Chapter 2
A Taste of Multimedia

Li, Drew, & Liu

2.1 Multimedia Tasks and Concerns

- More basic concerns will impact multimedia as it now appears in products e.g., we expect impact from Computer Vision: a camera owner might be encouraged to think like a computer scientist and ask:

  - "What is going on in an image?"

    - A less high-level question is "Where has this image been taken?" *(scene recognition)*

  - "Does the image contain a particular object?" *(object classification)*

  - "Where is an object of interest?" *(object detection)*

  - "Which object does each pixel belong to?" *(image segmentation)*

== a classic Computer Vision hierarchy of high-level to detailed description of an image, with scene recognition at the top and image segmentation at the bottom.

Li, Drew, & Liu

2.2 Multimedia Presentation

- **Graphics Styles**: Human visual dynamics impact how presentations must be constructed.

  a) **Color principles and guidelines**: Some color schemes and art styles are best combined with a certain theme or style. A general hint is to not use *too many* colors, as this can be distracting.

  b) **Fonts**: For effective visual communication in a presentation, it is best to use large fonts (i.e., 18 to 36 points), and no more than 6 to 8 lines per screen (*fewer than on this screen*!). Fig. 2.1 shows a comparison of two screen projections:
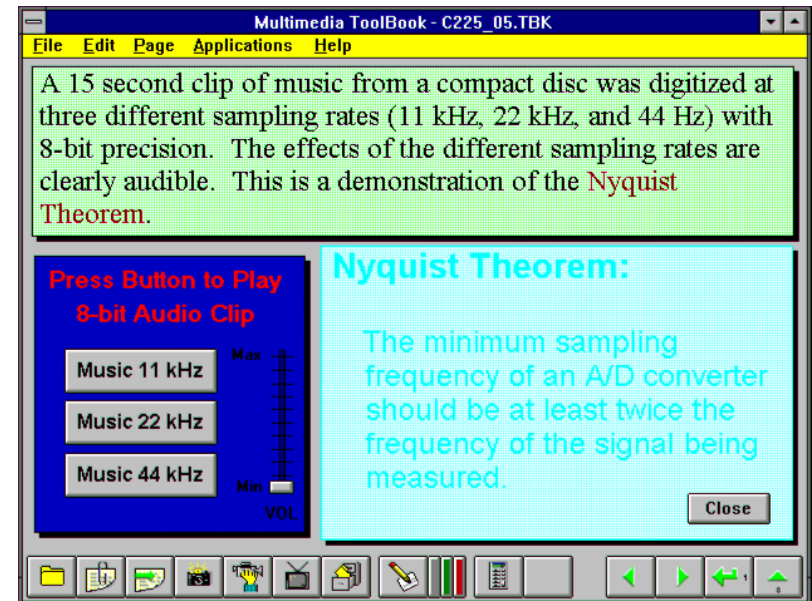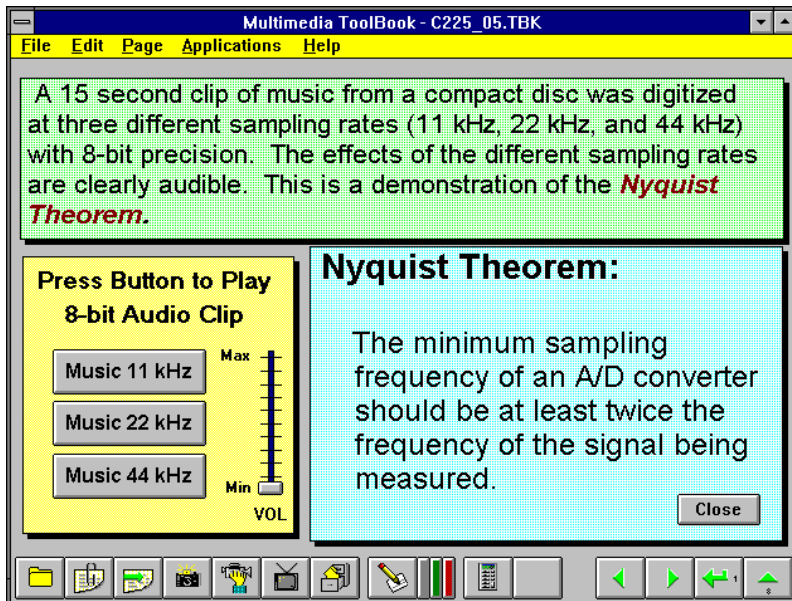
Li, Drew, & Liu

