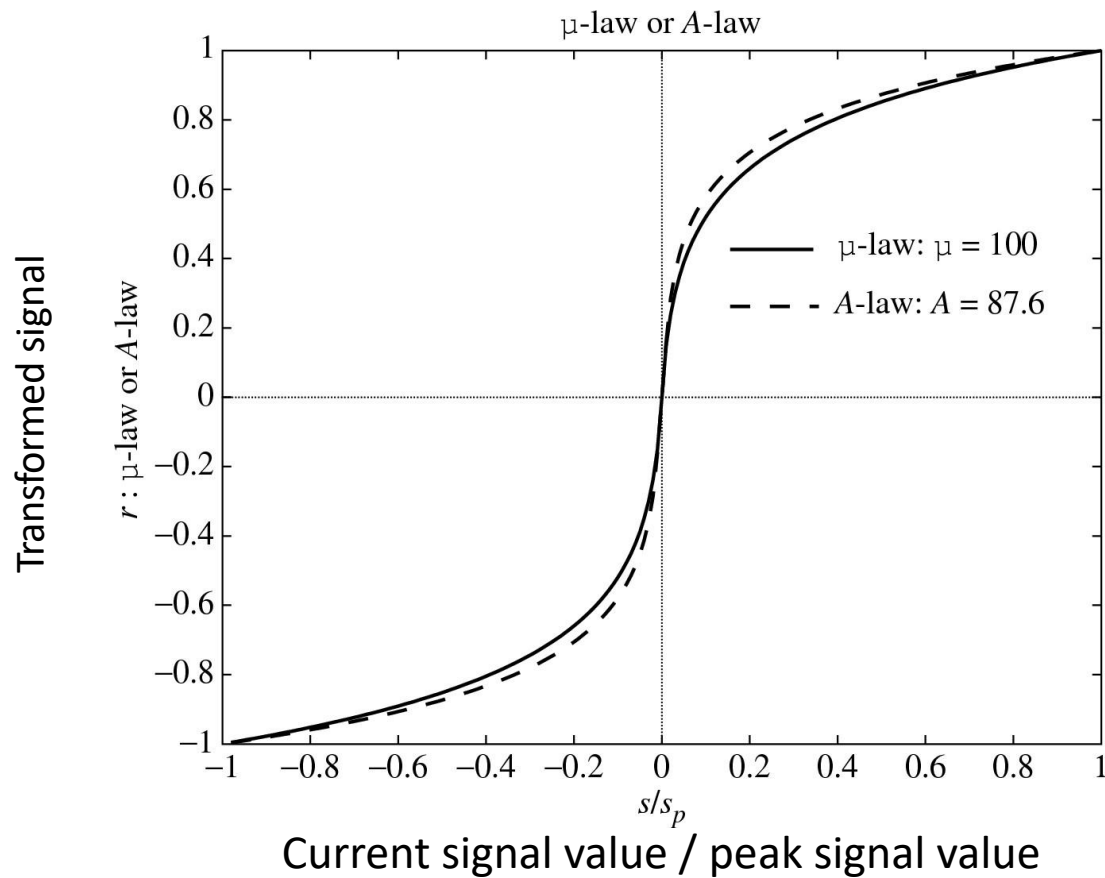
(a)



Time

(b)

- Fig. : Sampling and Quantization.



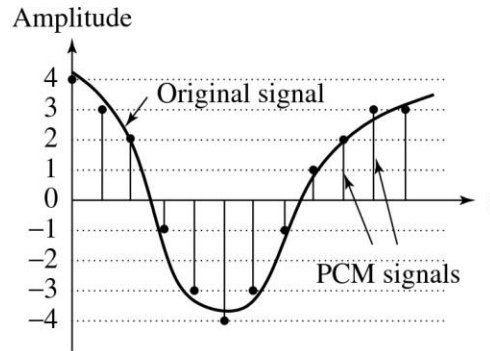
- Fig. : Nonlinear transform for audio signals
- The μ -law in audio is used to develop a non uniform quantization rule for sound: uniform quantization of r gives finer resolution in s at the quiet end (s/s_p near 0).
- Encode in this non-uniform space - can use fewer quantization levels for same perceptual quality (saw this in mat lab demo).

PCM in Speech Compression

- Assuming a bandwidth for speech from about 50 Hz to about 10 kHz, the Nyquist rate would dictate a sampling rate of 20 kHz.
 - (a) Using uniform quantization, the minimum sample size we could get away with would likely be about 12 bits. Hence for mono speech transmission the bit-rate would be 240 kbps.
 - (b) With non-uniform quantization, we can reduce the sample size down to about 8 bits with the same perceived level of quality, and thus reduce the bit-rate to 160 kbps.
 - (c) However, the standard approach to telephony in fact assumes that the highest-frequency audio signal we want to reproduce is only about 4 kHz. Therefore the sampling rate is only 8 kHz, and the compounded bit-rate.

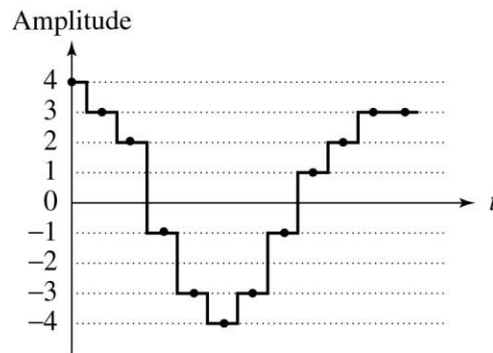
- However, there are two small wrinkles we must also address:
 1. Since only sounds up to 4 kHz are to be considered, all other frequency content must be noise. Therefore, we should remove this high-frequency content from the analog input signal. This is done using a band-limiting filter that blocks out high, as well as very low, frequencies.

—

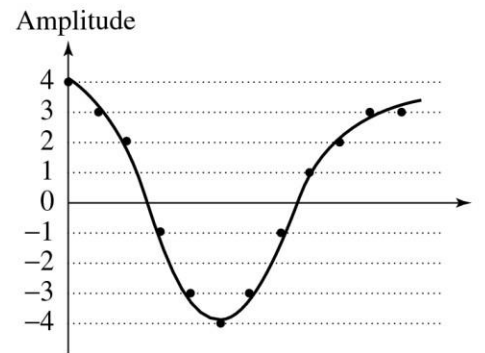


(a)

Signal decoded
from sample
points



(b)

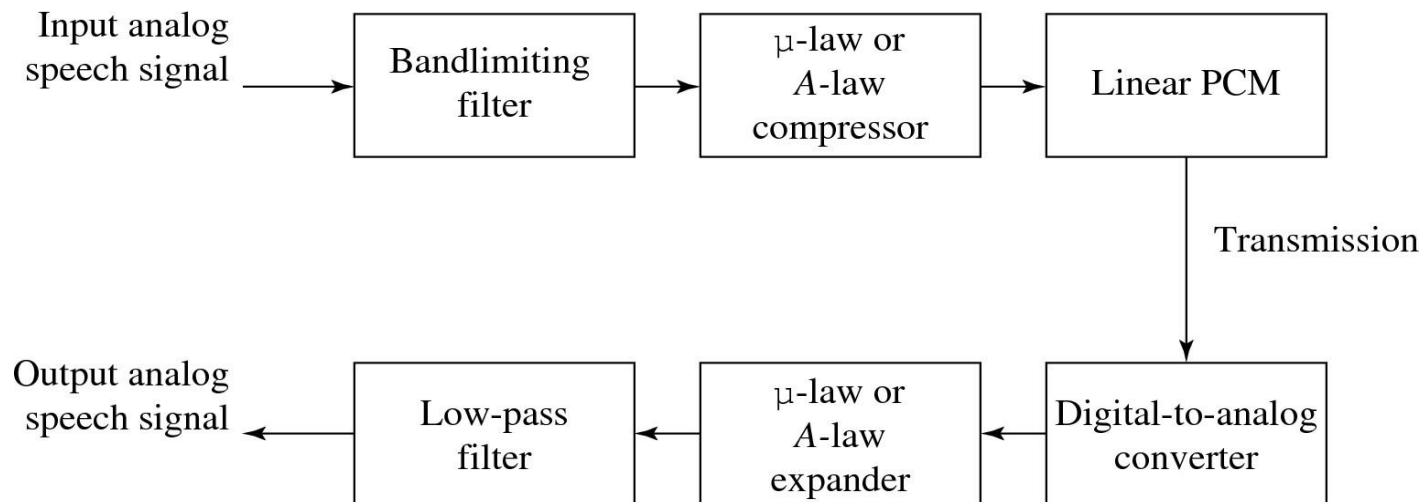


(c)

Reconstructed
signal after low-
pass filtering

•Fig. : Pulse Code Modulation (PCM). (a) Original analog signal and its corresponding PCM signals. (b) Decoded staircase signal. (c) Reconstructed signal after low-pass filtering.

The complete scheme for encoding and decoding telephony signals is shown as a schematic in Figure below. As a result of the low-pass filtering, the output becomes smoothed.



- Fig. : PCM signal encoding and decoding.

Differential Coding of Audio

- Audio is often stored not in simple PCM but instead in a form that exploits differences — which are generally smaller numbers, so offer the possibility of using fewer bits to store.
 - (a) If a time-dependent signal has some consistency over time (“temporal redundancy”), the difference signal, subtracting the current sample from the previous one, will have a more peaked histogram, with a maximum around zero.

- (b) For example, as an extreme case the histogram for a linear ramp signal that has constant slope is flat, whereas the histogram for the derivative of the signal (i.e., the differences, from sampling point to sampling point) consists of a spike at the slope value.
- (c) So if we then go on to assign bit-string code words to differences, we can assign short codes to prevalent values and long code words to rarely occurring ones.

Lossless Predictive Coding

- **Predictive coding:** simply means transmitting differences — predict the next sample as being equal to the current sample; send not the sample itself but the difference between previous and next.
 - (a) Predictive coding consists of finding differences, and transmitting these using a PCM system.
 - (b) Note that differences of integers will be integers. Denote the integer input signal as the set of values f_n . Then we **predict** values \hat{f}_n as simply the previous value.

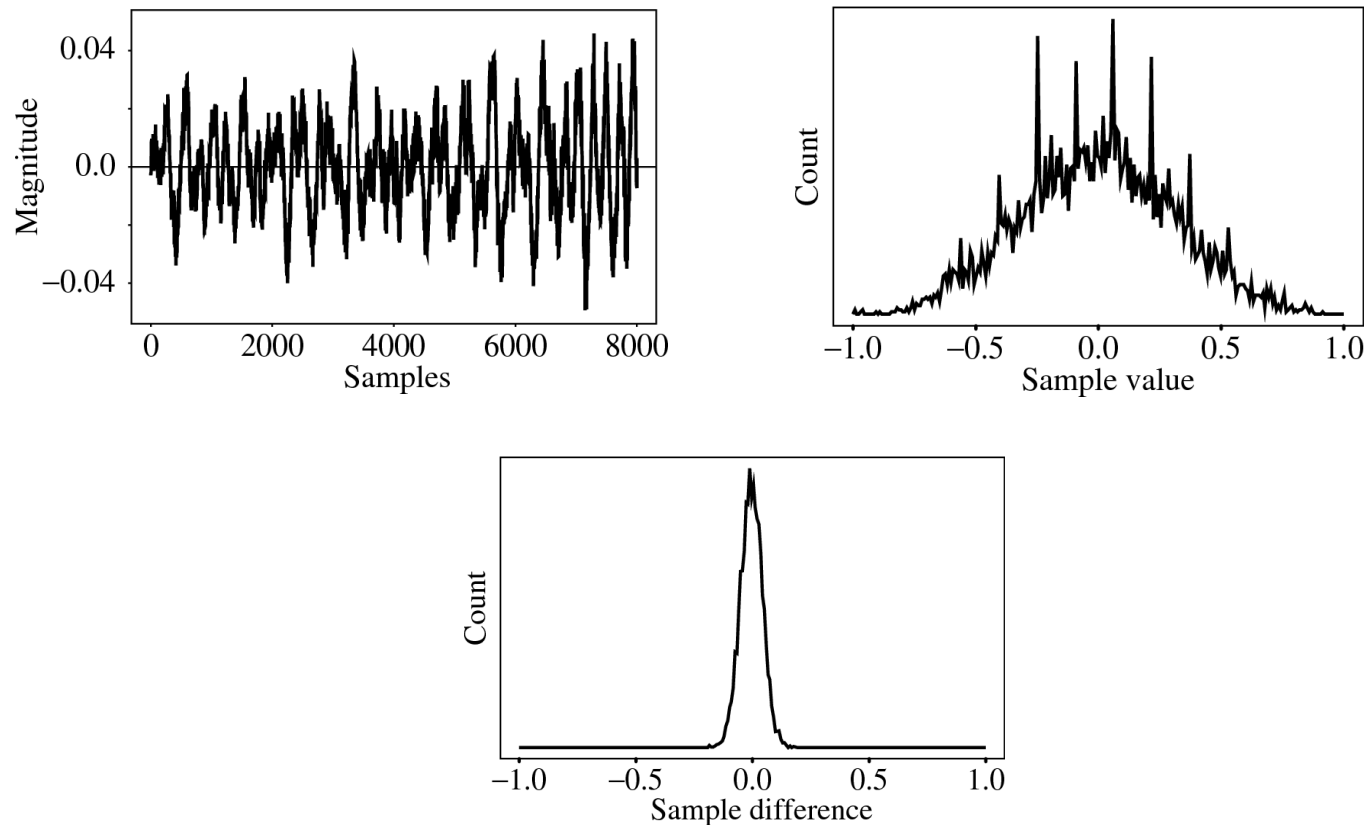
$$\hat{f}_n = f_{n-1}$$

$$e_n = f_n - \hat{f}_n$$

(c) But it is often the case that some function of a few of the previous values, $f_{n-1}, f_{n-2}, f_{n-3}$, etc., provides a better prediction. Typically, a linear predictor function is used:

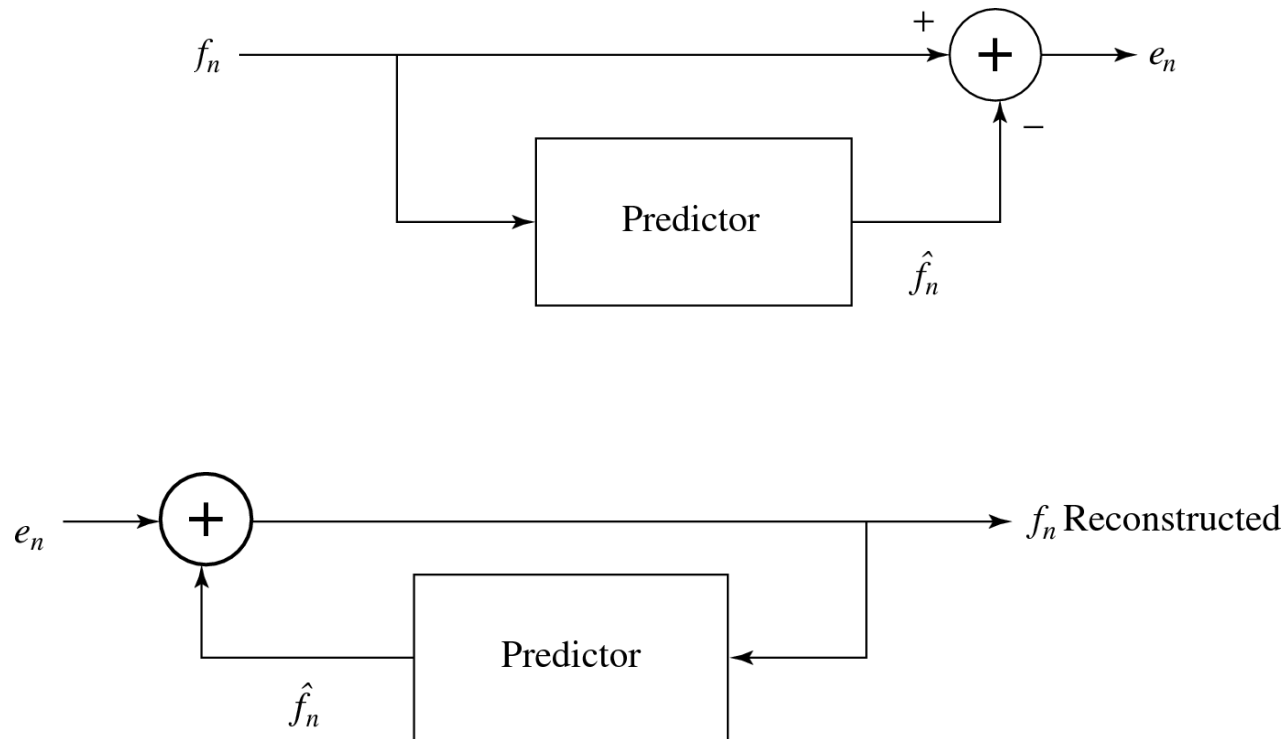
$$f_n = \sum_{k=1}^{2 \text{ to } 4} a_{n-k} f_{n-k}$$

- The idea of forming differences is to make the histogram of sample values more peaked.
 - (a) For example, Fig.(a) plots 1 second of sampled speech at 8 kHz, with magnitude resolution of 8 bits per sample.
 - (b) A histogram of these values is actually centered around zero, as in Fig.(b).
 - (c) Fig.(c) shows the histogram for corresponding speech signal differences. The difference values are much more clustered around zero than are sample values themselves.
 - (d) As a result, a method that assigns short code words to frequently occurring symbols will assign a short code to zero and do rather well: such a coding scheme will much more efficiently code sample differences than samples themselves.



•Fig.: Differencing concentrates the histogram. (a): Digital speech signal. (b): Histogram of digital speech signal values. (c): Histogram of digital speech signal differences.

- One problem: suppose our integer sample values are in the range $0..255$. Then differences could be as much as $-255..255$ —we've increased our dynamic range (ratio of maximum to minimum) by a factor of two → need more bits to transmit some differences.
 - (a) A clever solution for this: define two new codes, denoted SU and SD, standing for Shift-Up and Shift-Down. Some special code values will be reserved for these.
 - (b) Then we can use code words for only a limited set of signal differences, say only the range $-15..16$. Differences which lie in the limited range can be coded as is, but with the extra two values for SU, SD, a value outside the range $-15..16$ can be transmitted as a series of shifts, followed by a value that is indeed inside the range $-15..16$.



- Fig. : Schematic diagram for Predictive Coding encoder and decoder.

Introduction

- **Compression:** the process of coding that will effectively reduce the total number of bits needed to represent certain information.
- Figure 7.1 depicts a general data compression scheme, in which compression is performed by an encoder and decompression is performed by a decoder.

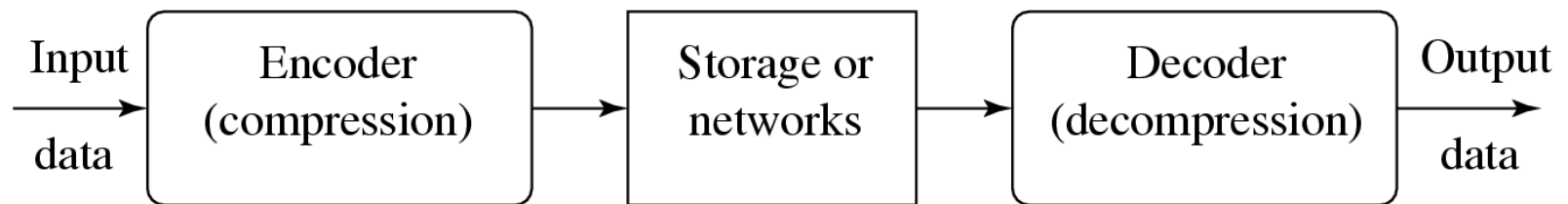
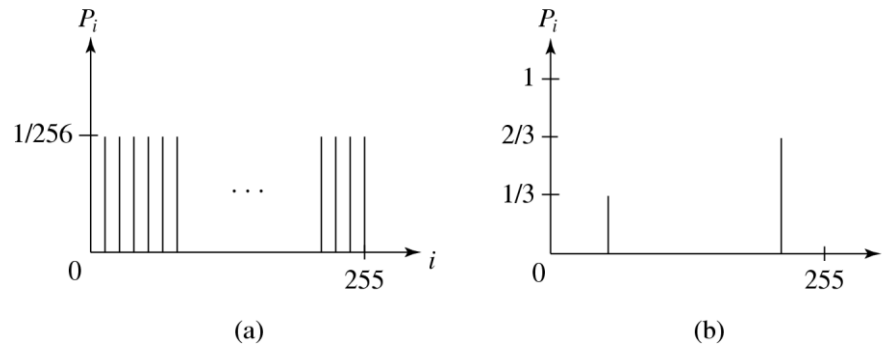


Fig.: A General Data Compression Scheme.

Basics of Information Theory

- What is entropy? is a measure of the number of specific ways in which a system may be arranged, commonly understood as a measure of the disorder of a system.
- As an example, if the information source S is a gray-level digital image, each s_i is a gray-level intensity ranging from 0 to $(2^k - 1)$, where k is the number of bits used to represent each pixel in an uncompressed image.
- We need to find the entropy of this image; which the number of bits to represent the image after compression.

Distribution of Gray-Level Intensities



- Fig.: Histograms for Two Gray-level Images.
- Fig. (a) shows the histogram of an image with *uniform* distribution of gray-level intensities, i.e., $\forall i \ p_i = 1/256$. Hence, the entropy of this image is:
 - $\log_2 256 = 8$
- Fig. (b) shows the histogram of an image with two possible values (binary image). Its entropy is 0.92.

Run-Length Coding

- RLC is one of the simplest forms of data compression.
- The basic idea is that if the information source has the property that symbols tend to form continuous groups, then such symbol and the length of the group can be coded.
- Consider a screen containing plain black text on a solid white background.
- There will be many long runs of white pixels in the blank space, and many short runs of black pixels within the text. Let us take a hypothetical single scan line, with B representing a black pixel and W representing white:

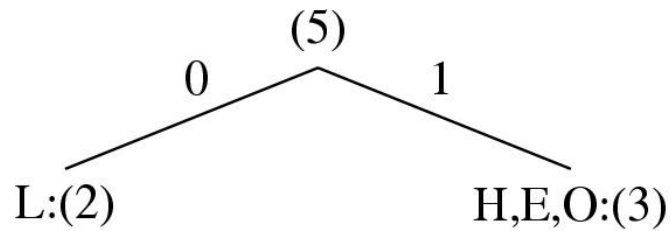
Shannon–Fano Algorithm

- To illustrate the algorithm, let us suppose the symbols to be coded are the characters in the word HELLO.
- The frequency count of the symbols is Symbol H E L O
Count 1 1 2 1
- The encoding steps of the Shannon–Fano algorithm can be presented in the following *top-down manner*:
- 1. Sort the symbols according to the frequency count of their occurrences.
- 2. Recursively divide the symbols into two parts, each with approximately the same number of counts, until all parts contain only one symbol.

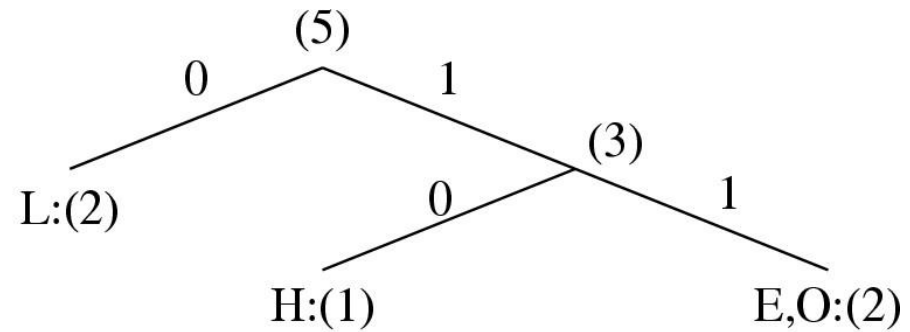
Shannon–Fano Algorithm

- A natural way of implementing the above procedure is to build a binary tree.
- As a convention, let us assign bit 0 to its left branches and 1 to the right branches.
- Initially, the symbols are sorted as LHEO.
- As Fig. shows, the first division yields two parts: L with a count of 2, denoted as L:(2); and H, E and O with a total count of 3, denoted as H, E, O:(3).
- The second division yields H:(1) and E, O:(2).
- The last division is E:(1) and O:(1).

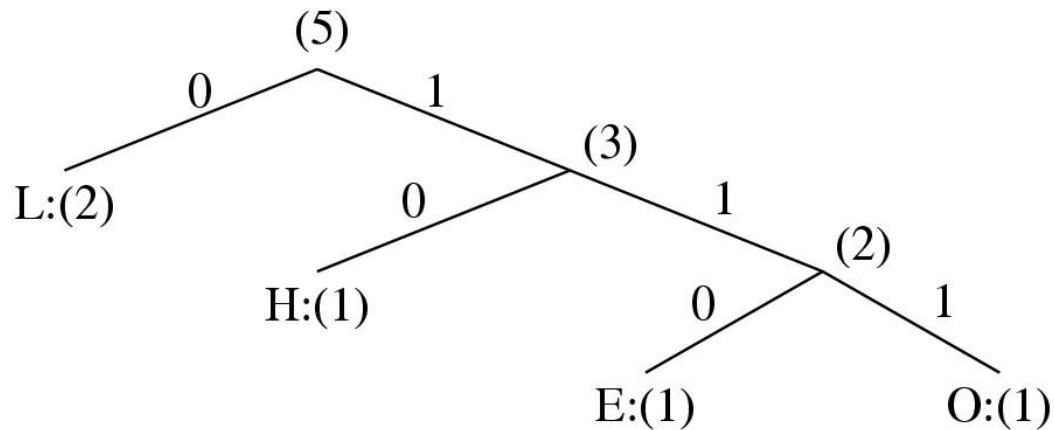
Shannon–Fano Algorithm



(a)



(b)

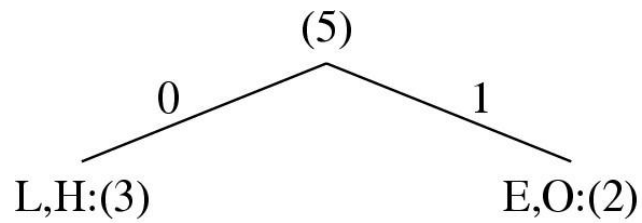


(c)

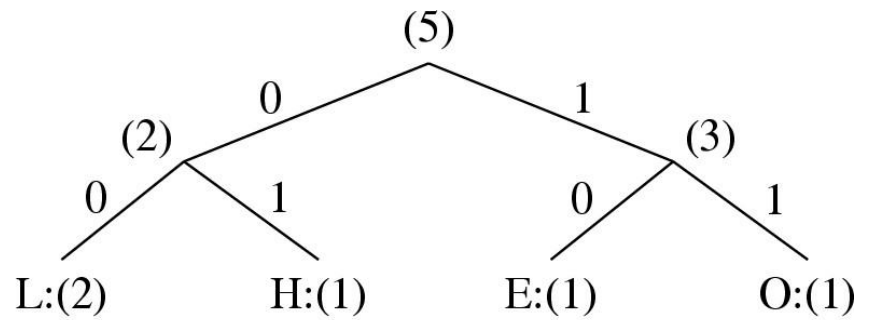
Fig.: Coding Tree for HELLO by Shannon-Fano.

Table : Result of Performing Shannon-Fano on HELLO

Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	# of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
TOTAL # of bits:				10



(a)



(b)

Fig. Another coding tree for HELLO by Shannon- Fano.

Table : Another Result of Performing Shannon-Fano

- on HELLO (see Fig.)**

Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	# of bits used
L	2	1.32	00	4
H	1	2.32	01	2
E	1	2.32	10	2
O	1	2.32	11	2
TOTAL # of bits:				10

Shannon–Fano Algorithm

- The Shannon–Fano algorithm delivers satisfactory coding results for data compression, but it was soon outperformed and overtaken by the Huffman coding method.
- The Huffman algorithm requires prior statistical knowledge about the information source, and such information is often not available.
- This is particularly true in multimedia applications, where future data is unknown before its arrival, as for example in live (or streaming) audio and video.
- Even when the statistics are available, the transmission of the symbol table could represent heavy overhead
- The solution is to use *adaptive* Huffman coding *compression algorithms, in which statistics are* gathered and updated dynamically as the data stream arrives.

Dictionary-Based Coding

- Unlike variable-length coding, in which the lengths of the code words are different, LZW uses fixed-length code words to represent variable length strings of symbols/characters that commonly occur together, such as words in English text.
- As in the other adaptive compression techniques, the LZW encoder and decoder builds up the same dictionary dynamically while receiving the data—the encoder and the decoder both develop the same dictionary.

Dictionary-Based Coding

- LZW proceeds by placing longer and longer repeated entries into a dictionary, then emitting (sending) the *code for an element rather than the string itself, if the element has* already been placed in the dictionary.
- Remember, the LZW is an adaptive algorithm, in which the encoder and decoder independently build their own string tables. Hence, there is no overhead involving transmitting the string table.
- LZW is used in many applications, such as UNIX compress, GIF for images, WinZip, and others.

compression

- *compression ratio for image data using lossless compression* techniques (e.g., Huffman Coding, Arithmetic Coding, LZW) is low when the image histogram is relatively flat.
- For image compression in multimedia applications, where a higher compression ratio is required, lossy methods are usually adopted.
- In lossy compression, the compressed image is usually not the same as the original image but is meant to form a close approximation to the original image *perceptually*

Distortion Measures

- To quantitatively describe how close the approximation is to the original data, some form of distortion measure is required.
- A *distortion measure* is a mathematical quantity that specifies how close an approximation is to its original, using some distortion criteria.
- When looking at compressed data, it is natural to think of the distortion in terms of the numerical difference between the original data and the reconstructed data.

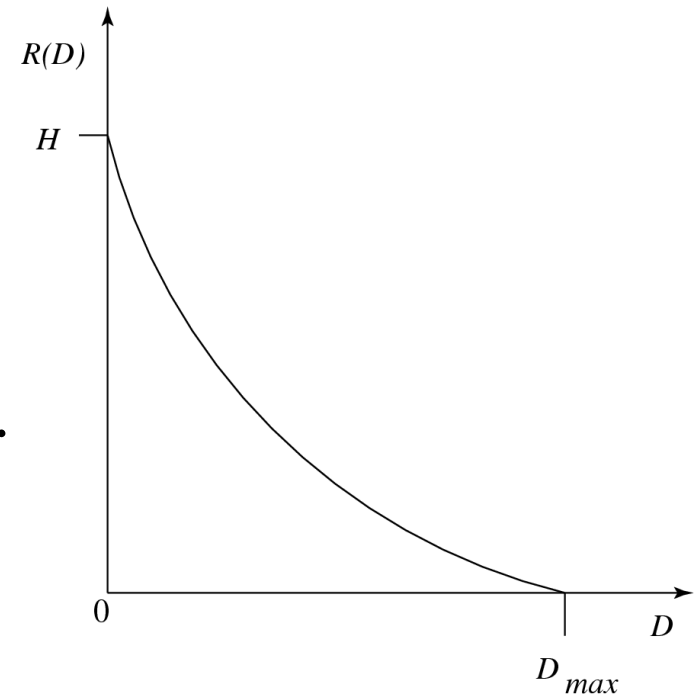
- This section introduces some basic video compression techniques and illustrates them in standards H.261 and H.263—two video compression standards aimed mostly at videoconferencing.
- The next two chapters further introduce several MPEG video compression standards and the latest, H.264 and H.265.

Lossy Compression

- Lossless compression algorithms do not deliver *compression ratios* that are high enough. Hence, most multimedia compression algorithms are *lossy*.
- What is *lossy compression*?
 - The compressed data is not the same as the original data, but a close approximation of it.
 - Yields a much higher compression ratio than that of lossless compression.

The Rate-Distortion Theory

- Provides a framework for
- the study of tradeoffs between
- Rate and Distortion.
- Rate: Average number of bits
- required to represent each symbol.



- Fig. : Typical Rate Distortion Function.

Quantization

- Reduce the number of distinct output values to a much smaller set. It is the main source of the “loss” in lossy compression.
- Three different forms of quantization:
 - Uniform Quantization.
 - Non uniform Quantization.
 - Vector Quantization.

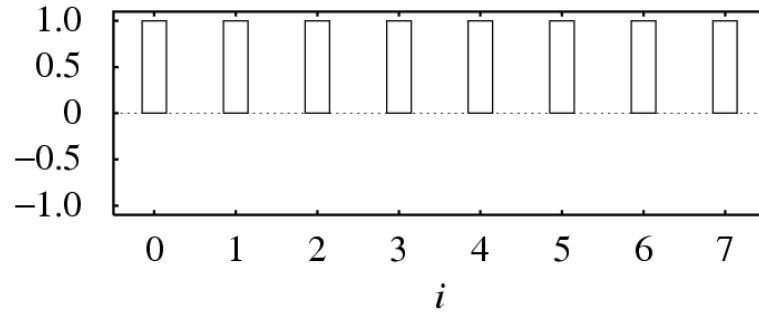
Transform Coding: DCT

- If \mathbf{Y} is the result of a linear transform \mathbf{T} of the input vector \mathbf{X} in such a way that the components of \mathbf{Y} are much less correlated, then \mathbf{Y} can be coded more efficiently than \mathbf{X} .
- If most information is accurately described by the first few components of a transformed vector, then the remaining components can be coarsely quantized, or even set to zero, with little signal distortion.

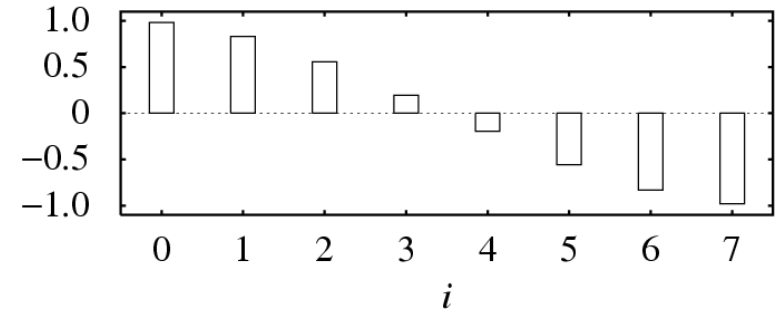
Spatial Frequency and DCT

- *Spatial frequency* indicates how many times pixel values change across an image block.
- The DCT formalizes this notion with a measure of how much the image contents change in correspondence to the number of cycles of a cosine wave per block.
- The role of the DCT is to *decompose* the original signal into its DC and AC components; the role of the IDCT is to *reconstruct* (re-compose) the signal.

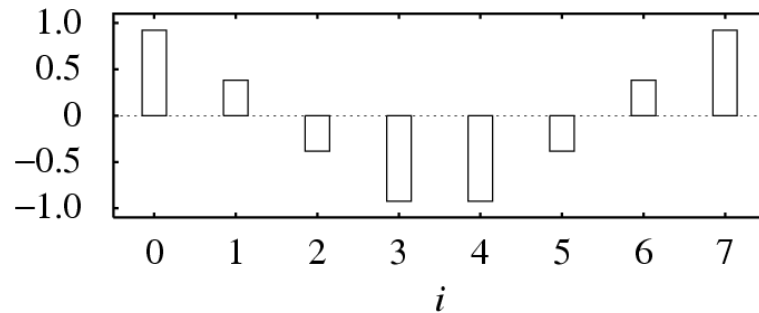
The 0th basis function ($u = 0$)



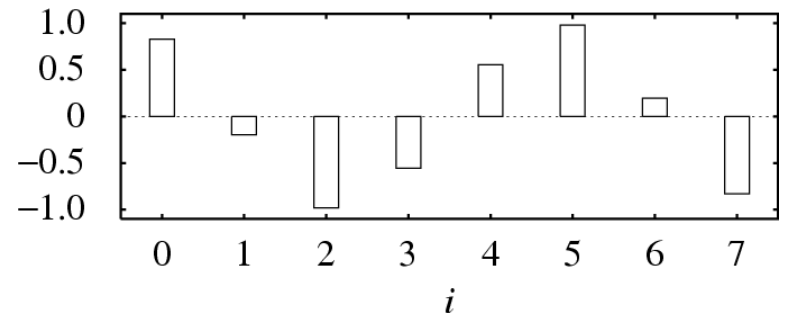
The 1st basis function ($u = 1$)



The 2nd basis function ($u = 2$)

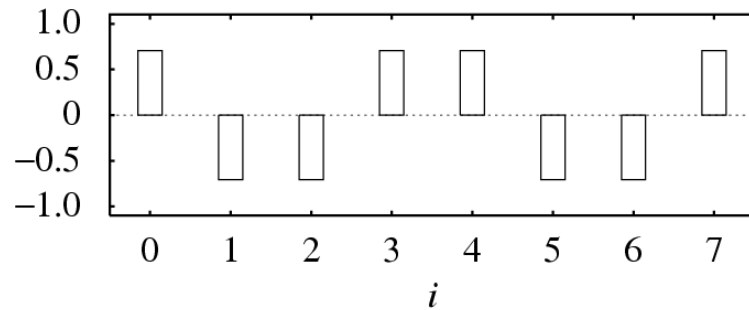


The 3rd basis function ($u = 3$)

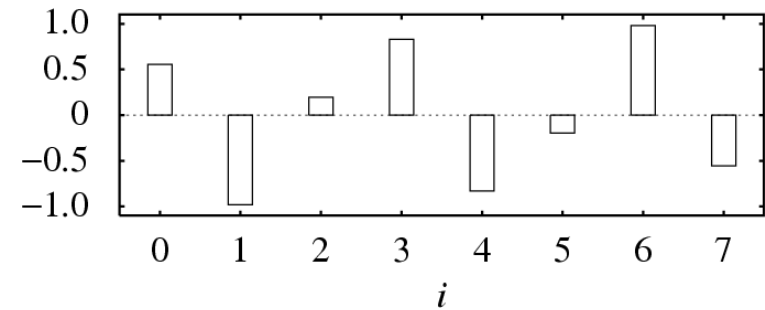


- Fig. : The 1D DCT basis functions.

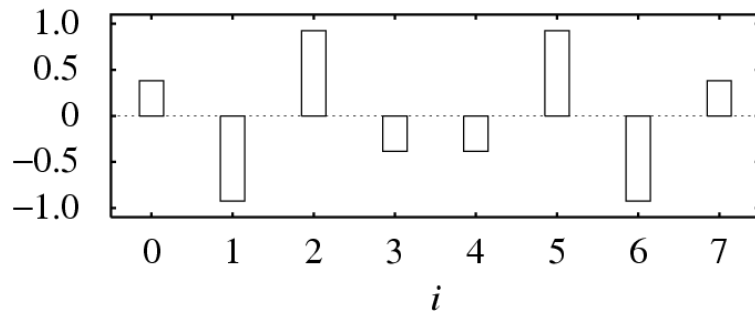
The 4th basis function ($u = 4$)



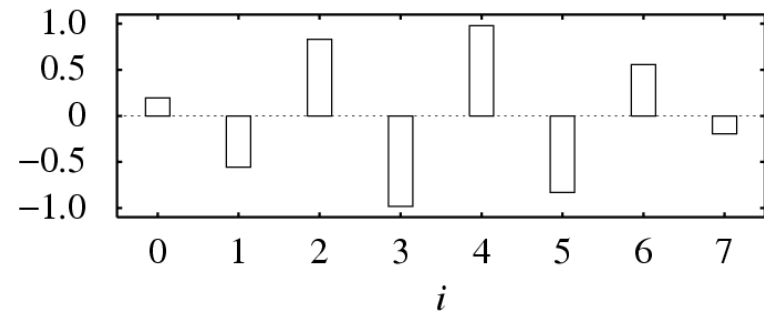
The 5th basis function ($u = 5$)



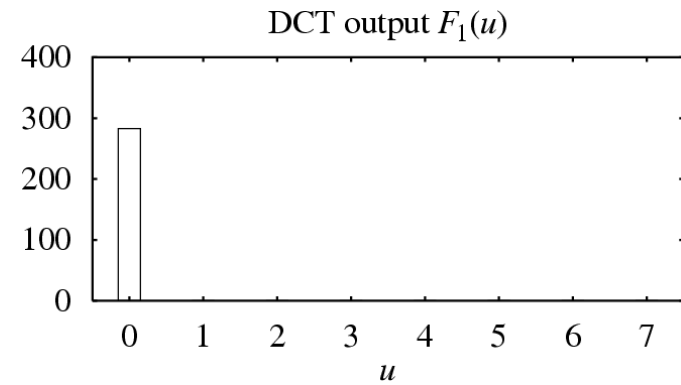
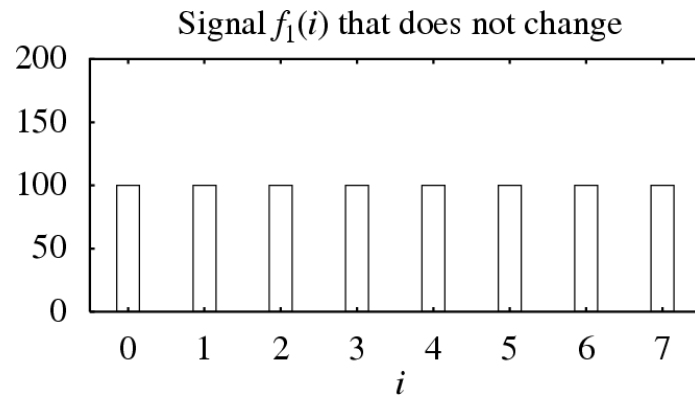
The 6th basis function ($u = 6$)



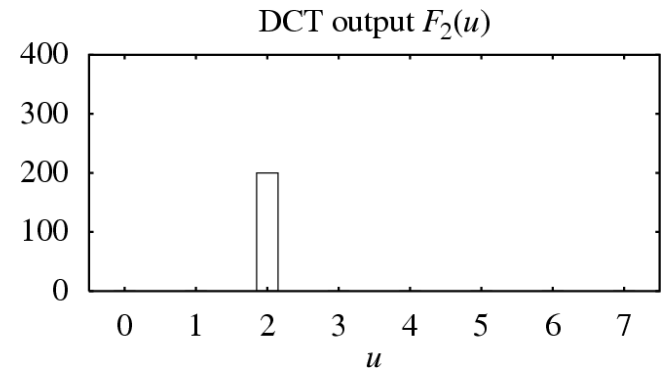
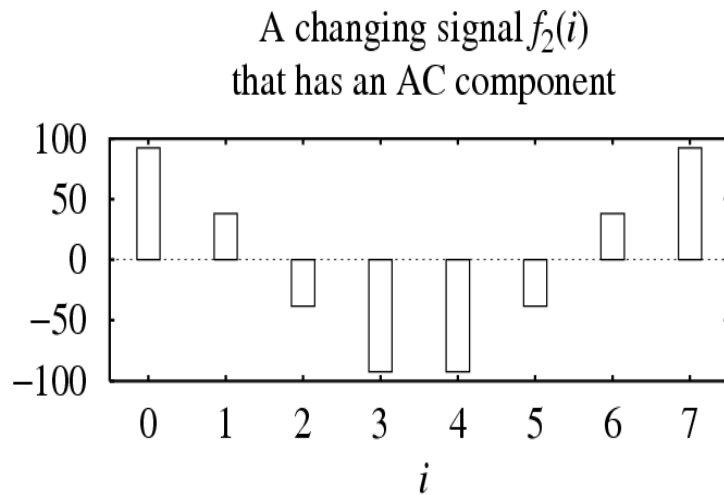
The 7th basis function ($u = 7$)



- Fig. (cont'd): The 1D DCT basis functions.

