11

MULTIMEDIA PRESENTATION AND AUTOMATIC AUTHORING

11.1 Objectives
11.2 Introduction
11.3 Multimedia Presentation
11.4 Automatic Authoring
11.5 Summary
11.6 Unit End Exercises

11.1 OBJECTIVES

After studying this Unit, you will be able to:

1. Understand graphic styles and fonts.
2. Learn Externalization versus Linearization.
3. Note Color Principles and Guidelines.

11.2 INTRODUCTION

Multimedia authoring is the creation of multimedia productions, sometimes called "movies" or "presentations". Since we are interested in this subject from a computer science point of view, we are mostly interested in interactive applications. Also, we need to consider still-image editors, such as Adobe Photoshop, and simple video editors, such as Adobe Premiere, because these applications help us create interactive multimedia projects. In this unit, we briefly outline some effects to keep in
mind for presenting multimedia content as well as some useful guidelines for content design.

11.3 MULTIMEDIA PRESENTATION

Graphics Styles
Careful thought has gone into combinations of color schemes and how lettering is perceived in a presentation. Many presentations are meant for business displays, rather than appearing on a screen. Human visual dynamics are considered in regard to how such presentations must be constructed. Most of the observations here are drawn from Vetter et al., as in Figure 11.1.
Fig. 11.1: Colors and fonts. (This figure also appears in the color insert section.) Courtesy, Ron Vetter.

Color Principles and Guidelines
Some color schemes and art styles are best combined with a certain theme or style. Color schemes could be, for example, natural and floral for outdoor scenes and solid colors for indoor scenes. Examples of art styles are oil paints, watercolors, colored pencils, and pastels. A general hint is not to use too many colors, as this can be distracting. It helps to be consistent with the use of color - then color can be used to signal changes in theme.

Fonts
For effective visual communication, large fonts (18 to 36 points) are best, with no more than six to eight lines per screen. As shown in Figure 11.1, sans serif fonts work better than serif fonts (serif fonts are those with short lines stemming from and at an angle to the upper and lower ends of a
letter's strokes). Figure 11.1 shows a comparison of twoscreen projections, (Figure 2 and 3 from Vetter, Ward and Shapiro).

The top figure shows good use of color and fonts. It has a consistent color scheme, uses large and all sans-serif (Arial) fonts. The bottom figure is poor, in that too many colors are used, and they are inconsistent. The red adjacent to the blue is hard to focus on, because the human retina cannot focus on these colors simultaneously. The serif (Times New Roman) font is said to be hard to read in a darkened, projection setting. Finally, the lower right panel does not have enough contrast - pretty pastel colors are often usable only if their background is sufficiently different.

A Color Contrast Program
Seeing the results of Vetter et al.'s research, we constructed a small Visual Basic program to investigate how readability of text colors depends on foreground color and the color of the background.

The simplest approach to making readable colors on a screen is to use the principal complementary color as the background for text. For color values in the range 0 to 1 (or, effectively, 0 to 255), if the text color is some triple (R, G, B), a legible color for the background is likely given by that color subtracted from the maximum:

\[(R,G,B) \rightarrow (1-R,1-G,1-B)\]

That is, not only is the color "opposite" in some sense (not the same sense as artists use), but if the text is bright, the background is dark, and vice versa.

In the Visual Basic program given below, sliders can be used to change the background color. As the background changes, the text changes to equal the principal complementary color. Clicking on the background brings up a color-picker as an alternative to the sliders.
If you feel you can choose a better color combination, click on the text. This brings up a color picker not tied to the background color, so you can experiment. (The text itself can also be edited.) A little experimentation shows that some color combinations are more pleasing than others - for example, a pink background and forest green foreground, or a green background and mauve foreground. Figure 11.2 shows this small program in operation.