Fig. 2.1: Colours and fonts [from Ron Vetter].

c) **A color contrast program**: If the text color is some triple (R,G,B), a legible color for the background is that color subtracted from the maximum (here assuming max=1):

$$(R, \ G, \ B) \rightarrow (1 - R, \ 1 - G, \ 1 - B) \qquad (2.1)$$

- Some color combinations are more pleasing than others; e.g., a pink background and forest green foreground, or a green background and mauve foreground. Fig. 2.2 shows a small VB program (`textcolor.exe`) in operation:

  → [Link to TextColor src.zip](#)
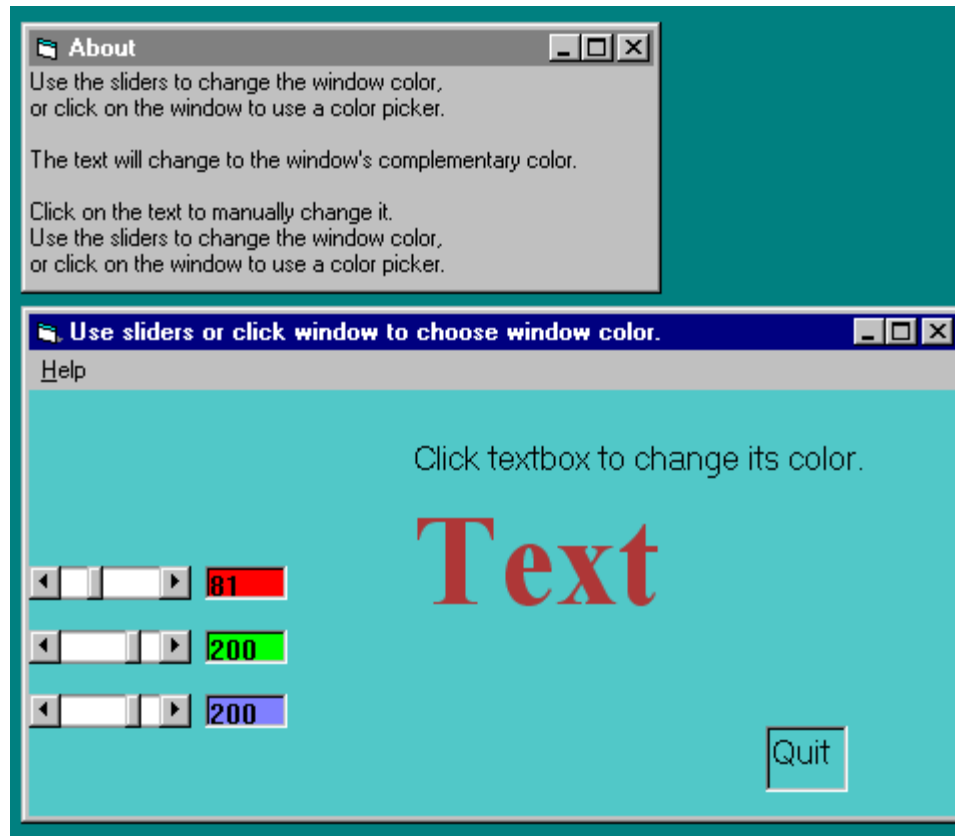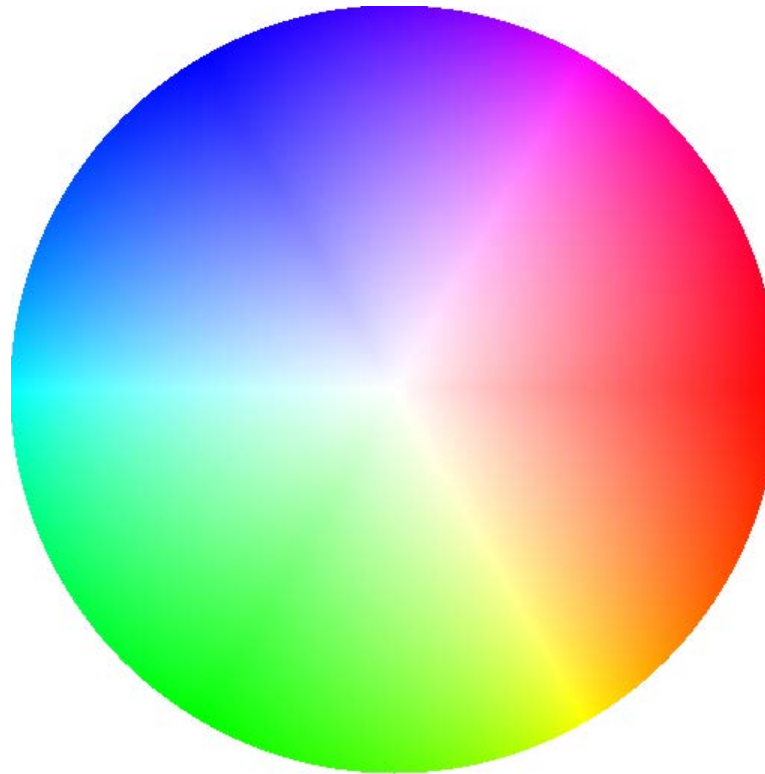
  → [Link to textcolor.exe](#)