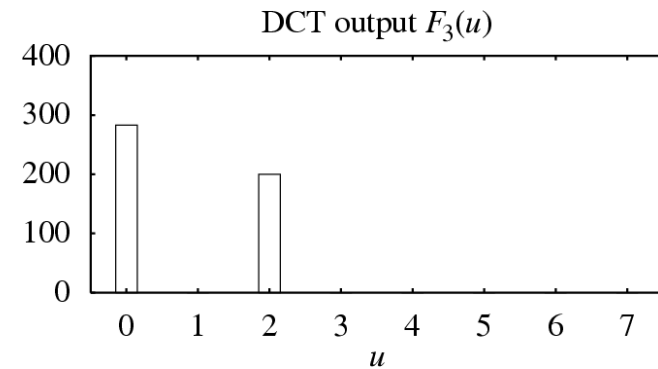
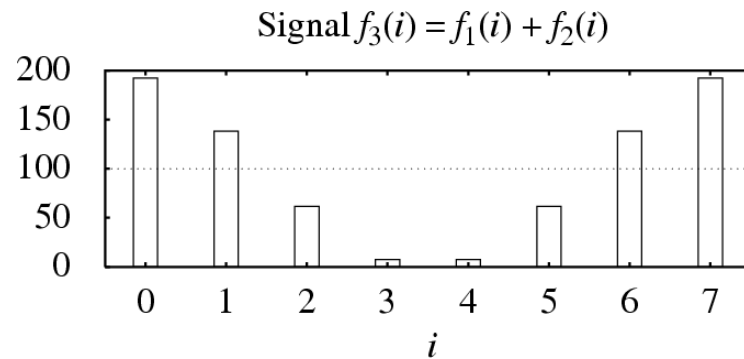


(a)

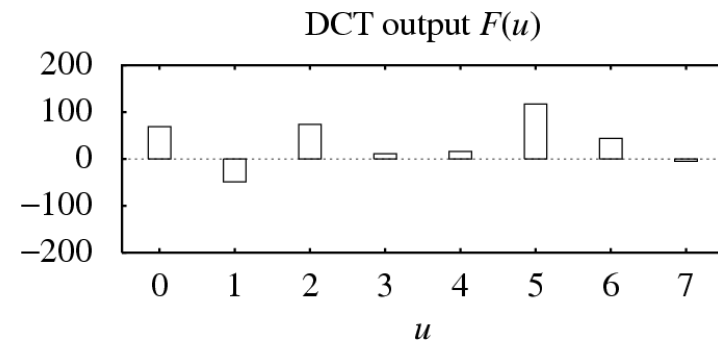
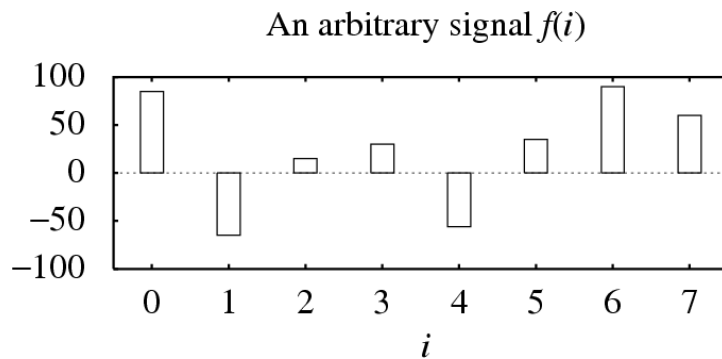


(b)

•Fig. : Examples of 1D Discrete Cosine Transform: (a) A DC signal $f_1(i)$, (b) An AC signal $f_2(i)$.



(c)



(d)

•Fig. (cont'd): Examples of 1D Discrete Cosine Transform: (c) $f_3(i) = f_1(i) + f_2(i)$, and (d) an arbitrary signal $f(i)$.

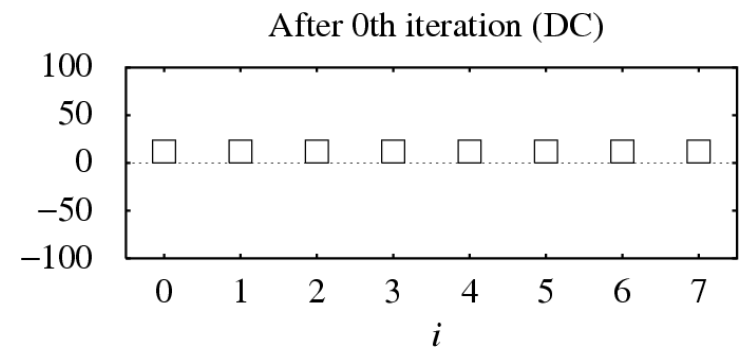
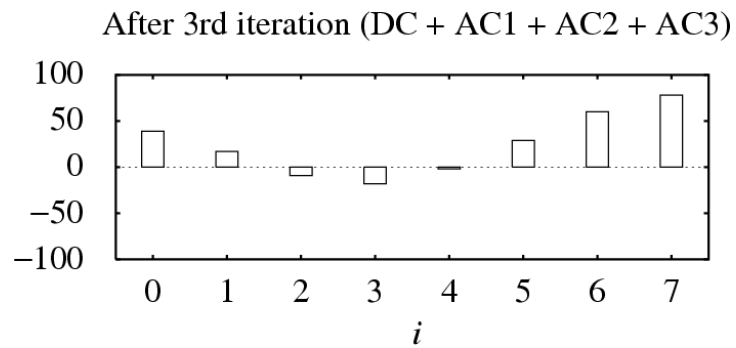
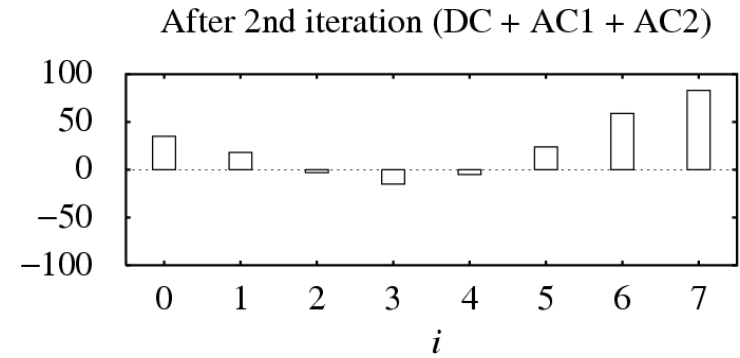
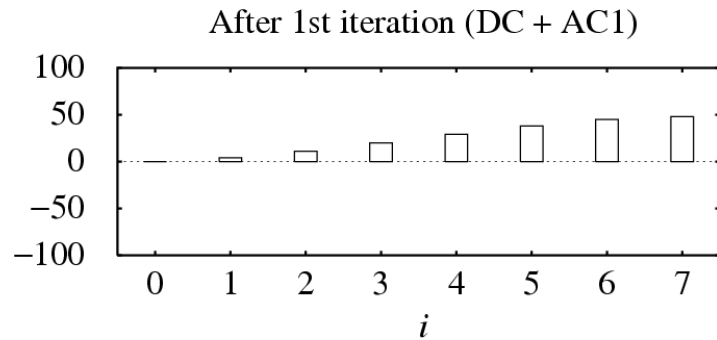
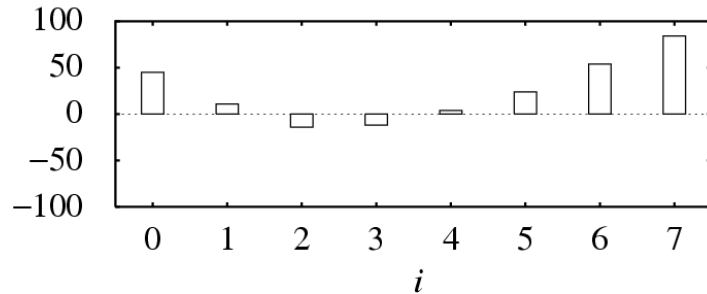
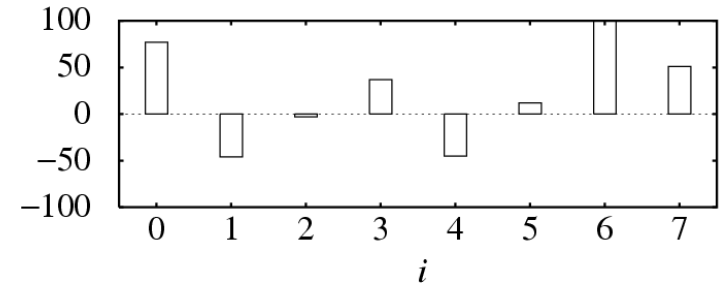


Fig.: An example of 1D IDCT.

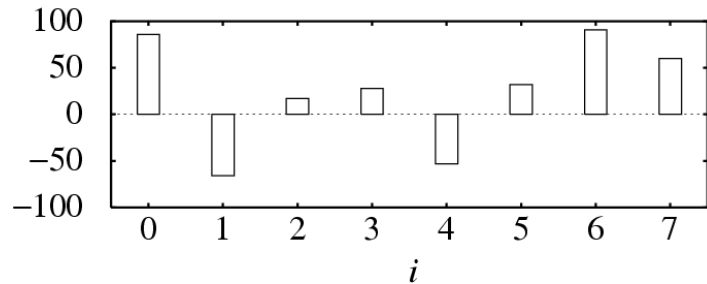
After 4th iteration (DC + AC1 + ... + AC4)



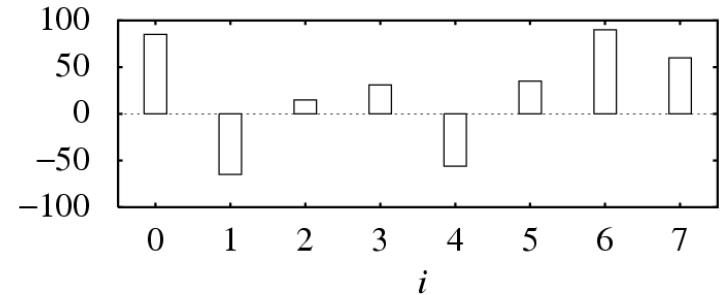
After 5th iteration (DC + AC1 + ... + AC5)



After 6th iteration (DC + AC1 + ... + AC6)



After 7th iteration (DC + AC1 + ... + AC7)



- Fig.:(cont'd): An example of 1D IDCT.

Introduction to Video Compression

- A video consists of a time-ordered sequence of frames, i.e., images.
- An obvious solution to video compression would be predictive coding based on previous frames.

Compression proceeds by subtracting images: subtract in time order and code the residual error.

- It can be done even better by searching for just the right parts of the image to subtract from the previous frame.

Video Compression with Motion Compensation

- Consecutive frames in a video are similar — temporal redundancy exists.
- Temporal redundancy is exploited so that not every frame of the video needs to be coded independently as a new image.
- The difference between the current frame and other frame(s) in the sequence will be coded — small values and low entropy, good for compression.
- Steps of Video compression based on *Motion Compensation (MC)*:
 1. Motion Estimation (motion vector search).
 2. MC-based Prediction.
 3. Derivation of the prediction error, i.e., the difference.

Motion Compensation

- Each image is divided into *macro blocks* of size $N \times N$.
 - By default, $N = 16$ for luminance images. For chrominance images,
 $N = 8$ if 4:2:0 chroma sub sampling is adopted.
- Motion compensation is performed at the macro block level.
 - The current image frame is referred to as *Target Frame*.
 - A match is sought between the macro block in the Target Frame and the most similar macro block in previous and/or future frame(s) (referred to as *Reference frame(s)*).
 - The displacement of the reference macro block to the target macro block is called a *motion vector* MV.
 - Figure 10.1 shows the case of *forward prediction* in which the Reference frame is taken to be a previous frame.

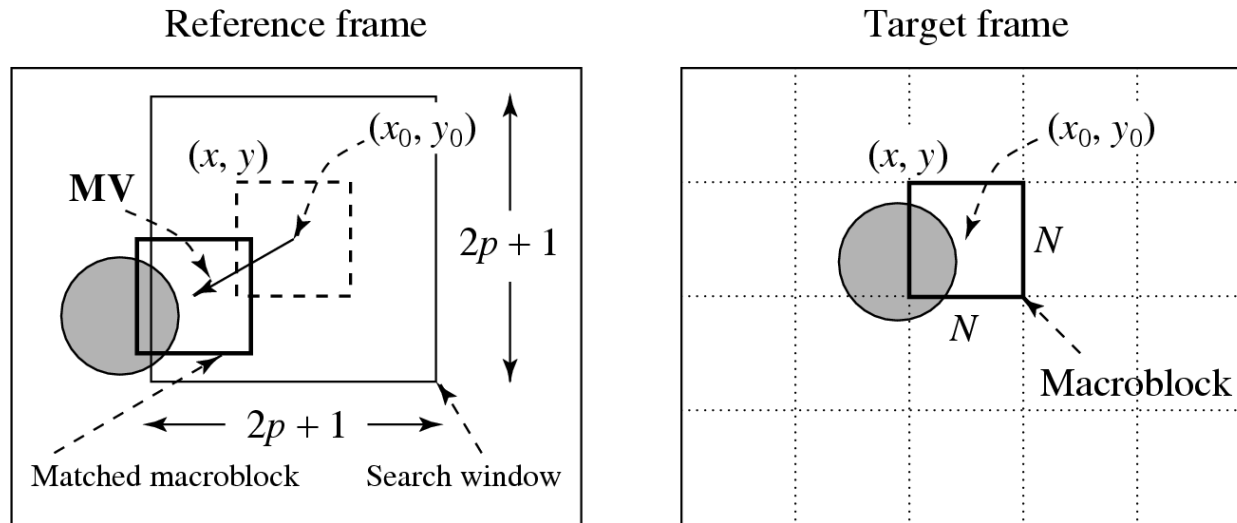


Fig. : Macroblocks and Motion Vector in Video Compression.

H.261

- **H.261:** An earlier digital video compression standard, its principle of MC-based compression is retained in all later video compression standards.
 - - The standard was designed for videophone, video conferencing and other audiovisual services over ISDN.
 - - The video codec supports bit-rates of $p \times 64$ kbps, where p ranges from 1 to 30 (Hence also known as $p * 64$).
 - - Require that the delay of the video encoder be less than 150 msec so that the video can be used for real-time bidirectional video conferencing.

Table :Video Formats Supported by H.261

Video format	Luminance image resolution	Chrominance image resolution	Bit-rate (Mbps) (if 30 fps and uncompressed)	H.261 support
QCIF	176×144	88×72	9.1	required
CIF	352×288	176×144	36.5	optional

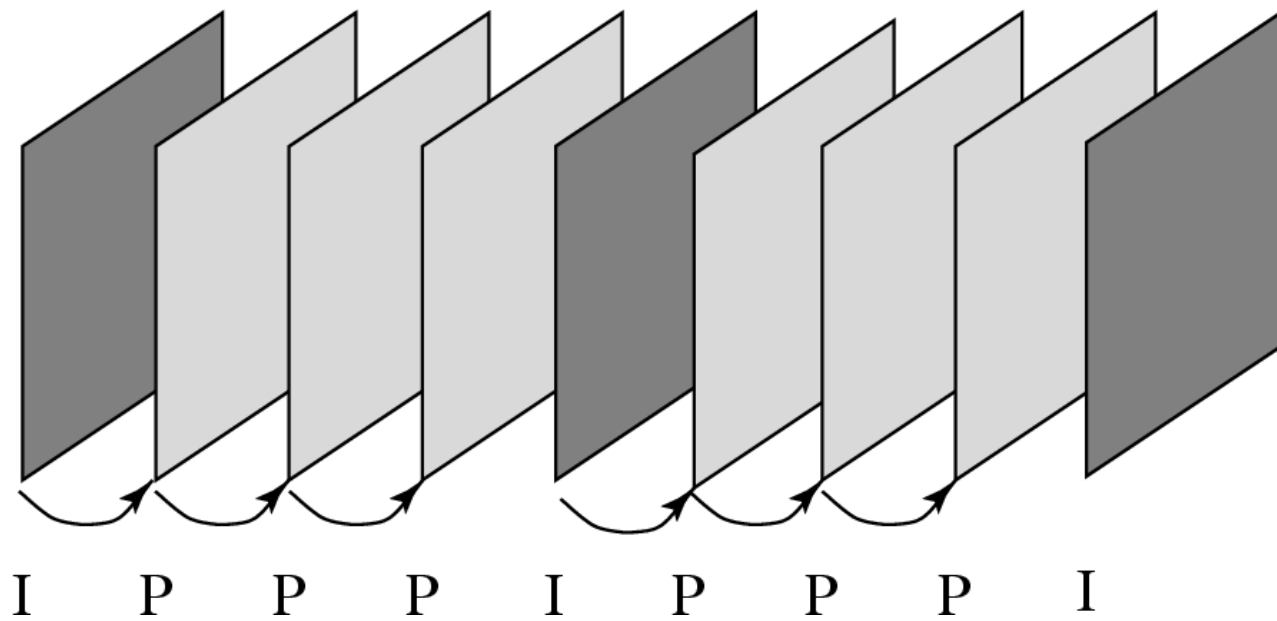
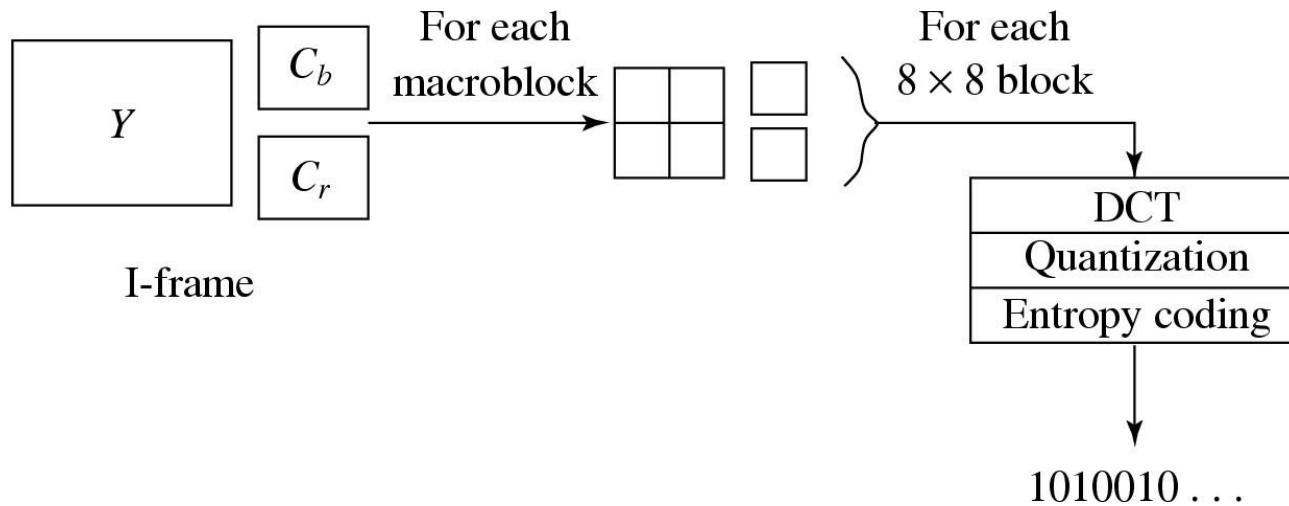


Fig. : H.261 Frame Sequence.

H.261 Frame Sequence

- Two types of image frames are defined: Intra-frames (I-frames) and Inter-frames (P-frames):
 - - I-frames are treated as independent images. Transform coding method similar to JPEG is applied within each I-frame, hence “Intra”.
 - - P-frames are not independent: coded by a forward predictive coding method (prediction from a previous P-frame is allowed — not just from a previous I-frame).
 - - Temporal redundancy removal is included in P-frame coding, whereas I-frame coding performs only spatial redundancy removal.
- To avoid propagation of coding errors, an I-frame is usually sent a couple of times in each second of the video.
- Motion vectors in H.261 are always measured in units of full pixel and they have a limited range of ± 15 pixels, i.e., $p = 15$.

Intra-frame (I-frame) Coding



Inter-frame (P-frame) Predictive Coding

- Figure shows the H.261 P-frame coding scheme based on motion compensation:
 - For each macro block in the Target frame, a motion vector is allocated by one of the search methods discussed earlier.
 - After the prediction, a *difference macro block* is derived to measure the *prediction error*.
 - Each of these 8 x 8 blocks go through DCT, quantization, zigzag scan and entropy coding procedures.

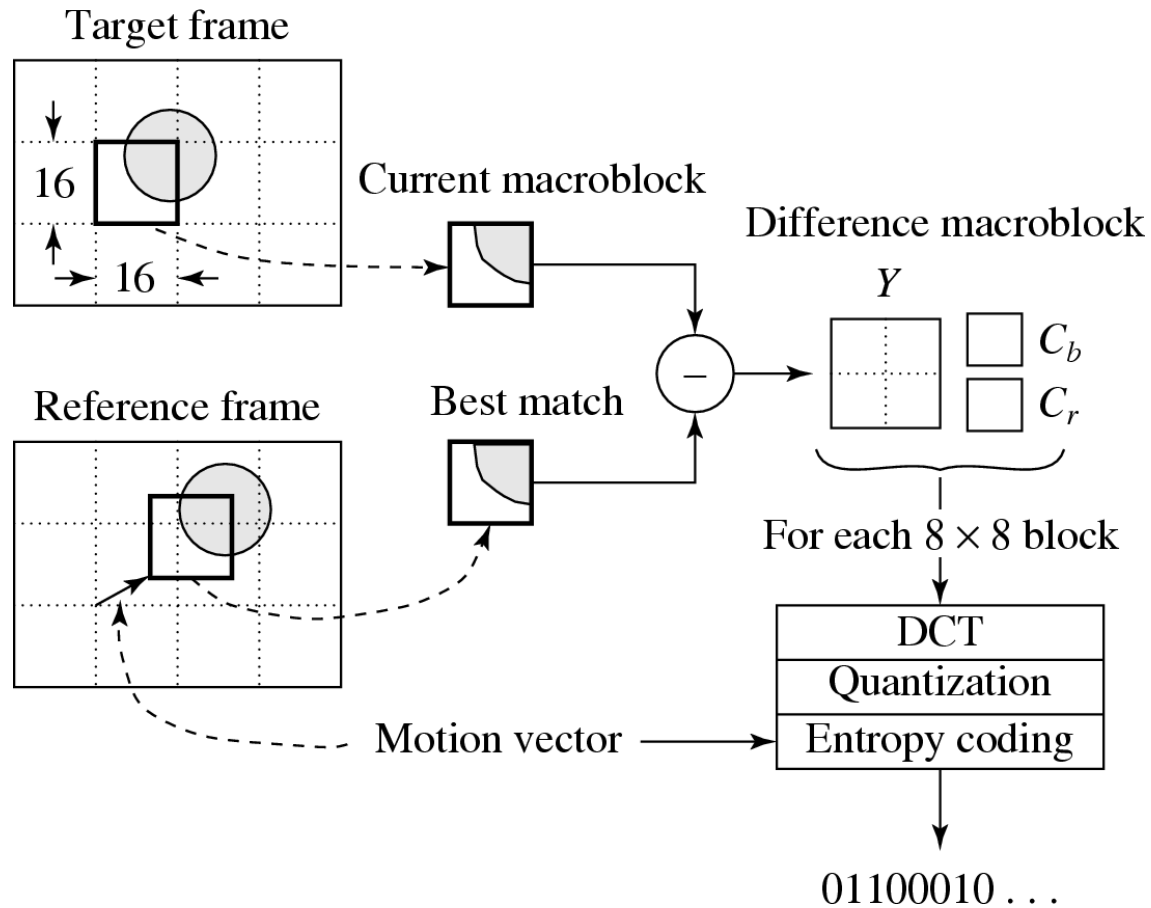
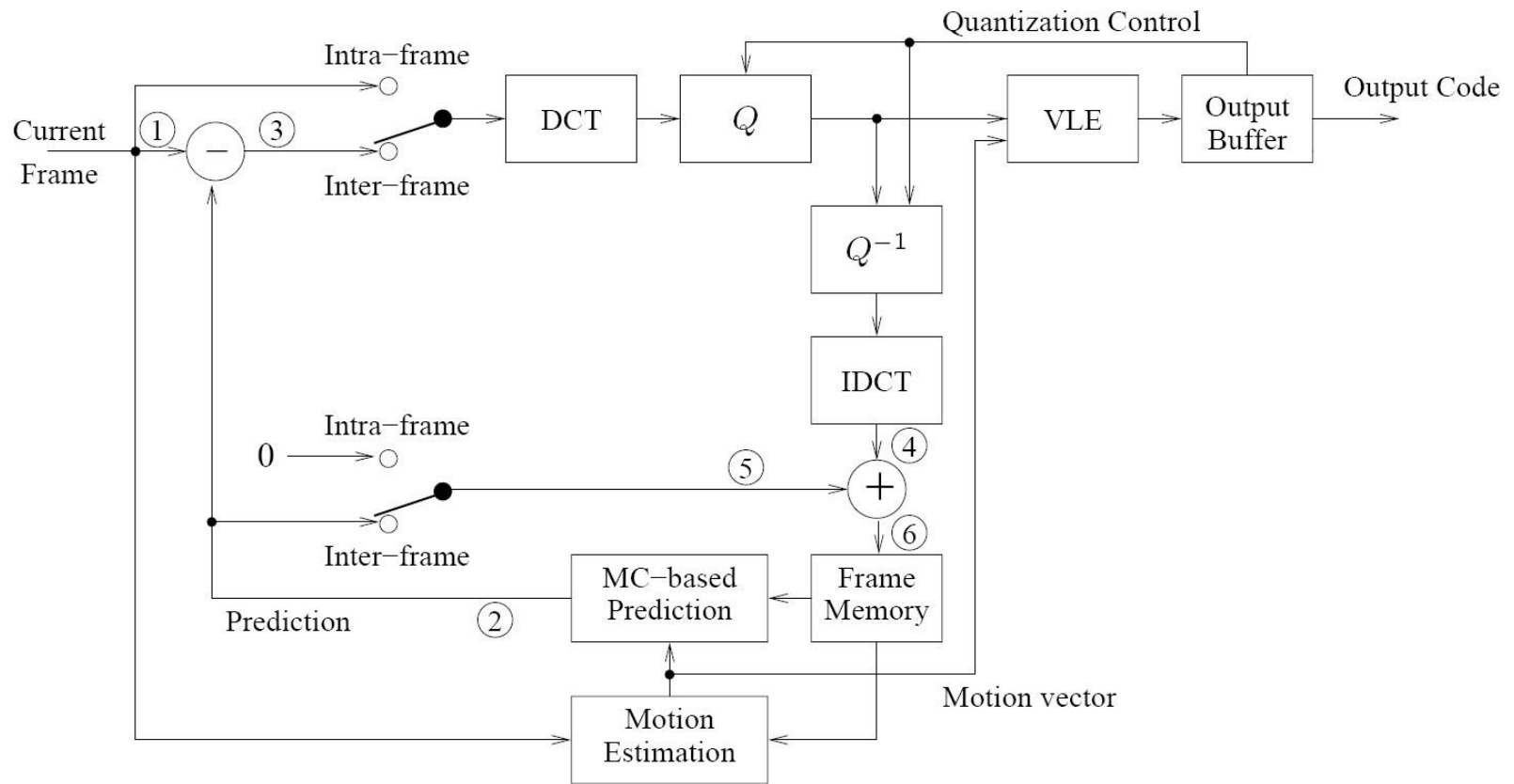
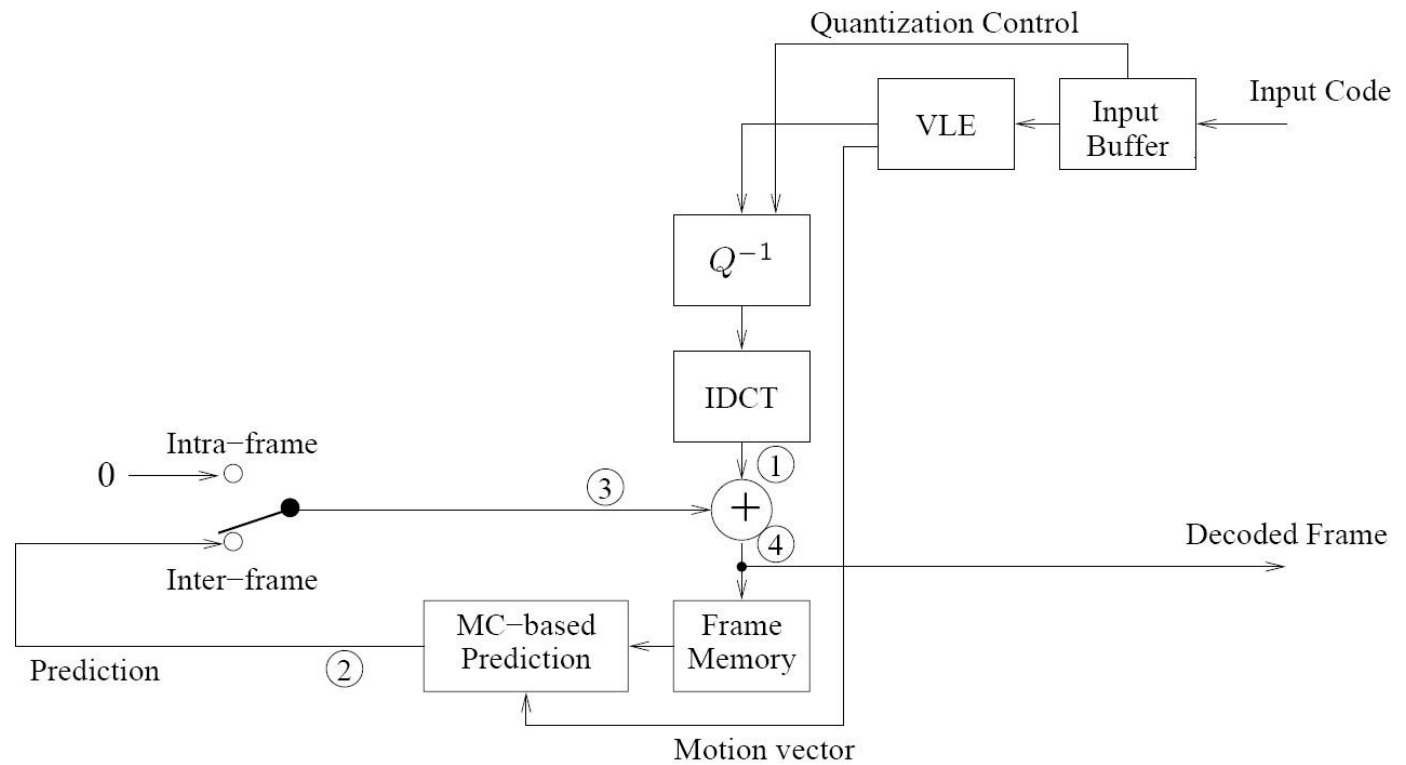


Fig. : H.261 P-frame Coding Based on Motion Compensation.



(a) Encoder

Fig. : H.261 Encoder and Decoder.



(b) Decoder

Fig. (Cont'd): H.261 Encoder and Decoder.

MPEG

- **MPEG:** *Moving Pictures Experts Group*, established in 1988 for the development of digital video.
- It is appropriately recognized that proprietary interests need to be maintained within the family of MPEG standards:
 - Accomplished by defining only a compressed bit stream that implicitly defines the decoder.
 - The compression algorithms, and thus the encoders, are completely up to the manufacturers.

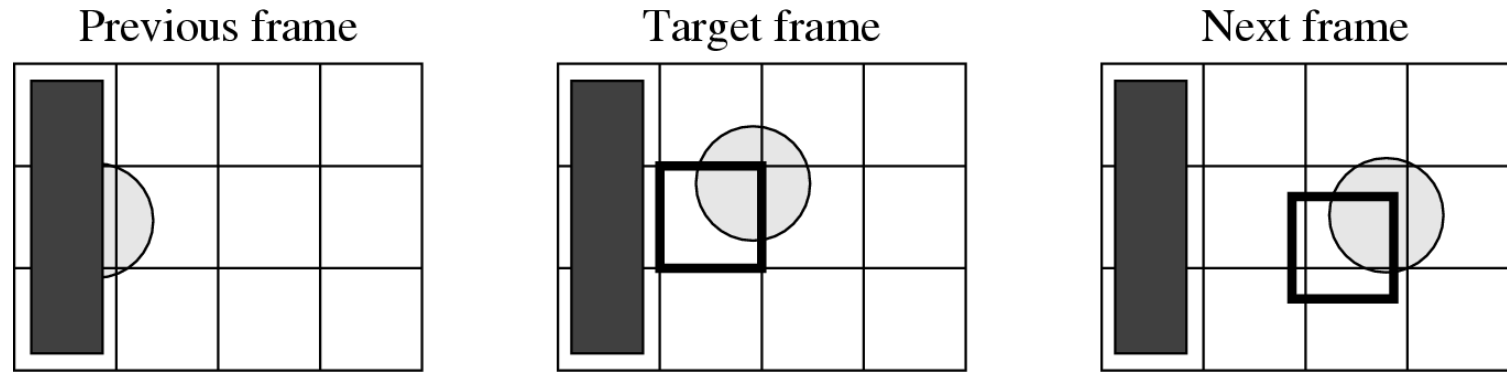


Fig : The Need for Bidirectional Search.

The MB containing part of a ball in the Target frame cannot find a good matching MB in the previous frame because half of the ball was occluded by another object. A match however can readily be obtained from the next frame.

Motion Compensation in MPEG-1 (Cont'd)

- MPEG introduces a third frame type — *B-frames*, and its accompanying bi-directional motion compensation.
- The MC-based B-frame coding idea is illustrated in Fig. :
 - Each MB from a B-frame will have up to *two* motion vectors (MVs) (one from the forward and one from the backward prediction).
 - If matching in both directions is successful, then two MVs will be sent and the two corresponding matching MBs are averaged (indicated by ‘%’ in the figure) before comparing to the Target MB for generating the prediction error.
 - If an acceptable match can be found in only one of the reference frames, then only one MV and its corresponding MB will be used from either the forward or backward prediction.

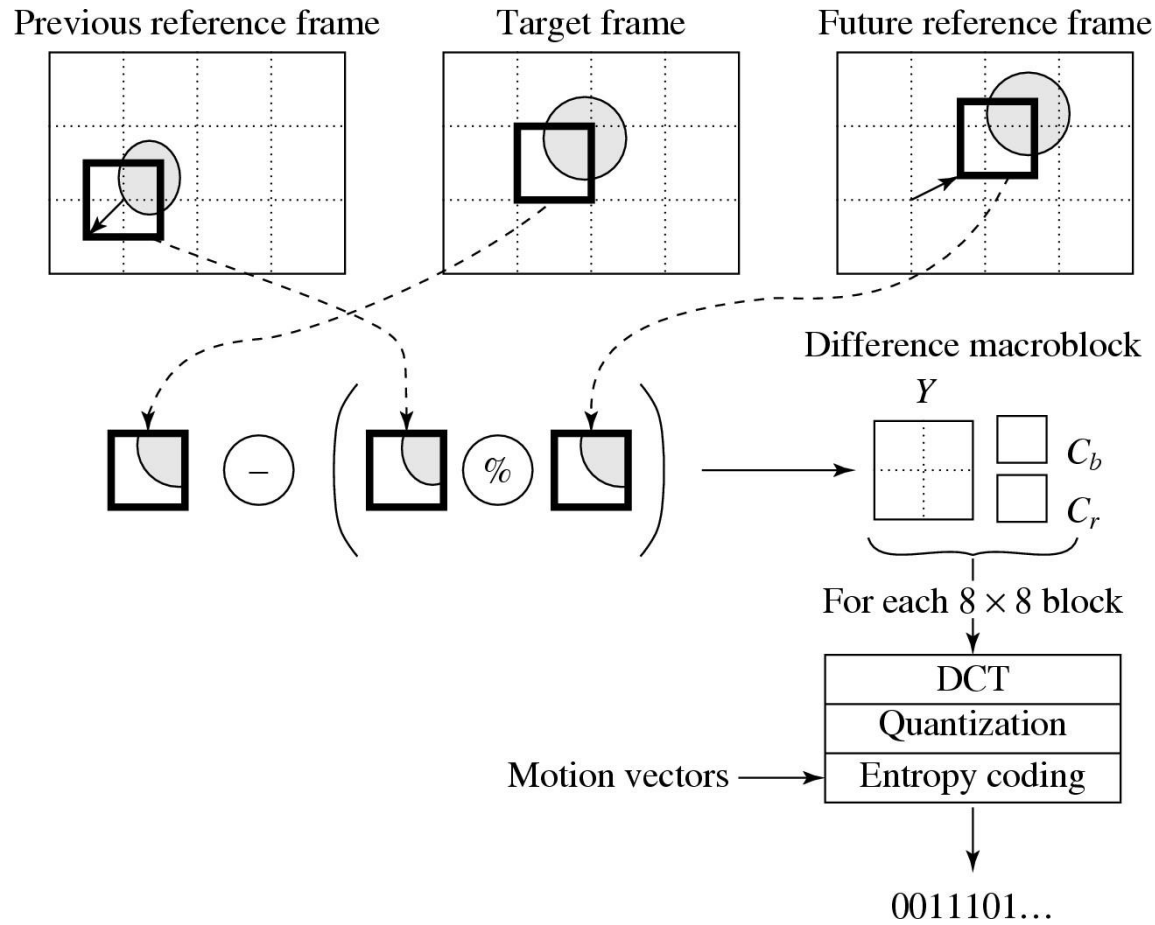


Fig : B-frame Coding Based on Bidirectional Motion Compensation.

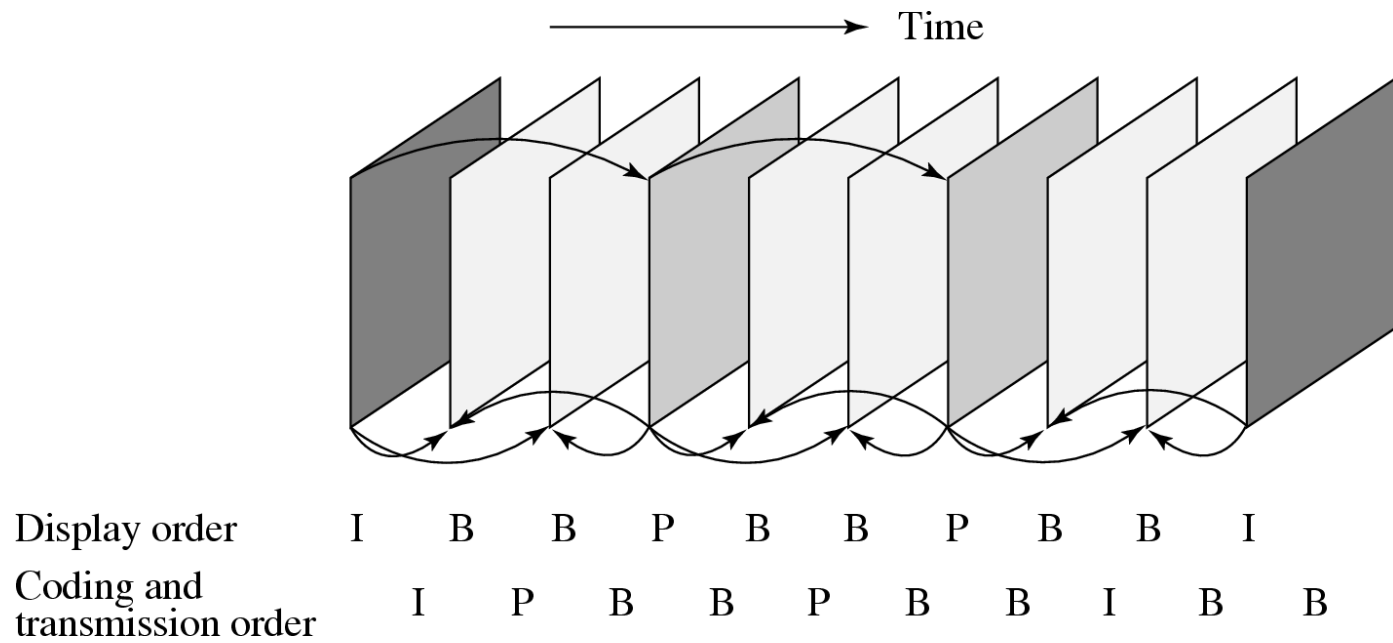


Fig : MPEG Frame Sequence.

Typical Sizes of MPEG-1 Frames

- The typical size of compressed P-frames is significantly smaller than that of I-frames — because temporal redundancy is exploited in inter-frame compression.
- B-frames are even smaller than P-frames — because of (a) the advantage of bi-directional prediction and (b) the lowest priority given to B-frames.

Table : Typical Compression Performance of MPEG-1

Type	Size	Compression
I	18kB	7:1
P	6kB	20:1
B	2.5kB	50:1
Avg	4.8kB	27:1

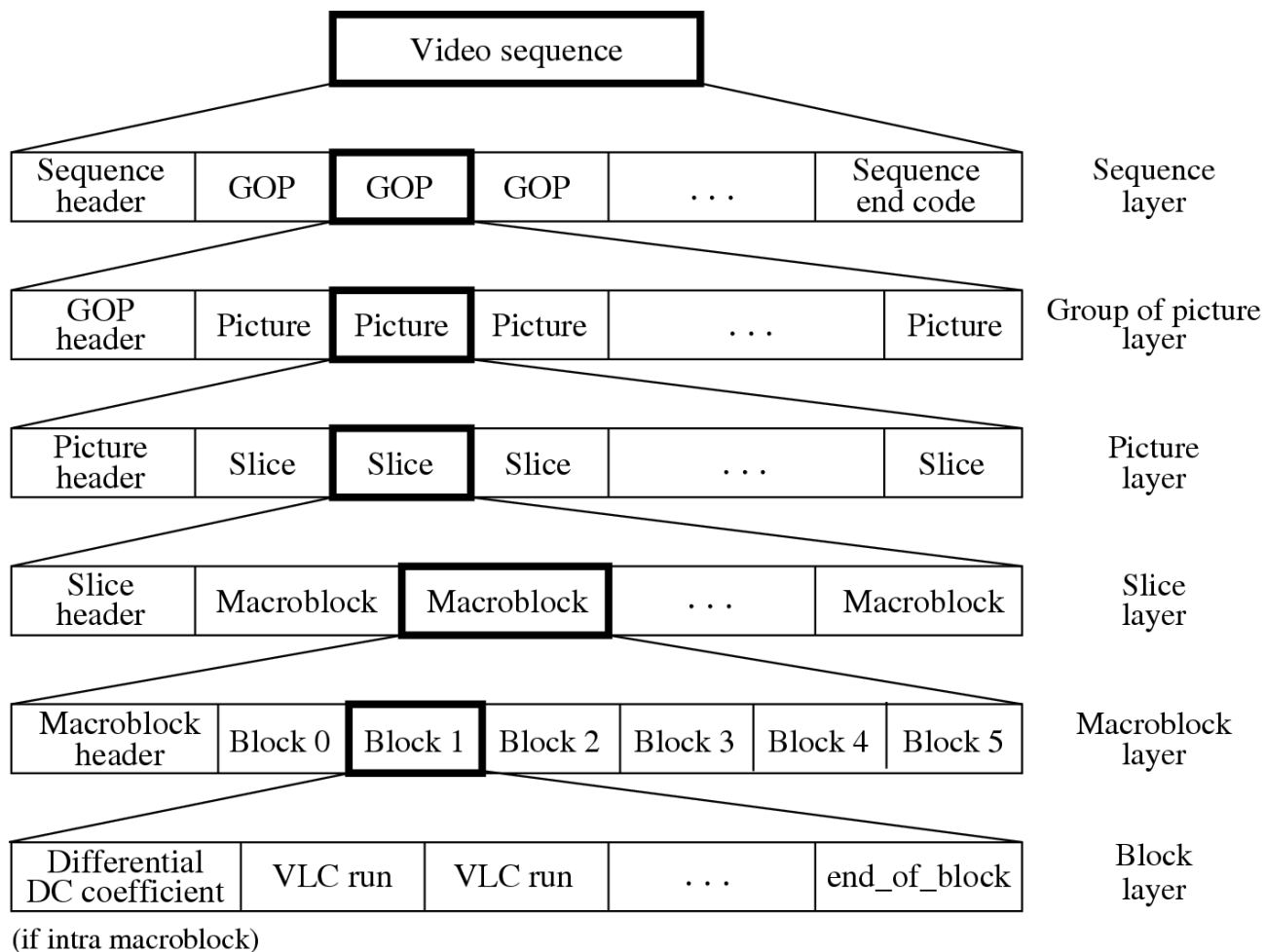


Fig : Layers of MPEG-1 Video Bitstream.

MPEG-4

- MPEG-4: a newer standard. Besides compression, pays great attention to issues about user interactivities.
- MPEG-4 departs from its predecessors in adopting a new object-based coding:
 - Offering higher compression ratio, also beneficial for digital video composition, manipulation, indexing, and retrieval.
 - Figure illustrates how MPEG-4 videos can be composed and manipulated by simple operations on the visual objects.

