Chapter 2
Multimedia Authoring and Tools

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**

– 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. E.g., a small Toolbook program is as below:

```
-- load an MPEG file
extFileName of MediaPlayer "theMpegPath" =
       "c:\windows\media\home33.mpg";
-- play
extPlayCount of MediaPlayer "theMpegPath" = 1;
-- put the MediaPlayer in frames mode (not time mode)
extDisplayMode of MediaPlayer "theMpegPath" = 1;
-- if want to start and end at specific frames:
extSelectionStart of MediaPlayer "theMpegPath" = 103;
extSelectionEnd of MediaPlayer "theMpegPath" = 1997;
-- start playback
get extPlay() of MediaPlayer "theMpegPath";
```

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

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):

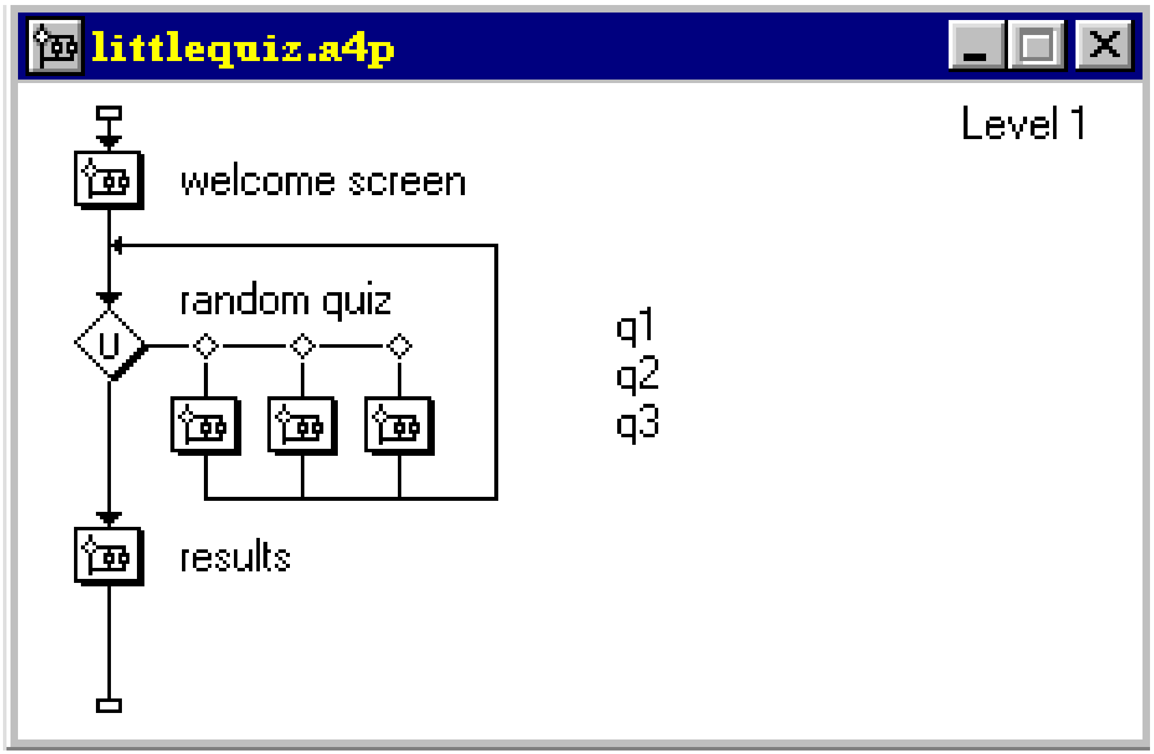


Fig. 2.1: Authorware flowchart

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):