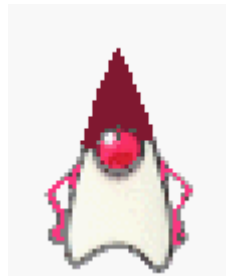**Fig. 2.2**: Program to investigate colors and readability.

- Fig. 2.3, shows a "colour wheel", with opposite colors equal to (1-R, 1-G, 1-B)

Fig. 2.3: Colour wheel

Sprite Animation

- **The basic idea**: Suppose we have an animation figure, as in Fig. 2.4 (a). Now create a 1-bit mask *M*, as in Fig. 2.4 (b), black on white, and accompanying sprite *S*, as in Fig. 2.4 (c).

(a)   (b)   (c)

Fig. 2.4: Sprite creation: Original, mask image *M*, and sprite *S* (*"Duke" figure courtesy of Sun Microsystems.*)

- We can overlay the sprite on a colored background B, as in Fig. 2.5 (a) by first **AND**ing B and M, and then **OR**ing the result with S, with final result as in Fig. 2.5 (e).

(a)   (b)   (c)

(d)   (e)

Fig. 2.5: Sprite animation: (a): Background B. (b): Mask M (c): B **AND** M.   (d): Sprite S (e): B **AND** M **OR** S

Video Transitions

- **Video transitions**: to signal "scene changes".

- Many different types of transitions:

1. **Cut**: an abrupt change of image contents formed by abutting two video frames consecutively. This is the simplest and most frequently used video transition.

2. **Wipe**: a replacement of the pixels in a region of the viewport with those from another video. Wipes can be left-to-right, right-to-left, vertical, horizontal, like an iris opening, swept out like the hands of a clock, etc.

3. **Dissolve**: replaces every pixel with a mixture over time of the two videos, gradually replacing the first by the second. Most dissolves can be classified as two types: **cross dissolve** and **dither dissolve**.

Type I: Cross Dissolve
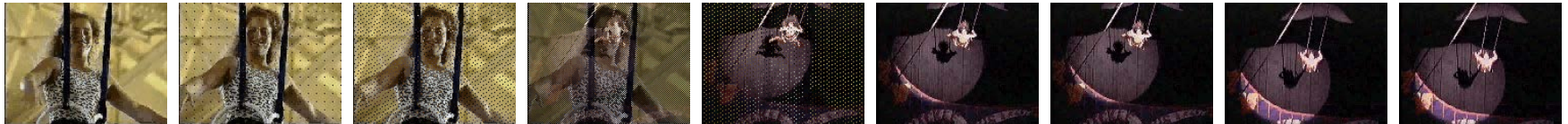
- Every pixel is affected gradually. It can be defined by:

$$\mathbf{D} = (1 - \alpha(t)) \cdot \mathbf{A} + \alpha(t) \cdot \mathbf{B}$$

(2.2)

where **A** and **B** are the color 3-vectors for video **A** and video **B**. Here, $\alpha(t)$ is a transition function, which is often linear:

$$\alpha(t) = k \cdot t, \quad \text{with } k \cdot t_{max} \equiv 1$$

(2.3)

Li, Drew, & Liu

Type II: Dither Dissolve

- Determined by $\alpha(t)$, increasingly more and more pixels in video A will abruptly (instead of gradually as in Type I) change to video B.

- Fade-in and fade-out are special types of Type I dissolve: video A or B is black (or white). Wipes are special forms of Type II dissolve in which changing pixels follow a particular geometric pattern.

- Build-your-own-transition: Suppose we wish to build a special type of wipe which slides one video out while another video slides in to replace it: a *slide* (or *push*).

(a) Unlike a wipe, we want each video frame not be held in place, but instead move progressively farther into (out of) the viewport.

(b) Suppose we wish to slide $Video_L$ in from the left, and push out $Video_R$. Figure 2.6 shows this process:

(a)   (b)   (c)

Fig. 2.6: (a): $Video_L$. (b): $Video_R$. (c): $Video_L$ sliding into place and pushing out $Video_R$.

Slide Transition (Cont'd)

- As time goes by, the horizontal location x_T for the transition boundary moves across the viewport from $x_T = 0$ at $t = 0$ to $x_T = x_{max}$ at $t = t_{max}$. Therefore, for a transition that is linear in time,

$$x_T = (t/t_{max})x_{max}$$

- So for any time t the situation is as shown in Fig. 2.7 (a). Let's assume that dependence on y is implicit since we use the same y as in the source video. Then for the red channel (and similarly for the green and blue), $R = R(x, t)$.

Li, Drew, & Liu

- Suppose that we have determined that pixels should come from Video$_L$.

  Then the *x*-position x_L in the unmoving video should be, $x_L = x + (x_{max} - x_T)$, where x is the position we are trying to fill in the viewport, x_T is the position in the viewport that the transition boundary has reached, and x_{max} is the maximum pixel position for any frame.

- From Fig. 2.7(b), we can calculate the position x_L in Video$_L$'s coordinate system as the sum of the distance *x*, in the viewport, plus the difference $x_{max} - x_T$.
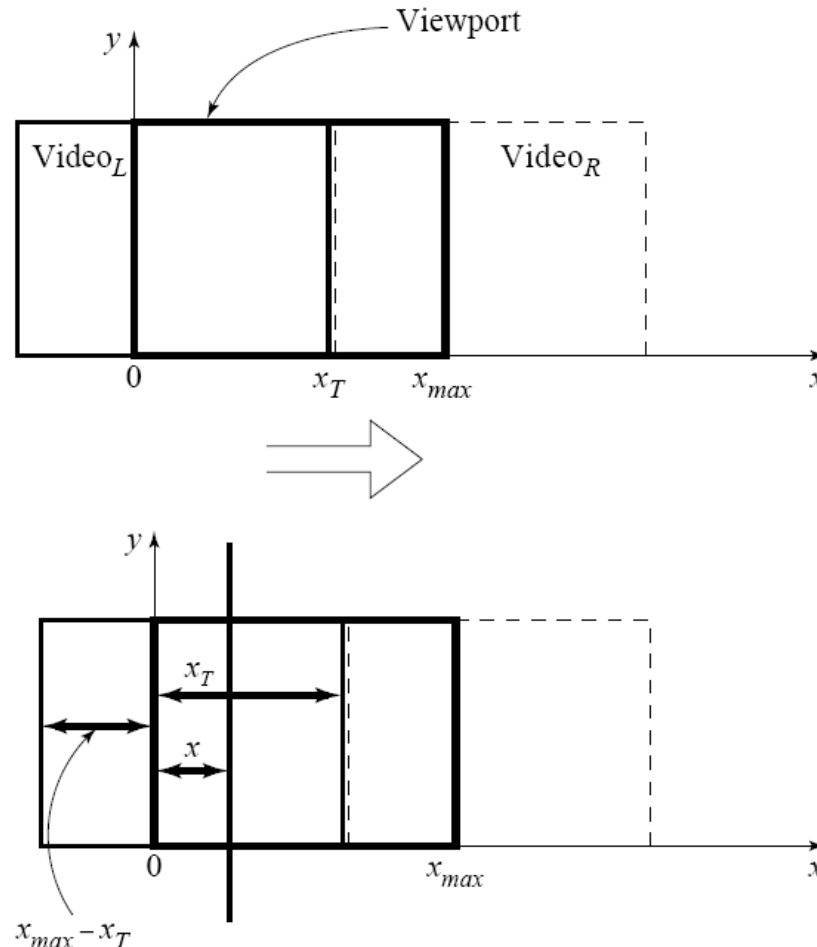
Fig. 2.7:
a) Geometry of Video$_L$ pushing out Video$_R$.
b) Calculating position in Video$_L$ from where pixels are copied to the viewport.

Slide Transition (Cont'd)

- Substituting the fact that the transition moves linearly with time, $x_T = x_{max}(t/t_{max})$, a pseudocode solution in shown in Fig. 2.8.