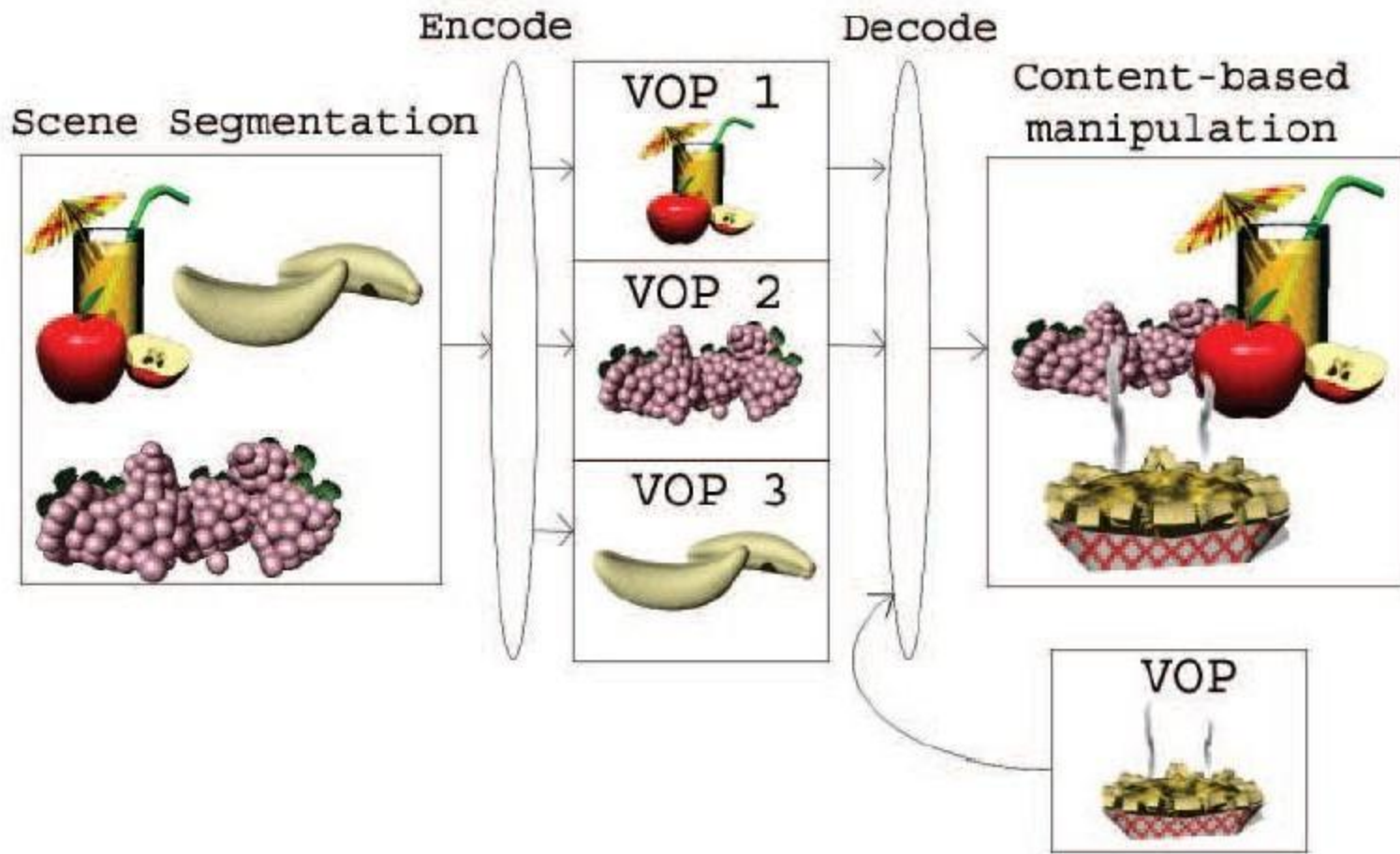
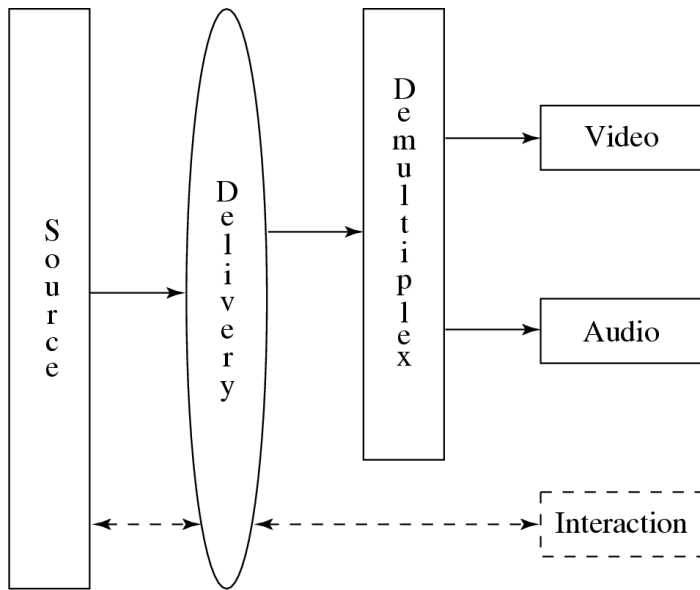


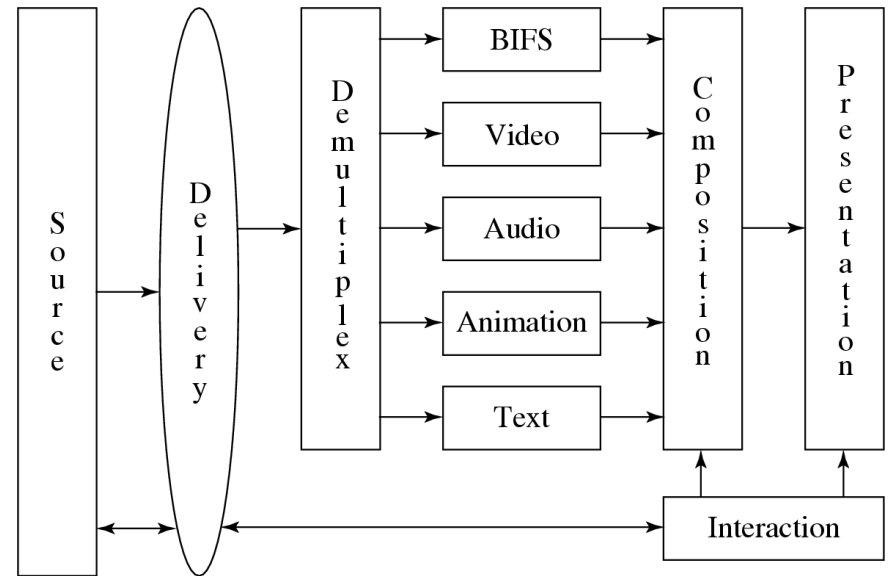
Fig. : Composition and Manipulation of MPEG-4 Videos.

MPEG-4 (Cont'd)

- MPEG-4 (Fig. (b)) is an entirely new standard for:
 - (a) Composing media objects to create desirable audiovisual scenes.
 - (b) Multiplexing and synchronizing the bit streams for these media data entities so that they can be transmitted with guaranteed Quality of Service (QoS).
 - (c) Interacting with the audiovisual scene at the receiving end — provides a toolbox of advanced coding modules and algorithms for audio and video compressions.



(a)



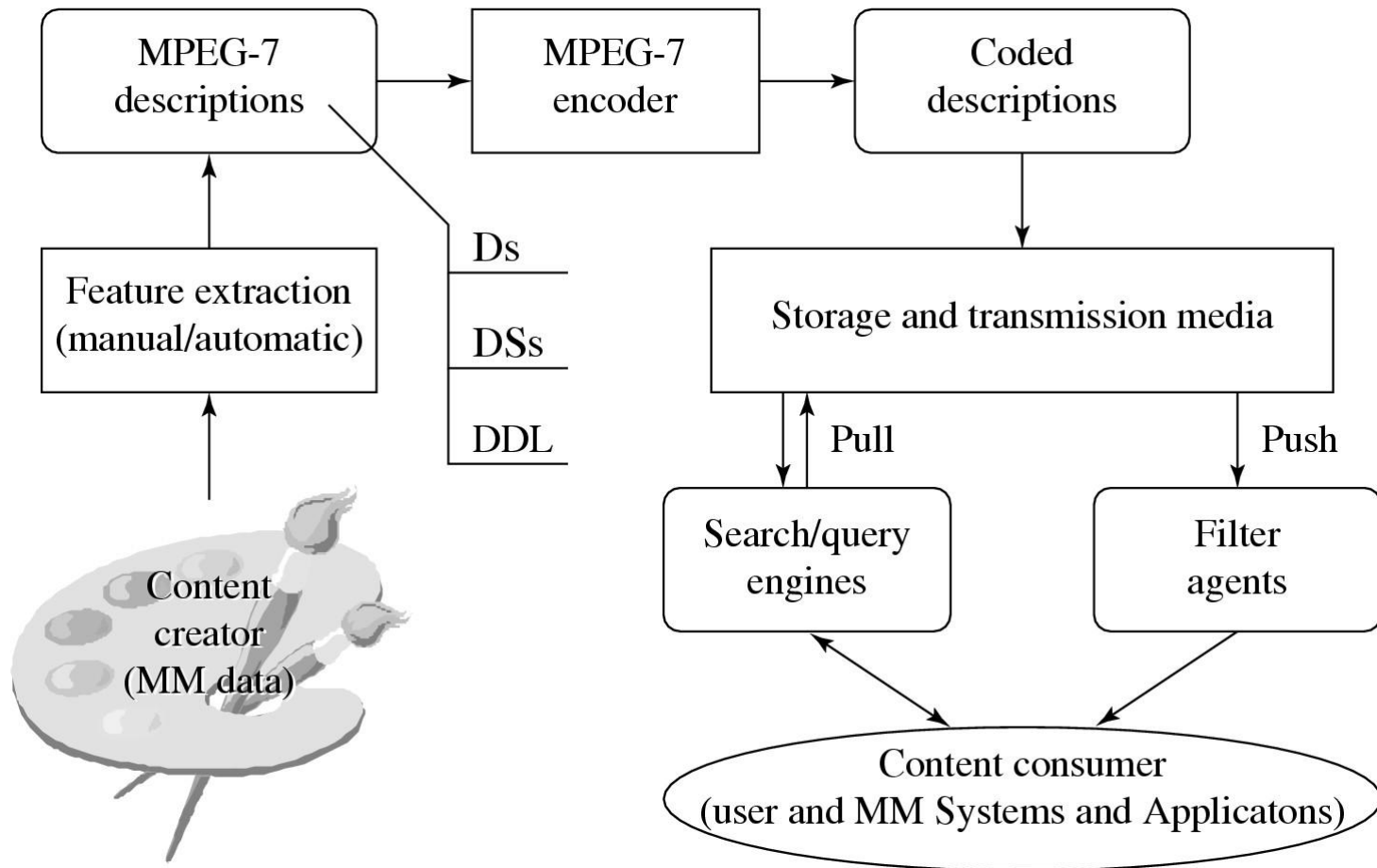
(b)

MPEG-7

- The main objective of MPEG-7 is to serve the need of audio-visual content-based retrieval (or audiovisual object retrieval) in applications such as digital libraries.
- Nevertheless, it is also applicable to any multimedia applications involving the generation (*content creation*) and usage (*content consumption*) of multimedia data.
- MPEG-7 became an International Standard in September 2001 with the formal name Multimedia Content Description Interface.

Applications Supported by MPEG-7

- MPEG-7 supports a variety of multimedia applications. Its data may include still pictures, graphics, 3D models, audio, speech, video, and composition information (how to combine these elements).
- These MPEG-7 data elements can be represented in textual format, or binary format, or both.
- Fig. illustrates some possible applications that will benefit from the MPEG-7 standard.



- Fig. :Possible Applications using MPEG-7.

MPEG-21

- The development of the newest standard, MPEG-21: Multimedia Framework, started in June 2000, and was expected to become International Standard by 2003.
- The *vision* for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities.
- The seven key elements in MPEG-21 are:
 - Digital item declaration — to establish a uniform and flexible abstraction and interoperable schema for declaring Digital items.
 - Digital item identification and description— to establish a framework for standardized identification and description of digital items regardless of their origin, type or granularity.

- **Content management and usage** — to provide an interface and protocol that facilitate the management and usage (searching, caching, archiving, distributing, etc.) of the content.
- **Intellectual property management and protection (IPMP)** — to enable contents to be reliably managed and protected.
- **Terminals and networks** — to provide interoperable and transparent access to content with Quality of Service (QoS) across a wide range of networks and terminals.
- **Content representation** — to represent content in an adequate way for pursuing the objective of MPEG-21, namely “content anytime anywhere”.
- **Event reporting** — to establish metrics and interfaces for reporting *events* (user interactions) so as to understand performance and alternatives.

UNIT - III

- MPEG-1 adopts the CCIR601 digital TV format also known as SIF (*Source Input Format*).
 - 352×240 for NTSC video at 30 fps
 - 352×288 for PAL video at 25 fps
 - It uses 4:2:0 chroma sub sampling
- The MPEG-1 standard is also referred to as ISO/IEC 11172. It has five parts: 11172-1 Systems, 11172-2 Video, 11172-3 Audio, 11172-4 Conformance, and 11172-5 Software.

MotionCompensatiin MPEG-1

- Motion Compensation (MC) based video encoding in H.261 works as follows:
 - In Motion Estimation (ME), each macro block (MB) of the Target P-frame is assigned a best matching MB from the previously coded I or P frame - prediction.
 - prediction error: The difference between the MB and its matching MB, sent to DCT and its subsequent encoding steps.
 - The prediction is from a previous frame — forward prediction.

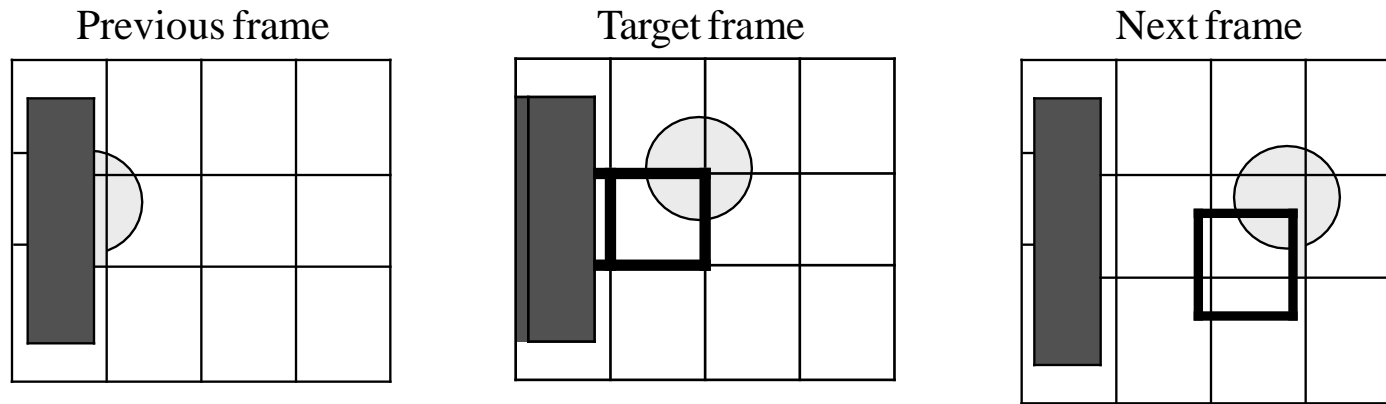


Fig : The Need for Bidirectional Search.

The MB containing part of a ball in the Target frame cannot find a good matching MB in the previous frame because half of the ball was occluded by another object. A match however can readily be obtained from the next frame.

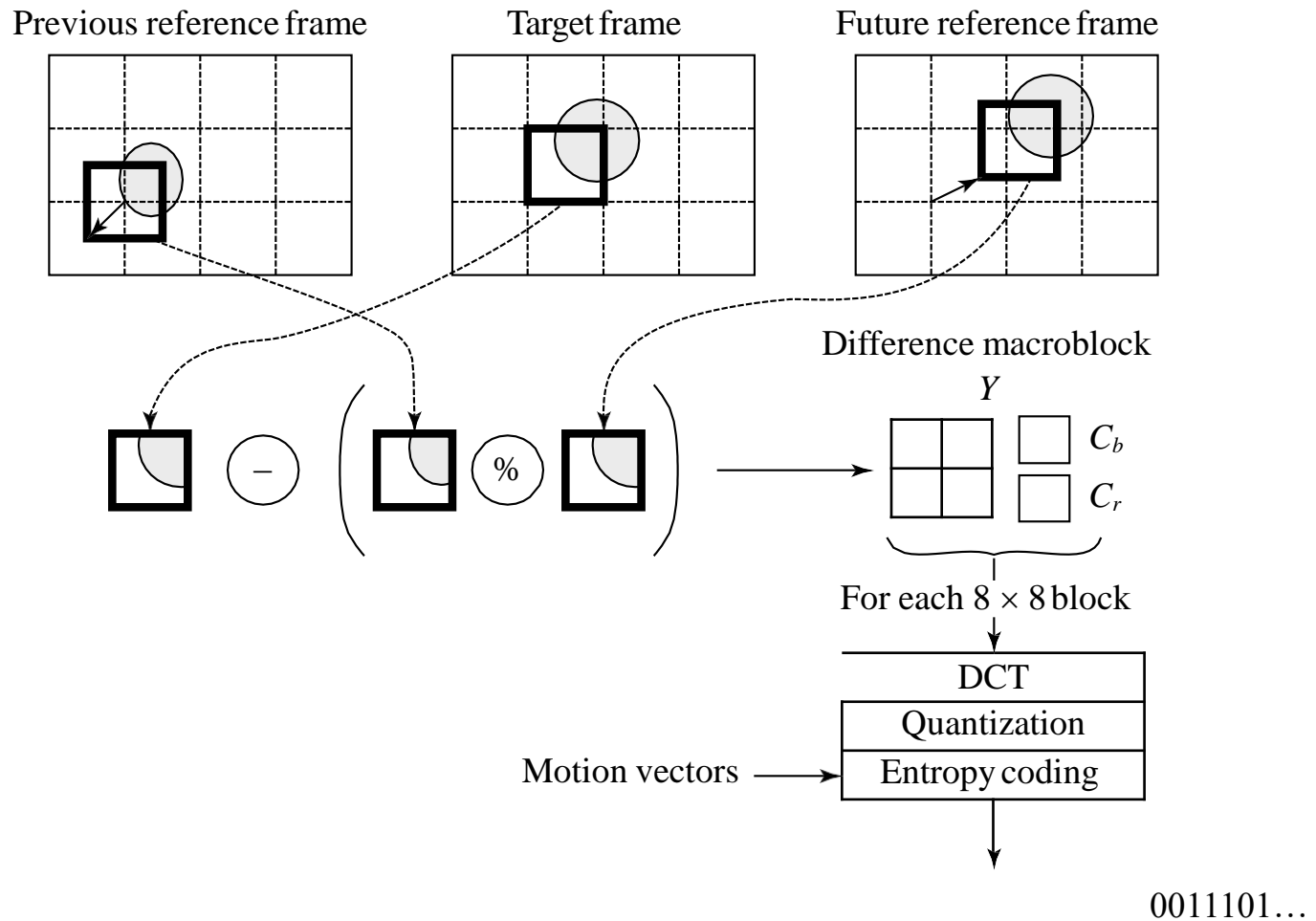
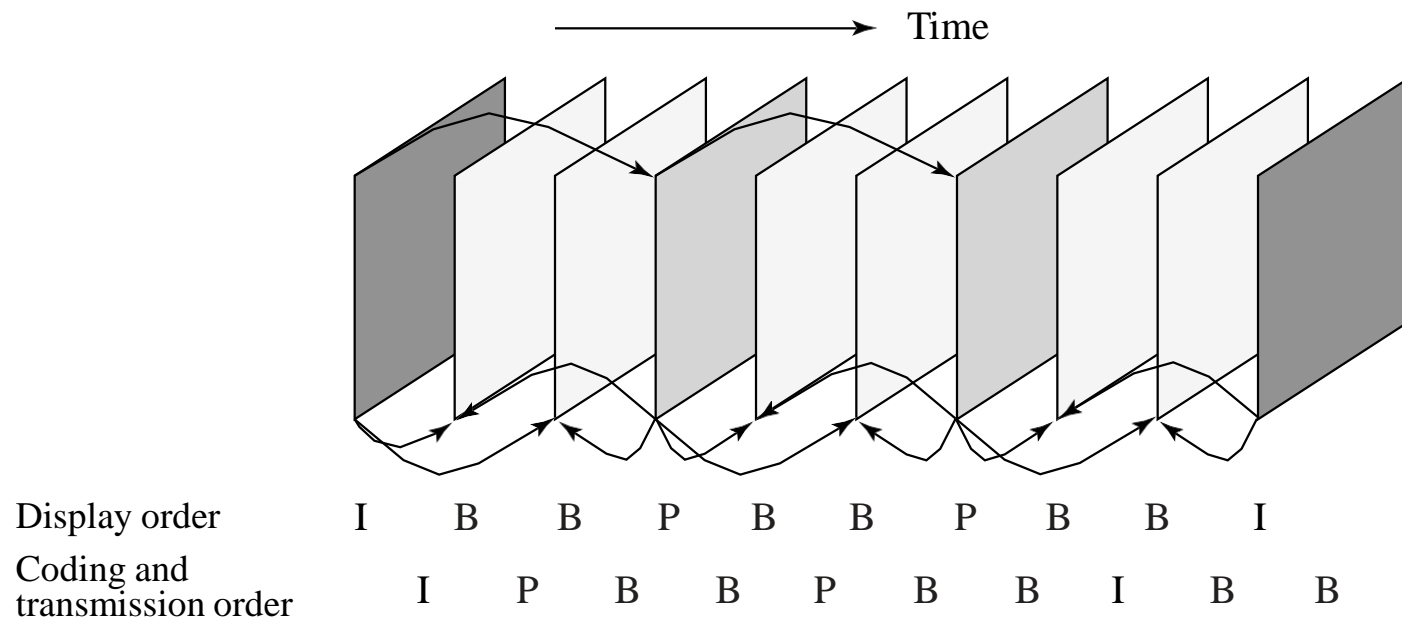
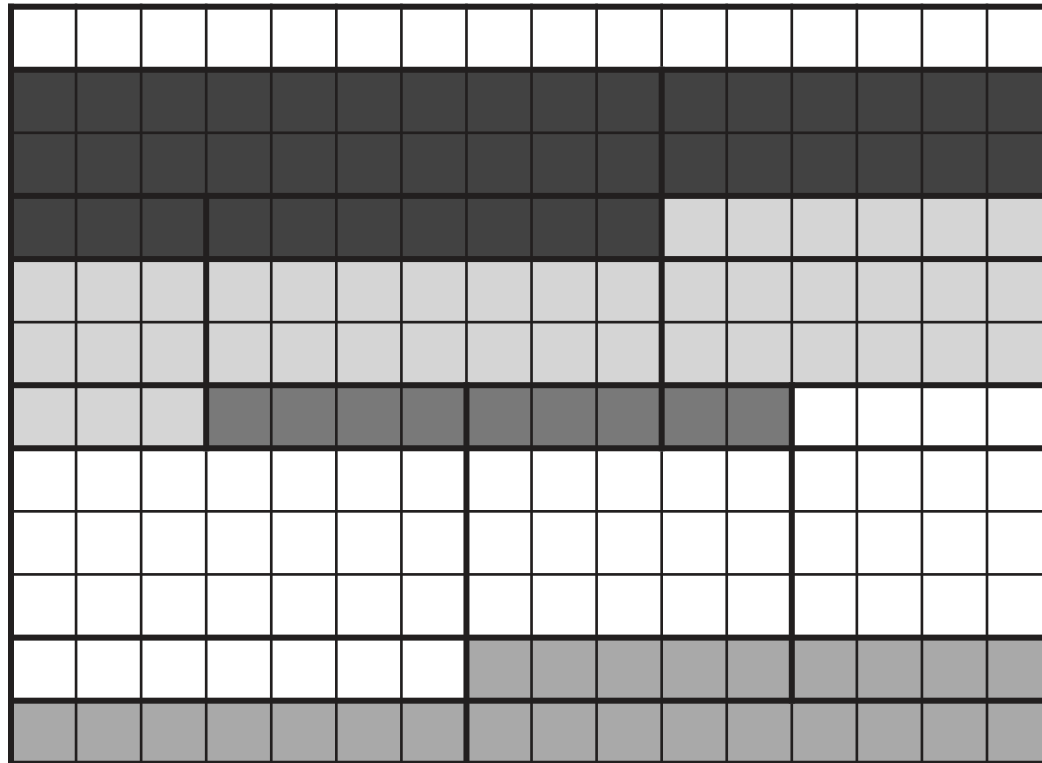


Fig : B-frame Coding Based on Bidirectional Motion Compensation.



MPEG Frame Sequence.



Slices in an MPEG-1 Picture.

Table : Default Quantization Table (Q_1) for Intra-Coding

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Table : Default Quantization Table (Q_2) for Inter-Coding

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Typical Sizes of MPEG-1 Frames

- The typical size of compressed P-frames is significantly smaller than that of I-frames — because temporal redundancy is exploited in inter-frame compression.
- B-frames are even smaller than P-frames — because of (a) the advantage of bi-directional prediction and (b) the lowest priority given to B-frames.

Table : Typical Compression Performance of MPEG-1 Frames

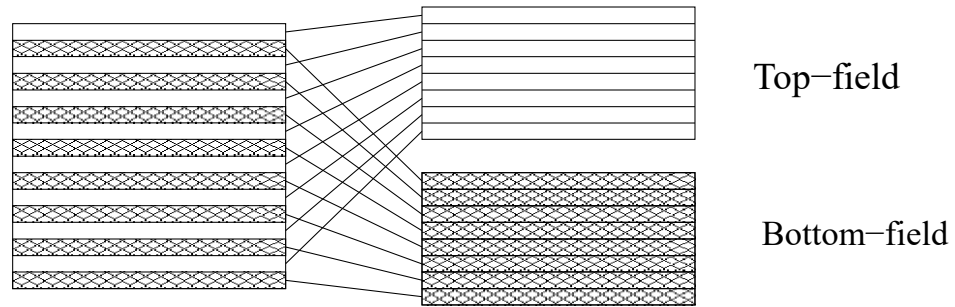
Type	Size	Compression
I	18 kB	7:1
P	6 kB	20:1
B	2.5 kB	50:1
Avg	4.8 kB	27:1

Table : Profiles and Levels in MPEG-2

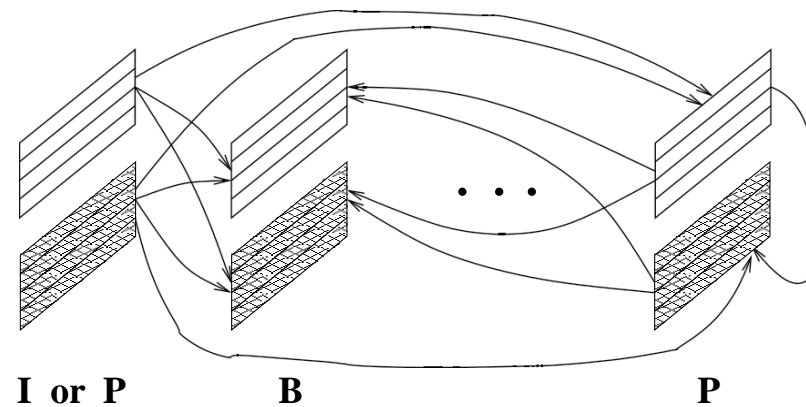
Level	Simple Profile	Main Profile	SNR Scalable Profile	Spatially Scalable Profile	High Profile	4:2:2 Profile	Multiview Profile
High		*			*		
High 1440	*	*	*	*	*	*	*
Main							
Low		*	*				

Table : Four Levels in the Main Profile of MPEG-2

Level	Max Resolution	Max fps	Max Pixels/sec	Max coded Data Rate (Mbps)	Application
High	1,920 × 1,152	60	62.7 × 10 ⁶	80	film production
High 1440	1,440 × 1,152	60	47.0 × 10 ⁶	60	consumer HDTV
Main	720 × 576	30	10.4 × 10 ⁶	15	studio TV
Low	352 × 288	30	3.0 × 10 ⁶	4	consumer tape equiv.

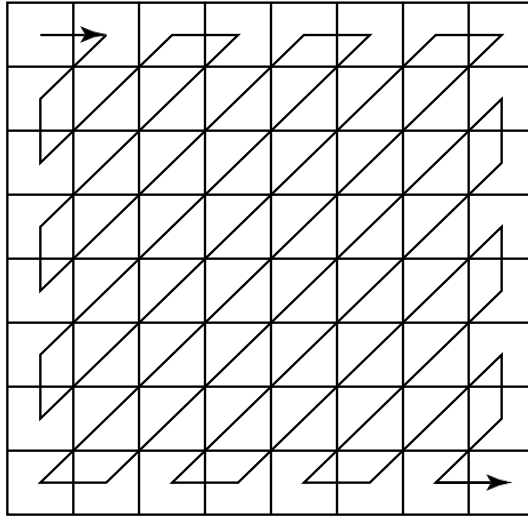


(a) Frame-picture vs. Field-pictures

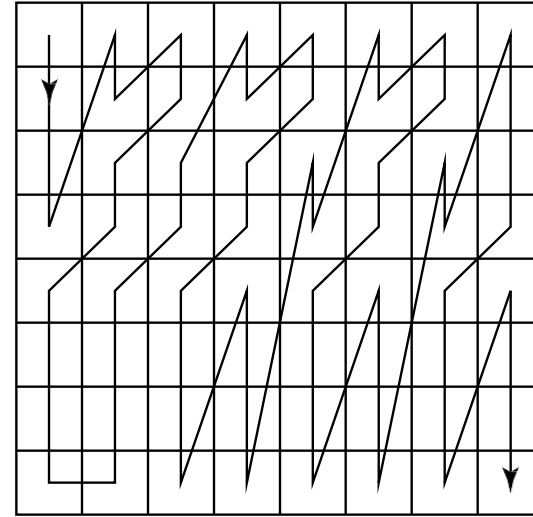


(b) Field Prediction for Field-pictures

Fig.: Field pictures and Field-prediction for Field-pictures in MPEG-2.



(a)



(b)

Fig : Zigzag and Alternate Scans of D C T Coefficients for Progressive and Interlaced Videos in M P E G - 2 .

MPEG-2 Scalabilities (Cont'd)

- MPEG-2 supports the following scalabilities:
 1. SNR Scalability — enhancement layer provides higher SNR.
 2. Spatial Scalability — enhancement layer provides higher spatial resolution.
 3. Temporal Scalability — enhancement layer facilitates higher frame rate.
 4. Hybrid Scalability — combination of any two of the above three scalabilities.
 5. Data Partitioning — quantized DCT coefficients are split into partitions.

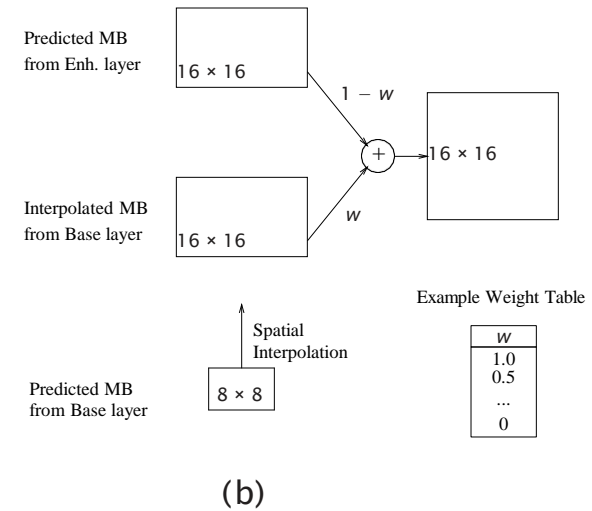
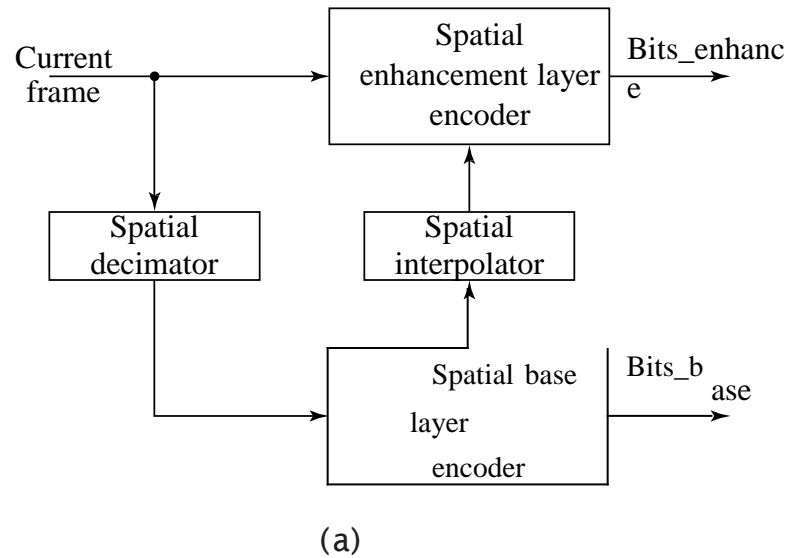


Fig. :Encoder for M P E G – 2 Spatial Scalability. (a) Block Diagram.
(b) Combining Temporal and Spatial Predictions for Encoding at Enhancement Layer.

Temporal Scalability

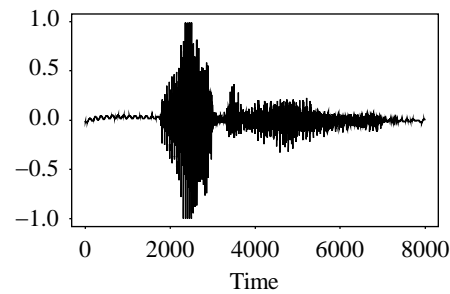
- The input video is temporally de multiplexed into two pieces, each carrying half of the original frame rate.
- Base Layer Encoder carries out the normal single-layer coding procedures for its own input video and yields the output bits stream Bits base.
- The prediction of matching MBs at the Enhancement Layer can be obtained in two ways:
 - Interlayer MC (Motion-Compensated) Prediction.
 - Combined MC Prediction and Interlayer MC Prediction.

Hybrid Scalability

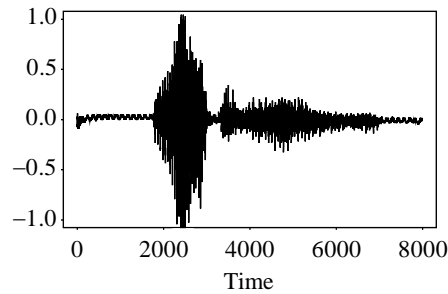
- Any two of the above three scalabilities can be combined to form hybrid scalability:
 1. Spatial and Temporal Hybrid Scalability.
 2. SNR and Spatial Hybrid Scalability.
 3. SNR and Temporal Hybrid Scalability.
- Usually, a three-layer hybrid coder will be adopted which consists of Base Layer, Enhancement Layer 1, and Enhancement Layer 2.

ADPCM in Speech Coding

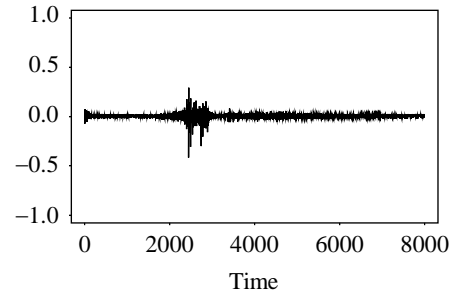
- ADPCM forms the heart of the ITU's speech compression standards G.721, G.723, G.726, and G.727.
- The difference between these standards involves the bit-rate (from 3 to 5 bits per sample) and some algorithm details.
- The default input is μ -law coded PCM 16-bit samples.



(a)



(b)



(c)

Fig. Waveform of Word "Audio": (a) Speech sample, linear PCM at 8 kHz/16 bits per sample. (b) Speech sample, restored from G.721-compressed audio at 4 bits/sample. (c) Difference signal between (a) and (b).

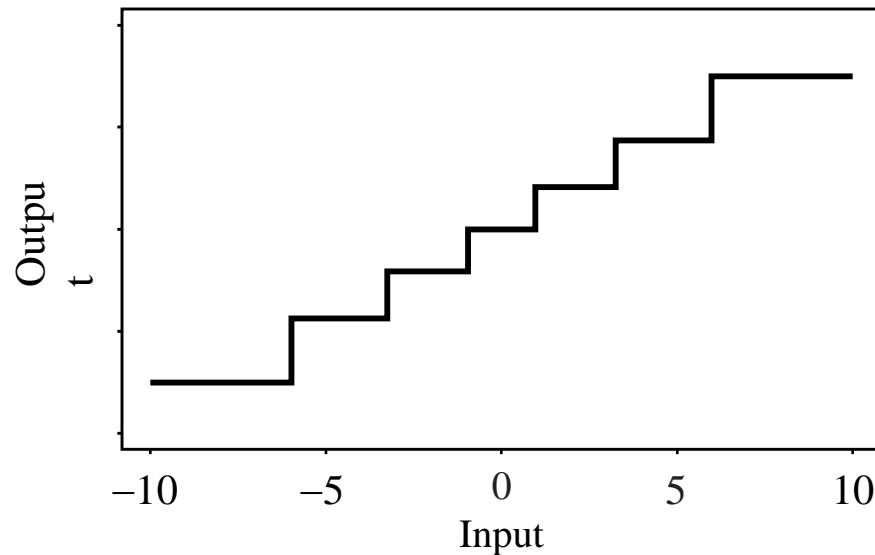


Fig.: G.726 Quantizer

- The input value is a ratio of a difference with the factor α .
- By changing α , the quantizer can adapt to change in the range of the difference signal — a *backward adaptive* quantizer.

Backward Adaptive Quantizer

- **Backward adaptive** works in principle by noticing either of the cases:
 - too many values are quantized to values far from zero – would happen if quantizer step size in f were too small.
 - too many values fall close to zero too much of the time
- would happen if the quantizer step size were too large.
- **Jayant quantizer** allows *one* to adapt a backward quantizer step size after receiving just one single output.
- Jayant quantizer simply expands the step size if the quantized input is in the outer levels of the quantizer, and reduces the step size if the input is near zero.

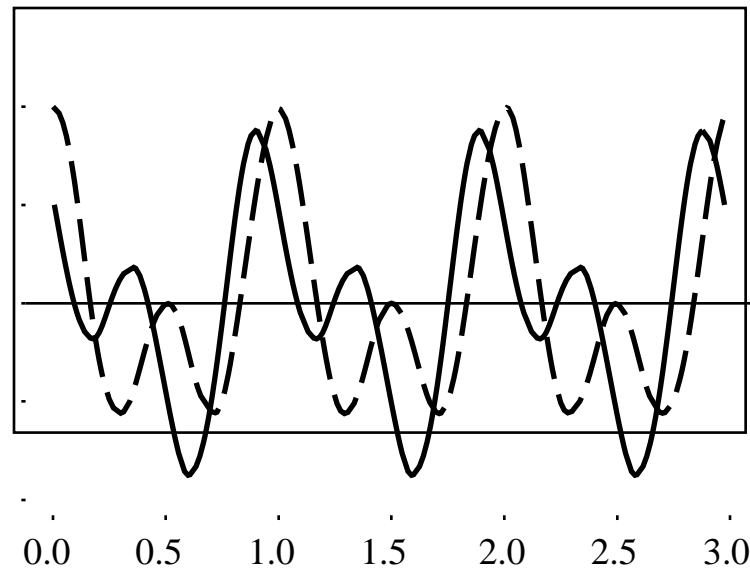
The Step Size of Jayant Quantizer

- Jayant quantizer assigns *multiplier values* M_k to each level, with values smaller than unity for levels near zero, and values larger than 1 for the outer levels.
- For signal f_n , the quantizer step size Δ is changed according to the quantized value k , for the previous signal value f_{n-1} , by the simple formula

$$\Delta \leftarrow M_k \Delta$$

Phase Insensitivity

- A complete reconstituting of speech waveform is really unnecessary, perceptually: all that is needed is for the amount of energy at any time to be about right, and the signal will sound about right.
- **Phase** is a shift in the time argument inside a function of time
 - Suppose we strike a piano key, and generate a roughly sinusoidal sound $\cos(\omega t)$, with $\omega = 2\pi f$.
 - Now if we wait sufficient time to generate a phase shift $\pi/2$ and then strike another key, with sound $\cos(2\omega t + \pi/2)$, we generate a waveform like the solid line in Fig. 13.3.
 - This waveform is the sum $\cos(\omega t) + \cos(2\omega t + \pi/2)$.



If we did not wait before striking the second note, then our waveform would be $\cos(\omega t) + \cos(2\omega t)$. But perceptually, the two notes would sound the same sound, even though in actuality they would be shifted in phase.