**Fig. 11.2:** Program to investigate colors and readability

Figure 11.3 shows a "color wheel", with opposite colors equal to (1-R, 1-G, 1-B). An artist's color wheel will not look the same, as it is based on feel rather than on an algorithm. In the traditional artist's wheel, for example, yellow is opposite magenta, instead of opposite blue as in Figure 11.3, and blue is instead opposite orange.
**Sprite Animation**

Sprites are often used in animation. For example, in Macromedia Director, the notion of a sprite is expanded to an instantiation of any resource. However, the basic idea of sprite animation is simple. Suppose we have produced an animation figure, as in Figure 11.4(a). Then it is a simple matter to create a 1-bit mask $M$, as in Figure 11.4(b), black on white, and the accompanying sprite $S$, as in Figure 11.4(c).

Now we can overlay the sprite on a colored background $E$, as in Figure 11.5(a), by first ANDing Band $M$, then ORing the result with $S$, with the final result as in Figure 11.5(e). Operations are available to carry out these simple compositing manipulations at frame rate and so produce a simple 2D animation that moves the sprite around the frame but does not change the way it looks.
**Video Transitions**

Video transitions can be an effective way to indicate a change to the next section. Video transitions are syntactic means to signal "scene changes" and often carry semantic meaning. Many different types of transitions exist; the main types are cuts, wipes, dissolves, fade-ills and fade-outs.
A cut, as the name suggests, carries out an abrupt change of image contents in two consecutive video frames from their respective clips. It is the simplest and most frequently used video transition.

A wipe is a replacement of the pixels in a region of the viewport with those from another video. If the boundary line between the two videos moves slowly across the screen, the second video gradually replaces the first. Wipes can be left-to-right, right-to-left, vertical, and horizontal, like an iris opening, swept out like the hands of a clock, and so on.

A dissolve replaces every pixel with a mixture over time of the two videos, gradually changing the first to the second. A fade-out is the replacement of a video by black (or white), and fade-in is its reverse.

Some Technical Design Issues

Technical parameters that affect the design and delivery of multimedia applications include computer platform, video format and resolution, memory and disk space, delivery methods.

Computer Platform

Usually we deal with machines that are either some type of UNIX box (such as a Sun) or else a PC or Macintosh. While a good deal of software is ostensibly "portable", much cross-platform software relies on runtime modules that may not work well across systems.

```python
for t in 0..t_max
    for x in 0..x_max
        if (x/x_max < t/t_max)
            R = R_L (x + x_max * [1 - t/t_max], t)
        else
            R = R_H (x - x_max * t/t_max, t)
```

Fig. 11.8:- Pseudo code for slide video transition
Video Format and Resolution
The most popular video formats are NTSC, PAL, and SECAM. They are not compatible, so conversion is required to play a video in a different format. The graphics card, which displays pixels on the screen, is sometimes referred to as a “video card”. In fact, some cards are able to perform “frame grabbing”, to change analog signals to digital for video. This kind of card is called a “video capture card”.

The graphics card's capacity depends on its price. An old standard for the capacity of a card is S-VGA, which allows for a resolution of 1280x1024 pixels in a displayed image and as many as 65,536 colors using 16-bit pixels or 16.7 million colors using 24-bit pixels. Nowadays, graphics cards that support higher resolution, such as 1600x1200, and 32-bit pixels or more are common.

Memory and Disk Space Requirement
Rapid progress in hardware alleviates the problem, but multimedia software is generally greedy. Nowadays, at least 1 Gigabyte of RAM and 20 gigabytes of hard disk space should be available for acceptable performance and storage for multimedia programs.

Delivery Methods
Once coding and all other work is finished, how shall we present our clever work? Since we have presumably purchased a large disk, so that performance is good and storage is not an issue, we could simply bring along our machine and show the work that way. However, we likely wish to distribute the work as a product. Presently, rewritable DVD drives are not the norm, and CD-ROMs may lack sufficient storage capacity to hold the presentation. Also, access time for CD-ROM drives is longer than for hard-disk drives.