Fig 2.8: Pseudocode for slide video transition

1: **for** t in $0 .. t_{max}$ **do**
2:     **for** x in $0 .. x_{max}$ **do**
3:         **if** $\left(\frac{x}{x_{max}} < \frac{t}{t_{max}} \right)$ **then**
4:             $R = R_L(x + x_{max} * [1 - \frac{t}{t_{max}}], \ t)$
5:         **else**
6:             $R = R_R(x - x_{max} * \frac{t}{t_{max}}, \ t)$

Li, Drew, & Liu

Table 2.1: Uncompressed video sizes.

Standard Definition Video		
640×480 full color	=	922 kB/frame
@ 30 frames/sec	=	28 MB/sec
	=	221 Mb/sec
× 3,600 sec/hour	=	100 GB/hour
High Definition Video		
1,920×1,080 full color	=	6.2 MB/frame
@ 30 frames/sec	=	187 MB/sec
	=	1.5 Gb/sec
× 3,600 sec/hour	=	672 GB/hour

(a)   (b)

Fig. 2.9: JPEG compression: (a) original uncompressed image; (b) JPEG compression with Quality Factor Q=75 (the typical default).

(c)
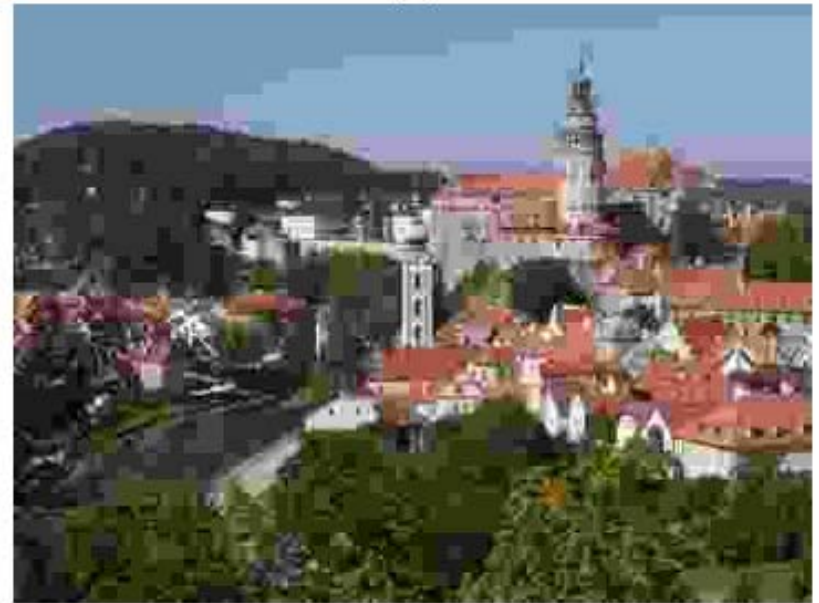
(d)

Fig. 2.9: JPEG compression: (c) Factor Q=25 and (d) Factor Q=5.

Table 2.2: JPEG file sizes (bytes) and percentage size of data for JPEG compression with Quality Factor Q=75, 25, and 5.

Quality Factor	Compressed File Size	Percentage of Original
-	529,620	100%
75	37,667	7.11%
25	16,560	3.13%
5	5,960	1.13%

- How expensive image and video processing is in terms of CPU cycles?

- Suppose we have an image whose pixels we wish to darken, by a factor of 2:

Algorithm 2.1: Darken by a factor of 2

1: **for** $x = 1$ to *columns* **do**
2:     **for** $y = 1$ to *rows* **do**
3:         image[x,y].red $/= 2$
4:         image[x,y].green $/= 2$
5:         image[x,y].blue $/= 2$

On a RISC machine, 12 instructions per pixel, i.e., per 3 bytes.

Li, Drew, & Liu

2.4 Multimedia Production

- Multimedia production can easily involve an art director, graphic designer, production artist, producer, project manager, writer, user interface designer, sound designer, videographer, and 3D and 2D animators, as well as programmers.