Channel Vocoder

- Vcoders can operate at low bit-rates, 1–2 kbps. To do so, a *channel vocoder* first applies a filter bank to separate out the different frequency components:

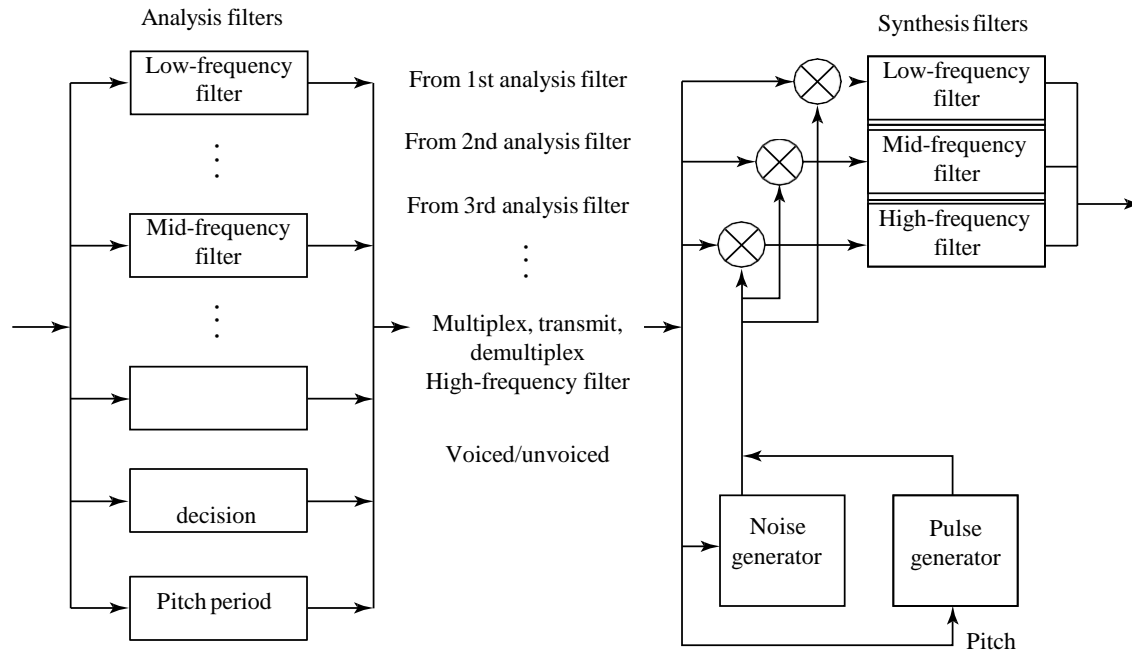


Fig : Channel Vocoder

LPC Coding Process (cont'd)

- Since $\varphi(i, j)$ can be defined as $\varphi(i, j) = R(|i - j|)$, and when $R(0) \geq 0$, the matrix $\{\varphi(i, j)\}$ is positive symmetric, there exists a fast scheme to calculate the LP coefficients:

$E(0) = R(0)$, $i = 1$
while $i \leq p$

$$\begin{aligned}
 k_i &= [R(i) - \sum_{j=1}^{i-1} a_j^{i-1} R(i-j)] / E(i-1) \\
 a_i^{i-1} &= k_i \\
 \text{for } j &= 1 \text{ to } i-1 \\
 a_j^i &= a_j^{i-1} - k_i a_{i-j}^{i-1} \\
 E(i) &= (1 - k_i^2) E(i-1) \\
 i &\leftarrow i+1 \quad \text{for} \\
 i &= 1 \text{ to } p \\
 a_j &= a_j^i
 \end{aligned}$$

The Predictors for CELP

- CELP coders contain two kinds of prediction:
 - LTP (Long time prediction): try to reduce redundancy in speech signals by finding the basic periodicity or pitch that causes a waveform that more or less repeats
 - STP (Short Time Prediction): try to eliminate the redundancy in speech signals by attempting to predict the next sample from several previous ones

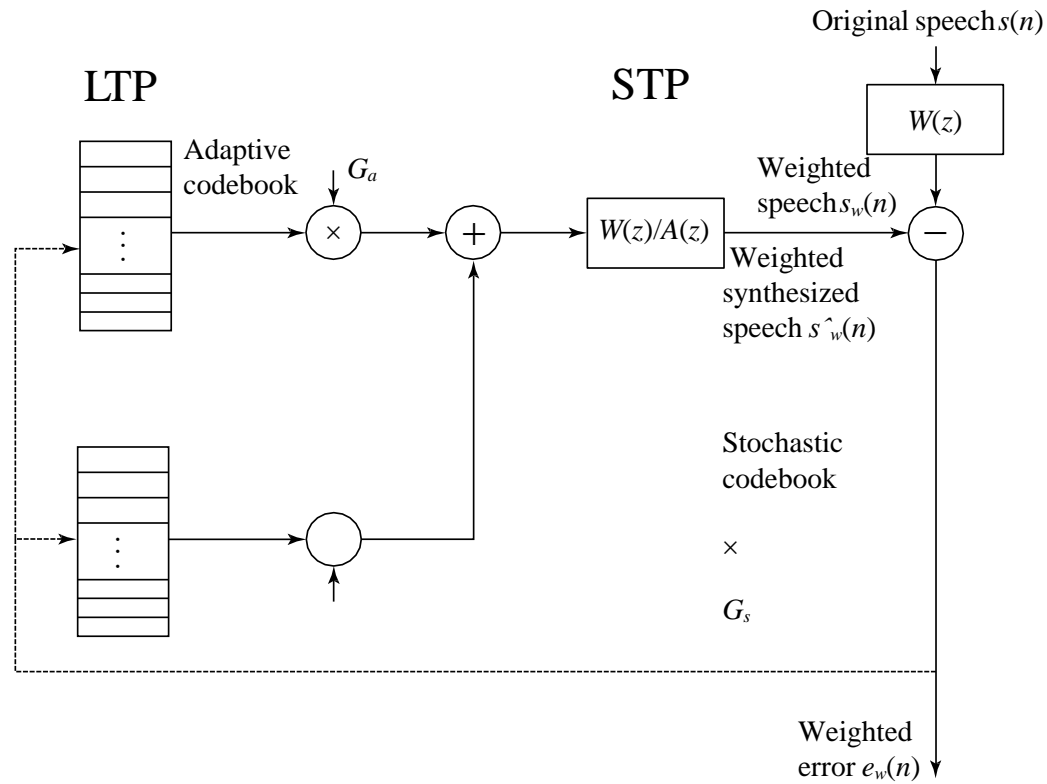


Fig CELP Analysis Model with Adaptive and Stochastic Codebooks

Adaptive Codebook Searching

- **Rationale:**

- Look in a codebook of waveforms to find one that matches the current sub frame
- *Codeword*: a shifted speech residue segment indexed by the lag τ corresponding to the current speech frame or sub frame in the adaptive codebook
- The gain corresponding to the codeword is denoted as g_0

LZW Close-Loop Codeword Searching

- Closed-loop search is more often used in CELP coders — also called *Analysis-By-Synthesis* (A-B-S)
- speech is reconstructed and perceptual error for that is minimized via an adaptive codebook search, rather than simply considering sum-of-squares
- The best candidate in the adaptive codebook is selected to minimize the distortion of locally reconstructed speech
- Parameters are found by minimizing a measure of the difference between the original and the reconstructed speech

MBE Vocoder

- MBE utilizes the A-B-S scheme in parameter estimation:
 - The parameters such as basic frequency, spectrum envelope, and sub-band U/V decisions are all done via closed-loop searching
 - The criterion of the closed-loop optimization is based on minimizing the perceptually weighted reconstructed speech error, which can be represented in frequency domain as

$$\varepsilon = \frac{1}{2\pi} \int_{-\pi}^{+\pi} tt(\omega) |S_w^+(\omega) - S_{wr}(\omega)| d\omega$$

$S_w(\omega)$ - original speech short-time spectrum

$S_{wr}(\omega)$ - reconstructed speech short-time spectrum

$tt(\omega)$ - spectrum of the perceptual weighting filter

MELP Vocoder

- **MELP:** also based on LPC analysis, uses a multiband soft-decision model for the excitation signal
- The LP residue is band passed and a voicing strength parameter is estimated for each band
- Speech can be then reconstructed by passing the excitation through the LPC synthesis filter
- Differently from MBE, MELP divides the excitation into five fixed bands of 0-500, 500-1000, 1000-2000, 2000-3000, and 3000-4000 Hz

Web 2.0

Network effects from user contributions are the key to market dominance in the Web 2.0 era.

—Tim O'Reilly

Link by link, click by click, search is building possibly the most lasting, ponderous, and significant cultural artifact in the history of humankind: the Database of Intentions.

—John Battelle, *The Search*

Web 2.0 is a massive social experiment...this is an opportunity to build a new kind of international understanding...citizen to citizen, person to person.

—Lev Grossman, *TIME*

One of the powerful things about networking technology like the Internet or the Web or the Semantic Web...is that the things we've just done with them far surpass the imagination of the people who invented them.

—Tim Berners-Lee, interviewed by Peter Moon, *IDG Now*

Introduction

- Mosaic browser introduced in 1993 → web exploded in popularity.
- Continued to experience tremendous growth throughout the 1990s—“dot-com bubble”
- Bubble burst in 2001
- In 2003, noticeable shift in how people and businesses were using the web and developing web-based applications
 - Web 2.0 = companies use the web as a platform to create collaborative, community-based sites (e.g., social networking sites, blogs, wikis, etc.)
- Growth of Web 2.0 key factors
 - Hardware keeps getting cheaper and faster, with memory capacities and speeds increasing at a rapid rate
 - Broadband Internet use has exploded
 - Availability of abundant open source software has resulted in cheaper (and often free) customizable software options
 - Makes it easier to start new Web 2.0 companies and greatly decreases the cost of failure
 - Unlike Web 1.0, there are many easy-to-employ models available to monetize Web 2.0 business

User-Generated Content

- Key to success for many of today's leading Web 2.0 companies = user-generated content
 - articles
 - home videos
 - Photos
 - implicitly generated
- *Collective Intelligence*
 - Collaboration can result in smart ideas
- *Wikis*
 - Allow users to edit existing content and add new information
 - Wikipedia
 - Wikia
 - Media•Wiki open source software
 - SocialText
 - Using wikis for project collaboration reduces e-mails and phone calls between employees, while allowing the ability to closely track a project's changes
- *Collaborative Filtering*
 - Users might submit false or faulty information
 - Wikipedia → people deliberately adding false information to entries
 - Web 2.0 companies rely on the community to help police their sites
 - Collaborative filtering lets users promote valuable material and flag offensive or inappropriate material

User-Generated Content (Cont.)

- *Craigslist*
 - Popular classified ads website that has radically changed the classified advertising market
 - Ad postings on Craigslist are free
 - Newspapers have experienced a decline in classified ad sales
- *Wisdom of Crowds*
 - Large diverse groups of people can be smarter than a small group of specialists

Blogging

- *History of Blogging*
 - Blogs are websites consisting of entries listed in reverse chronological order
 - Grown exponentially in recent years because of easy-to-use blogging software and increasingly economical Internet access
 - Blogs can also now incorporate media, such as music or videos
 - Xanga or LiveJournal
- *Blog Components*
 - Reader comments
 - Trackbacks
 - Blogroll
- *Blogging and Journalism*
 - Encouraged citizen journalism
 - Significant news resource
 - Many bloggers are recognized as members of the media

Social Networking

- Social networking sites
 - Allow users to keep track of their existing interpersonal relationships and form new ones
- *Network Effects*
 - Increased value of a network as its number of users grows
 - Example = eBay—the more buyers and sellers that use the site, the more valuable the site becomes to its users
 - Set the user preferences to default to share content so users will automatically contribute to the value of the network
 - Network effects make it difficult to break into markets already claimed by successful companies
- *Friendster*
 - Early leader in social networking
- *MySpace*
 - Most popular social networking site
 - Pages are personal and customizable
 - News Corp, which acquired MySpace in 2005 for \$580 million

Social Networking (Cont.)

- *Facebook*
 - Hitwise named Facebook the “preferred network among college students
 - Facebook held an 85% market share of four-year U.S. universities and had over 31 million users
- *LinkedIn*
 - Business-oriented social networking site
 - stay in touch with professional contacts
 - network with new contacts
 - check references
 - find a job or a potential employee
 - privacy concerns are more
- *Xing*
 - Xing is a professional networking site based out of Germany and popular in Europe
- *Second Life*
 - Second Life, developed by Linden Labs, is a 3D virtual world with millions of inhabitants
 - Users create avatars, digital representations of themselves that they can use to meet other users with similar interests, conduct business, participate in group activities, take classes and more
 - Users can create objects and add scripts (to animate the objects) in the virtual world
 - Users to maintain rights to whatever they create, a dynamic marketplace has emerged that does millions of dollars in transactions

Social Networking (Cont.)

- *Gaia Online*
 - Popular teen virtual world.
 - Play games, make friends and express their creativity
- *Mobile Social Networking*
 - Google's Dodgeball.com provides users with mobile access to a network of friends in many cities.

Social Media

- Social media = any media shared online (e.g., videos, music, photos, news, etc)
- *YouTube*
 - Launched in late 2005 and is the leading Internet video site
 - Entire site is based on user-generated content
 - Can browse videos by category, tag, or by following “related video” links
 - Users can subscribe to other users’ content, share videos with friends by e-mail, or embed videos directly into their blogs or other websites
 - YouTube was acquired by Google for \$1.65 billion.
- *Internet TV*
 - Many mass-media companies now offer full-length episodes of popular television shows
 - Limited by copyright issues
 - Internet TV allows advertisers to target their markets more precisely than with broadcast television
- *Digg*
 - Features news, videos and podcasts, all posted and rated by users
 - Gained popularity by allowing users to “digg” or “bury” posts and user comments
 - Digg uses collaborative filtering
- *Last.fm*
 - Last.fm is an Internet radio website that uses Web 2.0 concepts to make music recommendations and build communities

Social Media (Cont.)

- *Digital Rights Management (DRM)*
 - Add software to media files to prevent them from being misused
 - Protect digital products from illegal distribution
- *Podcasting*
 - Popularized by Apple's iPod portable media player.
 - Podcast is a digital audio file (e.g., an .mp3) that often takes on the characteristics of a radio talk show
 - Introduced a more democratic form of radio broadcasting

Tagging

- *History of Tagging*
 - Tagging, or labeling content, is part of the collaborative nature of Web 2.0
 - Tag is any user-generated word or phrase that helps organize web content and label it in a more human way
- *Tag Clouds*
 - Visual displays of tags weighted by popularity.
- *Folksonomies*
 - Classifications based on tags
 - Formed on sites such as Flickr, Technorati and del.icio.us
- *Flickr*
 - Flickr—a popular photo-sharing site—was launched in February 2004 and acquired by Yahoo! in 2005
 - Key content-tagging site
- *Technorati*
 - Social media search engine that uses tags to find relevant blogs and other forms of social media

Social Bookmarking

- Social bookmarking sites = share your Internet bookmarks (e.g., your favorite websites, blogs, and articles) through a website.
 - del.icio.us
 - Ma.gnolia
 - Blue Dot
 - StumbleUpon
 - Simpy
 - Furl

Software Development

- Key to Web 2.0 software development
 - KIS (keep it simple; keep it small)
 - Important given the “attention economy” (too much information, too little time)
- *The Webtop*
 - Web has now become an application, development, delivery, and execution platform
 - Webtop, or web desktop, allows you to run web applications in a desktop-like environment in a web browser
 - Operating-system-independent applications
- *Software as a Service (SaaS)*
 - Application software that runs on a web server rather than being installed on the client computer
 - Many benefits
 - Fewer demands on internal IT departments
 - Increased accessibility for out-of-the-office use
 - Easy way to maintain software on a large scale
 - Examples: Most Google software and Microsoft’s Windows Live and Office Live.
 - Collaborating on projects with co-workers across the world is easier
 - Information stored on a web server instead of on a single desktop

Software Development

- *Perpetual Beta and Agile Development*
 - Shift away from the traditional software release cycle (i.e., new software releases take months or years)
 - Now a greater focus on agile software development, which refers to development of fewer features at a time with more frequent releases
 - Made possible by using the web as a platform
 - The Internet is a dynamic medium
 - Should not “overuse” betas
- *Open Source*
 - Not always free, but the source code is available (under license) to developers, who can customize it to meet their unique needs
 - Linux operating systems Red Hat or Ubuntu
 - Because the source code is available to everyone, users can look to the community for bug fixes and plug-ins
 - Over 150,000 open source projects are under development
 - Examples: Firefox web browser, the Apache web server, the MySQL database system, DotNetNuke and PHPNuke

Software Development

- *Licensing: GNU Licenses and Creative Commons*
 - GNU General Public License (GPL)
 - Allows redistribution of the project provided the source code is included and the copyright information is left intact

Others: GNU Lesser General Public License and the GNU Free Documentation License, BSD license and the MIT license

- Creative Commons
 - Deals with licensing issues for all types of digital media

Rich Internet Applications (RIAs)

- Rich Internet Applications (RIAs)
 - Web applications that offer the responsiveness, “rich” features and functionality approaching that of desktop applications
- *Ajax*
 - Asynchronous JavaScript and XML
 - Allows partial page
 - Creates a more responsive GUI, allowing users to continue interacting with the page as the server processes requests
 - Technologies that make up Ajax—XHTML, CSS, JavaScript, the DOM, XML, and the XMLHttpRequest object
- *Dojo*
 - Dojo is an open source JavaScript
- *Flex*
 - RIA framework that allows you to build scalable, cross-platform, multimedia-rich applications that can be delivered over the Internet

Rich Internet Applications (RIAs) (Cont.)

- *Silverlight*
 - Microsoft app formerly known as Windows Presentation Foundation Everywhere (WPF/E)
 - Competitor to Flex and Flash
 - Uses a compact version of the .NET framework
 - User interfaces built in Extensible Application Markup Language (XAML)—Microsoft's XML-based format for describing user interfaces
- *JavaFX*
 - Sun Microsystems' counterpart to Flex and Silverlight
 - Consists of the JavaFX Script and JavaFX Mobile (for mobile devices)
- *Ruby on Rails*
 - Open source framework based on the Ruby scripting language that allows you to build database-intensive applications quickly, easily, and with less code
- *Script.aculo.us*
 - Library for creating “eye candy” effects
 - Built on the Prototype JavaScript framework
 - Encapsulates the DOM and provides cross-browser processing capabilities
 - Core effects include opacity, scale, morph, move, highlight and parallel
- *JavaServer Faces*
 - Java-based web application framework
 - Separates design elements from business logic and provides a set of user-interface components (JSF components) that make developing RIAs simple

Rich Internet Applications (RIAs) (Cont.)

- *ASP.NET Ajax*
 - Extension of the .NET framework for creating Ajax-enabled applications
- *Adobe Integrated Runtime and Google Gears*
 - AIR allows users to run Flex web applications on their desktops even when they are *not* connected to the Internet
 - Google Gears allows use of web applications while offline

Web Services, Mashups, Widgets and Gadgets

- Incorporating web services into new programs allows people to develop new applications quickly
- *APIs*
- Provide applications with access to external services and databases
 - Examples: Sun's Java API and Web Services APIs
- *Mashups*
 - Combine content or functionality from existing web services, websites and RSS feeds to serve a new purpose
 - Housingmaps.com
 - Yahoo! Pipes

Web Services, Mashups, Widgets and Gadgets (Cont.)

- *Widgets and Gadgets*
 - Mini applications designed to run either as stand-alone applications or as add-on features in web pages
 - Personalize your Internet experience by displaying real-time weather conditions, aggregating RSS feeds, viewing maps, receiving event reminders, providing easy access to search engines and more.
- *Amazon Web Services*
 - Amazon is a leading provider of web services
- *REST (Representational State Transfer)-Based Web Services*
 - Architectural style for implementing web services
 - Identified by a unique URL

Location-Based Services

- Location-Based Services (LBS)
 - Applications that take your geographic location (city, state, location of your mobile device, etc.) into consideration
 - Global Positioning System (GPS)
 - Local search
- *Global Positioning System (GPS)*
 - Uses numerous satellites that send signals to a GPS receiver to determine its exact location.
- *Mapping Services*
 - Google Maps is one of the most popular mapping applications available online.
 - Google Earth provides satellite images of virtually any location on the planet
 - MapQuest provides similar mapping services
 - Additional mapping services include Yahoo! Local Maps and MSN Live Search
 - Companies such as NAVTEQ and Tele Atlas provide digital map data for in-vehicle and portable navigation devices, websites, location-based services and more
- *GeoRSS and Geotagging*
 - Set of standards for representing geographical information in an RSS feed (GeoRSS)
 - Geotagging can be used to add location information (longitude, latitude, etc.)

XML, RSS, Atom, JSON and VoIP

- *XML*
 - Extensible Markup Language that is a markup language that allows you to label data based on its meaning
 - Describes data in a way that is meaningful to both humans and computers
 - Document Type Definition (DTD) or a schema, which defines the structure for the document
 - XML Vocabularies
 - XHTML for web content
 - CML for chemistry
 - MathML for mathematical content and formulas
 - XBRL for financial data
- *RSS and Atom*
 - Sites that offer RSS and Atom feeds can maintain an “open connection” with their readers
 - Most major web browsers support RSS and Atom feeds
- *JSON*
 - JavaScript Object Notation (JSON)
 - Text-based data interchange format used to represent JavaScript objects as strings and transmit them over a network
 - Commonly used in Ajax applications
- *VoIP*
 - Voice over Internet Protocol (VoIP) is the technology used to make free or inexpensive phone calls over the Internet.

Web 2.0 Monetization Models

- Many Web 1.0 businesses discovered that popularity (“eyeballs”) was not the same as financial success.
- Web 2.0 companies are paying more attention to monetizing their traffic.
- Web 2.0 monetization is heavily reliant on advertising
 - Example: Google’s AdSense.

Web 2.0 Business Models

- Technologies and collaborative nature of Web 2.0 have opened up new business models

Future of the Web

- Computers have a hard time deciphering meaning from XHTML content
- Web today involves *users'* interpretations of what pages and images mean, but the future entails a shift from XHTML to a more sophisticated system based on XML, enabling *computers* to better understand meaning.
- Web 2.0 companies use “data mining” to extract as much meaning as they can from XHTML-encoded pages
- *Tagging and Folksonomies*
 - Early hints a “web of meaning.”
 - “loose” classification system
- *Semantic Web*
 - Next generation in web development,
 - “web of meaning”
 - Depends heavily on XML and XML-based technologies
- *Micro formats*
 - Standard formats for representing information aggregates that can be understood by computers, enabling better search results and new types of applications

UNIT-IV

Rich Internet Applications (RIAS) with Adobe

Flash: Adobe Flash Introduction, Flash Movie Development, Learning Flash with Hands-on Examples, Publish your flash movie, Creating special effects with Flash, Creating a website splash screen, action script, web sources.

Rich Internet Applications (RIAs) with Flex 3 –

Introduction, Developing with Flex 3, Working with Components, Advanced Component Development, Visual Effects and Multimedia

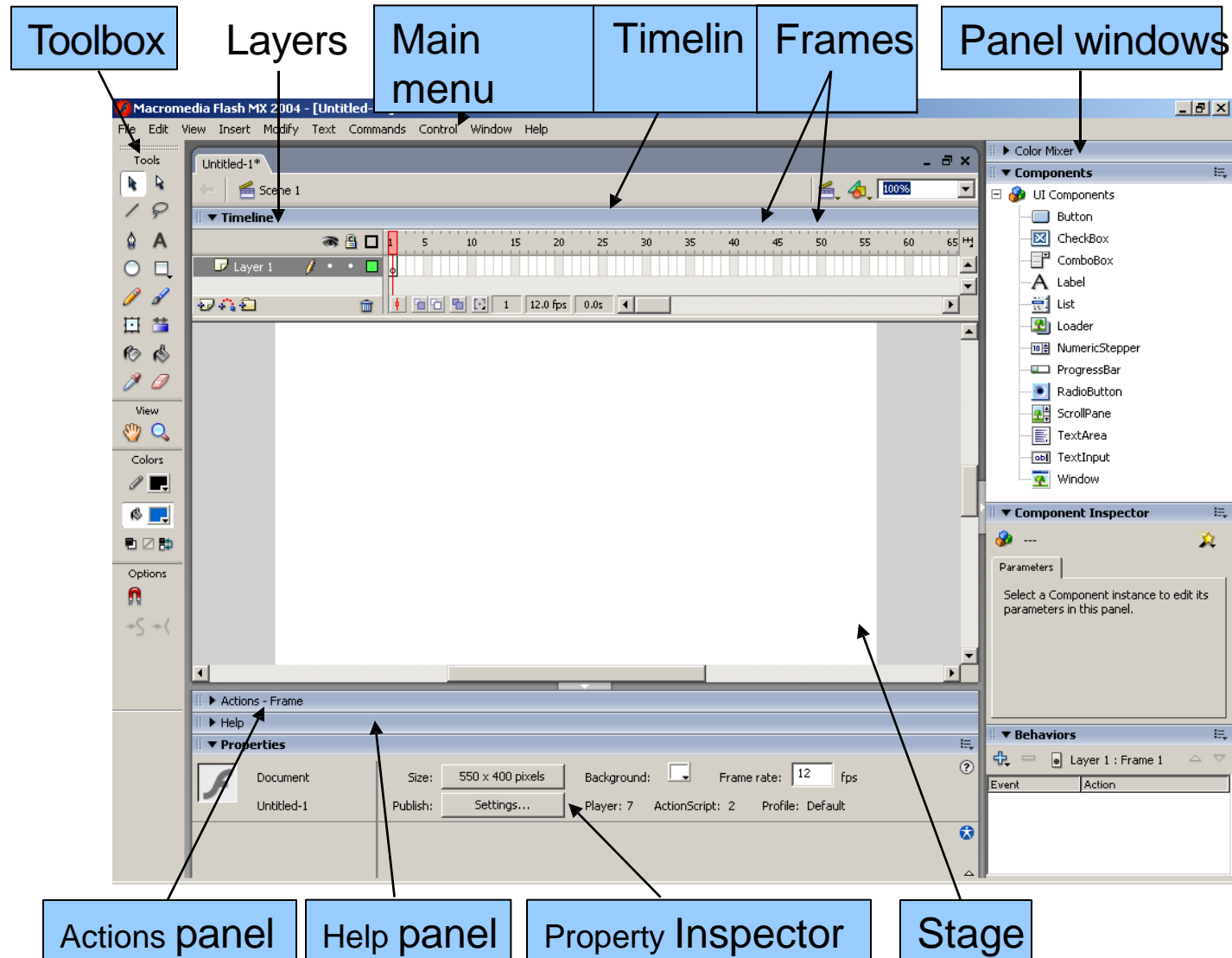
Introduction

- Flash
 - Produce interactive, animated movies
 - Web-based banner advertisements
 - Interactive Web sites
 - Games
 - Web-based applications
 - Provides tools for drawing graphics, generating animation and adding sound and video
 - Tools for coding in its scripting language, Action Script
- Flash Player plug-in
 - Installed in a Web browser to play flash movies

Flash Movie Development

- **Start page**
 - Contains a number of helpful options
 - **Create From Template**
 - **Open a Recent Item**
- Creating blank Flash document
 - Click **Flash Document** under the **Create New** heading
- **Tools** section
 - Contains tools that select, add and remove graphics from Flash movies
- **View** section
 - Contains two tools that modify what portion of stage
- **Colors** section
 - Provides colors for shapes, lines, and filled areas
- **Options** section
 - Contains settings for the active tool

Flash Movie Development



Flash MX development environment.

Flash Movie Development

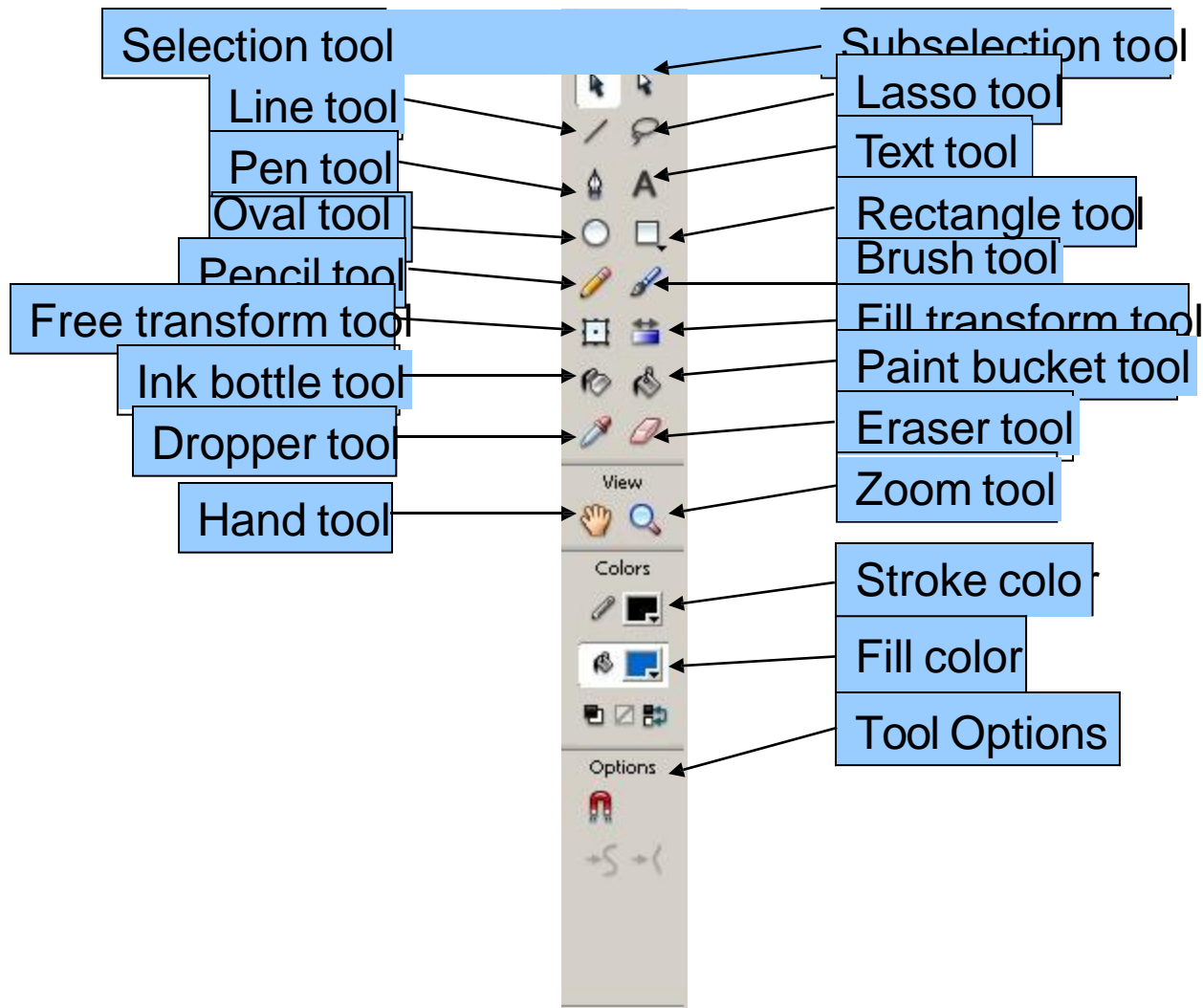


Fig.: Flash MX Toolbox.

Learning Flash with Hands-On Examples

- Open a new Flash movie file
 - Select **New** from the **File** menu
 - In the **New Document** dialog, select **Flash Document** under **General** tab and click **OK**
 - Choose **Save As...** from **File** menu
- Movie options
 - Select **Document Properties**
 - **Frame Rate**
 - The speed at which movie frames display
 - **Dimensions**
 - Define size of movie as it displays on screen
 - **Background Color**
 - Stage color
 - Click **Background Color** box to select background color

Learning Flash with Hands-On Examples

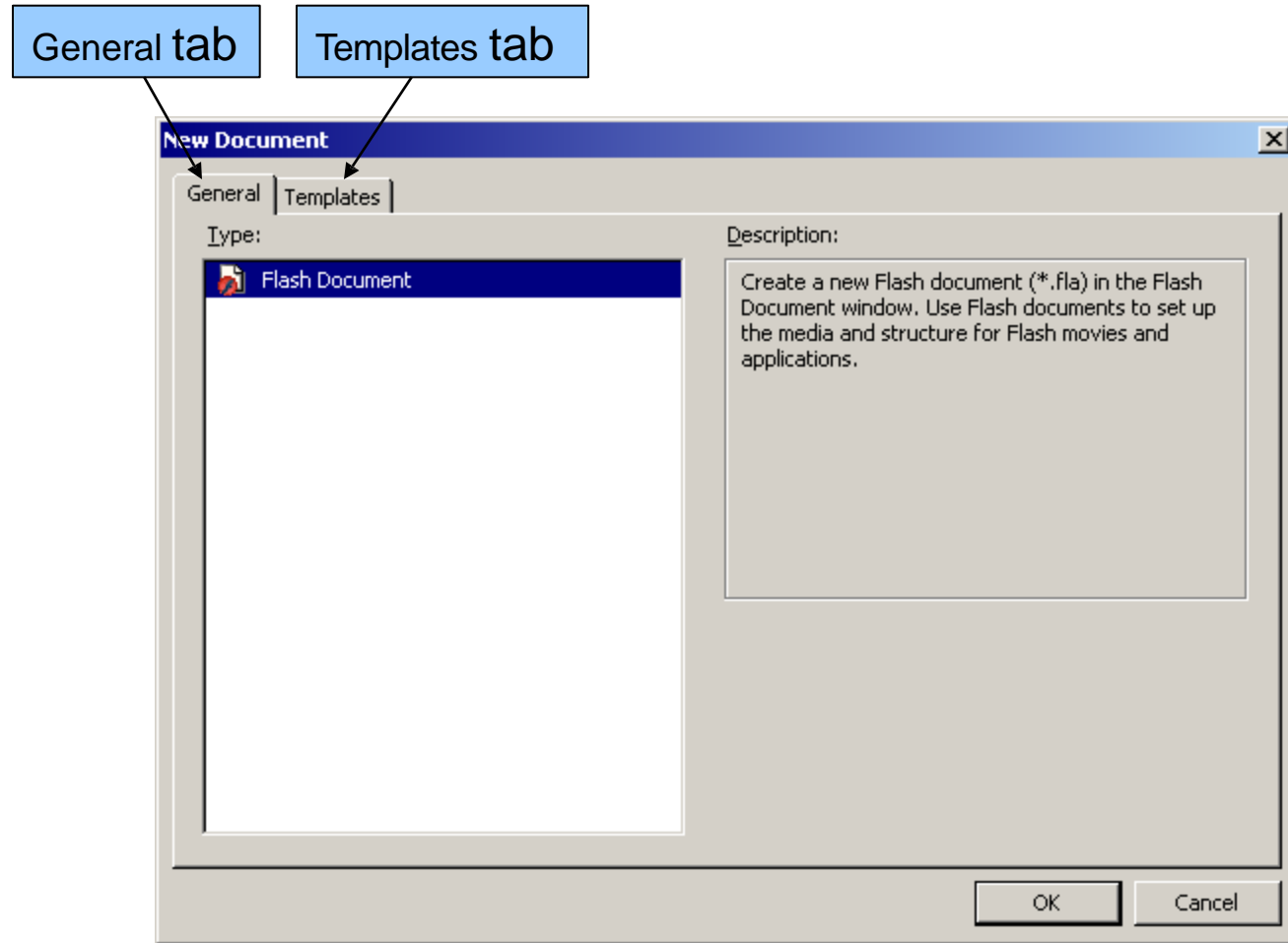


Fig.: New Document dialog.

Learning Flash with Hands-On Examples

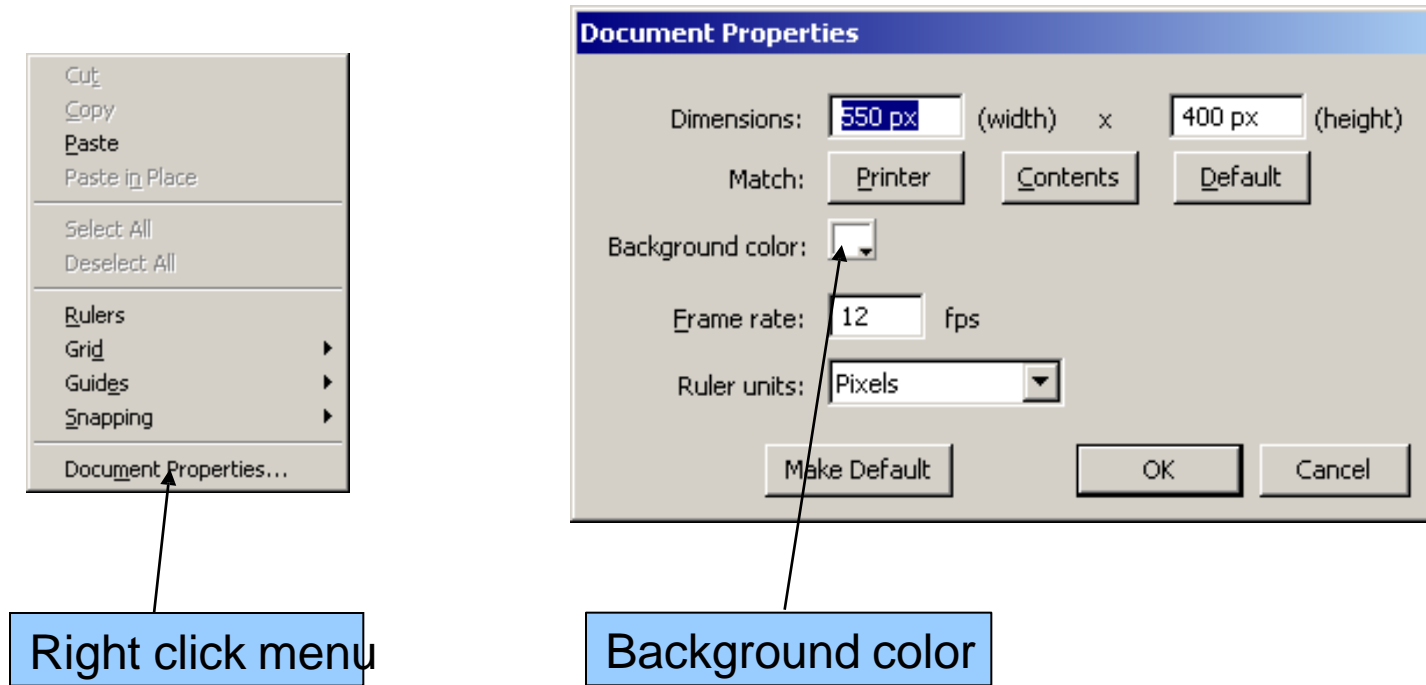


Fig.: Flash MX 2004 Document Properties dialog.

Learning Flash with Hands-On Examples

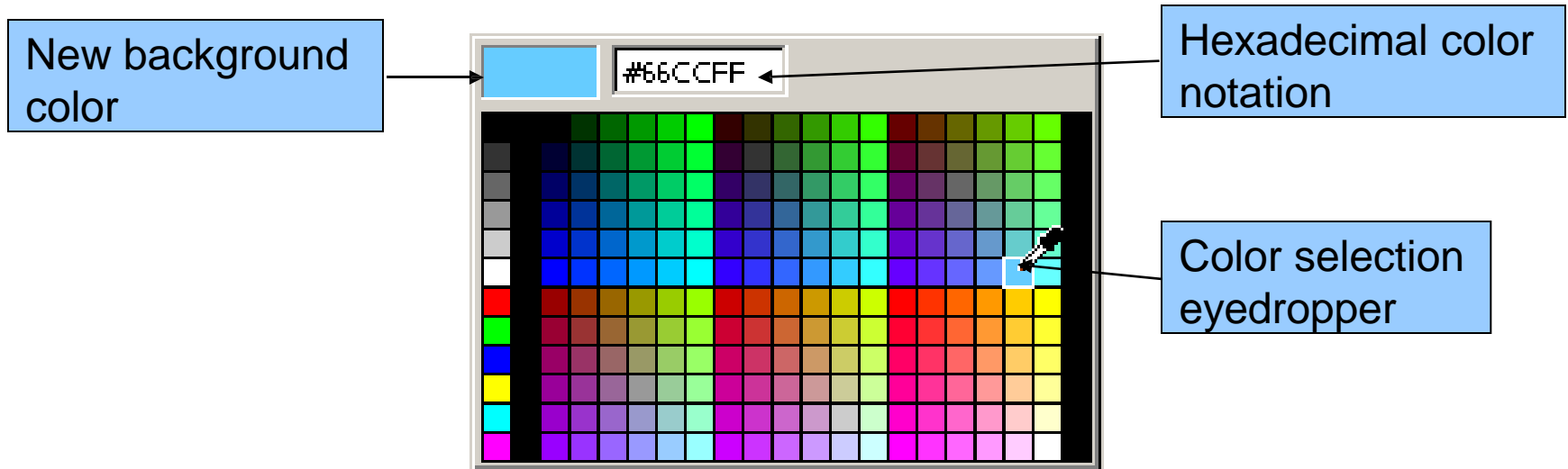


Fig: Selecting a background color.

Creating a Shape with the Oval Tool

- Flash creates shapes using vectors
 - Vectors are mathematical equations that define size, shape and color
- Some graphics applications create raster graphics
 - Defined by areas of colored pixels
- Oval
 - Stroke color
 - Color of a shape's outline
 - Fill color
 - Color that fills the shape
 - **Black and White** button
 - Resets the stroke color to black and the fill color to white
 - **Swap Colors**
 - Switches the stroke and fill color

Creating a Shape with the Oval Tool

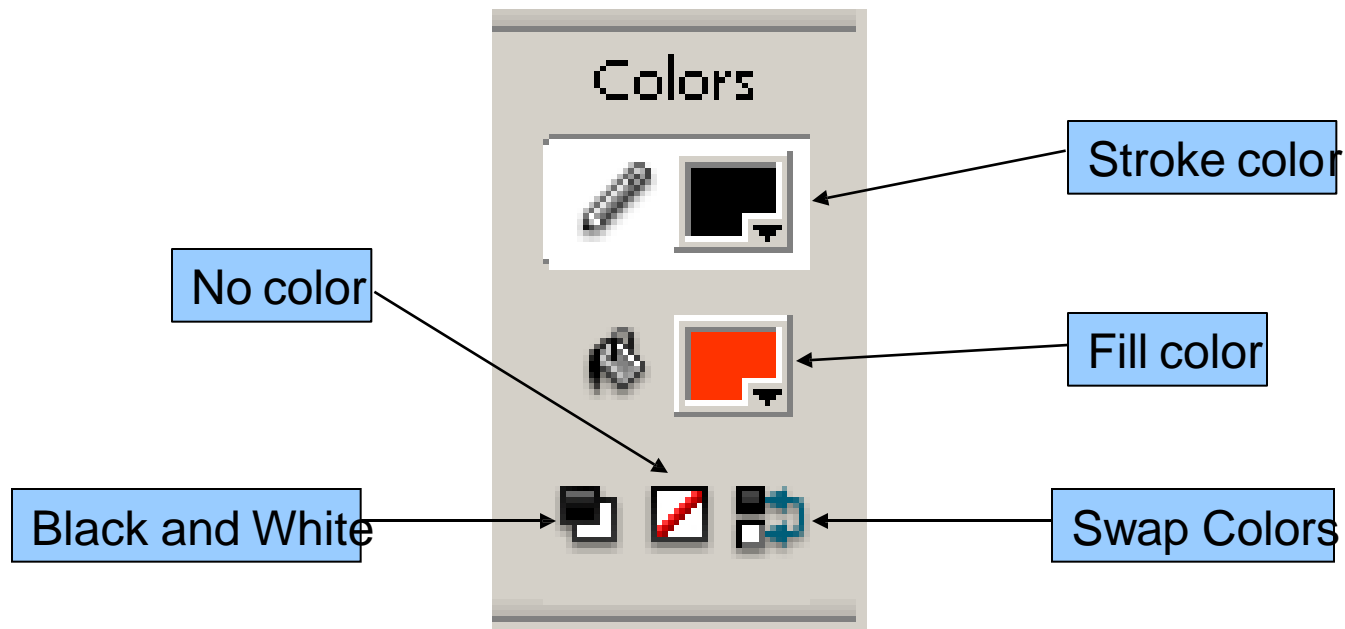


Fig. Setting the fill and stroke colors.

Creating a Shape with the Oval Tool

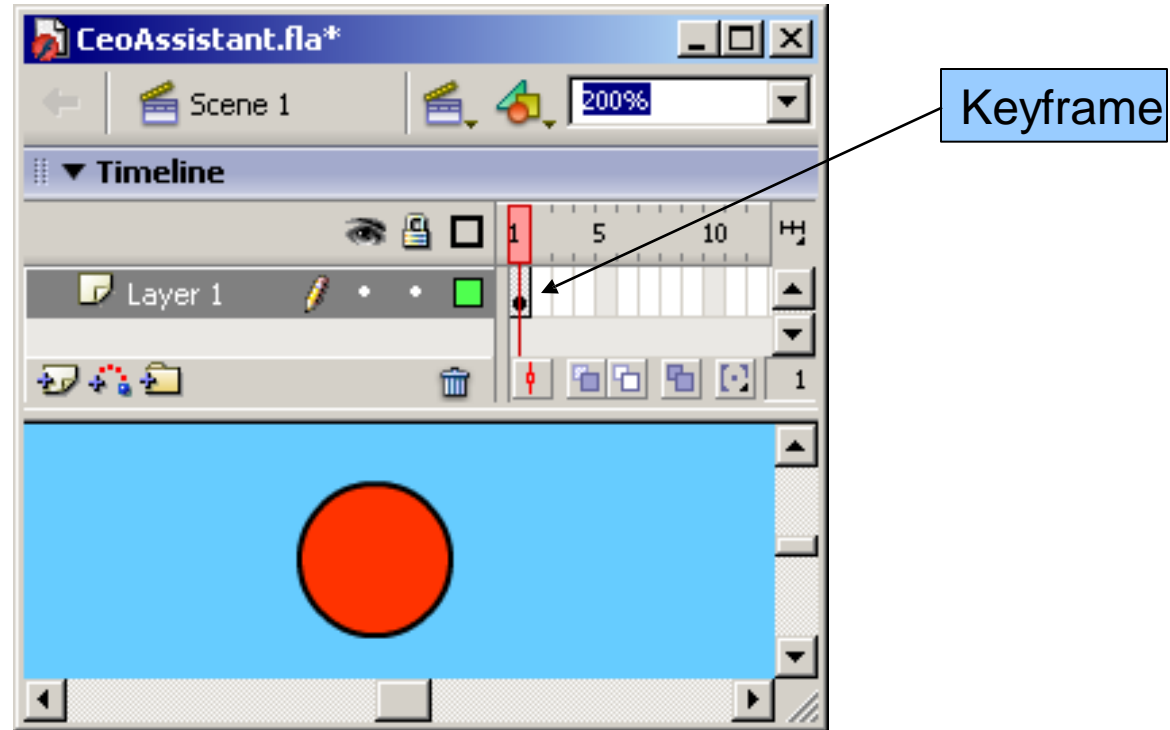
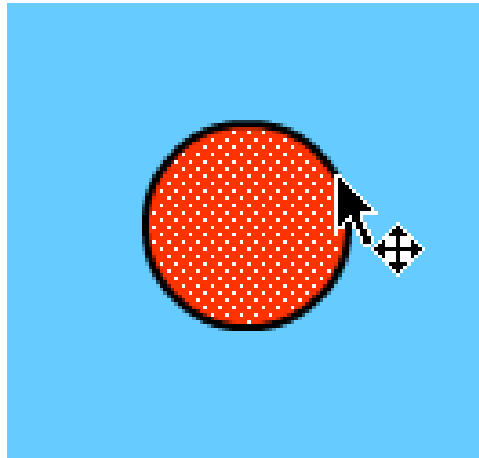


Fig.: Keyframe added to the timeline.

Creating a Shape with the Oval Tool



Making multiple selections with the selection tool.

Creating a Shape with the Oval Tool

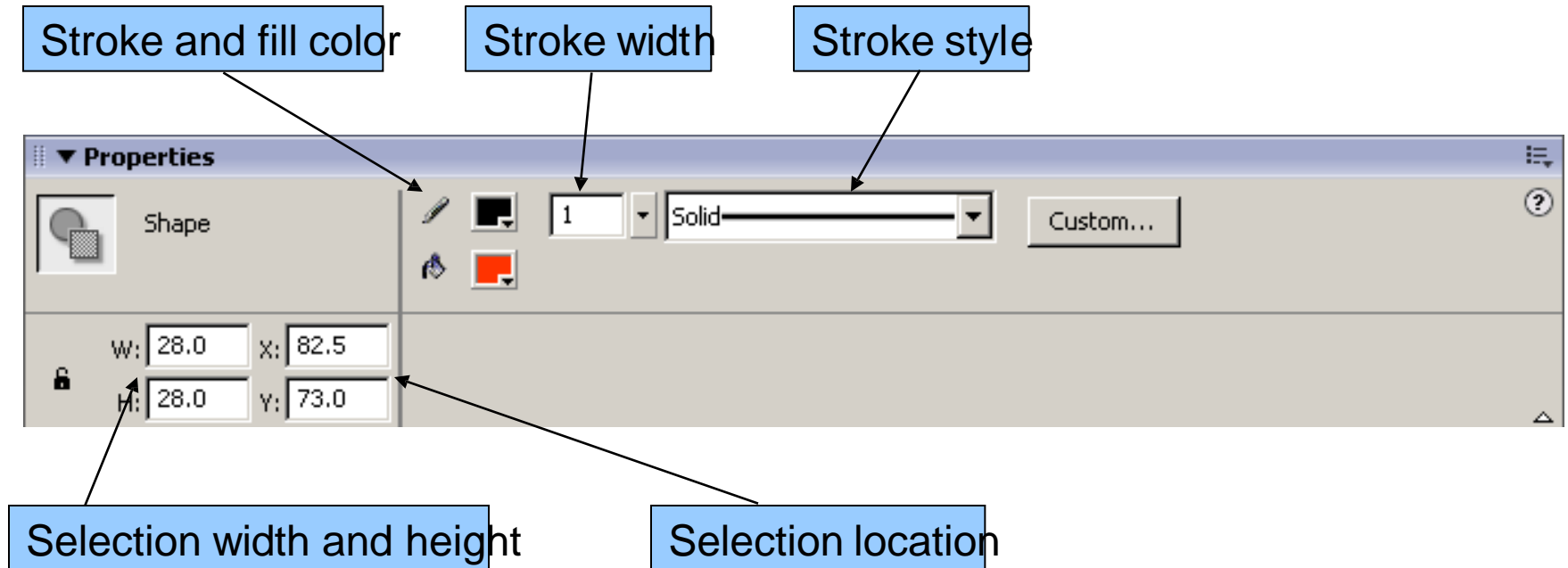
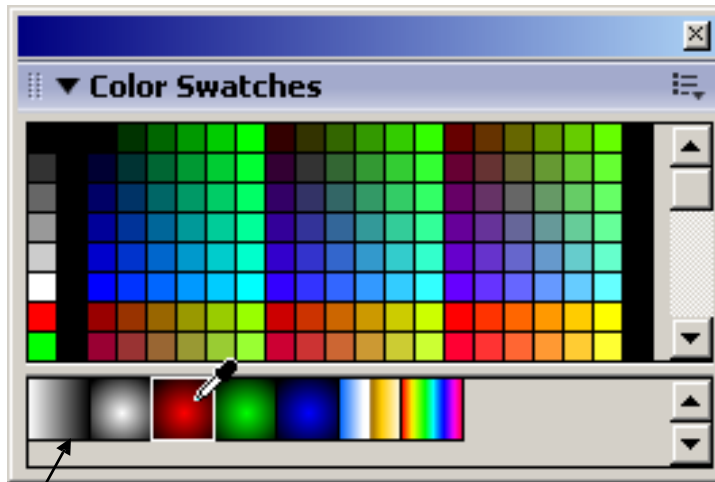
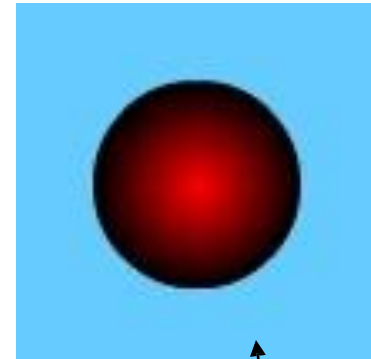


Fig.: Modifying the size of a shape with the Property Inspector.

Creating a Shape with the Oval Tool



Gradient fills



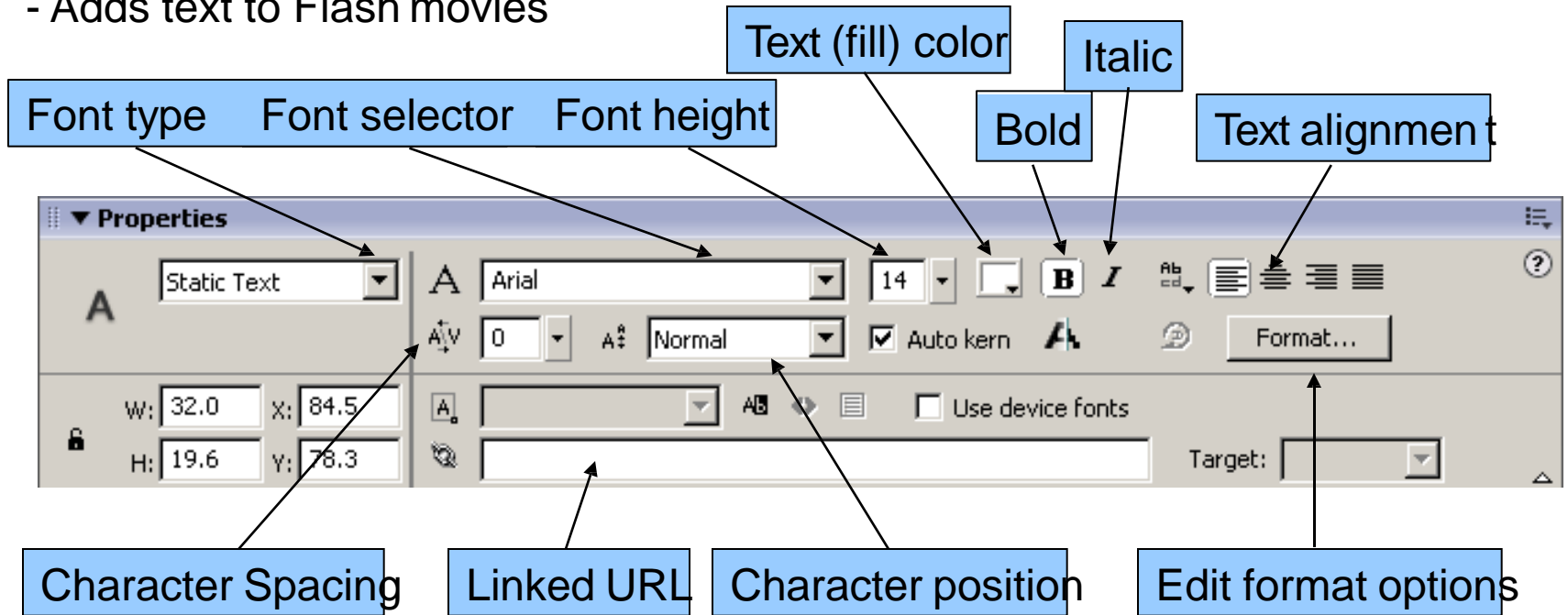
Red radial
gradient fill

Choosing a gradient fill.

Adding Text to a Button

Text tool

- Adds text to Flash movies



Setting the font face, size, weight and color with the Property Inspector.

Adding Text to a Button

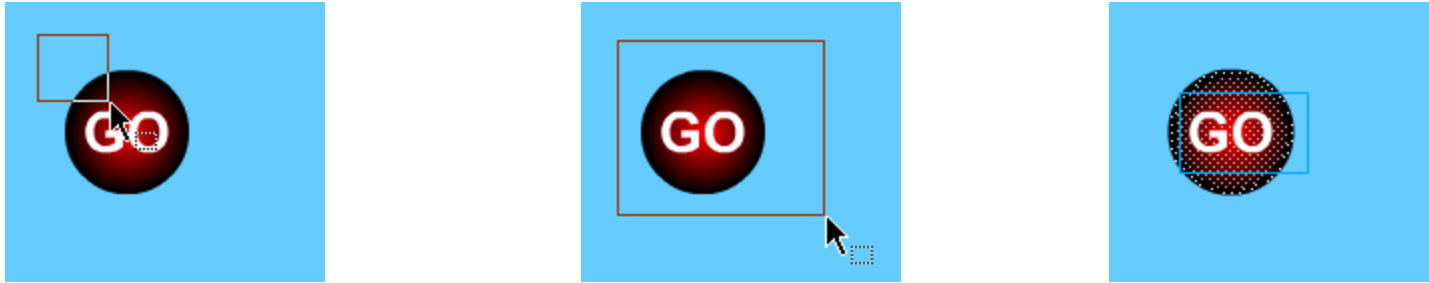


Adding text to the button.

Converting a Shape into a Symbol

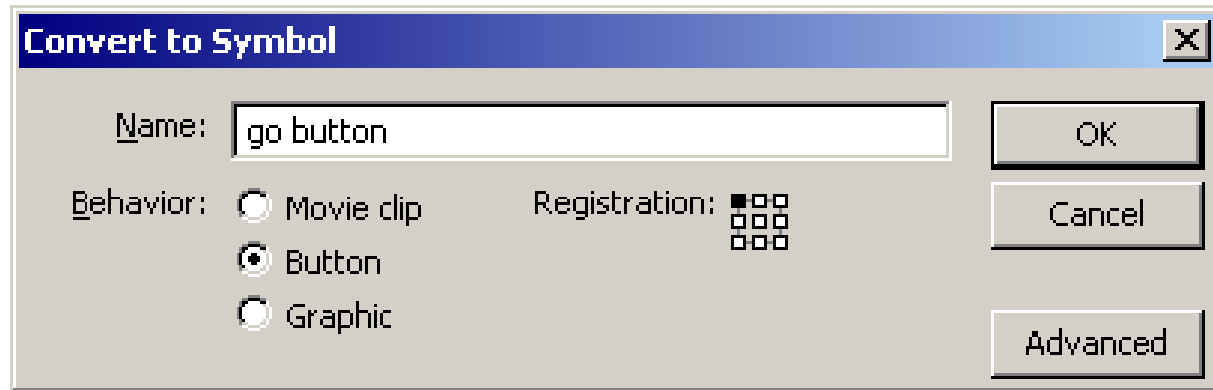
- Flash movie
 - Parent movie
 - A scene
 - Contains the entire movie including all graphics and symbols
 - Reusable movie elements
 - Graphics
 - » Ideal for static images and basic animations
 - Buttons
 - » Objects that perform button actions
 - Movie clips
 - » Ideal for recurring animations
 - Movie explorer
 - Displays the movie structure

Converting a Shape into a Symbol



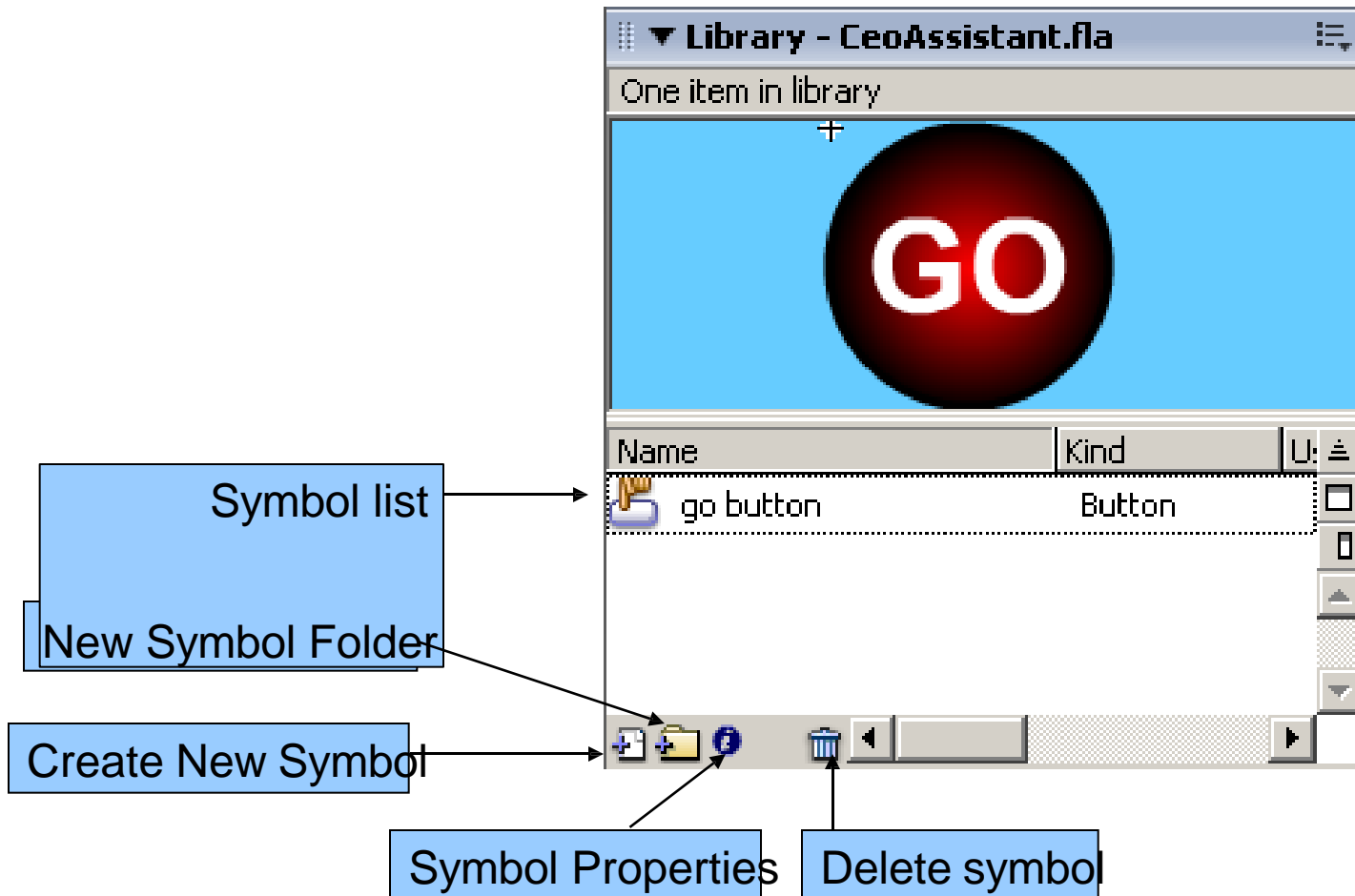
Selecting an object with the selection tool.

Converting a Shape into a Symbol



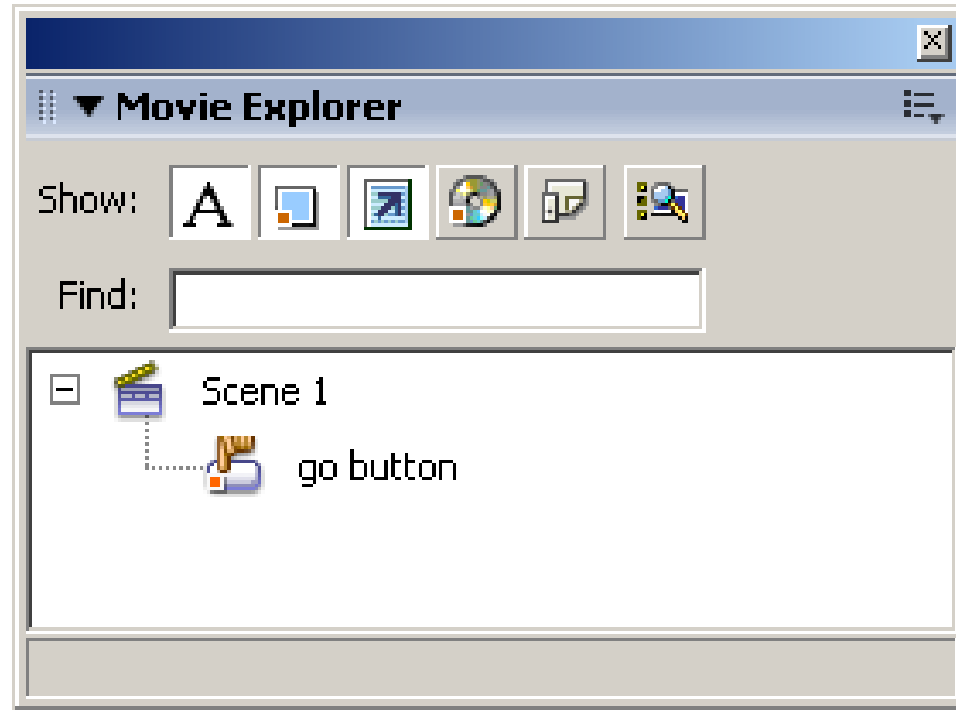
Creating a new symbol with the Convert to Symbol dialog.

Converting a Shape into a Symbol



Library panel.

Converting a Shape into a Symbol

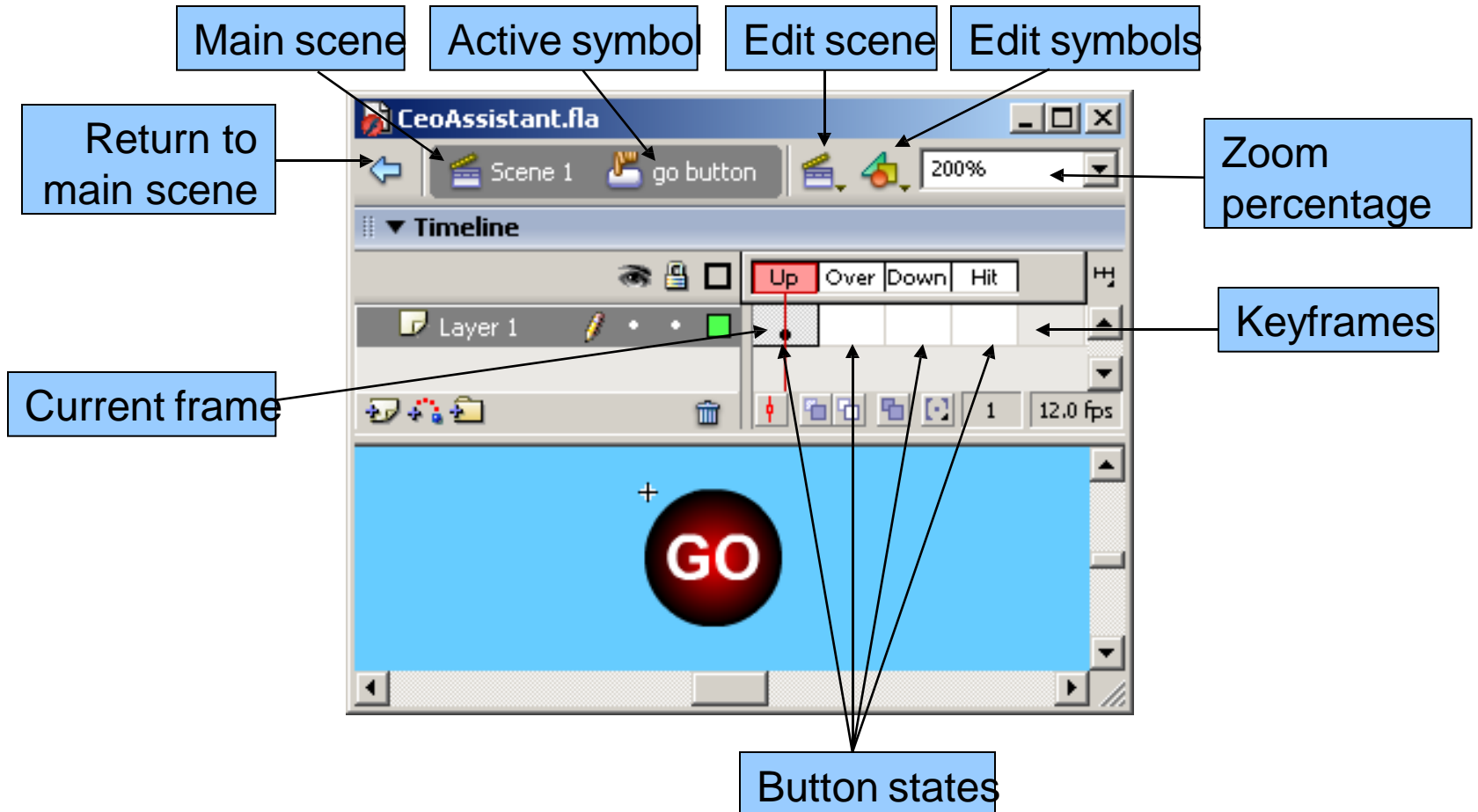


Movie Explorer for ceoassist.fla.

Editing Button Symbols

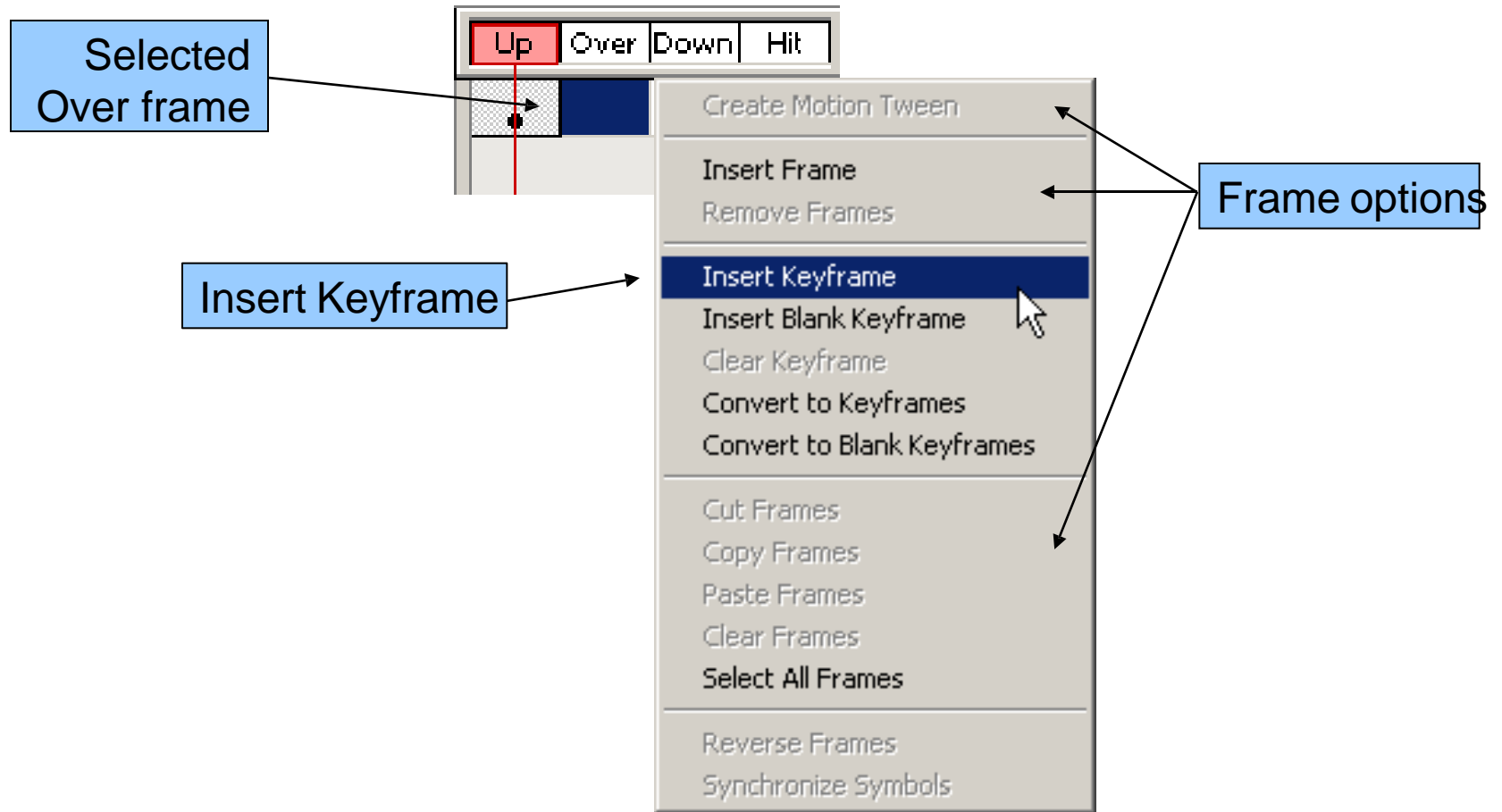
- **Edit Symbols** button
 - Four frames
 - Button states
 - **Up** state
 - » Default state before user presses the button or rolls over with mouse
 - **Over** state
 - » User rolls over the button with mouse
 - **Down** state
 - » Plays when user presses a button
 - **Hit** state
 - Not visible when viewing the movie
 - Defines active area of the button

Editing Button Symbols



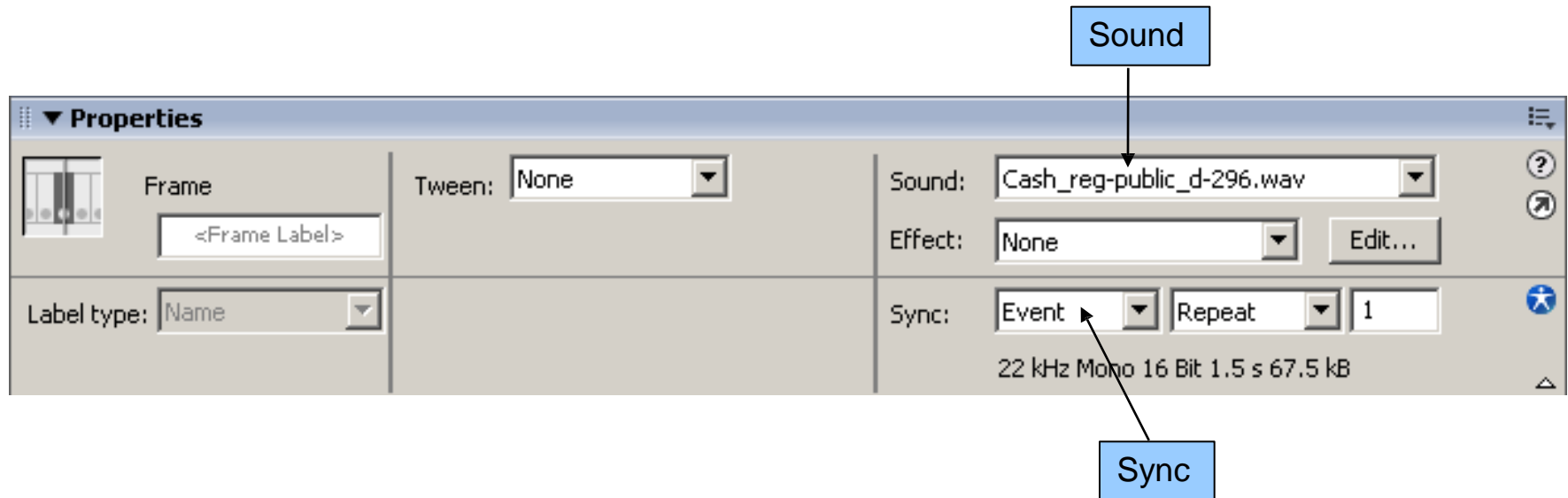
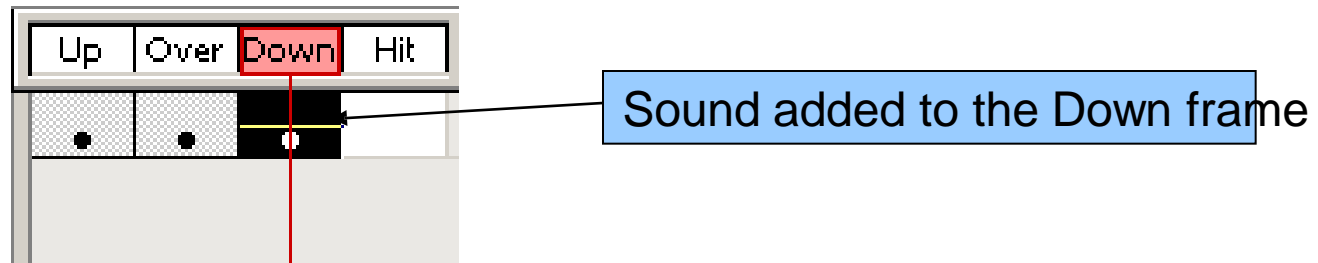
Modifying button states with a button's editing stage.

Adding Key frames



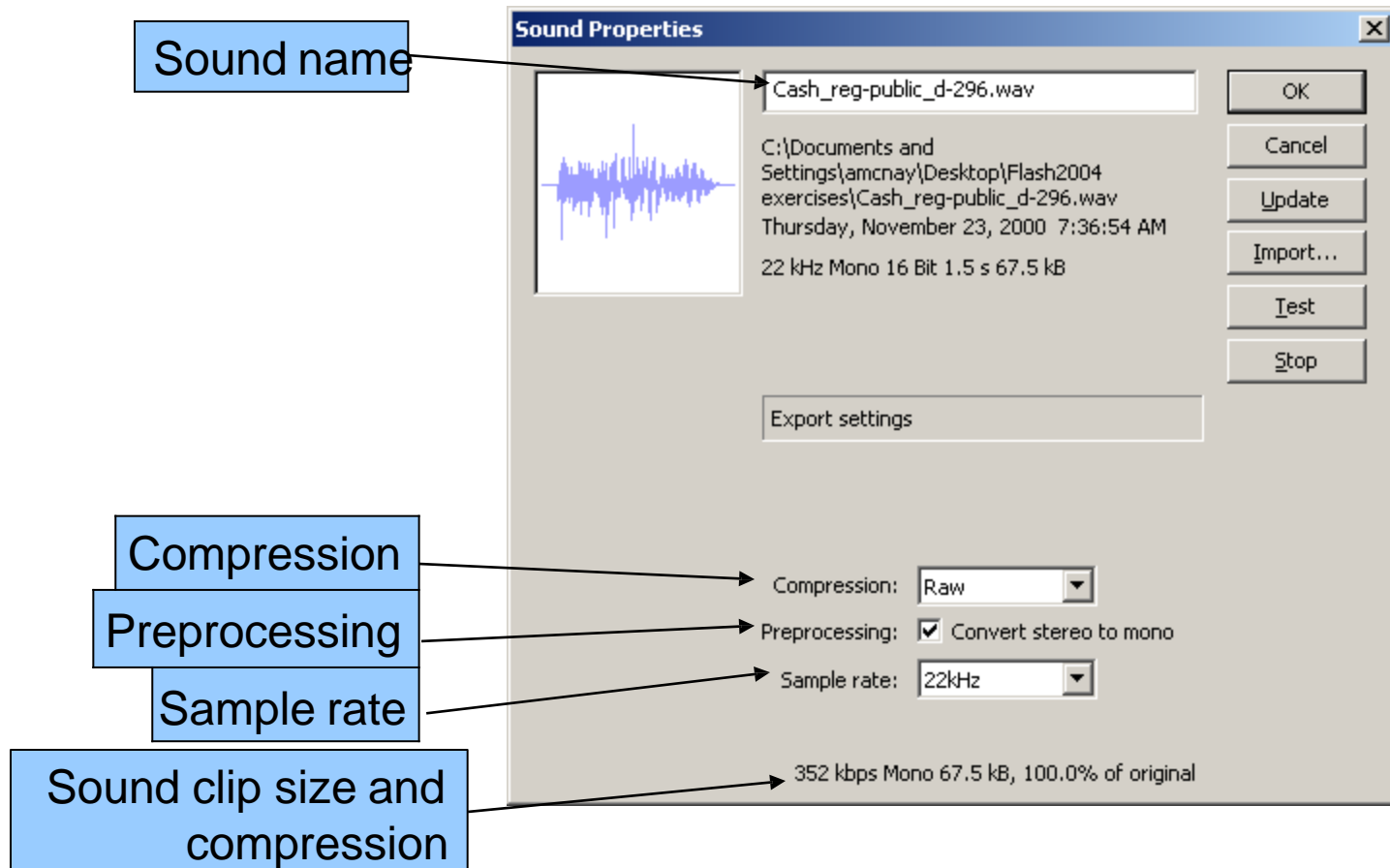
Inserting a key frame.

Adding Sound to a Button



Adding sound to a button.

Adding Sound to a Button



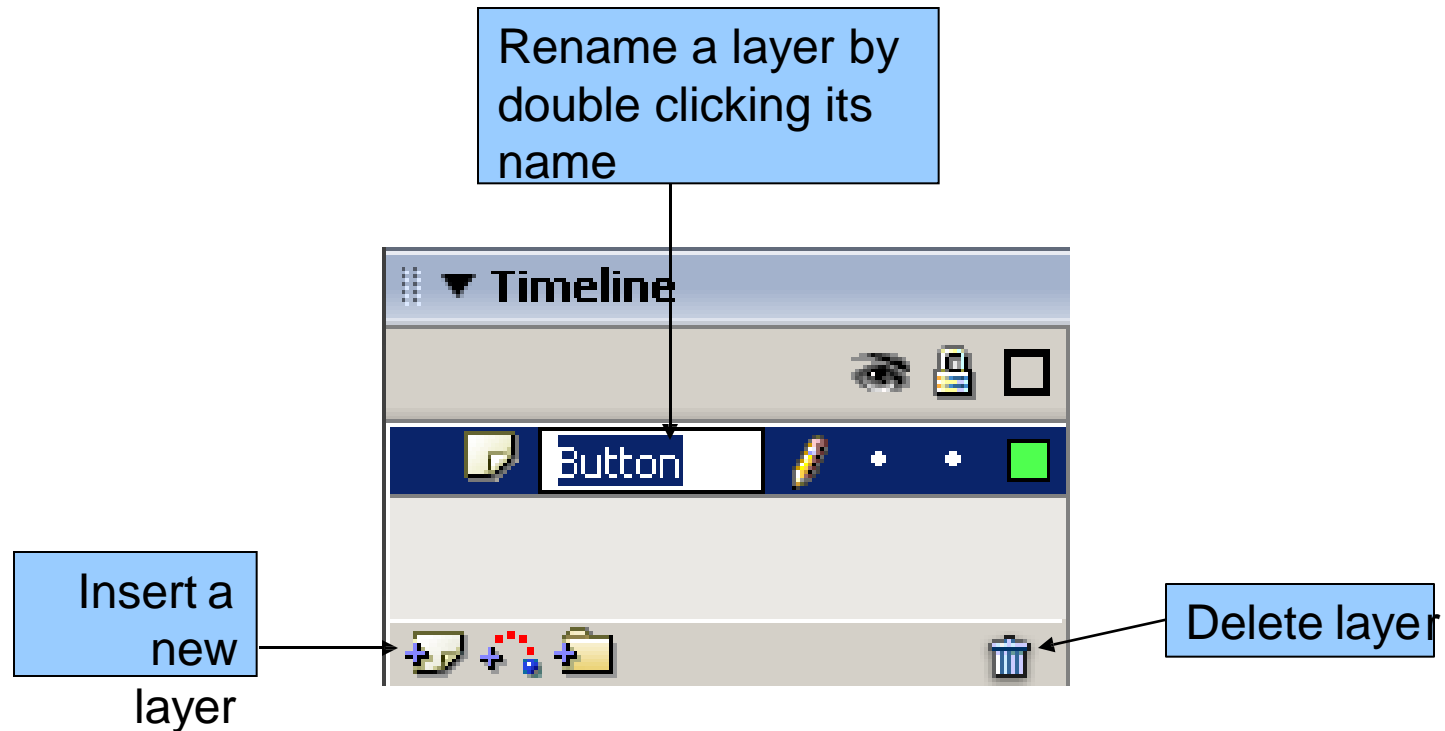
Optimizing sound with the Sound Properties dialog.

Verifying Changes with Test Movie



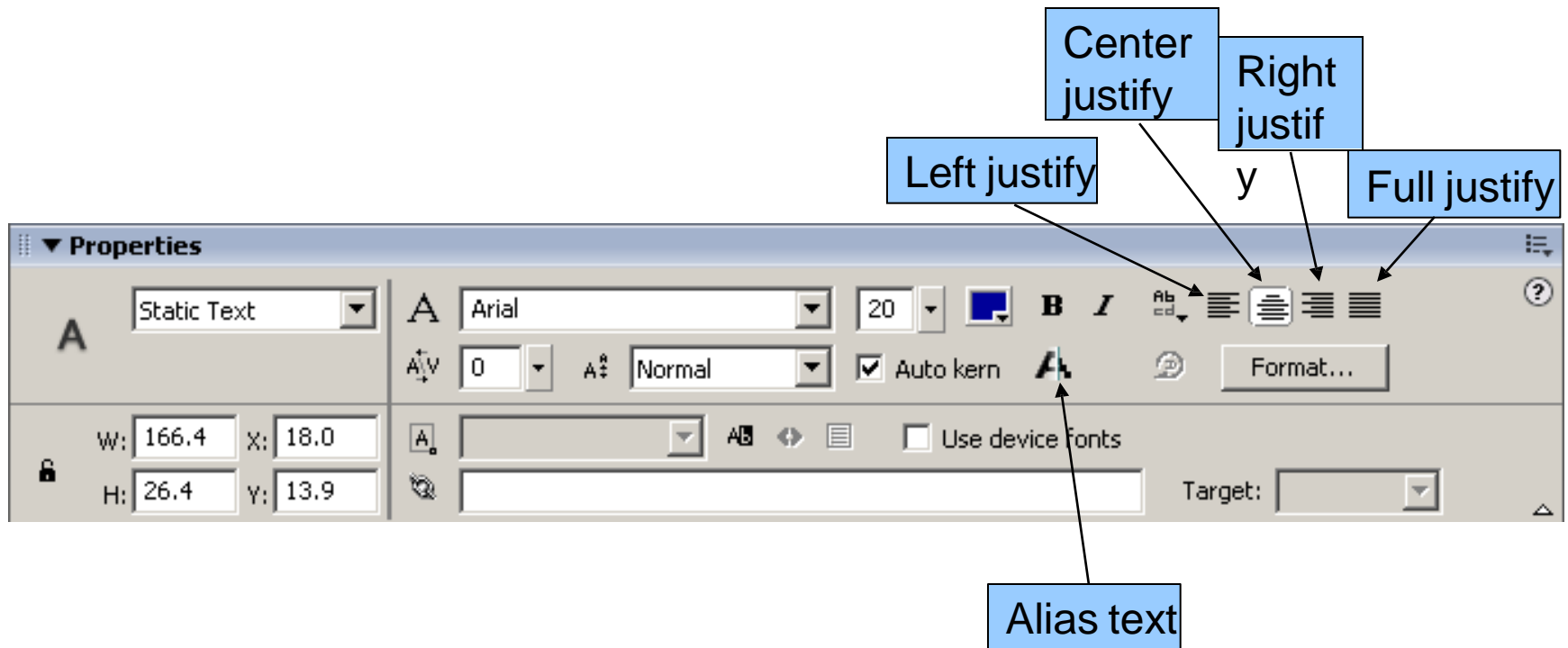
GO button in its up and over states.

Adding Layers to a Movie



Renaming a layer.

Adding Layers to a Movie

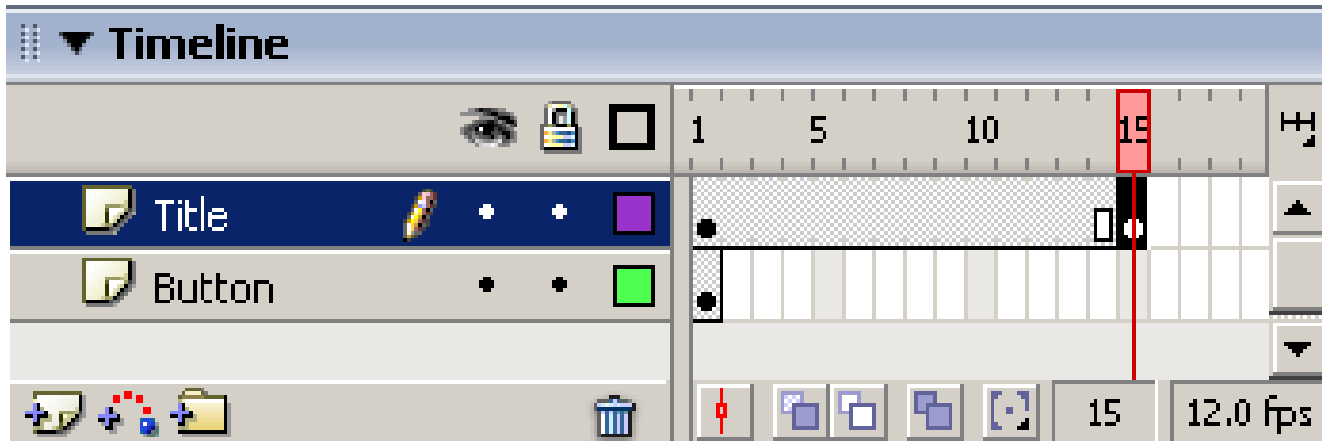


Setting text alignment with the Property Inspector.