Electronic delivery is an option, but this depends on network bandwidth at the userside (and at our server). A streaming option may be available,
depending on the presentation. No perfect mechanism currently exists to distribute large multimedia projects. Nevertheless, using such tools as PowerPoint or Director, it is possible to create acceptable presentations that fit on a single CD-ROM.

### 11.4 AUTOMATIC AUTHORING

Thus far, we have considered notions developed for authoring new multimedia. Nevertheless, a tremendous amount of legacy multimedia documents exists, and researchers have been interested in methods to facilitate automatic authoring. By this term is meant either an advanced helper for creating new multimedia presentations or a mechanism to facilitate automatic creation of more useful multimedia documents from existing sources.

**Hypermedia Documents**

Let us start by considering hypermedia documents. Generally, three steps are involved in producing documents meant to be viewed nonlinearly: information generation or capture, authoring, and publication. A question that can be asked is, how much of this process can be automated?

The first step, capture of media, be it from text or using an audio digitizer or videoframe-grabber, is highly developed and well automated. The final step, presentation, is the objective of the multimedia tools we have been considering. But the middle step (authoring) is most under consideration here.

Essentially, we wish to structure information to support access and manipulation of the available media. Clearly, we would be well advised to consider the standard computing science data structures in structuring this information: lists, trees, or networks (graphs). However, here we would
like to consider how best to structure the data to support multiple views, rather than a single, static view.

**Externalization versus Linearization**

Figure 11.9 shows the essential problem involved in communicating ideas without using a hypermedia mechanism: the author's ideas are “linearized” by setting them down in linear order on paper. In contrast, hyperlinks allow us the freedom to partially mimic the author's thought process (i.e., externalization).

---

**Fig. 11.9:** Communication using hyperlinks. Courtesy of David Lowe; (© 1995 IEEE).
Now, using Microsoft Word, say, it is trivial to create a hypertext version of one's document, as Word simply follows the layout already set up in chapters, headings, and so on. But problems arise when we wish to extract semantic content and find links and anchors, even considering just text and not images. Figure 11.10 displays the problem: while it is feasible to mentally manage a few information nodes, once the problem becomes large, we need automatic assistants.

![Complex information space: (a) complexity: manageable; (b) complexity: overwhelming. Courtesy of David Lowe; (@1995 IEEE).]

Once a dataset becomes large, we should employ database methods. The issues become focused on scalability (to a large dataset), maintainability, addition of material, and reusability. The database information must be set up in such a way that the “publishing” stage, presentation to the user, can be carried out just-in-time, presenting information in a user-defined view from an intermediate information structure.

**Semiautomatic Migration of Hypertext**

The structure of hyperlinks for text information is simple: "nodes" represent semantic information and are anchors for links to other pages. Figure 11.11 illustrates these concepts.
For text, the first step for migrating paper-based information to hypertext is to automatically convert the format used to HTML. Then, sections and chapters can be placed in a database. Simple versions of data mining techniques, such as word stemming, can easily be used to parse titles and captions for keywords - for example, by frequency counting. Keywords found can be added to the database being built. Then a helper program can automatically generate additional hyperlinks between related concepts.

A semiautomatic version of such a program is most likely to be successful, making suggestions that can be accepted or rejected and manually added to. A database management system can maintain the integrity of links when new nodes are inserted. For the publishing stage, since it may be impractical to re-create the underlying information structures, it is best to delay imposing a viewpoint on the data until as late as possible.

Hyperimages

Matters are not nearly so straightforward when considering image or other multimedia data. To treat an image in the same way as text, we would wish to consider an image to be a node that contains objects and other anchors, for which we need to determine image entities and rules. What we
desire is an automated method to help us produce true hypermedia, as in Figure 11.12.

**Fig. 11.12:** Structure of hypermedia. *Courtesy of David Lowe.*

It is possible to manually delineate syntactic image elements by masking image areas. These can be tagged with text, so that previous text-based methods can be brought into play.

## 11.5 SUMMARY

## 11.6 UNIT END EXERCISES

Reference: