Chapter 2
Multimedia Authoring and Tools

Multimedia Authoring
2.1 Multimedia Authoring

- **Multimedia authoring**: creation of multimedia productions, sometimes called “movies” or “presentations”.

  - we are mostly interested in **interactive** applications.

  - For practicality, we also have a look at still-image editors such as Adobe Photoshop, and simple video editors such as Adobe Premiere.

- In this section, we take a look at:

  - Multimedia Authoring Metaphors
  - Multimedia Production
  - Multimedia Presentation
  - Automatic Authoring
2.1.1 Multimedia Authoring Metaphors

1. Authoring is the process of creating multimedia applications.
2. Metaphors are methodologies employed to create multimedia applications.
3. Some common authoring metaphors are:
   1. Scripting language metaphor
   2. Slide show metaphor
   3. Hierarchical metaphor
   4. Iconic/flow-control metaphor
   5. Frames metaphor
   6. Card/scripting metaphor
   7. Cast/score/scripting metaphor
Multimedia Authoring Metaphors

1. **Scripting Language Metaphor**: use a special language to enable interactivity (buttons, mouse, etc.), and to allow conditionals, jumps, loops, functions/macros etc.

2. **Slide Show Metaphor**: A linear presentation by default, although tools exist to perform jumps in slide shows, EX. PowerPoint or ImageQ.

3. **Hierarchical Metaphor**: User-controllable elements are organized into a tree structure — often used in menu-driven applications.
4. **Iconic/Flow-control Metaphor**: Graphical icons are available in a toolbox, and authoring proceeds by creating a flow chart with icons attached (Fig. 2.1), EX. Authorware by Macromedia. As well as simple flowchart elements, such as an IF statement, a CASE statement, group of elements using MAP (subroutine). Also, simple animation is possible.
5. **Frames Metaphor**: Like Iconic/Flow-control Metaphor; however links between icons are more conceptual, rather than representing the actual flow of the program (Fig. 2.2), EX. Quest by Allen Communication. Flowchart consists of modules composed of frames. Frames are constructed from objects, such as text, graphics, audio, animations and video, all of which can respond to events.
6. **Card/Scripting Metaphor**: Uses a simple index-card structure — easy route to producing applications that use hypertext or hypermedia; used in schools.

![Fig. 2.3: Two Cards in a Hypermedia Stack](image-url)
7. Cast/Score/Scripting Metaphor:

• Time is shown horizontally; like a spreadsheet: rows, or tracks, represent instantiations of characters in a multimedia production.

• Multimedia elements are drawn from a cast of characters, and scripts are basically event-procedures or procedures that are triggered by timer events.

• Director, by Macromedia, is the chief example of this metaphor. Director uses the Lingo scripting language, an object-oriented event-driven language.
2.1.2 Multimedia Production

• People produce multimedia 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 programmer.

• Multimedia production design phase consists of storyboarding, flowcharting, prototyping and user testing as well as a parallel production of media.
  • A storyboard depicts the initial idea content of a multimedia concepts in a series of sketches (keyframes).
  • A flowchart organizes the storyboards by inserting navigation information (multimedia concept’s structure and user interaction).
  • A prototype includes development of detailed functional specification (walk-through, screen action, user interface).
  • User testing is extremely important before final development phase (tools may be used).
2.1.3 Multimedia Presentation

• Features that affect presenting multimedia content:
  1. **Graphics Styles**: Human visual dynamics are considered in regard to how such presentations must be constructed.
  2. **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.
  3. **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. 2.4 shows a comparison of two screen projections:
Fig. 2.4: Colours and fonts [from Ron Vetter].
4. **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):

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

(2.1)

- This is called the principal complementary color, for color values in the range 0 to 1 (or effectively 0 to 255)
- 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. 2.5 shows a small VB program (textcolor.exe) in operation:
Fig. 2.5: Program to investigate colours and readability.
• A “colour wheel”, with opposite colours equal to $(1-R, 1-G, 1-B)$
5. Sprite Animation

• The basic idea: Suppose we have an animation figure, as in Fig. 2.7 (a). Now create a 1-bit mask $M$, as in Fig. 2.7 (b), black on white, and accompanying sprite $S$, as in Fig. 2.7 (c).

Fig. 2.7: 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. 2.8 (a) by first ANDing $B$ and $M$, and then ORing the result with $S$, with final result as in Fig. 2.8 (e).

Fig. 2.8: Sprite animation: (a): Background $B$. (b): Mask $M$. (c): $B$ AND $M$. (d): Sprite $S$. (e): $B$ AND $M$ OR $S$
6. Video Transitions

- **Video transitions**: is an effective way to indicate a change to the next section. It is a semantic means to signal “scene changes” and often carry semantic meaning. 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:

\[
D = (1 - \alpha(t)) \cdot A + \alpha(t) \cdot B
\]  

(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)
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.
Slide Transition

• 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*).