Animating Text with Tweening

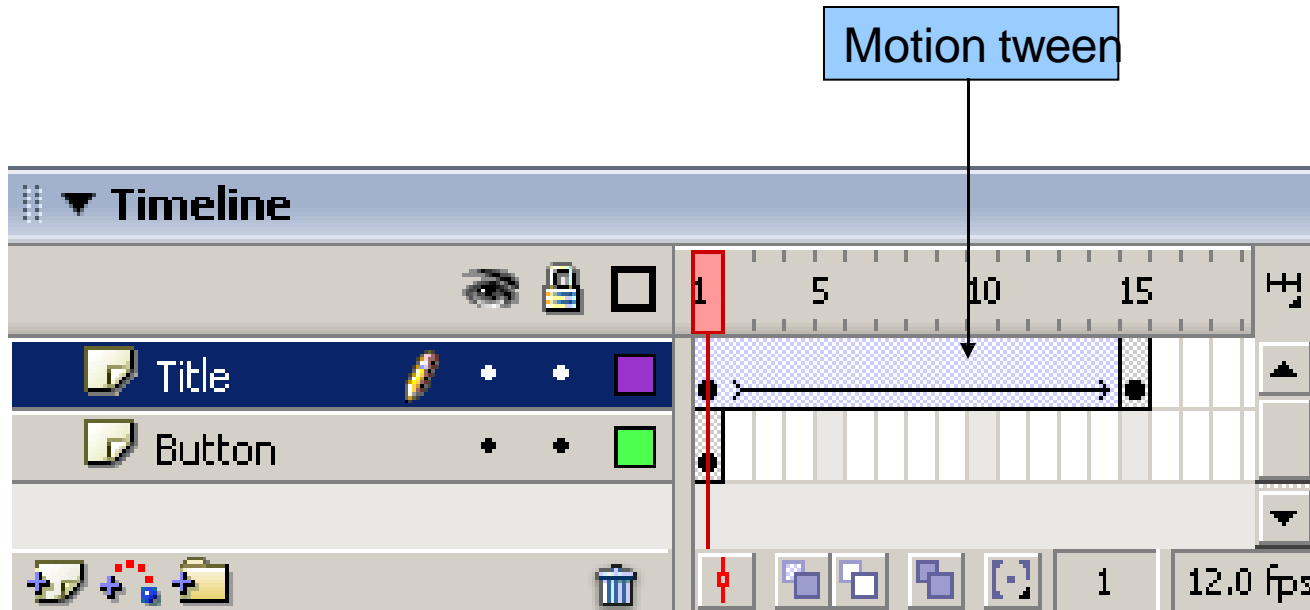
- Two methods to tween objects
 - Shape tweening
 - Morphs an object from one shape to another
 - Motion tweening
 - Moves objects around the stage
 - Can be applied to symbols or grouped objects

Animating Text with Tweening



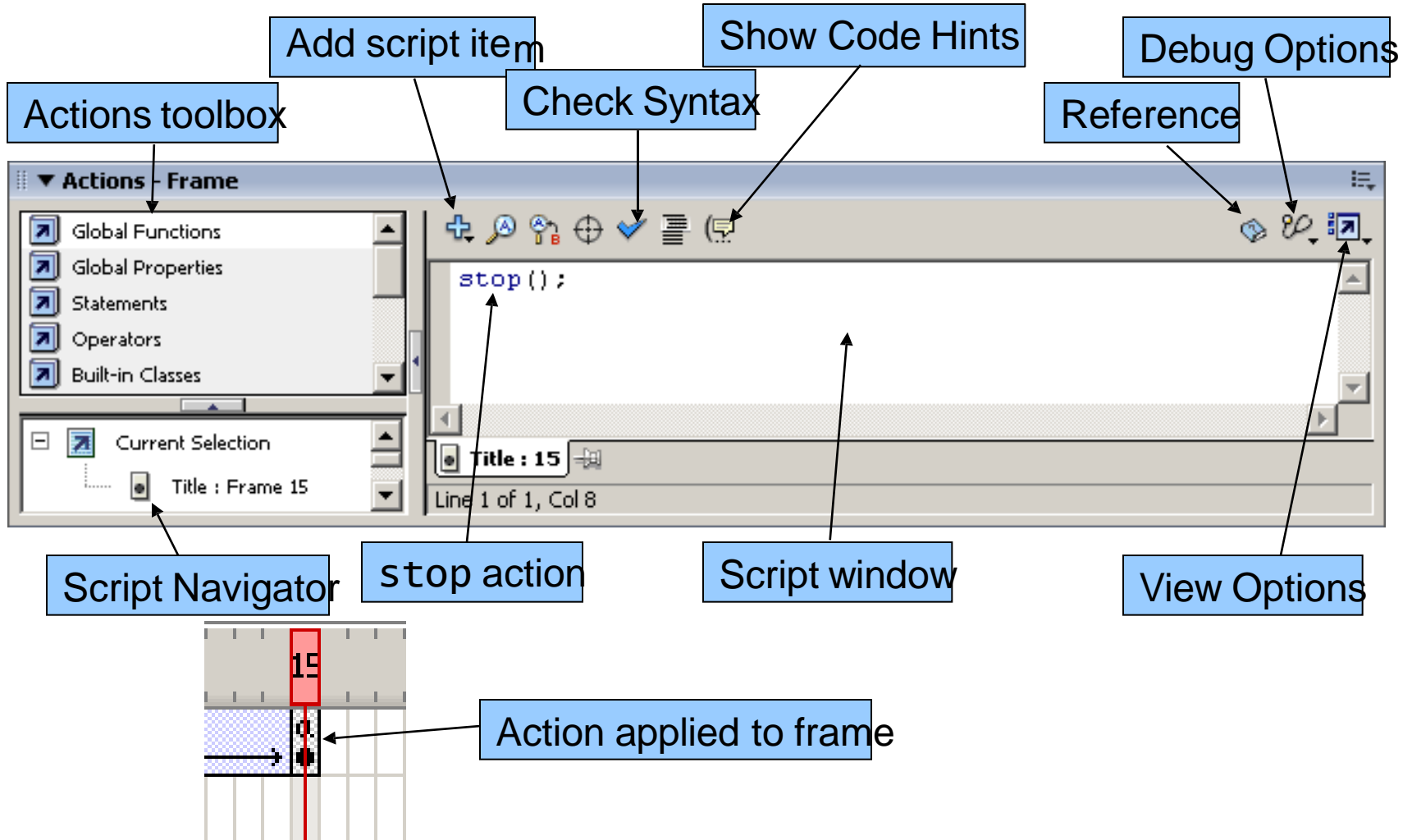
Adding a keyframe to create an animation.

Animating Text with Tweening



Creating a motion tween.

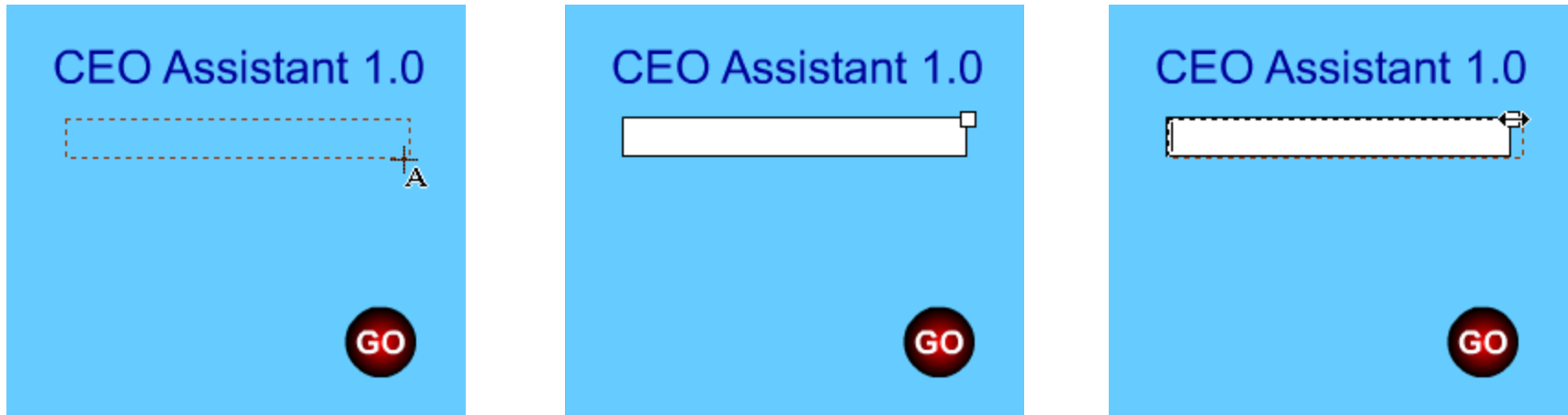
Animating Text with Tweening



Adding ActionScript to a frame with the Actions panel.

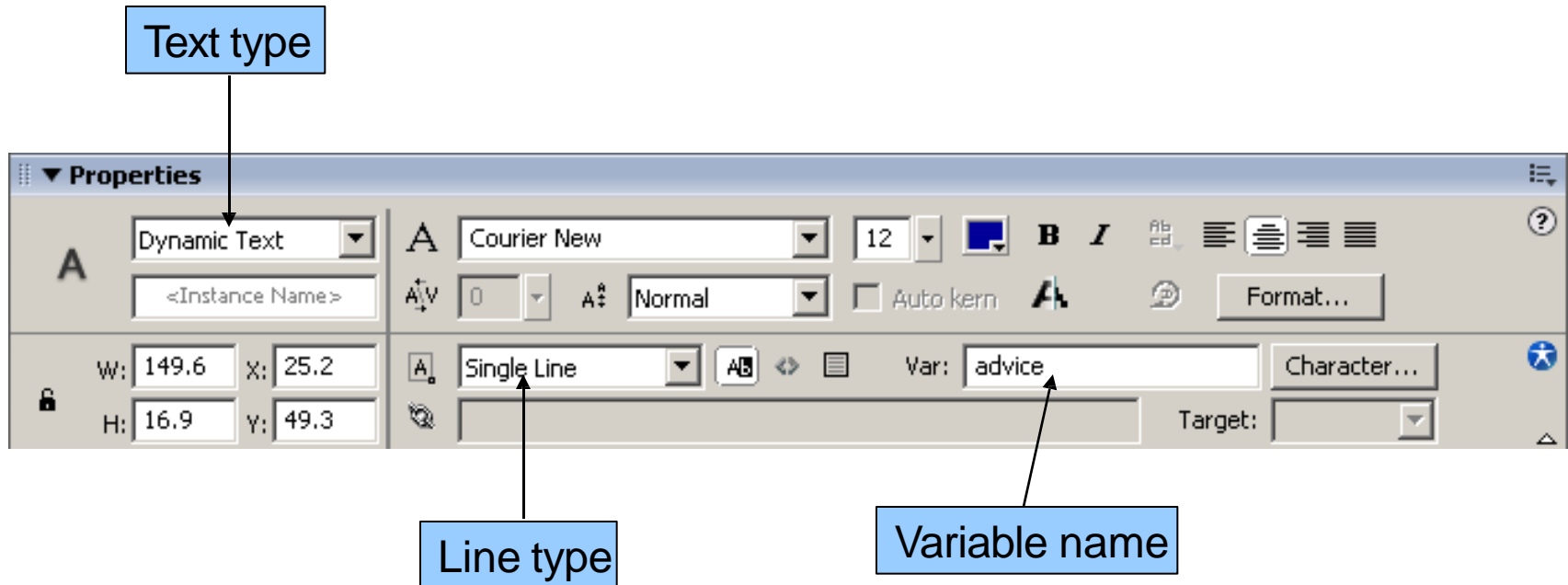
Adding a Text Field

Text field -Contains a string that changes every time the user presses the button



Creating a text field.

Adding a Text Field

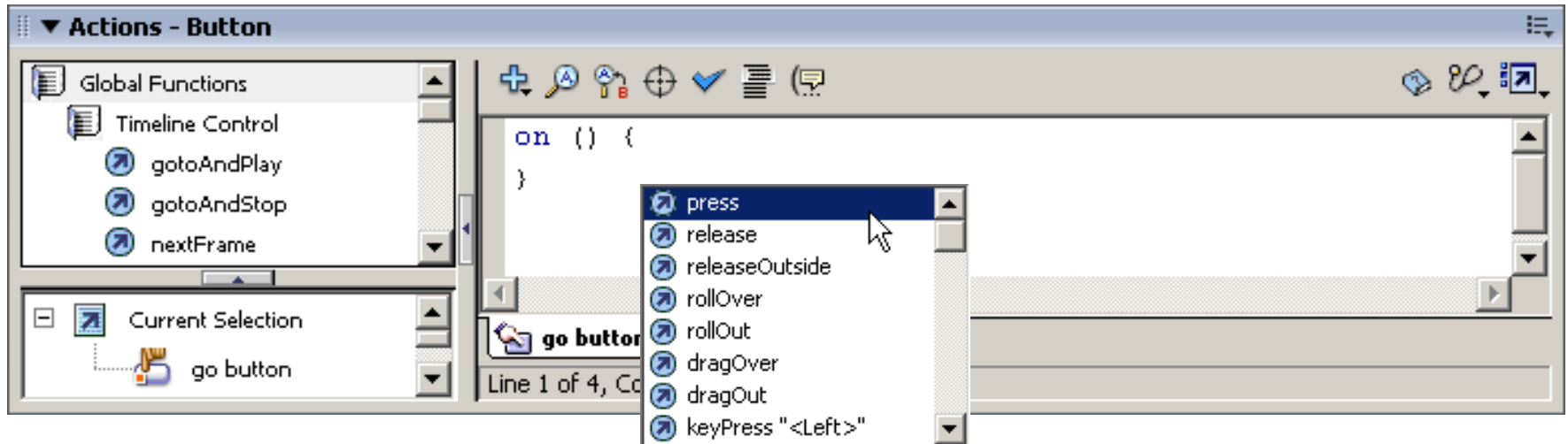


Creating a dynamic text field with the Property Inspector.

Adding Action Script

Add Action Script to the button

Change the contents of the text field every time a user clicks the button

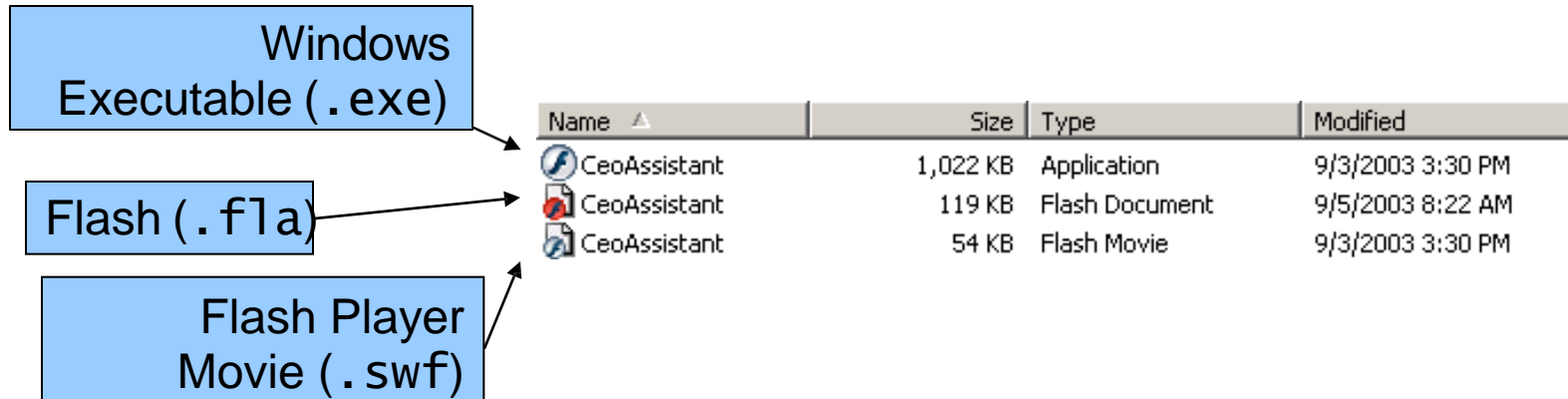


Adding an action to a button with the Actions panel.

Creating a Projector (.exe) File with Publish

- Publish Flash in two formats
 - Select **Publish Settings...** from **File** menu to open the **Publish Settings** dialog
 - Select **Flash** and **Windows Projector** checkboxes and uncheck all others
 - Click **OK** to enable the new settings
 - Select **Publish** from **File** menu

Creating a Projector (.exe) File with Publish



Published Flash files.

Manually Embedding a Flash Movie in a Web Page

- Add Flash movies to Web sites
 - object
 - Makes movie viewable in Internet Explorer
 - embed
 - Makes movie viewable in Netscape

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5  <!-- Embedding a Flash movie into a Web site.: embedFlash.html
6  <!-- Embedding a Flash movie into a Web site -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9
10     <head>
11         <title>Adding Flash to your web site</title>
12     </head>
13
14     <body>
15
16         <!-- The following object tag tells the      -->
17         <!-- Microsoft Internet Explorer browser to -->
18         <!-- play the Flash movie and where to find -->
19         <!-- the Flash Player plug-in if it is not   -->
20         <!-- installed.                             -->
21
```