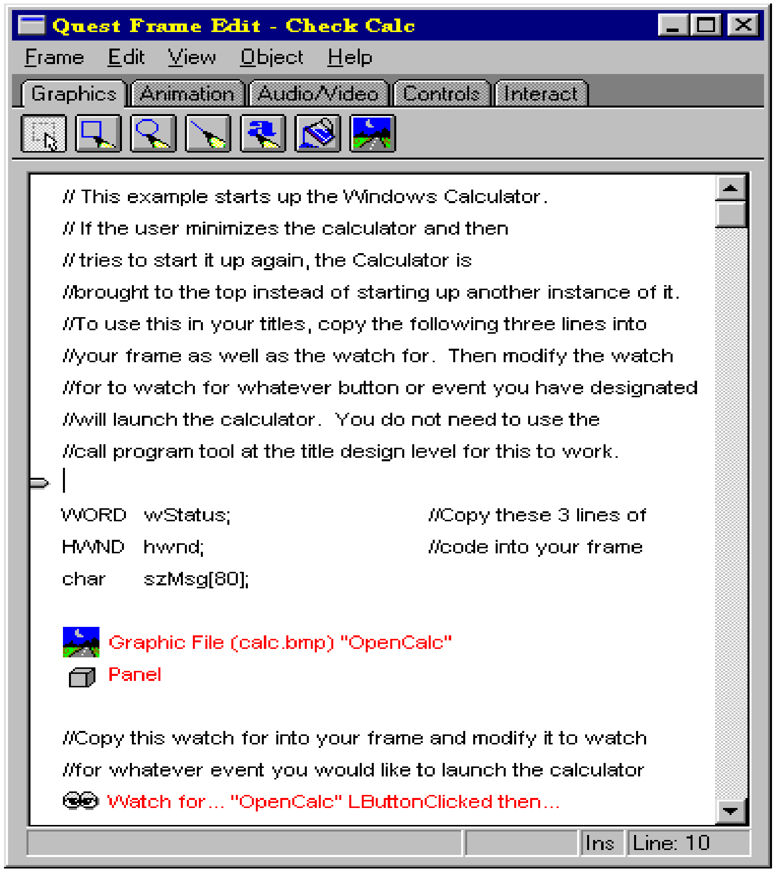


Fig. 2.2: Quest Frame

6. **Card/Scripting Metaphor**: Uses a simple index-card structure — easy route to producing applications that use hypertext or hypermedia; used in schools.

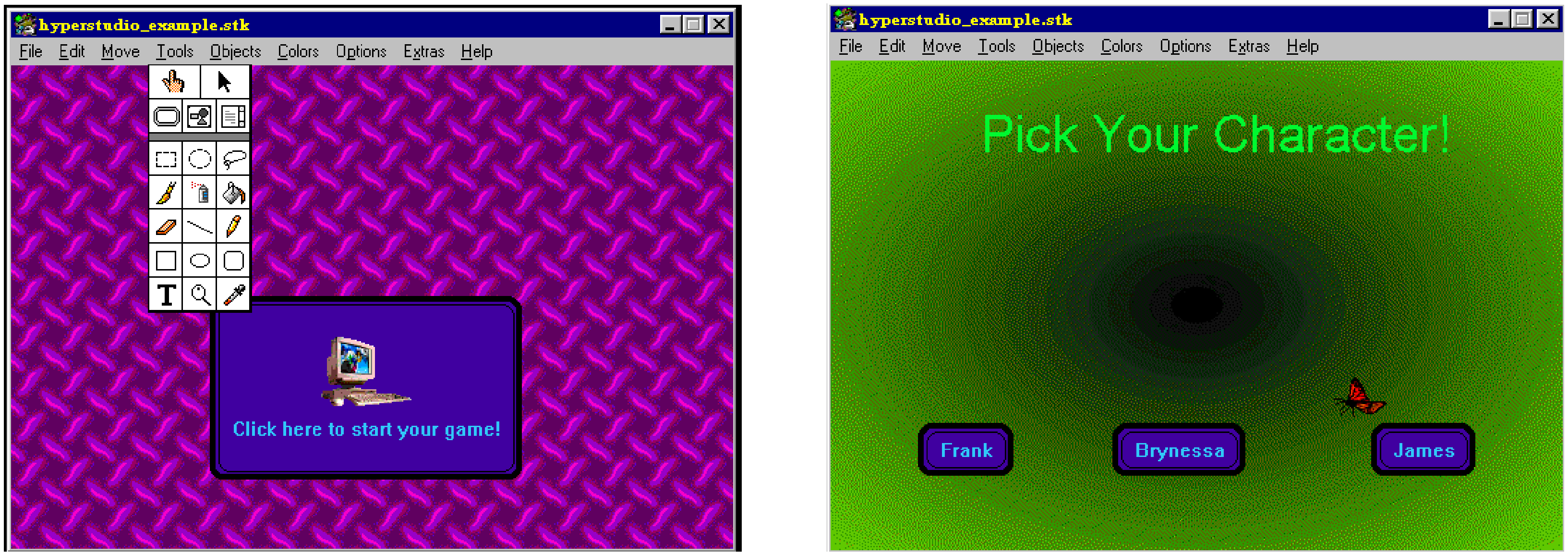


Fig. 2.3: Two Cards in a Hypermedia Stack

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.