• Each Video\(_L\) and Video\(_R\) (shown in Fig 2.9), has its own values R, G and B. R is a function of position in the frame \((x,y)\) as well as of time \(t\).

• Since this is video and not collection of images of various sizes, each of the two videos has the same maximum extent, \(x_{\text{max}}\)
(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 2.9 shows this process:

Fig. 2.9: (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 $x_T = x_{\text{max}}$ at $t = t_{\text{max}}$. Therefore, for a transition that is linear in time, $x_T = (t/t_{\text{max}})x_{\text{max}}$.

- So for any time $t$ the situation is as shown in Fig. 2.10 (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), $R = R(x, t)$. 
• Suppose that we have determined that pixels should come from Video\textsubscript{L}. Then the \( x \)-position \( x_L \) in the \textit{unmoving} video should be \( x_L = x + (x_{\text{max}} - x_T) \), 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 \( x_{\text{max}} \) is the maximum pixel position for any frame.

• From Fig. 2.10(b), we can calculate the position \( x_L \) in Video\textsubscript{L}'s coordinate system as the sum of the distance \( x \), in the viewport, plus the difference \( x_{\text{max}} - x_T \).
Fig. 2.10: (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, $x_T = x_{\text{max}}(t/t_{\text{max}})$, a pseudocode solution is shown in Fig. 2.11.

```
for t in 0..t_{\text{max}}
    for x in 0..x_{\text{max}}
        if (x/x_{\text{max}} < t/t_{\text{max}})
            R = R_L ( x + x_{\text{max}} \times [1 - t/t_{\text{max}}], t)
        else
            R = R_R ( x - x_{\text{max}} \times t/t_{\text{max}}, t)
```

Fig. 2.11: Pseudocode for slide video transition
Some Technical Design Issues

1. **Computer Platform**: Much software is ostensibly “portable” but cross-platform software relies on run-time modules which may not work well across systems.

2. **Video format and resolution**: The most popular video formats — NTSC, PAL, and SECAM — are not compatible, so a conversion is required before a video can be played on a player supporting a different format.

3. **Memory and Disk Space Requirement**: At least 128 MB of RAM and 20 GB of hard-disk space should be available for acceptable performance and storage for multimedia programs.
4. Delivery Methods:

• Not everyone has rewriteable DVD drives, as yet.

• CD-ROMs: may be not enough storage to hold a multimedia presentation. As well, access time for CD-ROM drives is longer than for hard-disk drives.

• Electronic delivery is an option, but depends on network bandwidth at the user side (and at server). A streaming option may be available, depending on the presentation.
2.1.4 Automatic Authoring

- Facilitating automatic authoring is either by creating new multimedia presentations or by automatic creation of more useful multimedia documents from existing resources.

- **Hypermedia documents**: Generally, three steps are to produce documents to be viewed nonlinearly:
  
  1. **Capture of media**: From text or using an audio digitizer or video frame-grabber; is highly developed and well automated.
  2. **Authoring**: How best to structure the data in order to support multiple views of the available data, rather than a single, static view.
  3. **Publication**: i.e. Presentation, is the objective of the multimedia tools we have been considering.

- Standard computing science data structures are recommended in structuring this information to support multiple views.
• **Externalization versus linearization:**

(a) Fig. 2.12(a) shows the essential problem involved in communicating ideas without using a hypermedia mechanism.

(b) In contrast, hyperlinks allow us the freedom to partially mimic the author’s thought process (i.e., externalization).

(c) Using, e.g., Microsoft Word, creates a hypertext version of a document by following the layout already set up in chapters, headings, and so on. But problems arise when we actually need to automatically extract **semantic** content and find *links* and anchors (even considering just text and not images etc.) Fig. 2.13 displays the problem.
Fig. 2.12: Communication using hyperlinks [from David Lowe].
(d) 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.

Fig. 2.13: Complex information space [from David Lowe].
Semi-automatic migration of hypertext

The structure of hyperlinks for text information is simple: “nodes” represent semantic information and these are anchors for links to other pages.

---

Textual data has been widely used to construct and manipulate information in multimedia systems. If we consider the use of textual data within multimedia we can see that text can be stored, analysed, manipulated, generated synthetically, and extracted for use elsewhere. Essentially textual information can be grouped into ‘nodes’. Each node can be treated as consisting of discrete entities (words, sentences, paragraphs etc.) which obey a series of syntactic and semantic rules describing the interrelationships.
• 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 replaced in a database.

• Simple versions of data mining techniques such as word stemming can easily be used to parse titles and captions for keywords.

• Keywords found can be added to the database being built. The a helper program can automatically generate additional hyperlinks between related concepts.
• **Hyperimages**

• We need an automated method to help us produce true hypermedia:

---

Fig. 2.15: Structure of hypermedia [from David Lowe].
Can manually delineate syntactic image elements by masking image areas. Fig. 2.16 shows a “hyperimage”, with image areas identified and automatically linked to other parts of a document:

It is interested to know what tools from database systems, data mining and artificial intelligence can be brought to assist production of full-blown multimedia systems.