2.5 Multimedia Sharing and Distribution

Fig. 2.10: The webpage for uploading a YouTube video. The video, titled "Edison Phonograph Multimedia Textbook" is open to all users (Privacy settings: Public) and can be searched in the YouTube homepage using the title or tags. Note that video thumbnails are automatically generated by YouTube.
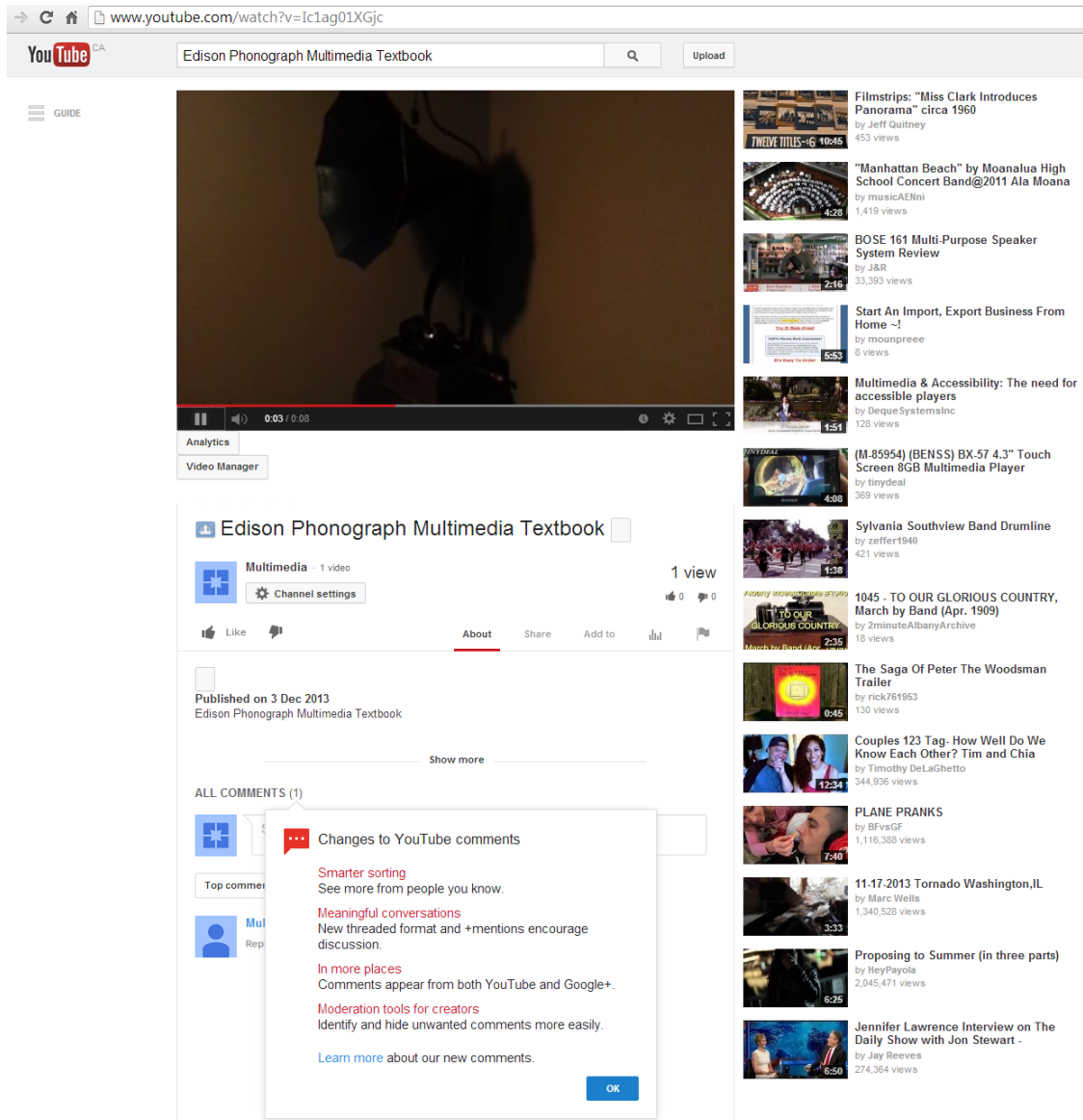
Fig. 2.11: The YouTube page for the video uploaded. The list of related videos are shown on the right side, and users can post their comments, as well.

2.6 Some Useful Editing and Authoring Tools

- One needs real vehicles for showing understanding principles   of and creating multimedia. And straight programming in C++ or Java is not always the best way of showing your knowledge and creativity.

- Some popular authoring tools include the following:
    - **Adobe Premiere**
    - **Adobe Director**
    - **Adobe Flash**
    - **Cakewalk Pro Audio**

- **Hint for studying this section**: Hands-on work in a laptop or Lab environment, with reference to the text.