− 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. 2.4 shows a comparison of two screen projections:

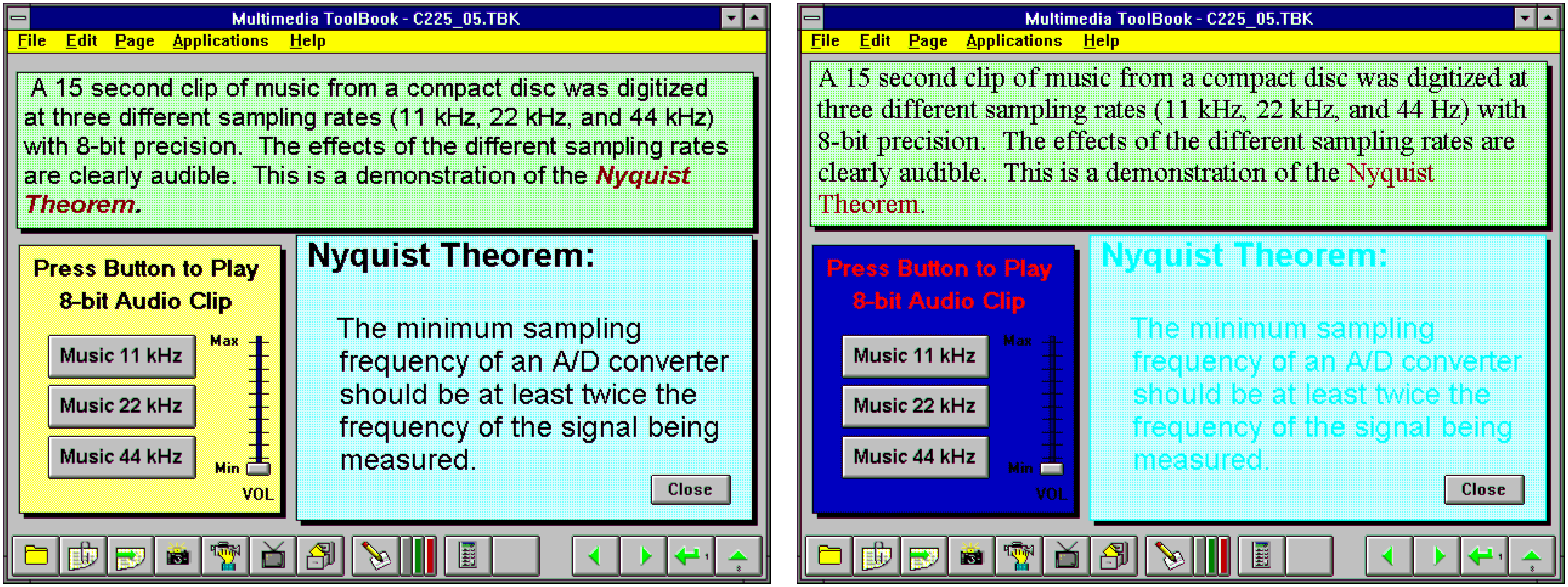


Fig 2.4: Colors and fonts [from Ron Vetter].