```
22 <object classid =
23     "clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
24     codebase = "http://download.macromedia.com/pub/shockwave/
25     cabs/flash/swflash.cab#version=7,0,0,0">
26     <param name = "movie" value = "CeoAssistant.swf" />
27
28     <!-- The following embed tag tells the Netscape -->
29     <!-- browser to play the Flash movie and where -->
30     <!-- to find the Flash Player plug-in if it is -->
31     <!-- not installed. -->
32
33     <embed src = "CeoAssistant.swf" pluginspage =
34         "http://www.macromedia.com/go/getflashplayer">
35     </embed>
36
37     <!-- Non-Flash viewing page content -->
38     <noembed>
39         This web site contains the CEO Assistant 1.0
40         Flash movie. You must have the Flash Player
41         plug-in to view the Flash movie.
42     </noembed>
43
44 </object>
45
46 </body>
47 </html>
```

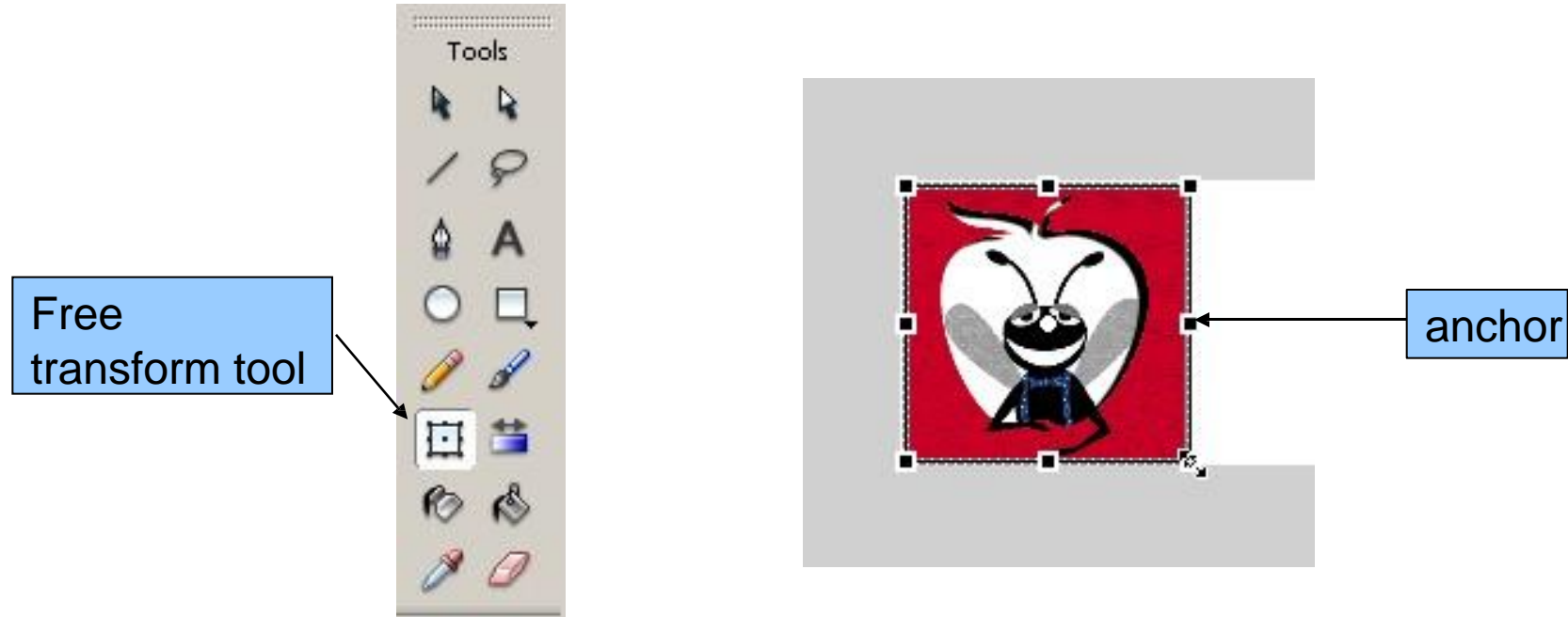
Importing and Manipulating Bitmaps

- Lasso tool
 - Selects areas of shapes
 - Magic wand
 - Selects areas of similar colors
 - Polygonal mode
 - Selects straight-edged areas
- Eraser tool
 - Removes shape areas by clicking and dragging across
- Paintbrush tool
 - Applies color the same way the eraser removes color
- Paint behind
 - Sets the tool to only paint in area void of color information
- Paint inside
 - Paints inside a line boundary

Creating an Advertisement Banner with Masking

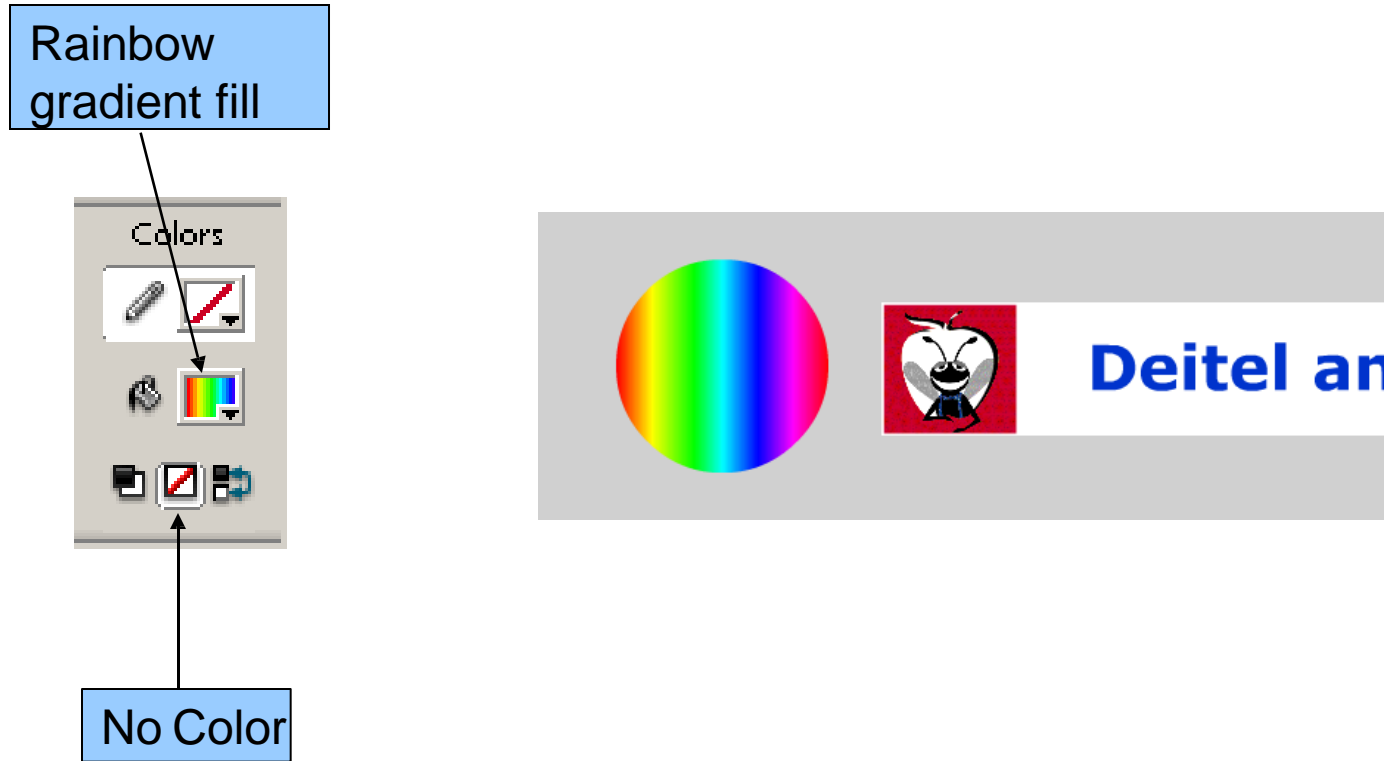
- Masking
 - Hides portion or layers
 - Masking layer
 - Hides objects in the layers beneath it

Creating an Advertisement Banner with Masking



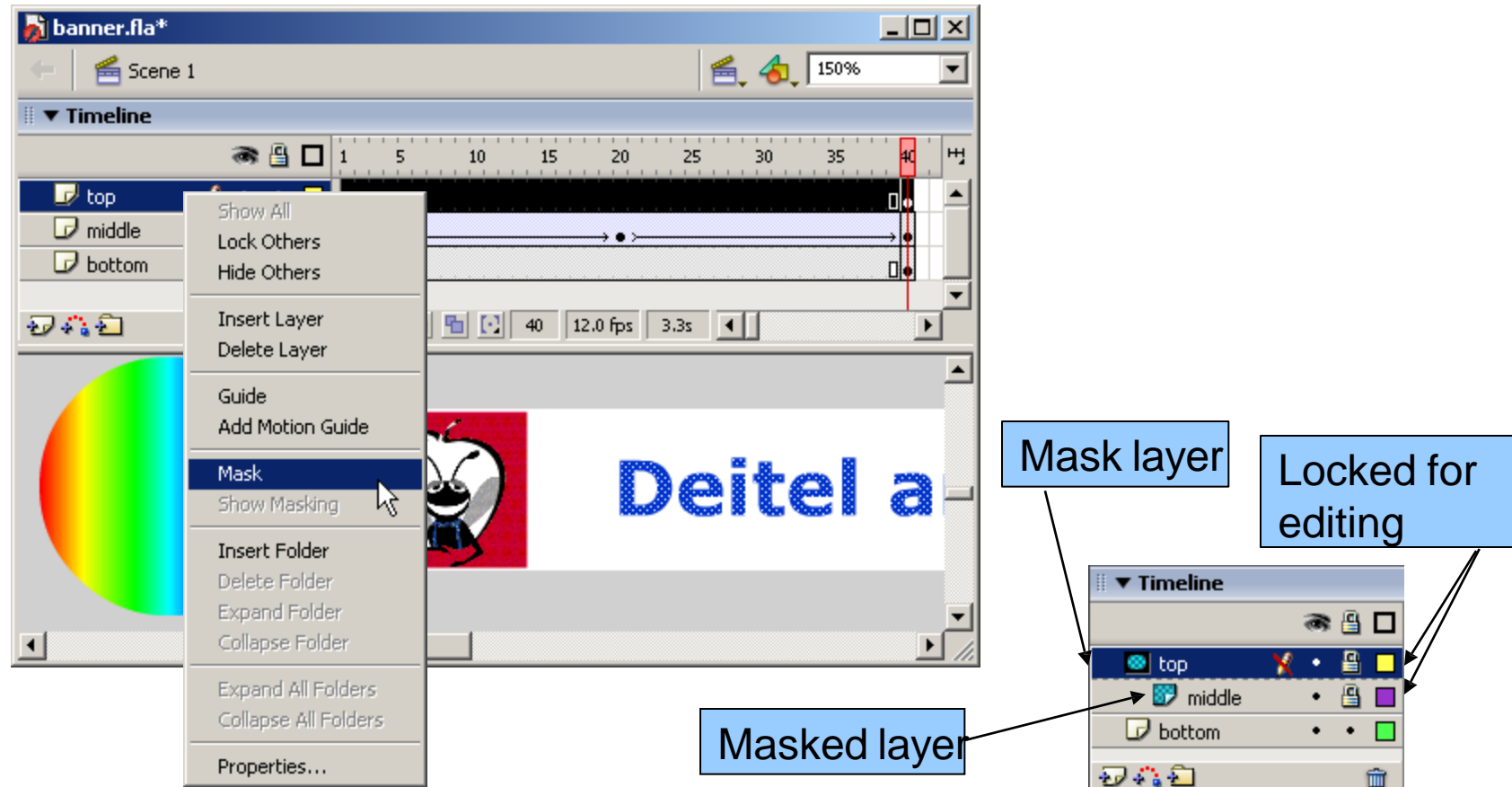
Resizing an image with the free transform tool.

Creating an Advertisement Banner with Masking



Creating the oval graphic.

Creating an Advertisement Banner with Masking



Creating a mask layer.

Adding Online Help to Forms

Bug2Bug.com

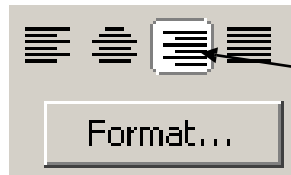
Registration Form

Name:
Member#:
Password:

Format Options

Indent: 0 px
Line spacing: 22 pt
Left margin: 0 px
Right margin: 0 px

OK
Cancel



Right justify

Line-spacing
adjustment

Adjusting the line spacing with the Format Options panel.

Adding Online Help to Forms

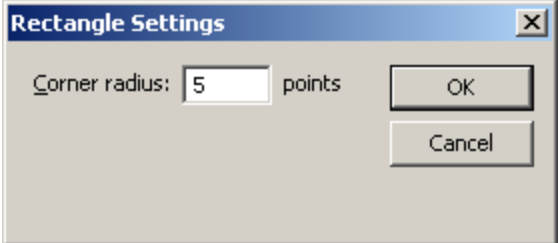
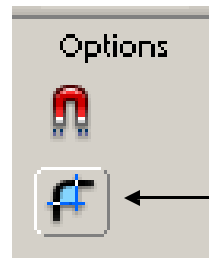
Bug2Bug.com

Registration Form

Name:

Member#:

Password:

A dialog box titled "Rectangle Settings" with a close button (X) in the top right corner. It contains a label "Corner radius:" followed by a text input field containing the number "5", and the word "points" to the right of the field. At the bottom right, there are two buttons: "OK" and "Cancel".

Round Rectangle
Radius option

Creating a rectangle with rounded corners.

Adding Online Help to Forms

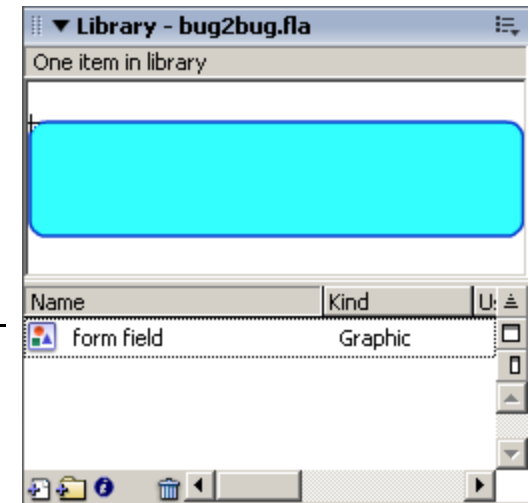
Bug2Bug.com

Registration Form

Name:

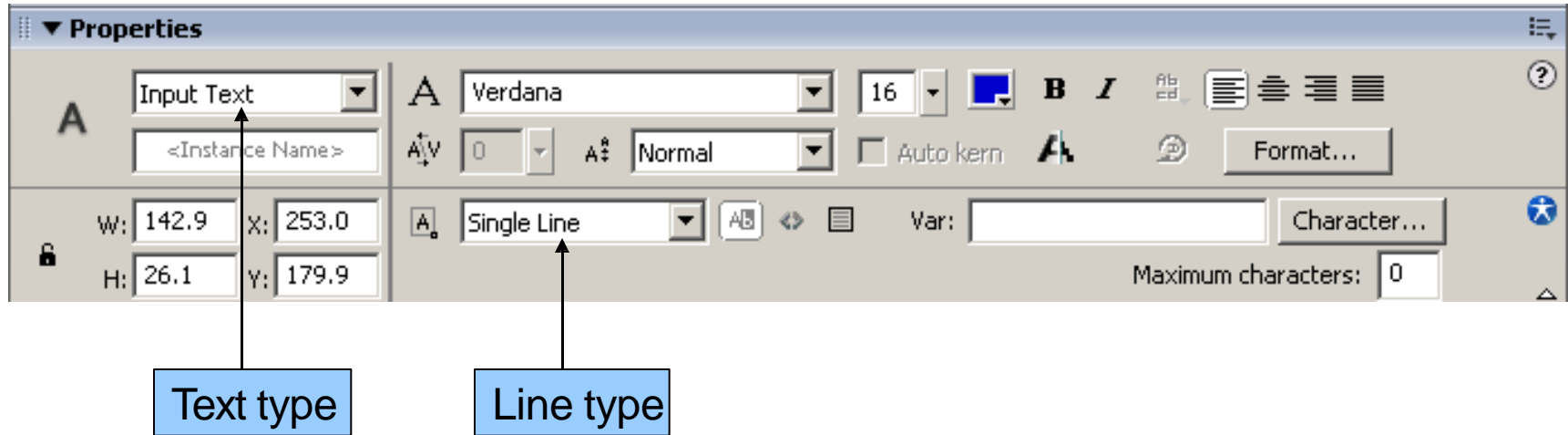
Member#:

Password:



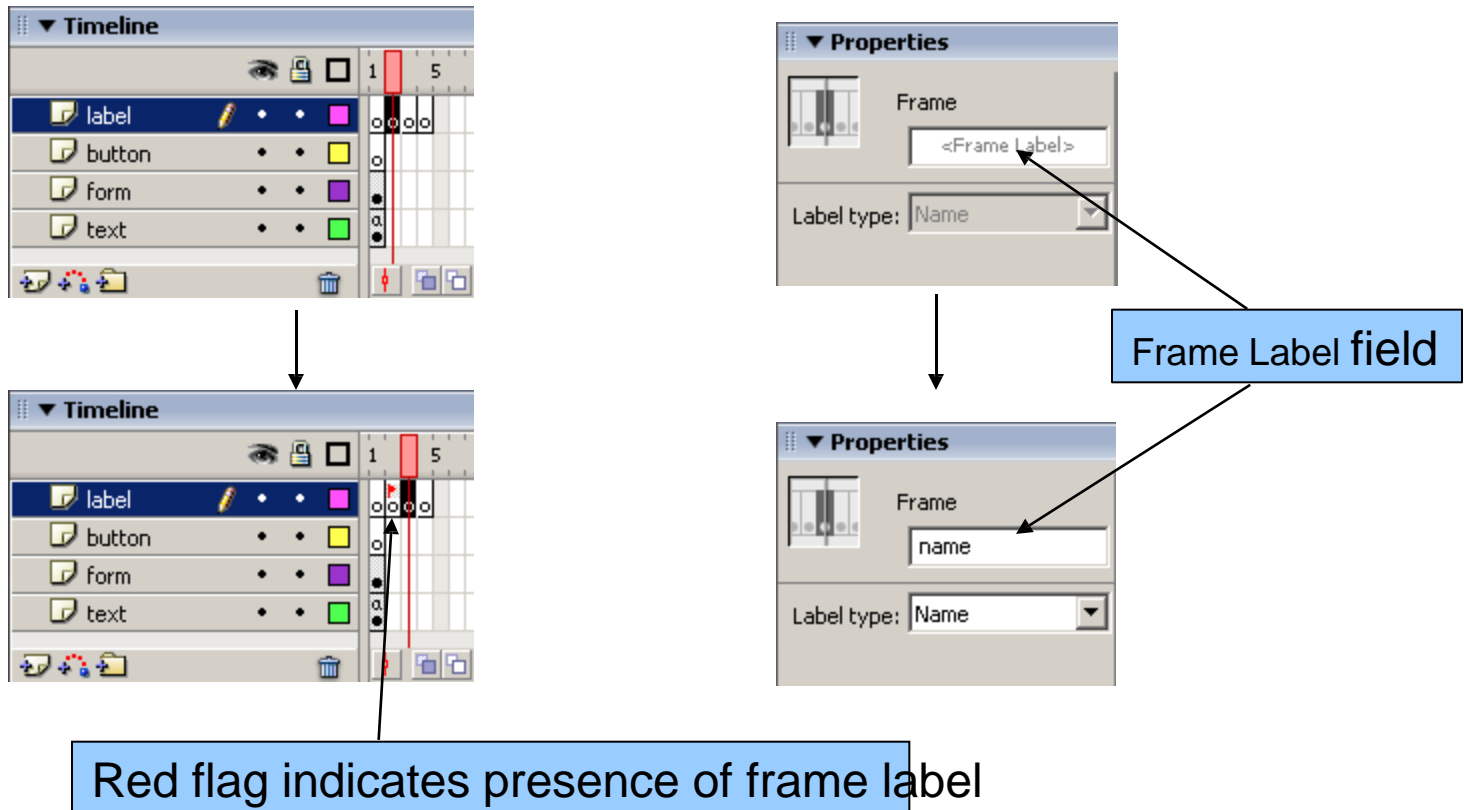
Creating multiple instances of a symbol with the Library panel.

Adding Online Help to Forms



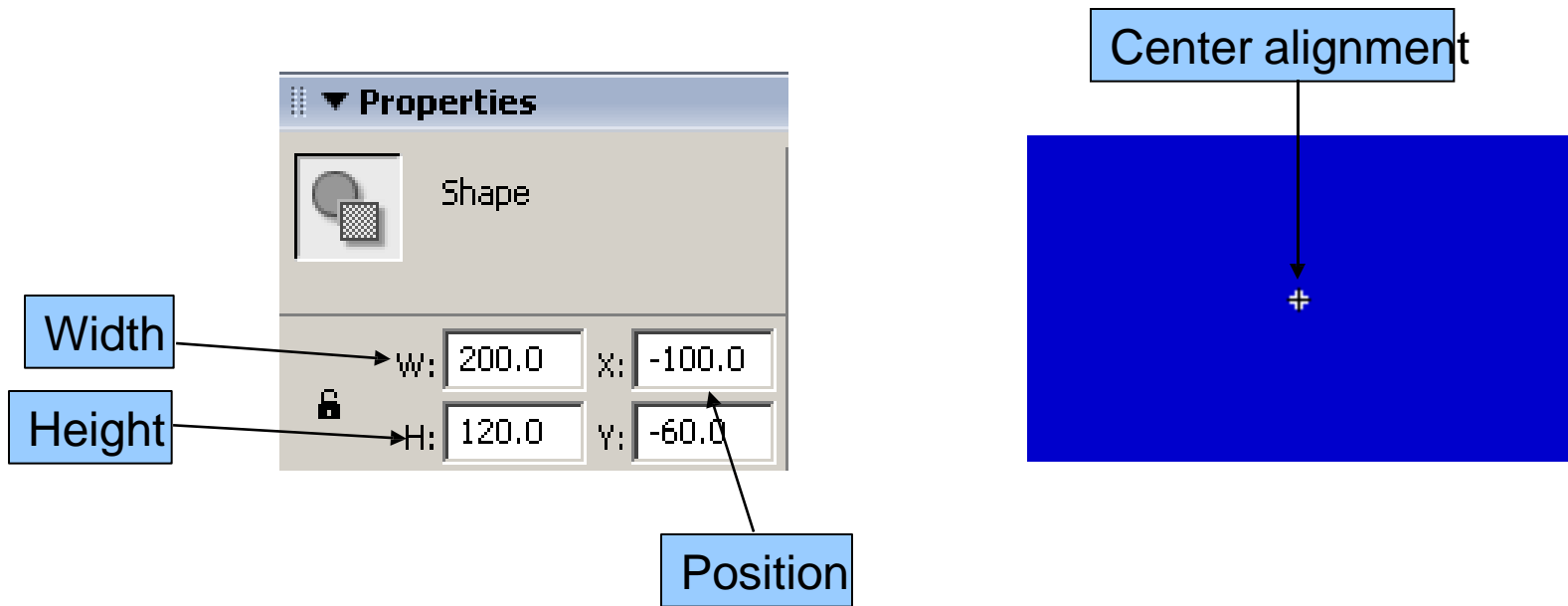
Input and password text field creation.

Adding Online Help to Forms



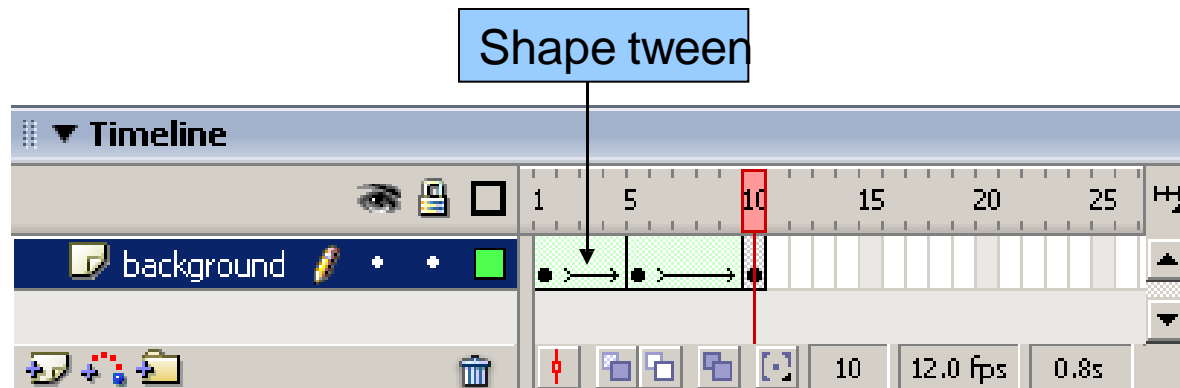
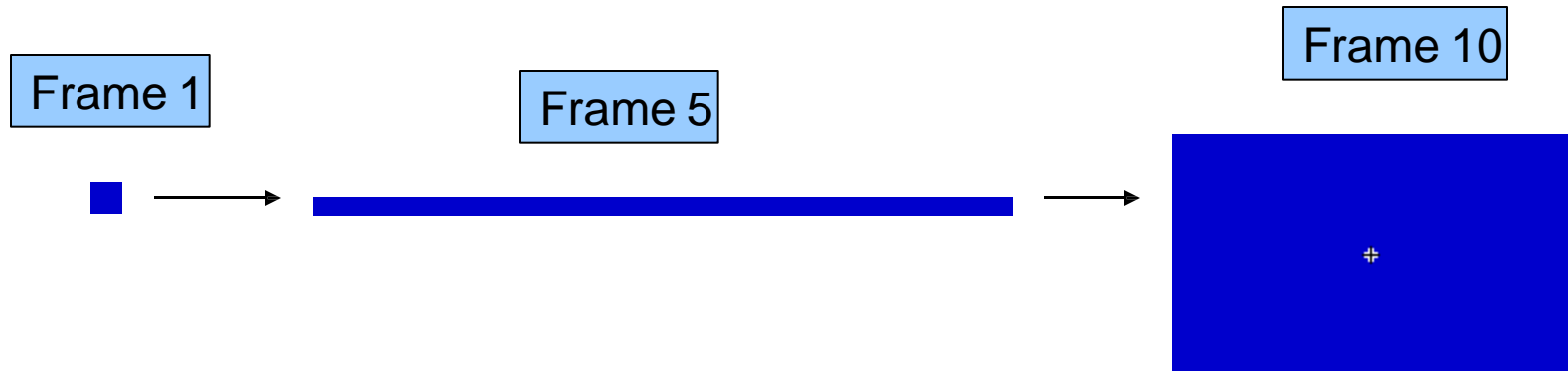
Adding Frame Labels using the Property Inspector.

Adding Online Help to Forms



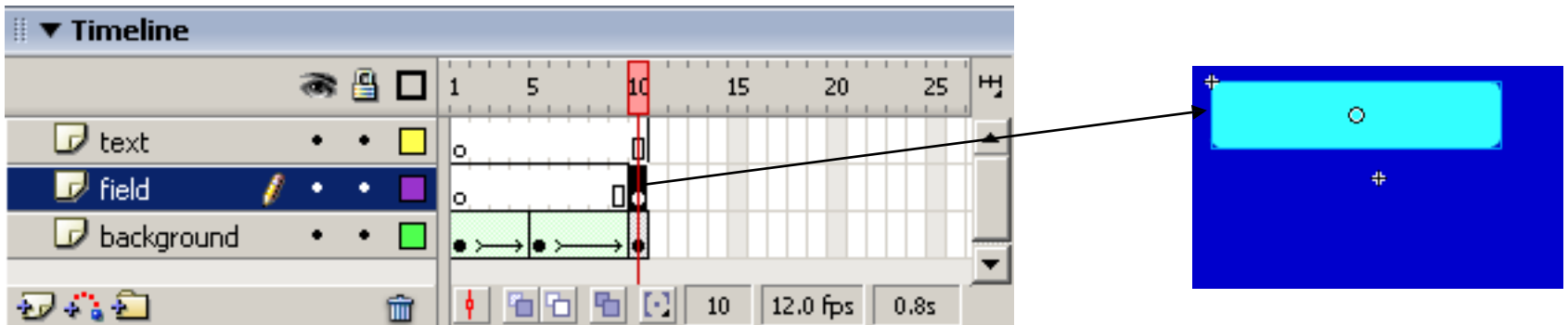
Centering an image on the stage with the Property Inspector.

Adding Online Help to Forms



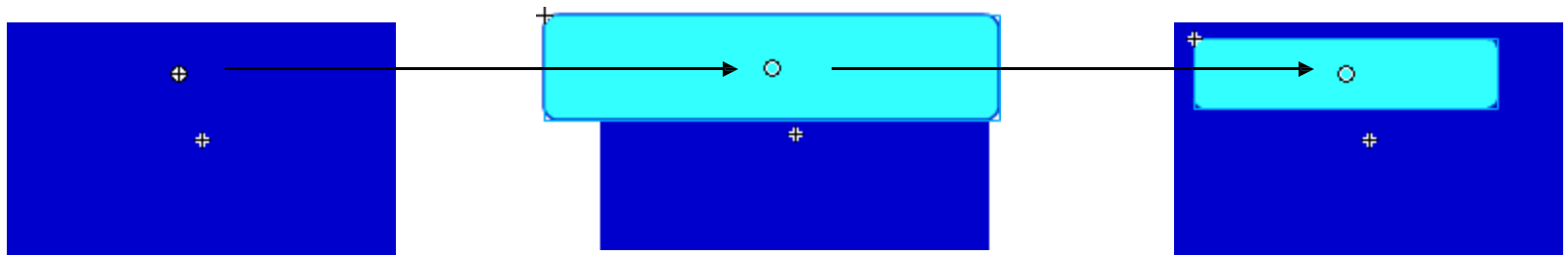
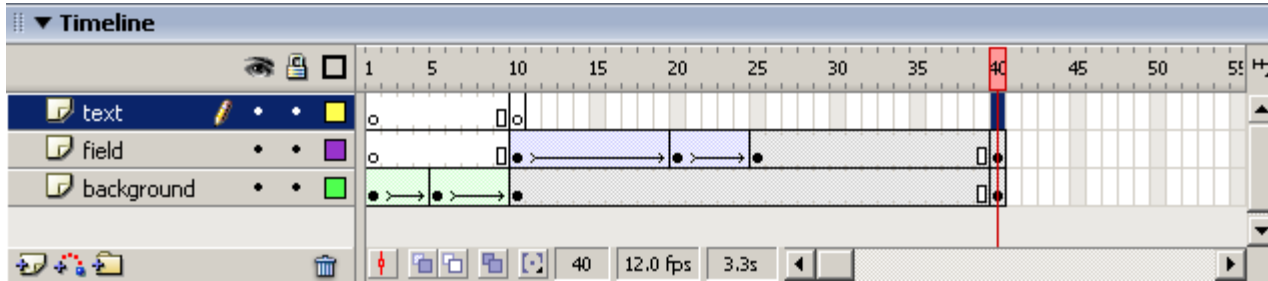
Creating a shape tween.

Adding Online Help to Forms



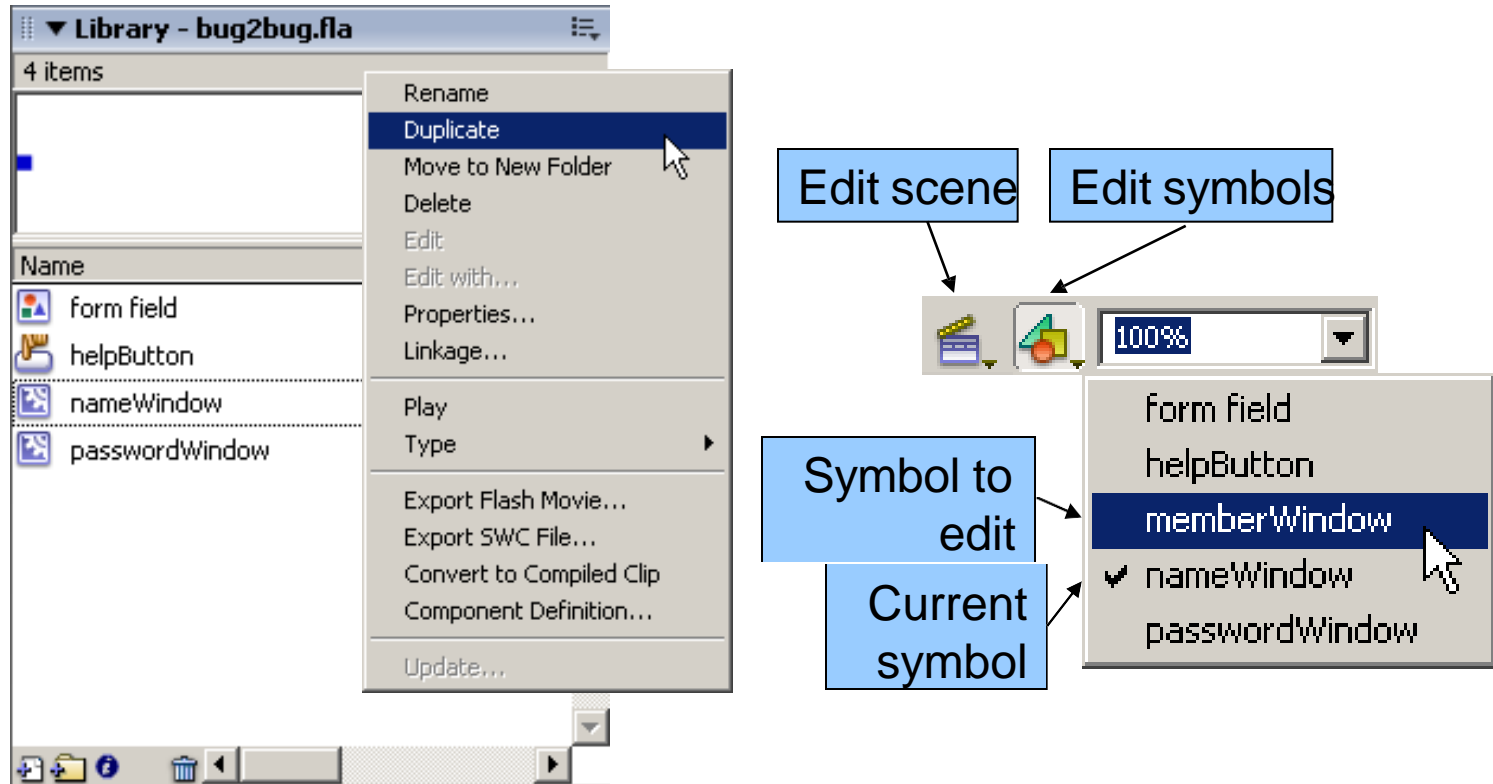
Adding the field symbol to the namewindow movie clip.

Adding Online Help to Forms



Creating an animation with the form field symbol.

Adding Online Help to Forms



Duplicating movie clip symbols with the Library panel.

Adding Online Help to Forms

The screenshot shows an animation software interface. On the left is a 'Timeline' panel with a list of layers: 'typedText' (pink), 'text' (yellow), 'field' (purple), and 'background' (green). The timeline itself has a ruler from 1 to 40 frames. A red vertical line marks frame 27. Arrows indicate the duration of each layer's animation. A callout box labeled 'Frames for animation' points to the timeline. Another callout box labeled 'Deleting a letter from each subsequent frame' points to a specific frame in the 'typedText' layer. To the right is a preview window showing a blue box with a text input field containing 'Joh' and the text 'Enter your name in this field. First name, Last name' below it.

Timeline

typedText

text

field

background

Frames for animation

Deleting a letter from each subsequent frame

Joh

Enter your name in this field. First name, Last name

Creating a frame-by-frame animation.

Adding Online Help to Forms

Bug2Bug.com

Registration Form

Name: ?

Member#: ?

Password: ?

John Doe

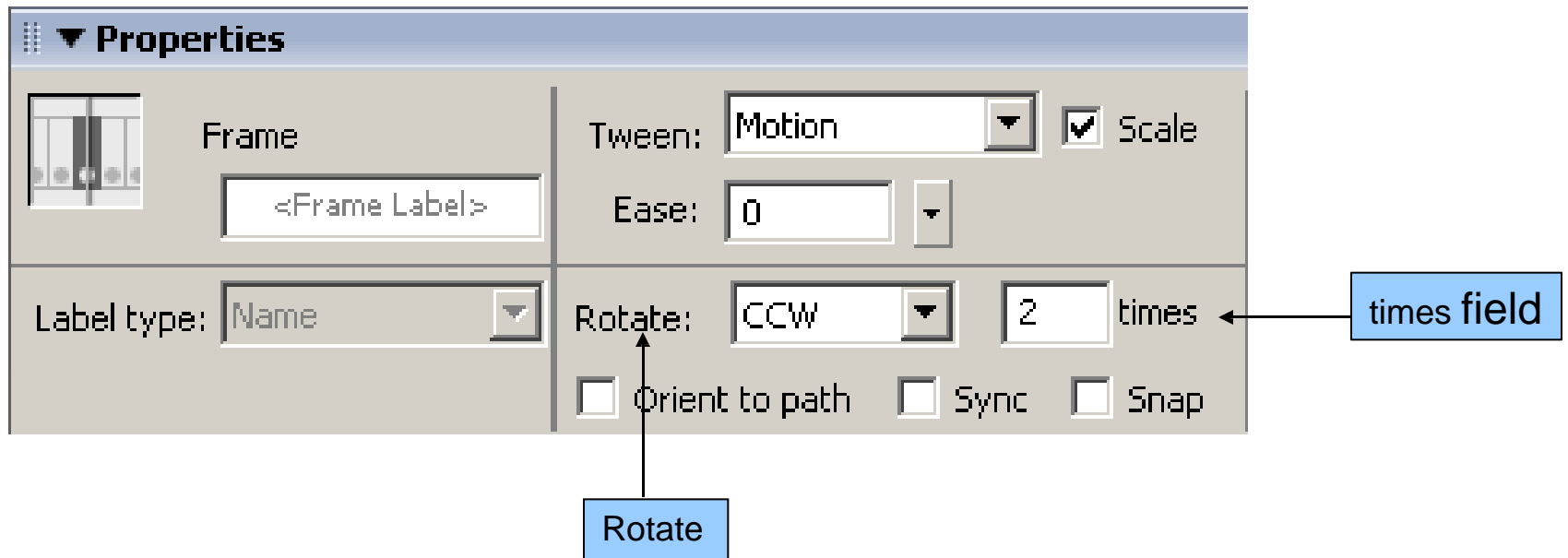
Enter your name in this field. First name, Last name

Bug2Bug . com help form.

Creating a Web-Site Introduction

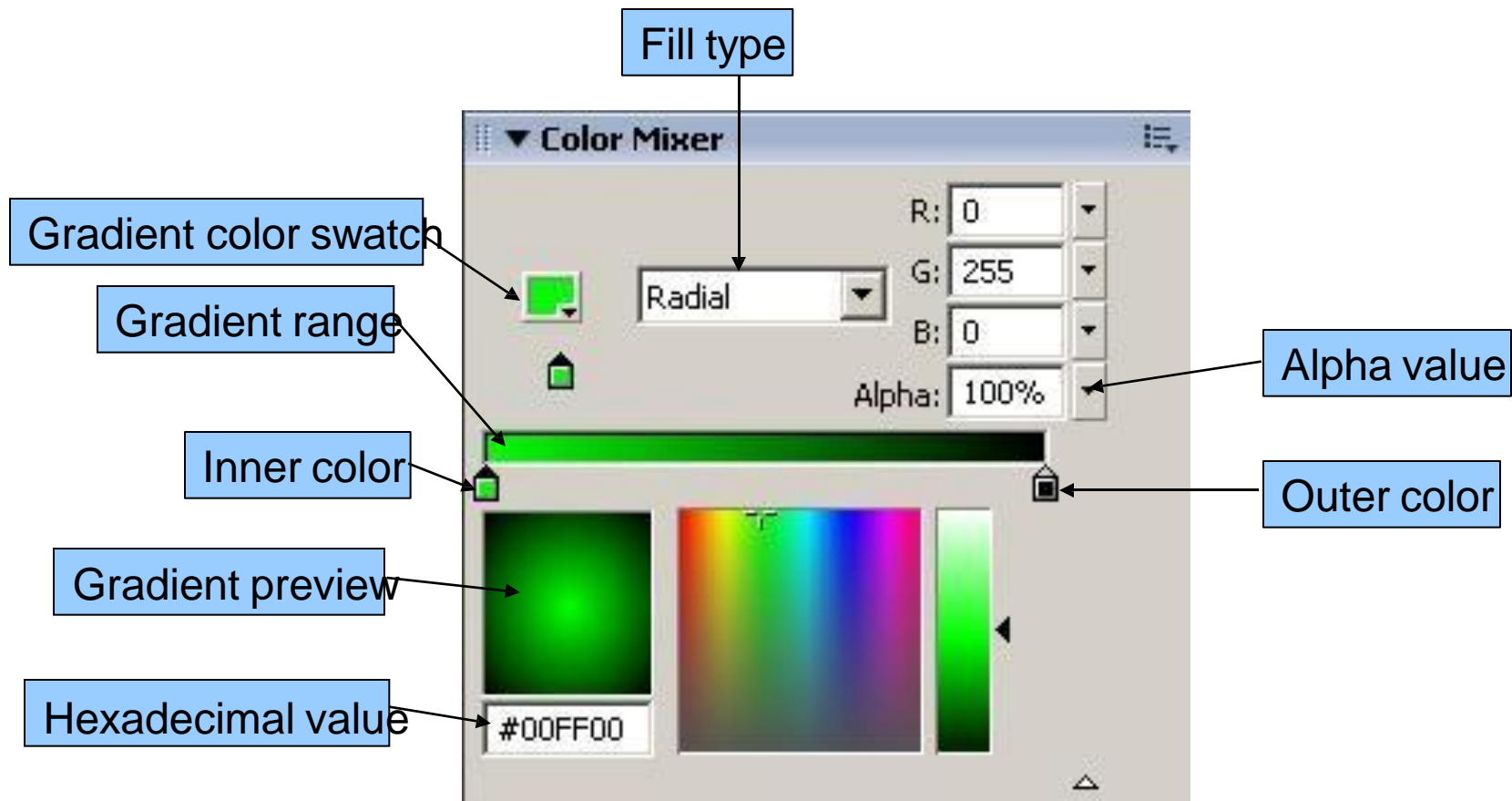
- Preloader
 - Simple animation that plays while the rest of the Web page is loading

Creating a Web-Site Introduction



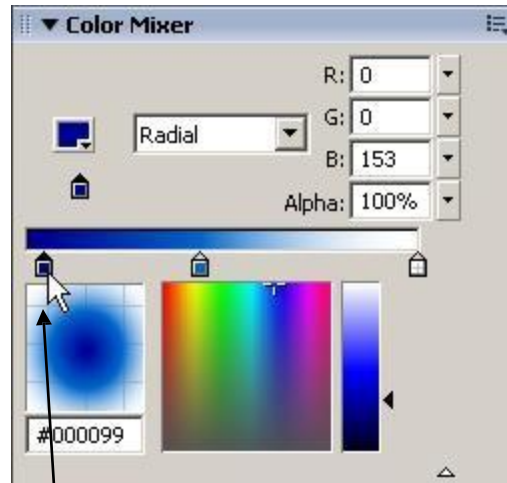
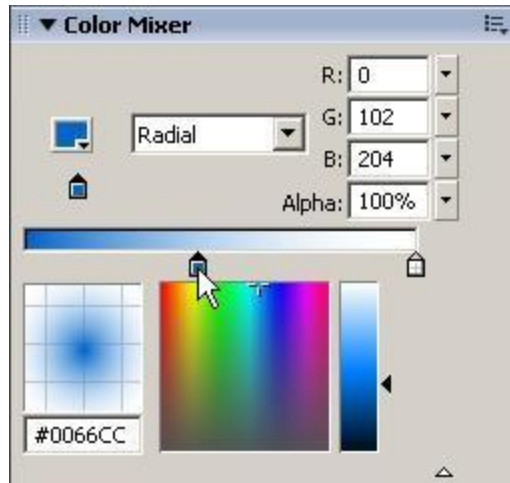
Creating a rotating object with the motion tween Rotate option.

Creating a Web-Site Introduction



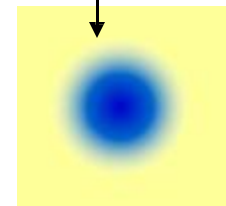
Changing gradient colors with the Color Mixer panel.

Creating a Web-Site Introduction



Click and drag to add or remove a color

Resulting gradient



Adding an intermediate color to a gradient.

Creating a Web-Site Introduction

A rectangular button with a yellow background and a thin blue border. The text "skip directly to the Deitel Web site" is centered in blue. A small black crosshair cursor is positioned over the text.

skip directly to the Deitel Web site

Up state

A rectangular button with a yellow background and a thin blue border. The text "skip directly to the Deitel Web site" is centered in blue. A small black crosshair cursor is positioned over the text.

skip directly to the Deitel Web site

Hit state

Defining the hit area of a button.

Creating a Web-Site Introduction



Creating an animation to preload images.

Action Script

- With the following functions, you can build some fairly complex Flash movies

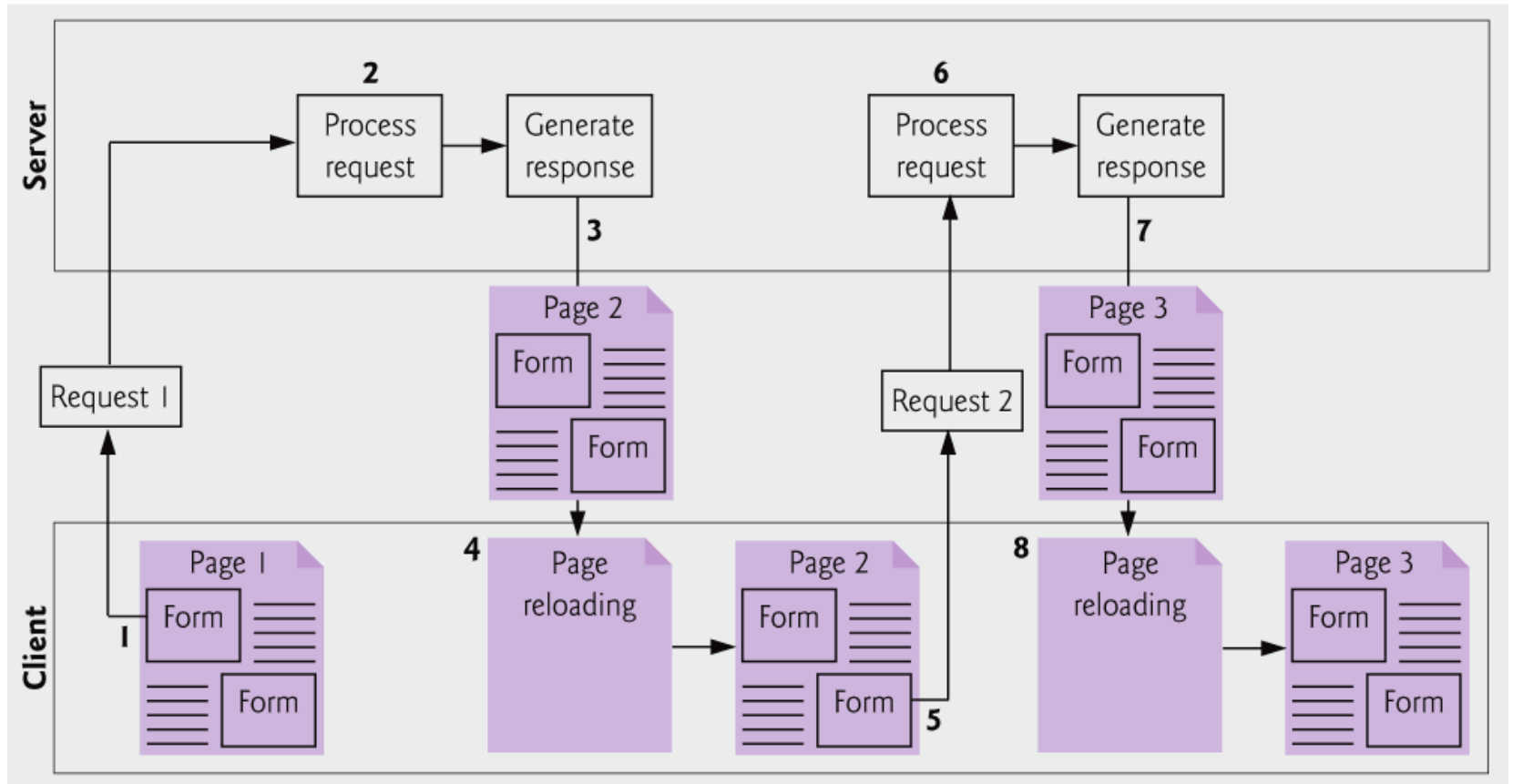
UNIT-V

Introduction

- Portability issues
 - Hidden by Ajax toolkits, such as Dojo, Prototype and Script.aculo.us
 - Toolkits provide powerful ready-to-use controls and functions that enrich web applications and simplify JavaScript coding by making it cross-browser compatible
- Achieve rich GUI in RIAs with
 - Ajax toolkits
 - RIA environments such as Adobe's Flex, Microsoft's Silverlight and Java Server Faces
 - Such toolkits and environments provide powerful ready-to-use controls and functions that enrich web applications.
- Client-side of Ajax applications
 - Written in XHTML and CSS
 - Uses JavaScript to add functionality to the user interface
- XML and JSON are used to structure the data passed between the server and the client
- XML Http Request
 - The Ajax component that manages interaction with the server

Traditional Web Applications vs. Ajax Applications

- Traditional web applications
 - User fills in the form's fields, then submits the form
 - Browser generates a request to the server, which receives the request and processes it
 - Server generates and sends a response containing the exact page that the browser will render
 - Browser loads the new page and temporarily makes the browser window blank
 - Client *waits* for the server to respond and *reloads the entire page* with the data from the response
- While a synchronous request is being processed on the server, the user cannot interact with the client web browser
- The synchronous model was originally designed for a web of hypertext documents
 - some people called it the “brochure web”
 - model yielded “choppy” application performance

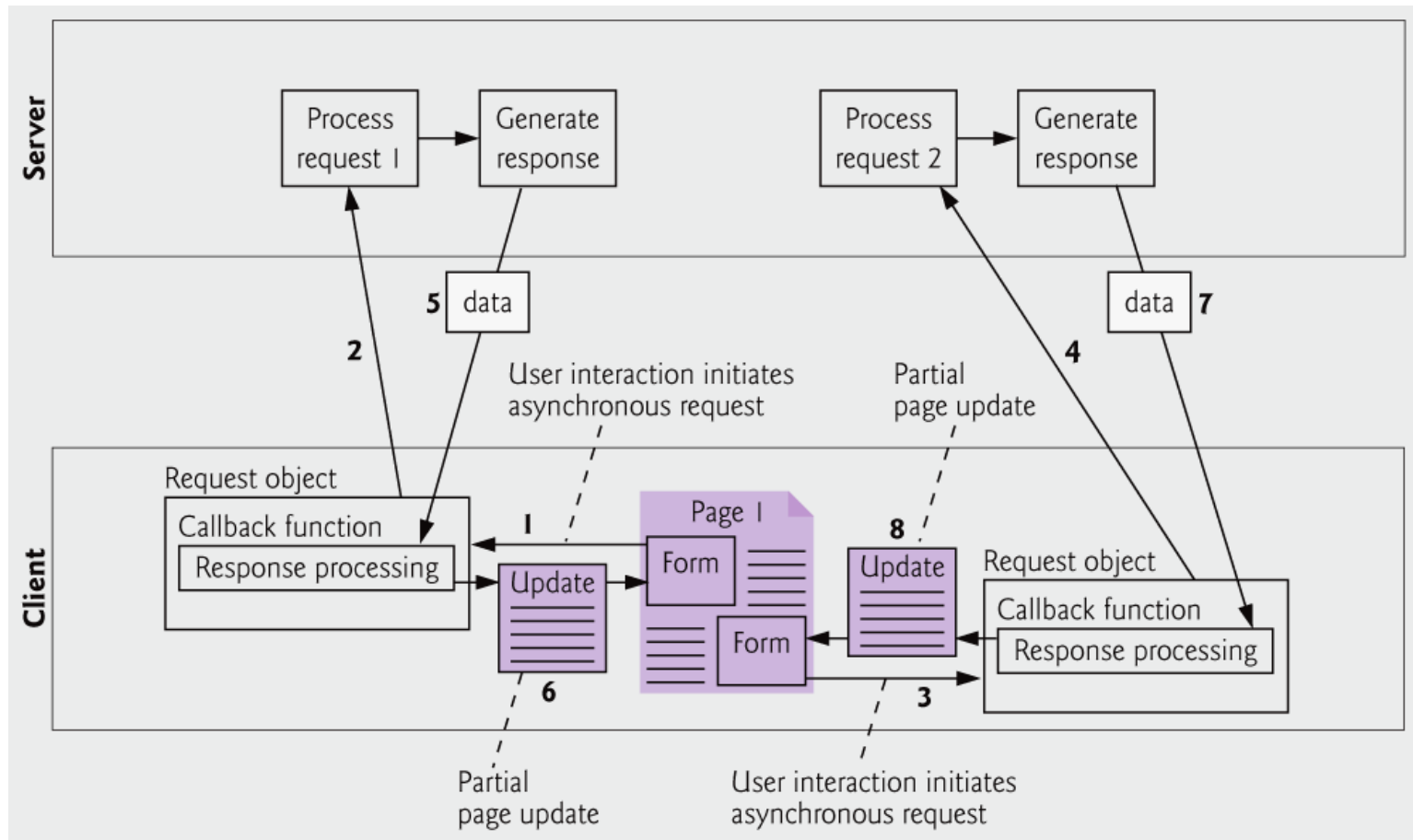


Classic web application reloading the page for every user interaction.

Traditional Web Applications Vs. Ajax Applications

(Cont.)

- Request object sends the request to and awaits the response from the server
 - Requests are asynchronous, allowing the user to continue interacting with the application while the server processes the request concurrently
 - When the server responds, the XMLHttpRequest object that issued the request invokes a callback function, which typically uses partial page updates to display the returned data in the existing web page *without reloading the entire page*
- Callback function updates only a designated part of the page
- Partial page updates help make web applications more responsive, making them feel more like desktop applications



Ajax-enabled web application interacting with the server asynchronously.

Rich Internet Applications (RIAs) with Ajax

- Classic XHTML registration form
 - Sends all of the data to be validated to the server when the user clicks the Register button
 - While the server is validating the data, the user cannot interact with the page
 - Server finds invalid data, generates a new page identifying the errors in the form and sends it back to the client—which renders the page in the browser
 - User fixes the errors and clicks the Register button again
 - Cycle repeats until no errors are found, then the data is stored on the server
 - Entire page reloads every time the user submits invalid data
- Ajax-enabled forms are more interactive
 - Entries are validated dynamically as the user enters data into the fields
 - Asynchronous requests could also be used to fill some fields based on

a) A sample registration form in which the user has not filled in the required fields, but attempts to submit the form anyway by clicking **Register**.

The screenshot shows a web browser window titled "Sample Registration Form - Windows Internet Explorer". The address bar displays "http://localhost:8080/WebComponents/". The page content includes a heading "This is a sample registration form" with a subtext "Please fill in all fields and click Register". The form is divided into three sections: "User Information" with fields for "First Name", "Last Name", "Email", and "Phone"; "Publications" with a dropdown menu currently showing "Internet & World Wide Web How to Program"; and "Operating System" with radio button options for "Windows Vista", "Windows XP", "Mac OS X", "Linux", and "Other". A "Register" button is located at the bottom of the form, with a mouse cursor hovering over it. The browser's status bar at the bottom shows "Done", "Local intranet", and "100%".

Fig. Classic XHTML form: User submits entire form to server, which validates the data entered (if any). Server responds indicating fields with invalid or missing data.

b) The server responds by indicating all the form fields with missing or invalid data. The user must correct the problems and resubmit the entire form repeatedly until all errors are corrected.

The screenshot shows a web browser window titled "Sample Registration Form - Windows Internet Explorer". The address bar displays "http://localhost:8080/WebComponents/". The page content includes a title "This is a sample registration form" and a instruction "Please fill in all fields and click Register". The form is divided into three sections: "User Information", "Publications", and "Operating System". The "User Information" section has four text input fields: "First Name", "Last Name", "Email", and "Phone". The "Publications" section has a dropdown menu for "Which book would you like information about?" with the selected value "Internet & World Wide Web How to Program". The "Operating System" section has five radio button options: "Windows Vista", "Windows XP", "Mac OS X", "Linux", and "Other". A "Register" button is at the bottom. Red validation error messages are displayed next to the "First Name", "Last Name", "Email", and "Operating System" fields, all stating "Validation Error: Value is required.".

Sample Registration Form - Windows Internet Explorer

http://localhost:8080/WebComponents/

Google

Google C Go Search USA Settings

Sample Registration Form

This is a sample registration form

Please fill in all fields and click Register

User Information

First Name form1:firstNameTextField: Validation Error: Value is required.

Last Name form1:lastNameTextField: Validation Error: Value is required.

Email form1:emailTextField: Validation Error: Value is required.

Phone

Publications Which book would you like information about?

Internet & World Wide Web How to Program

Click here to learn more about our books

Operating System What operating system are you using?

Windows Vista form1:osRadioGroup: Validation Error: Value is required.

Windows XP

Mac OS X

Linux

Other

Register

Done Local intranet 100%

Fig. Classic XHTML form: User submits entire form to server, which validates the data entered (if any). Server responds indicating fields with invalid or missing data.

The screenshot shows a Windows Internet Explorer browser window titled "Sample Registration Form - Windows Internet Explorer". The address bar displays "http://localhost:8080/WebComponents/". The page content includes a heading "This is a sample registration form" and a subheading "Please fill in all fields and click Register". The form is divided into three sections: "User Information", "Publications", and "Operating System". The "User Information" section has fields for "First Name" (Sally), "Last Name" (Blue), "Email" (NotaValidEmail), and "Phone". A red error message "Enter a valid email address, e.g. user@domain.com" is displayed next to the email field. The "Publications" section has a dropdown menu for "Which book would you like information about?" with the selected option "Internet & World Wide Web How to Program". Below this is a link "Click here to learn more about our books". The "Operating System" section has radio buttons for "Windows Vista", "Windows XP", "Mac OS X", "Linux", and "Other". A "Register" button is at the bottom of the form. The browser's status bar at the bottom shows "Done", "Local intranet", and "100%".

Sample Registration Form - Windows Internet Explorer

http://localhost:8080/WebComponents/

Google

Go Search USA Settings

Sample Registration Form

This is a sample registration form

Please fill in all fields and click Register

User Information

First Name Sally

Last Name Blue

Email NotaValidEmail Enter a valid email address, e.g. user@domain.com

Phone

Publications Which book would you like information about?

Internet & World Wide Web How to Program

Click here to learn more about our books

Operating System What operating system are you using?

☐ Windows Vista

☐ Windows XP

☐ Mac OS X

☐ Linux

☐ Other

Register

Done Local intranet 100%

Ajax-enabled form shows errors asynchronously when user moves to another field.

History of Ajax

- The term Ajax was coined by Jesse James Garrett of Adaptive Path in February 2005, when he was presenting the previously unnamed technology to a client
- Ajax technologies (XHTML, JavaScript, CSS, dynamic HTML, the DOM and XML) have existed for many years
 - In 1998, Microsoft introduced the XML Http Request object to create and manage asynchronous requests and responses
 - Popular applications like Flickr, Google's Gmail and Google Maps use the XML Http Request object to update pages dynamically
 - Ajax has quickly become one of the hottest technologies in web development, as it enables WebTop applications to challenge the dominance of established desktop applications

“Raw” Ajax Example using the XMLHttpRequest Object

- XML Http Request object
 - Resides on the client
 - Is the layer between the client and the server that manages asynchronous requests in Ajax applications
 - Supported on most browsers, though they may implement it differently
 - To initiate an asynchronous request
 - Create an instance of the XMLHttpRequest object
 - Use its open method to set up the request, and its send method to initiate the request
 - When an Ajax application requests a file from a server, the browser typically caches that file
 - Subsequent requests for the same file can load it from the browser’s cache
 - Security
 - XMLHttpRequest object does not allow a web application to request resources from servers other than the one that served the web application
 - Making a request to a different server is known as cross-site scripting (also known as XSS)
 - You can implement a server-side proxy—an application on the web application’s web server—that can make requests to other servers on the web application’s behalf
- When the third argument to XMLHttpRequest method open is true, the request is asynchronous

Software Engineering Observation

For security purposes, the XML HttpRequest object doesn't allow a web application to request resources from domain names other than the one that served the application. For this reason, the web application and its resources must reside on the same web server (this could be a web server on your local computer). This is commonly known as the same origin policy (SOP). SOP aims to close a vulnerability called cross-site scripting, also known as XSS, which allows an attacker to compromise a website's security by injecting a malicious script onto the page from another domain. To learn more about XSS visit en.wikipedia.org/wiki/XSS. To get content from another domain securely, you can implement a server-side proxy—an application on the web application's web server—that can make requests to other servers on the web application's behalf.

“Raw” Ajax Example using the XMLHttpRequest Object (Cont.)

- An exception is an indication of a problem that occurs during a program’s execution
- Exception handling enables you to create applications that can resolve (or handle) exceptions—in some cases allowing a program to continue executing as if no problem had been encountered
- try block
 - Encloses code that might cause an exception and code that should not execute if an exception occurs
 - Consists of the keyword try followed by a block of code enclosed in curly braces ({})
- When an exception occurs
 - try block terminates immediately
 - catch block catches (i.e., receives) and handles an exception
- catch block
 - Begins with the keyword catch
 - Followed by an exception parameter in parentheses and a block of code enclosed in curly braces
- Exception parameter’s name
 - Enables the catch block to interact with a caught exception object, which contains name and message properties
- A callback function is registered as the event handler for the XMLHttpRequest object’s onreadystatechange event
 - Whenever the request makes progress, the XMLHttpRequest calls the onreadystatechange event handler.
 - Progress is monitored by the readyState property, which has a value from 0 to 4
 - The value 0 indicates that the request is not initialized and the value 4 indicates that the request is complete.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 15.5: SwitchContent.html -->
6 <!-- Asynchronously display content without reloading the page. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9   <style type="text/css">
10     .box { border: 1px solid black;
11            padding: 10px }
12   </style>
13   <title>Switch Content Asynchronously</title>
14   <script type = "text/javascript" language = "JavaScript">
15     <!--
16     var asyncRequest; // variable to hold XMLHttpRequest object
17
18     // set up and send the asynchronous request
19     function getContent( url )
20     {
21       // attempt to create the XMLHttpRequest and make the request
22       try
23       {
24         asyncRequest = new XMLHttpRequest(); // create request object
25
26         // register event handler
27         asyncRequest.onreadystatechange = stateChange;
28         asyncRequest.open( 'GET', url, true ); // prepare the request
29         asyncRequest.send( null ); // send the request
30       } // end try
```

The program attempts to execute the code in the try block. If an exception occurs, the code in the catch block will be executed.

Set the event handler for the onreadystatechange event to the function stateChange.

The request will be a GET request for the page located at url, and it will be asynchronous.

```
31 catch ( exception )
32 {
33     alert( 'Request failed.' );
34 } // end catch
35 } // end function getContent
36
37 // displays the response data on the page
38 function stateChange()
39 {
40     if ( asyncRequest.readyState == 4 && asyncRequest.status == 200 )
41     {
42         document.getElementById( 'contentArea' ).innerHTML =
43             asyncRequest.responseText; // places text in contentArea
44     } // end if
45 } // end function stateChange
46
47 // clear the content of the box
48 function clearContent()
49 {
50     document.getElementById( 'contentArea' ).innerHTML = '';
51 } // end function clearContent
52 // -->
```

Notify the user that an error occurred

If the request has completed successfully, use the DOM to update the page with the `responseText` property of the request object


```

53 </script>
54 </head>
55 <body>
56 <h1>Mouse over a book for more information.</h1>
57 <img src =
58 "http://test.deitel.com/examples/iw3http4/ajax/thumbs/cpphttp6.jpg"
59 onmouseover = 'getContent( "cpphttp6.html" )'
60 onmouseout = 'clearContent()'/>
61 <img src =
62 "http://test.deitel.com/examples/iw3http4/ajax/thumbs/iw3http4.jpg"
63 onmouseover = 'getContent( "iw3http4.html" )'
64 onmouseout = 'clearContent()'/>
65 <img src =
66 "http://test.deitel.com/examples/iw3http4/ajax/thumbs/jhttp7.jpg"
67 onmouseover = 'getContent( "jhttp7.html" )'
68 onmouseout = 'clearContent()'/>
69 <img src =
70 "http://test.deitel.com/examples/iw3http4/ajax/thumbs/vbhttp3.jpg"
71 onmouseover = 'getContent( "vbhttp3.html" )'
72 onmouseout = 'clearContent()'/>
73 <img src =
74 "http://test.deitel.com/examples/iw3http4/ajax/thumbs/vcsharphttp2.jpg"
75 onmouseover = 'getContent( "vcsharphttp2.html" )'
76 onmouseout = 'clearContent()'/>

```

Outline

SwitchContent
.html

(3 of 5)

```

77 <img src =
78 "http://test.deitel.com/examples/iw3http4/ajax/thumbs/chtp5.jpg"
79 onmouseover = 'getContent( "chtp5.html" )'
80 onmouseout = 'clearContent()' />
81 <div class = "box" id = "contentArea">&nbsp;</div>
82 </body>
83 </html>

```

This div is updated with the description of the book that the mouse is currently

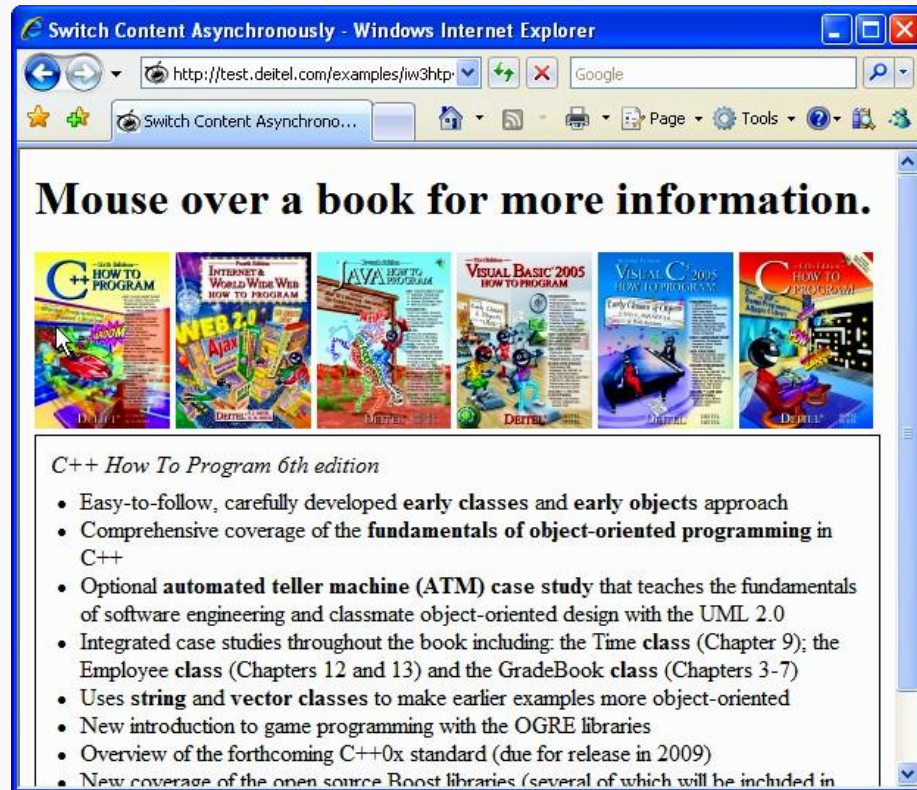
hovering over

Outline

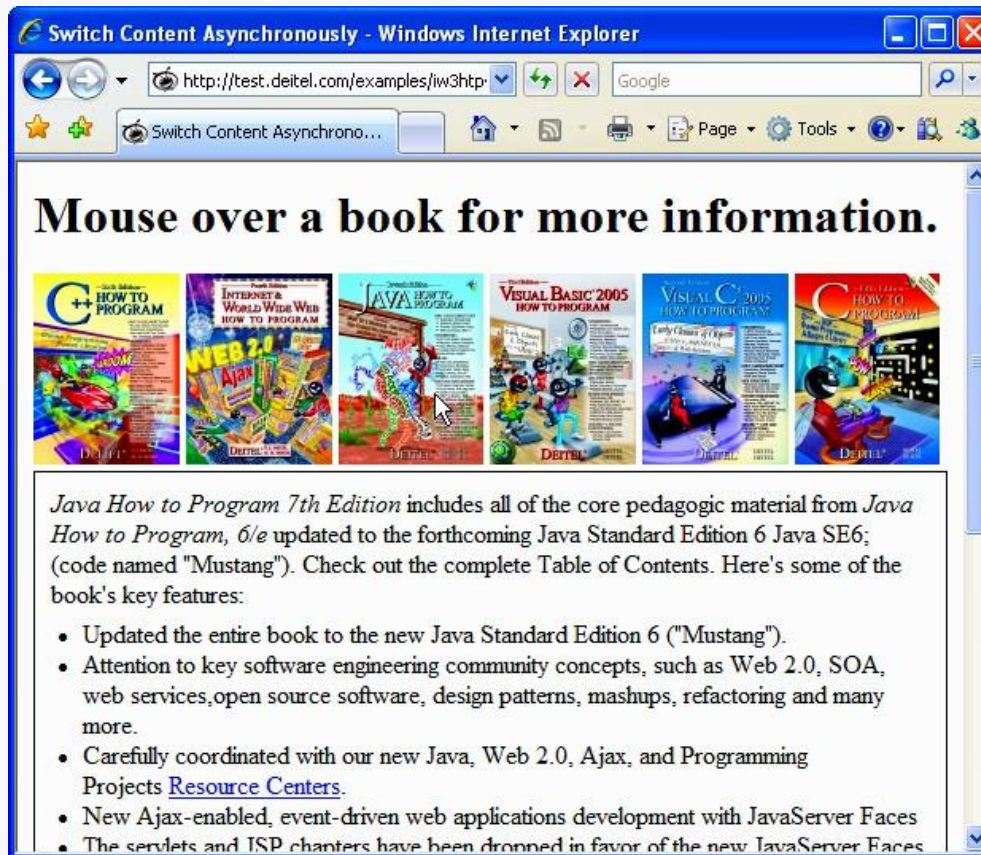
SwitchContent.html

(4 of 5)

a) User hovers over C++ *How to Program* book cover image, causing an asynchronous request to the server to obtain the book's description. When the response is received, the application performs a partial page update to display the description.



b) User hovers over *Java How to Program* book cover image, causing the process to repeat.



SwitchContent
.html

(5 of 5)

Using XML and the DOM

- When passing structured data between the server and the client, Ajax applications often use XML because it consumes little bandwidth and is easy to parse
- XML Http Request object response XML property
 - contains the parsed XML returned by the server
- DOM method create Element
 - Creates an XHTML element of the specified type
- DOM method set Attribute
 - Adds or changes an attribute of an XHTML element
- DOM method append Child
 - Inserts one XHTML element into another
- Inner HTML property of a DOM element
 - Can be used to obtain or change the XHTML that is displayed in a particular element

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 15.8: PullImagesOntoPage.html -->
6 <!-- Image catalog that uses Ajax to request XML data asynchronously. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title> Pulling Images onto the Page </title>
10 <style type = "text/css">
11     td { padding: 4px }
12     img { border: 1px solid black }
13 </style>
14 <script type = "text/javascript" language = "Javascript">
15     var asyncRequest; // variable to hold XMLHttpRequest object
16
17     // set up and send the asynchronous request to the XML file
18     function getImages( url )
19     {
20         // attempt to create the XMLHttpRequest and make the request try
21         {
22             asyncRequest = new XMLHttpRequest(); // create request object
23
24
25             // register event handler asyncRequest.onreadystatechange =
26             processResponse;
27             asyncRequest.open( 'GET', url, true ); // prepare the request
28             asyncRequest.send( null ); // send the request
29         } // end try
```

```
30 catch ( exception )
31 {
32     alert( 'Request Failed' );
33 } // end catch
34 } // end function getImages
35
36 // parses the XML response; dynamically creates a table using DOM and
37 // populates it with the response data; displays the table on the page
38 function processResponse()
39 {
40     // if request completed successfully and responseXML is non-null
41     if ( asyncRequest.readyState == 4 && asyncRequest.status == 200 &&
42         asyncRequest.responseXML )
43     {
44         clearTable(); // prepare to display a new set of images
45
46         // get the covers from the responseXML
47         var covers = asyncRequest.responseXML.getElementsByTagName(
48             "cover" );
49
50         // get base URL for the images
51         var baseUrl = asyncRequest.responseXML.getElementsByTagName(
52             "baseUrl" ).item( 0 ).firstChild.nodeValue;
53
54         // get the placeholder div element named covers
55         var output = document.getElementById( "covers" );
56
57         // create a table to display the images
58         var imageTable = document.createElement( 'table' );
59     }
```

The XMLHttpRequest
object's responseXML
property contains a DOM
document object for
the loaded XML document

Get a list of the covers
from the XML
document

Get the base URL for the
images to be displayed on the
page


```
60 // create the table's body
61 var tableBody = document.createElement( 'tbody' );
62
63 var rowCount = 0; // tracks number of images in current row
64 var imageRow = document.createElement( "tr" ); // create row
65
66 // place images in row
67 for ( var i = 0; i < covers.length; i++ )
68 {
69     var cover = covers.item( i ); // get a cover from covers array
70
71     // get the image filename
72     var image = cover.getElementsByTagName( "image" ).
73     item( 0 ).firstChild.nodeValue;
74
75     // create table cell and img element to display the image
76     var imageCell = document.createElement( "td" );
77     var imageTag = document.createElement( "img" );
78
79     // set img element's src attribute
80     imageTag.setAttribute( "src", baseUrl + escape( image ) );
81     imageCell.appendChild( imageTag ); // place img in cell
82     imageRow.appendChild( imageCell ); // place cell in row
83     rowCount++; // increment number of images in row
84 }
```

**Insert each image based
on its filename and
baseUrl**

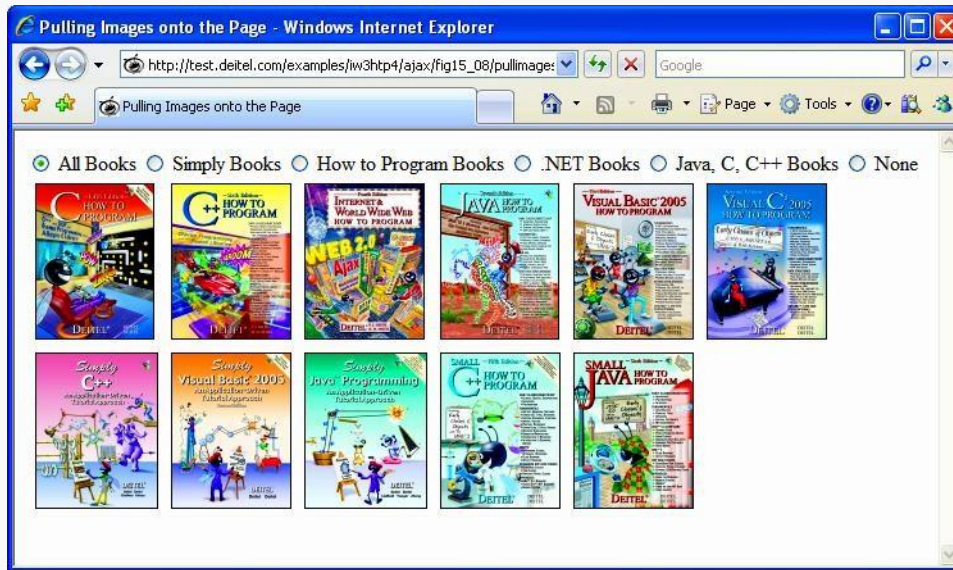
```
85 // if there are 6 images in the row, append the row to
86 // table and start a new row
87 if ( rowCount == 6 && i + 1 < covers.length )
88 {
89     tableBody.appendChild( imageRow ); imageRow =
90     document.createElement( "tr" );
91     rowCount = 0;
92 } // end if statement
93 } // end for statement
94
95 tableBody.appendChild( imageRow ); // append row to table body
96 imageTable.appendChild( tableBody ); // append body to table
97 output.appendChild( imageTable ); // append table to covers div
98 } // end if
99 } // end function processResponse
100
101 // deletes the data in the table.
102 function clearTable()
103 {
104     document.getElementById( "covers" ).innerHTML = '';
105 } // end function clearTable
```



```
106 </script>
107</head>
108<body>
109     <input type = "radio" checked = "unchecked" name ="Books" value =
        "all"
110     onclick = 'getImages( "all.xml" )' /> All Books
111     <input type = "radio" checked = "unchecked"
112     name = "Books" value = "simply"
113     onclick = 'getImages( "simply.xml" )' /> Simply Books
114     <input type = "radio" checked = "unchecked"
115     name = "Books" value = "howto"
116     onclick = 'getImages( "howto.xml" )' /> How to Program Books
117     <input type = "radio" checked = "unchecked" name = "Books"
118     value = "dotnet"
119     onclick = 'getImages( "dotnet.xml" )' /> NET Books
120     <input type = "radio" checked = "unchecked"
121     name = "Books" value = "javaccpp"
122     onclick = 'getImages( "javaccpp.xml" )' /> Java, C, C++ Books
123     <input type = "radio" checked = "checked" name = "Books" value = "none"
124     onclick = 'clearTable()' /> None
125 <br/>
126 <div id = "covers"></div>
127</body>
128</html>
```

**Load the correct XML
document when a radio
button is clicked**

a) User clicks the **All Books** radio button to display all the book covers. The application sends an asynchronous request to the server to obtain an XML document containing the list of book-cover filenames. When the response is received, the application performs a partial page update to display the set of book covers.

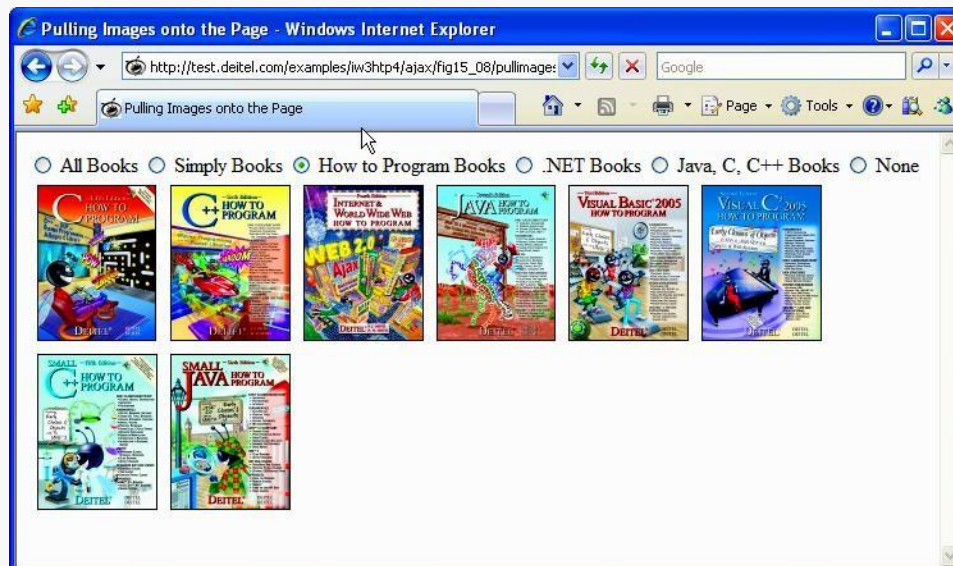


Outline

PullImagesOntoPage.html

(6 of 6)

b) User clicks the **How to Program Books** radio button to select a subset of book covers to display. Application sends an asynchronous request to the server to obtain an XML document containing the appropriate subset of book-cover filenames. When the response is received, the application performs a partial page update to display the subset of book covers.



```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 15.9 addressbook.html -->
6 <!-- Ajax enabled address book application. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9   <title>Address Book</title>
10  <link rel = "stylesheet" type = "text/css" href = "address.css" />
11  <script type = "text/javascript" src = "json.js"></script>
12  <script type = "text/javascript">
13    <!--
14    // URL of the web service
15    var webServiceUrl = '/AddressBookWebService/AddressService.asmx';
16
17    var phoneValid = false; // indicates if the telephone is valid
18    var zipValid = false; // indicates if the zip code is valid
19
20    // get a list of names from the server and display them
21    function showAddressBook()
22    {
23      // hide the "addEntry" form and show the address book
24      document.getElementById( 'addEntry' ).style.display = 'none';
25      document.getElementById( 'addressBook' ).style.display = 'block';
26
27      var params = "[]"; // create an empty object
28      callWebService( 'getAllNames', params, parseData );
29    } // end function showAddressBook
30
```

Hide the form for adding an entry

Get a list of all the names from the web service, and call parseData when it's loaded

```
31 // send the asynchronous request to the web service
32 function callWebService( method, paramString, callBack )
33 {
34     // build request URL string
35     var requestUrl = baseUrl + "/" + method;
36     var params = paramString.parseJSON();
37
38     // build the parameter string to add to the url
39     for ( var i = 0; i < params.length; i++ )
40     {
41         // checks whether it is the first parameter and builds
42         // the parameter string accordingly
43         if ( i == 0 )
44             requestUrl = requestUrl + "?" + params[ i ].param +
45                 "=" + params[ i ].value; // add first parameter to url
46         else
47             requestUrl = requestUrl + "&" + params[ i ].param +
48                 "=" + params[ i ].value; // add other parameters to url
49     } // end for
50
51     // attempt to send the asynchronous request
52     try
53     {
54         var asyncRequest = new XMLHttpRequest(); // create request
55
56         // set up callback function and store it
57         asyncRequest.onreadystatechange = function()
58         {
59             callBack( asyncRequest );
60         }; // end anonymous function
```

Call a particular method by
appending method to the base

Parse the
parameters

Build the
parameter string

Set the callback function for
the request to callback with
the request object as a
parameter

```
61 // send the asynchronous request
62 asyncRequest.open( 'GET', requestUrl, true );
63
64 asyncRequest.setRequestHeader("Accept",
65     "application/json; charset=utf-8" );
66 asyncRequest.send(); // send request
67 } // end try
68 catch ( exception )
69 {
70     alert ( 'Request Failed' );
71 } // end catch
72 } // end function callWebService
73
74 // parse JSON data and display it on the page function parseData(
75 asyncRequest )
76 {
77 // if request has completed successfully process the response
78 if ( asyncRequest.readyState == 4 && asyncRequest.status == 200 )
79 {
80 // convert the JSON string to an Object
81     var data = asyncRequest.responseText.parseJSON();
82     displayNames( data ); // display data on the page
83 } // end if
84 } // end function parseData
85
```

**Prepare and send
the request**

**Parse the JSON
text**

Outline

addressbook.html

(4 of 18)

Create
field
entry

Create an XHTML
fieldset element for

the entry
Insert the name into
the entry

Use this to give the
clicked element the
correct parameters

318

```
86 // use the DOM to display the retrieved address book entries
87 function displayNames( data )
88 {
89 // get the placeholder element from the page
90 var listBox = document.getElementById( 'Names' );
91 listBox.innerHTML = ''; // clear the names on the page
92
93 // iterate over retrieved entries and display them on the page
94 for ( var i = 0; i < data.length; i++ )
95 {
96 // dynamically create a div element for each entry
97 // and a fieldset element to place it in
98 var entry = document.createElement( 'div' );
99 var field = document.createElement( 'fieldset' );
100 entry.onclick = handleClick; // set onclick event handler
101 entry.id = i; // set the id
102 entry.innerHTML = data[ i ].First + ' ' + data[ i ].Last;
103 field.appendChild( entry ); // insert entry into field
104 listBox.appendChild( field ); // display the fieldset element
105 } // end for
106 } // end function displayAll
107
108 // event handler for entry's onclick event
109 function handleClick()
110 {
111 // call getAddress with the element's content as a parameter
112 getAddress( eval( 'this' ), eval( 'this.innerHTML' ) );
113 } // end function handleClick
```

].
the
element

Last;
field

handle

by

to

on the page

page

```
115 // search the address book for input
116 // and display the results on the page function search(
117 input )
118 {
119 // get the placeholder element and delete its content
120 var listBox = document.getElementById( 'Names' );
121 listBox.innerHTML = ''; // clear the display box
122
123 // if no search string is specified all the names are displayed
124 if ( input == "" ) // if no search value specified
125 {
126 showAddressBook(); // Load the entire address book
127 } // end if
128 else
129 {
130     var params = '[{"param": "input", "value": "' + input + '"}]';
131     callWebService( "search", params , parseData );
132 } // end else
133 } // end function search
134
135 // Get address data for a specific entry function getAddress(
136 entry, name )
137 {
138 // find the address in the JSON data using the element's id
139 // and display it on the page
140 var firstLast = name.split(" "); // convert string to array
141 var requestUrl = webServiceUrl + "/getAddress?first="
142     + firstLast[ 0 ] + "&last=" + firstLast[ 1 ];
143
```

**Create a JSON
parameter object to
search for input
in the list of names**

**Assemble the web
service call**


```
144 // attempt to send an asynchronous request
145 try
146 {
147     // create request object
148     var asyncRequest = new XMLHttpRequest();
149
150     // create a callback function with 2 parameters
151     asyncRequest.onreadystatechange = function()
152     {
153         displayAddress( entry, asyncRequest );
154     }; // end anonymous function
155
156     asyncRequest.open( 'GET', requestUrl, true );
157     asyncRequest.setRequestHeader("Accept",
158     "application/json; charset=utf-8"); // set response datatype
159     asyncRequest.send(); // send request
160 } // end try
161 catch ( exception )
162 {
163     alert ( 'Request Failed.' );
164 } // end catch
165 } // end function getAddress
166
```


Parse and display the
address details for an
entry

```
167 // clear the entry's data.
168 function displayAddress( entry, asyncRequest )
169 {
170     // if request has completed successfully, process the response
171     if ( asyncRequest.readyState == 4 && asyncRequest.status == 200 )
172     {
173         // convert the JSON string to an object
174         var data = asyncRequest.responseText.parseJSON();
175         var name = entry.innerHTML // save the name string
176         entry.innerHTML = name + '<br/>' + data.Street +
177             '<br/>' + data.City + ', ' + data.State
178             + ', ' + data.Zip + '<br/>' + data.Telephone;
179
180         // clicking on the entry removes the address
181         entry.onclick = function()
182         {
183             clearField( entry, name );
184             }; // end anonymous function
185
186     } // end if
187 } // end function displayAddress
188
```

**Clear the address
from an entry and
reset the event
handler to display
the address again
when clicked**

**Make the zip-code
validation web
service call**

```
189 // clear the entry's data
190 function clearField( entry, name )
191 {
192     entry.innerHTML = name; // set the entry to display only the name
193     entry.onclick = function() // set onclick event
194     {
195         getAddress( entry, name ); // retrieve address and display it
196     }; // end function
197 } // end function clearField
198
199 // display the form that allows the user to enter more data
200 function addEntry()
201 {
202     document.getElementById( 'addressBook' ).style.display = 'none';
203     document.getElementById( 'addEntry' ).style.display = 'block';
204 } // end function addEntry
205
206 // send the zip code to be validated and to generate city and state
207 function validateZip( zip )
208 {
209     // build parameter array
210     var params = '[{"param": "zip", "value": "' + zip + '"}]';
211     callWebService ( "validatezip", params, showCityState );
212 } // end function validateZip
213
```

Notify the user that the
zip code is being

checked

addressbook.html

(9 of 18)

```
214 // get city and state that were generated using the zip code
215 // and display them on the page function showCityState(
216 asyncRequest )
217 {
218     // display message while request is being processed
219     document.getElementById( 'validatezip' ).
220         innerHTML = "checking zip...";
221
222     // if request has completed successfully, process the response
223     if ( asyncRequest.readyState == 4 )
224     {
225         if ( asyncRequest.status == 200 )
226         {
227             // convert the JSON string to an object
228             var data = asyncRequest.responseText.parseJSON();
229
230             // update zip code validity tracker and show city and state
231             if ( data.Validity == 'Valid' )
232             {
233                 zipValid = true; // update validity tracker
234
235                 // display city and state
236                 document.getElementById( 'validatezip' ).innerHTML = '';
237                 document.getElementById( 'city' ).innerHTML = data.City;
238                 document.getElementById( 'state' ).
239                     innerHTML = data.State;
240             } // end if
```

```
241     else
242     {
243         zipValid = false; // update validity tracker
244         document.getElementById( 'validateZip' ).
245         innerHTML = data.ErrorText; // display the error
246
247         // clear city and state values if they exist
248         document.getElementById( 'city' ).innerHTML = '';
249         document.getElementById( 'state' ).innerHTML = '';
250     } // end else
251 } // end if
252 else if ( asyncRequest.status == 500 )
253 {
254     document.getElementById( 'validateZip' ).
255     innerHTML = 'zip validation service not available';
256 } // end else if
257 } // end if
258 } // end function showCityState
259
260 // send the telephone number to the server to validate format
261 function validatePhone( phone )
262 {
263     var params = '[{"param": "tel", "value": "' + phone + '"}]';
264     callWebService( "validateTel", params, showPhoneError );
265 } // end function validatePhone
266
```

Get the response from the request object so we can determine if the phone number is valid

```
267 // show whether the telephone number has correct format
268 function showPhoneError( asyncRequest )
269 {
270     // if request has completed successfully, process the response
271     if ( asyncRequest.readyState == 4 && asyncRequest.status == 200 )
272     {
273         // convert the JSON string to an object
274         var data = asyncRequest.responseText.parseJSON();
275
276         if ( data.ErrorText != "Valid Telephone Format" )
277         {
278             phoneValid = false; // update validity tracker
279             } // end if else
280
281         {
282             phoneValid = true; // update validity tracker
283             } // end else
284
285         document.getElementById( 'validatePhone' ).
286             innerHTML = data.ErrorText; // display the error
287         } // end if
288     } // end function showPhoneError
289
290 // enter the user's data into the database
```

```
291 function saveForm()
292 {
293     // retrieve the data from the form
294     var first = document.getElementById( 'first' ).value;
295     var last = document.getElementById( 'last' ).value;
296     var street = document.getElementById( 'street' ).value;
297     var city = document.getElementById( 'city' ).innerHTML;
298     var state = document.getElementById( 'state' ).innerHTML;
299     var zip = document.getElementById( 'zip' ).value;
300     var phone = document.getElementById( 'phone' ).value;
301
302     // check if data is valid
303     if ( !zipValid || !phoneValid )
304     {
305         // display error message
306         document.getElementById( 'success' ).innerHTML =
307             'Invalid data entered. Check form for more information';
308     } // end if
309     else if ( ( first == "" ) || ( last == "" ) )
310     {
311         // display error message
312         document.getElementById( 'success' ).innerHTML =
313             'First Name and Last Name must have a value.';
314     } // end if
```

```
315 else
316 {
317 // hide the form and show the addressbook
318 document.getElementById( 'addEntry' )
319 .style.display = 'none';
320 document.getElementById( 'addressBook' ).
321 style.display = 'block';
322
323 // build the parameter to include in the web service URL
324 params = '[{"param": "first", "value": "' + first +
325           '"}, {"param": "last", "value": "' + last +
326           '"}, {"param": "street", "value": "' + street +
327           '"}, {"param": "city", "value": "' + city +
328           '"}, {"param": "state", "value": "' + state +
329           '"}, {"param": "zip", "value": "' + zip +
330           '"}, {"param": "tel", "value": "' + phone + '"}]';
331
332 // call the web service to insert data into the database
333 callWebService( "addEntry", params, parseData );
334 } // end else
335 } // end function saveForm
336 //-->
```

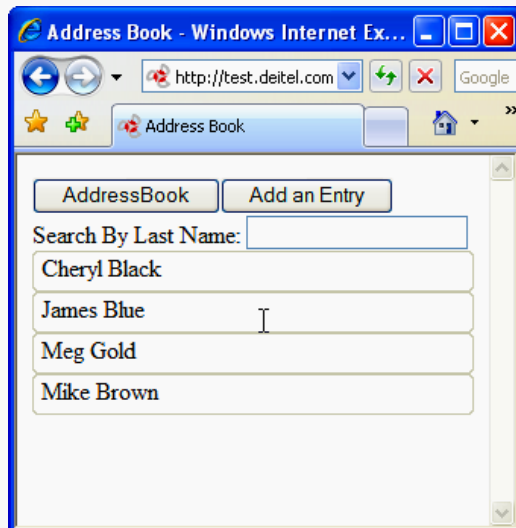
← Create a JSON object with all the data as parameters for the web service call to save the data

```
337 </script>
338</head>
339<body onload = "showAddressBook()">
340  <div>
341    <input type = "button" value = "Address Book"
342      onclick = "showAddressBook()"/>
343    <input type = "button" value = "Add an Entry"
344      onclick = "addEntry()"/>
345  </div>
346  <div id = "addressBook" style = "display : block;">
347    Search By Last Name:
348    <input onkeyup = "search( this.value )"/>
349    <br/>
350    <div id = "Names">
351    </div>
352  </div>
353  <div id = "addEntry" style = "display : none">
354    First Name: <input id = 'first' />
355    <br/>
356    Last Name: <input id = 'last' />
357    <br/>
358    <strong> Address: </strong>
359    <br/>
360    Street: <input id = 'street' />
361    <br/>
362    City: <span id = "city" class = "validator"></span>
363    <br/>
364    State: <span id = "state" class = "validator"></span>
365    <br/>
366    Zip: <input id = 'zip' onblur = 'validateZip( this.value )' />
```

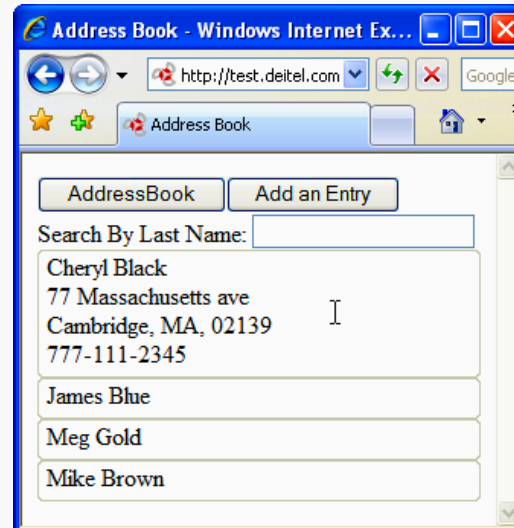
**The search is done for every
onkeyup event, giving the form
live search functionality**


```
367 <span id = "validateZip" class = "validator">
368 </span>
369 <br/>
370 Telephone:<input id = 'phone'
371     onblur = 'validatePhone( this.value )' />
372 <span id = "validatePhone" class = "validator">
373 </span>
374 <br/>
375 <input type = "button" value = "Submit"
376     onclick = "saveForm()" />
377 <br/>
378 <div id = "success" class = "validator">
379 </div>
380 </div>
381 </body>
382 </html>
```

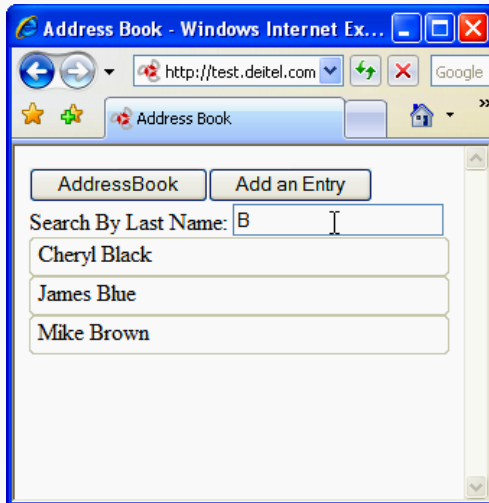
a) Page is loaded. All the entries are displayed.



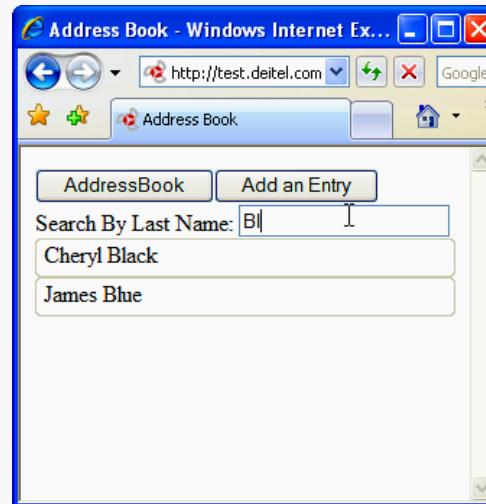
b) User clicks on an entry. The entry expands, showing the address and the telephone.



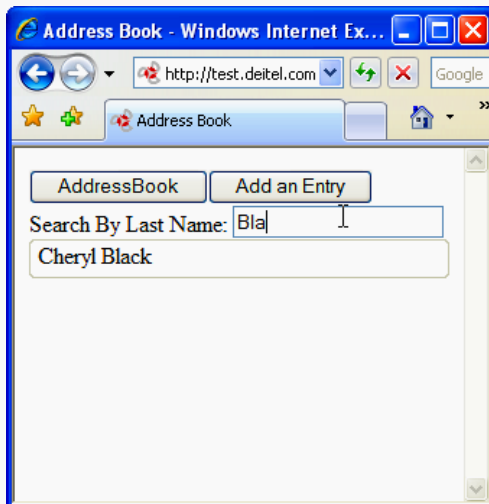
c) User types "B" in the search field. Application loads the entries whose last names start with "B".



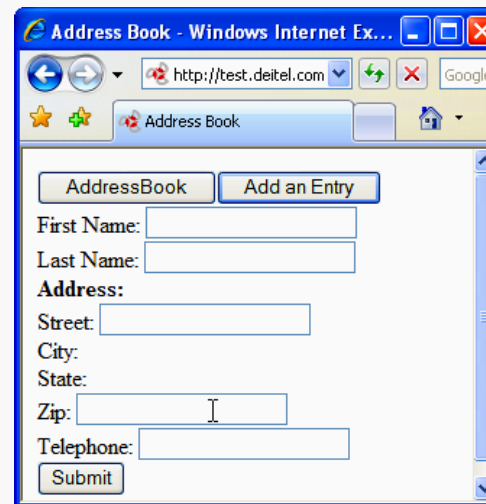
d) User types "Bl" in the search field. Application loads the entries whose last names start with "Bl".



e) User types "Bla" in the search field. Application loads the entries whose last names start with "Bla".



f) User clicks **Add an Entry** button. The form allowing user to add an entry is displayed.



g) User types in a nonexistent zip code. An error is displayed.

The screenshot shows the 'Address Book' form in a Windows Internet Explorer window. The form has fields for First Name, Last Name, Address (Street, City, State), Zip, and Telephone. The 'Zip' field contains '1910' and the text 'Zip code does not exist' is displayed next to it. The 'Submit' button is visible at the bottom.

h) User enters a valid zip code. While the server processes it, **Checking Zip...** is displayed on the page.

The screenshot shows the 'Address Book' form in a Windows Internet Explorer window. The 'Zip' field contains '19106' and the text 'Checking zip...' is displayed next to it. The 'Submit' button is visible at the bottom.

i) The server finds the city and state associated with the zip code entered and displays them on the page.

The screenshot shows the 'Address Book' form in a Windows Internet Explorer window. The 'Zip' field contains '19106' and the 'City' field is populated with 'Philadelphia' and the 'State' field is populated with 'PA'. The 'Submit' button is visible at the bottom.

j) The user enters a telephone number and tries to submit the data. The application does not allow this, because the First Name and Last Name are empty.

The screenshot shows the 'Address Book' form in a Windows Internet Explorer window. The 'First Name' and 'Last Name' fields are empty. The 'Zip' field contains '19106' and the 'Telephone' field contains '555-111-2222'. The text 'Valid' is displayed next to the telephone field. The 'Submit' button is highlighted, and the error message 'First Name and Last Name must have a value.' is displayed at the bottom.

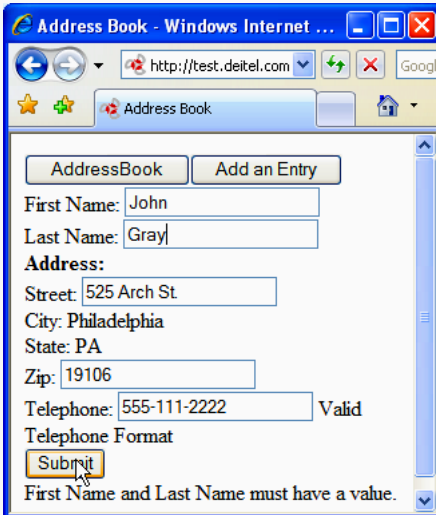
addressbook.html

(17 of 18)

addressbook.html

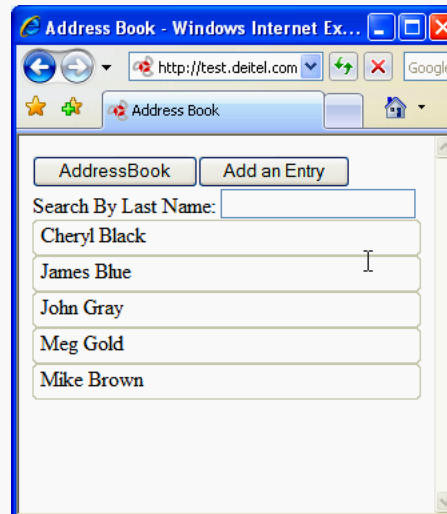
(18 of 18)

k) The user enters the last name and the first name and clicks the Submit button.



A screenshot of a web browser window titled "Address Book - Windows Internet ...". The address bar shows "http://test.deitel.com". The page has a navigation bar with "AddressBook" and "Add an Entry" buttons. The form contains the following fields: "First Name:" with the value "John", "Last Name:" with the value "Gray", "Address:" section with "Street:" (525 Arch St), "City:" (Philadelphia), "State:" (PA), "Zip:" (19106), and "Telephone:" (555-111-2222) with a "Valid" status. A "Submit" button is at the bottom, and a message below it reads "First Name and Last Name must have a value.".

l) The address book is redisplayed with the new name added in.



A screenshot of the same web browser window after submission. The form fields are replaced by a "Search By Last Name:" input field. Below it is a list of names in individual boxes: "Cheryl Black", "James Blue", "John Gray", "Meg Gold", and "Mike Brown". The "Submit" button is no longer visible.

Dojo Toolkit

- Cross-browser compatibility, DOM manipulation and event handling can be cumbersome, particularly as an application's size increases
- Dojo
 - Free, open source JavaScript library that simplifies Ajax development
 - Reduces asynchronous request handling to a single function call
 - Provides cross-browser DOM functions that simplify partial page updates
 - Provides event handling and rich GUI controls
- To install Dojo
 - Download the latest release from www.Dojotoolkit.org/downloads to your hard drive
 - Extract the files from the archive file to your web development directory or web server
 - To include the Dojo.js script file in your web application, place the following script in the head element of your XHTML document:
<script type = "text/javascript" src = "*path*/Dojo.js">
where *path* is the relative or complete path to the Dojo toolkit's files
- Edit-in-place
 - Enables a user to modify data directly in the web page
 - Common feature in Ajax applications

Dojo Toolkit (Cont.)

- Dojo is organized in packages of related functionality
- `dojo.require` method
 - Used to include specific Dojo packages
- `dojo.io` package functions communicate with the server
- `dojo.event` package simplifies event handling
- `dojo.widget` package provides rich GUI controls
- `dojo.dom` package contains DOM functions that are portable across many different browsers
- Dojo widget
 - Any predefined user interface element that is part of the Dojo toolkit
 - To incorporate an existing widget onto a page, set the `dojoType` attribute of any HTML element to the type of widget that you want it to be
- `dojo.widget.byId` method can be used to obtain a Dojo widget
- `dojo.events.connect` method links functions together

Outline

Calendar.html

(1 of 11)

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 15.11 Calendar.html -->
6 <!-- Calendar application built with dojo. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9   <script type = "text/javascript" src = "/dojo043/dojo.js"></script>
10  <script type = "text/javascript" src = "json.js"></script>
11  <script type = "text/javascript">
12    <!--
13    // specify all the required dojo scripts
14    dojo.require( "dojo.event.*" ); // use scripts from event package
15    dojo.require( "dojo.widget.*" ); // use scripts from widget package
16    dojo.require( "dojo.dom.*" ); // use scripts from dom package
17    dojo.require( "dojo.io.*" ); // use scripts from the io package
18
19    // configure calendar event handler
20    function connectEventHandler()
21    {
22      var calendar = dojo.widget.byId( "calendar" ); // get calendar
23      calendar.setDate( "2007-07-04" );
24      dojo.event.connect(
25        calendar, "onValueChanged", "retrieveItems" );
26    } // end function connectEventHandler
27
```

Import dojo
packages

Get the calendar, set the
date, and connect an
event handler so that
retrieveItems is
called when the calendar
is changed

```
28 // location of CalendarService web service
29 var webServiceUrl = "/CalendarService/CalendarService.asmx";
30
31 // obtain scheduled events for the specified date
32 function retrieveItems( eventData )
33 {
34 // convert date object to string in yyyy-mm-dd format
35 var date = dojo.date.toRfc3339( eventData ).substring( 0, 10 );
36
37 // build parameters and call web service
38 var params = '[{"param":"eventDate", "value":"' +
39     date + '"}]';
40 callWebService( 'getItemsByDate', params, displayItems );
41 } // end function retrieveItems
42
43 // call a specific web service asynchronously to get server data
44 function callWebService( method, params, callback )
45 {
46 // url for the asynchronous request
47 var requestUrl = webServiceUrl + "/" + method; var
48 params = paramString.parseJSON();
49
50 // build the parameter string to append to the url for ( var i
51 = 0; i < params.length; i++ )
52 {
53 // check if it is the first parameter and build
54 // the parameter string accordingly if ( i == 0 )
55 requestUrl = requestUrl + "?" + params[ i ].param +
56 "=" + params[ i ].value; // add first parameter to url
57
```

Build a JSON object
to hold the date
parameter and call

← the Import dojo
packages


```
58     else
59         requestUrl = requestUrl + "&" + params[ i ].param +
60             "=" + params[ i ].value; // add other parameters to url
61     } // end for
62
63     // call asynchronous request using dojo.io.bind
64     dojo.io.bind( { url: requestUrl, handler: callback,
65         accept: "application/json; charset=utf-8" } );
66 } // end function callWebService
67
68 // display the list of scheduled events on the page
69 function displayItems( type, data, event )
70 {
71     if ( type == 'error' ) // if the request has failed
72     {
73         alert( 'Could not retrieve the event' ); // display error
74     } // end if
75     else
76     {
77         var placeholder = dojo.byId( "itemList" ); // get placeholder
78         placeholder.innerHTML = ''; // clear placeholder
79         var items = data.parseJSON(); // parse server data
80
81         // check whether there are events;
82         // if none then display message if ( items == "" )
83         {
84             placeholder.innerHTML = 'No events for this date.';
85         }
86
87
```

dojo.io.bind
makes a call, sets a
callback handler, and
specifies what type of
data to

Get the placeholder
by its id

```
88 for ( var i = 0; i < items.length; i++ )
89 {
90     // initialize item's container
91     var item = document.createElement( "div" );
92     item.id = items[ i ].id; // set DOM id to database id
93
94     // obtain and paste the item's description
95     var text = document.createElement( "div" );
96     text.innerHTML = items[i].description;
97     text.id = 'description' + item.id;
98     dojo.dom.insertAtIndex( text, item, 0 );
99
100    // create and insert the placeholder for the edit button
101    var buttonPlaceholder = document.createElement( "div" );
102    dojo.dom.insertAtIndex( buttonPlaceholder, item, 1 );
103
104    // create the edit button and paste it into the container
105    var editButton = dojo.widget.
106        createWidget( "Button", {}, buttonPlaceholder );
107    editButton.setCaption( "Edit" );
108    dojo.event.connect(
109        editButton, 'buttonClick', handleEdit );
110
111    // insert item container in the list of items container
112    dojo.dom.insertAtIndex( item, placeholder, i );
113 } // end for
114 } // end else
115 } // end function displayItems
116
```

Use dojo's cross-browser DOM features to update the page

Create an edit button for the event

**Call the web service
to get the item to be
edited**

```
117 // send the asynchronous request to get content for editing and
118 // run the edit-in-place UI
119 function handleEdit( event )
120 {
121     var id = event.currentTarget.parentNode.id; // retrieve id
122     var params = ' [{ "param": "id", "value": "' + id + '" } ]';
123     callWebService( 'getItemById', params, displayForEdit );
124 } // end function handleEdit
125
126 // set up the interface for editing an item function
127 displayForEdit( type, data, event )
128 {
129     if ( type == 'error' ) // if the request has failed
130     {
131         alert( 'Could not retrieve the event' ); // display error
132     }
133     else
134     {
135         var item = data.parseJSON(); // parse the item
136         var id = item.id; // set the id
137
138         // create div elements to insert content
139         var editElement = document.createElement( 'div' );
140         var buttonElement = document.createElement( 'div' );
141
142         // hide the unedited content
143         var oldItem = dojo.byId( id ); // get the original element
144         oldItem.id = 'old' + oldItem.id; // change element's id
145         oldItem.style.display = 'none'; // hide old element
146         editElement.id = id; // change the "edit" container's id
```

```
147 // create a textbox and insert it on the page
148 var editArea = document.createElement( 'textarea' );
149 editArea.id = 'edit' + id; // set textbox id
150 editArea.innerHTML = item.description; // insert description
151 dojo.dom.insertAtIndex( editArea, editElement, 0 );
152
153
154 // create button placeholders and insert on the page
155 // these will be transformed into dojo widgets
156 var saveElement = document.createElement( 'div' ); var
157 cancelElement = document.createElement( 'div' );
158 dojo.dom.insertAtIndex( saveElement, buttonElement, 0 );
159 dojo.dom.insertAtIndex( cancelElement, buttonElement, 1 );
160 dojo.dom.insertAtIndex( buttonElement, editElement, 1 );
161
162 // create "save" and "cancel" buttons var saveButton =
163 dojo.widget.createWidget( "Button", {}, saveElement ); var
164 cancelButton =
165 dojo.widget.createWidget( "Button", {}, cancelElement );
166 saveButton.setCaption( "Save" ); // set saveButton label
167 cancelButton.setCaption( "Cancel" ); // set cancelButton text
168
169
170 // set up the event handlers for cancel and save buttons
171 dojo.event.connect( saveButton, 'buttonClick', handleSave );
172 dojo.event.connect(
173 cancelButton, 'buttonClick', handleCancel );
```

```
174         // paste the edit UI on the page
175         dojo.dom.insertAfter( editElement, oldItem );
176     } // end else
177 } // end function displayForEdit
178
179
180 // sends the changed content to the server to be saved
181 function handleSave( event )
182 {
183     // grab user entered data
184     var id = event.currentTarget.parentNode.parentNode.id;
185     var descr = dojo.byId( 'edit' + id ).value;
186
187     // build parameter string and call the web service
188     var params = '[{"param":"id", "value":"' + id +
189     '"}, {"param": "descr", "value":"' + descr + '"}]';
190     callWebService( 'save', params, displayEdited );
191 } // end function handleSave
192
193 // restores the original content of the item
194 function handleCancel( event )
195 {
196     var voidEdit = event.currentTarget.parentNode.parentNode;
197     var id = voidEdit.id; // retrieve the id of the item
198     dojo.dom.removeNode( voidEdit, true ); // remove the edit UI
199     var old = dojo.byId( 'old' + id ); // retrieve pre-edit version
200     old.style.display = 'block'; // show pre-edit version
201     old.id = id; // reset the id
202 } // end function handleCancel
203
```

← Cancel the edit by removing the form and showing the old item

```
204 // displays the updated event information after an edit is saved
205 function displayEdited( type, data, event )
206 {
207     if ( type == 'error' )
208     {
209         alert( 'Could not retrieve the event' );
210     }
211     else
212     {
213         editedItem = data.parseJSON(); // obtain updated description  var
214         id = editedItem.id; // obtain the id
215         var editElement = dojo.byId( id ); // get the edit UI
216         dojo.dom.removeNode( editElement, true ); // delete edit UI  var
217         old = dojo.byId( 'old' + id ); // get item container
218
219         // get pre-edit element and update its description
220         var oldText = dojo.byId( 'description' + id );
221         oldText.innerHTML = editedItem.description;
222
223         old.id = id; // reset id
224         old.style.display = 'block'; // show the updated item
225     } // end else
226 } // end function displayEdited
227
228 // when the page is loaded, set up the calendar event handler
229 dojo.addOnLoad( connectEventHandler );
230 // -->
```

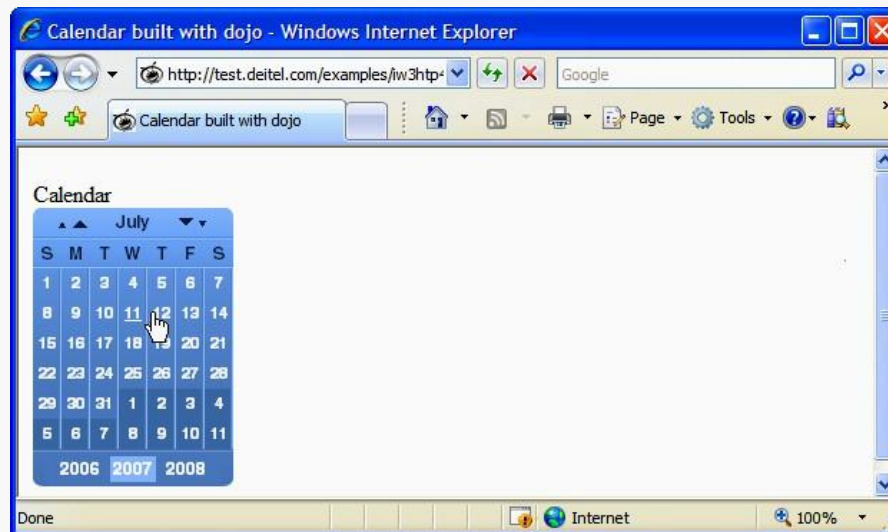
Set connectEventHandler
to be called onload

```

231 </script>
232 <title> Calendar built with dojo </title>
233</head>
234<body>
235   Calendar
236   <div dojoType = "datePicker" style = "float: left"
237       widgetID = "calendar"></div>
238   <div id = "itemList" style = "float: left"></div>
239</body>
240</html>

```

a) DatePicker Dojo widget
after the web page loads.

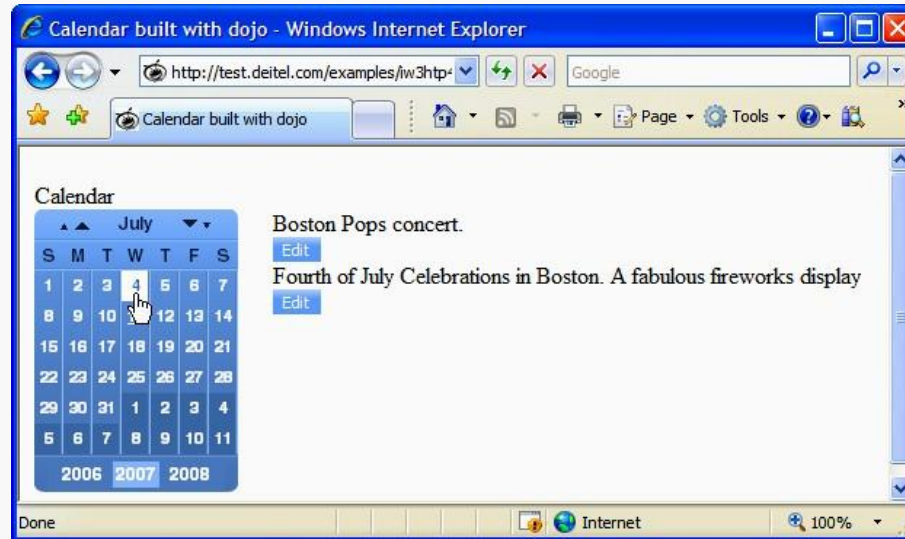


Outline

calendar.html

(9 of 11)

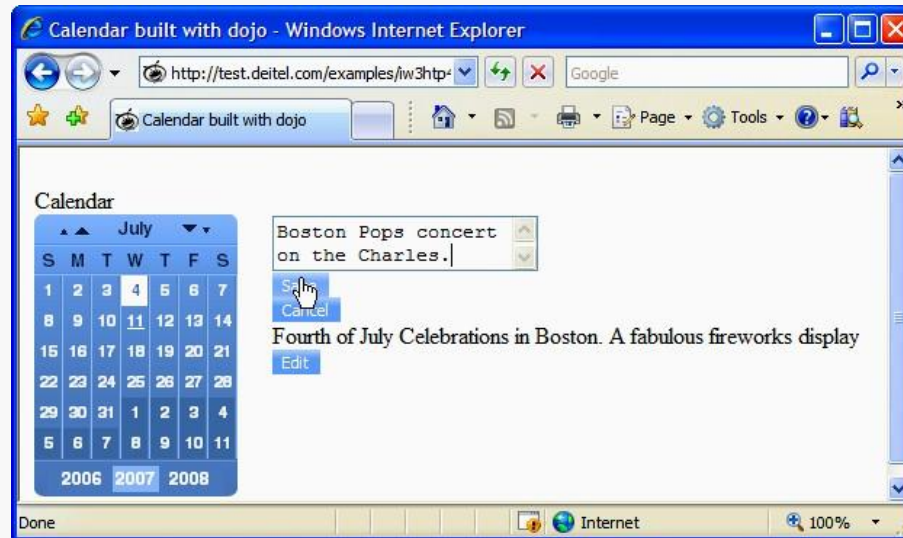
b) User selects a date and the application asynchronously requests a list of events for that date and displays the results with a partial page update.



c) User clicks the **Edit** button to modify an event's description.



d) Application performs a partial page update, replacing the original description and the **Edit** button with a text box, **Save** button and **Cancel** button. User modifies the event description and clicks the **Save** button.



e) The **Save** button's event handler uses an asynchronous request to update the server and uses the server's response to perform a partial page update, replacing the editing GUI components with the updated description and an **Edit** button.