(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):

$$(R, G, B) \Rightarrow (1 - R, 1 - G, 1 - B) \tag{2.1}$$

– 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. 2.5 shows a small VB program (`textcolor.exe`) in operation:

⟶ Link to TextColor_src.zip

⟶ Link to textcolor.exe

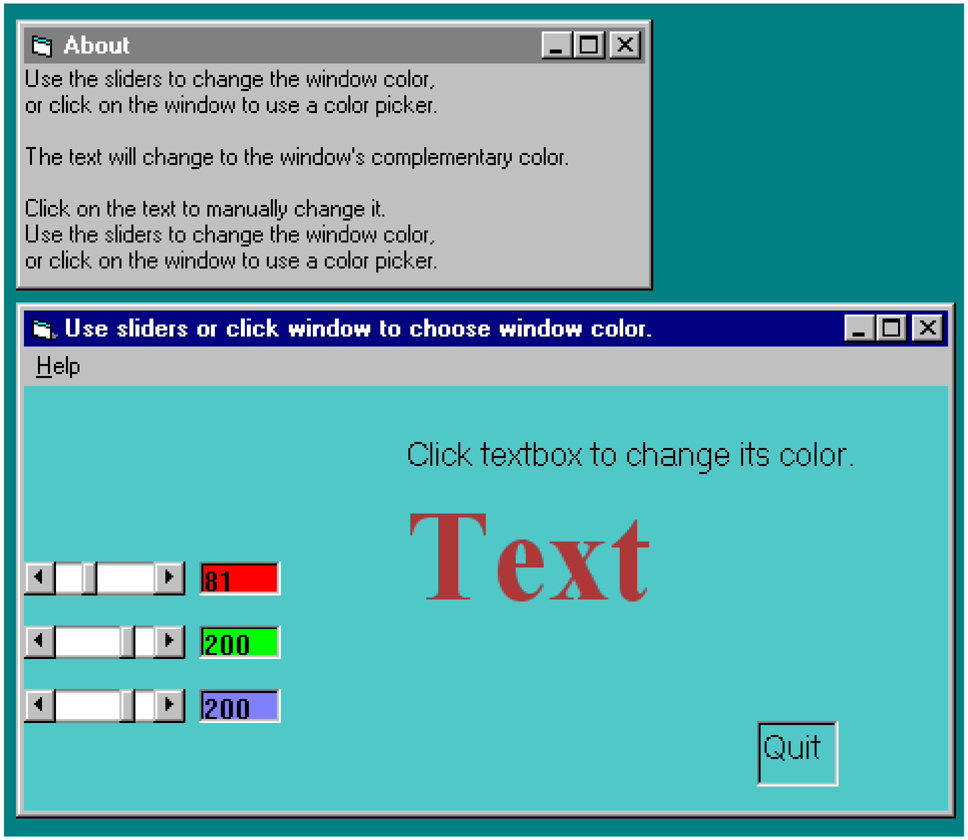


Fig. 2.5: Program to investigate colors and readability.

– Fig. 2.6, shows a "color wheel", with opposite colors equal to (1-R,1-G,1-B):

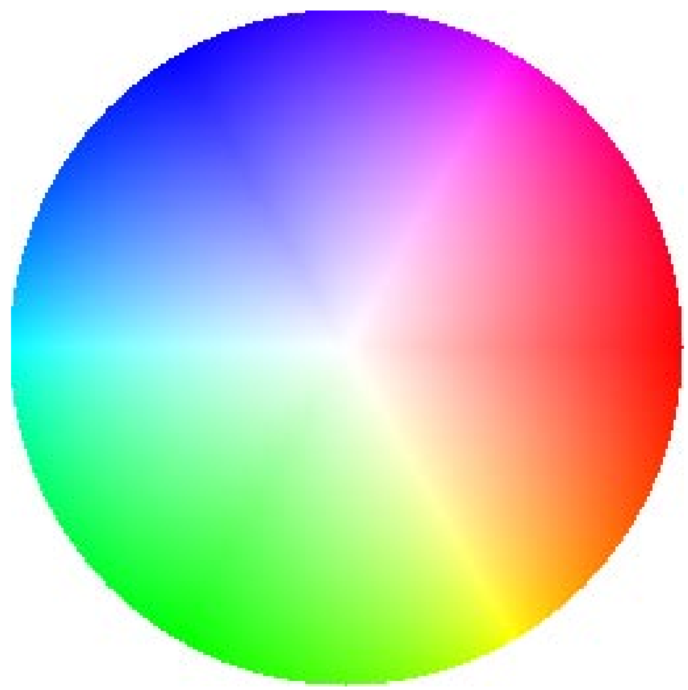


Fig. 2.6: Color wheel

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).

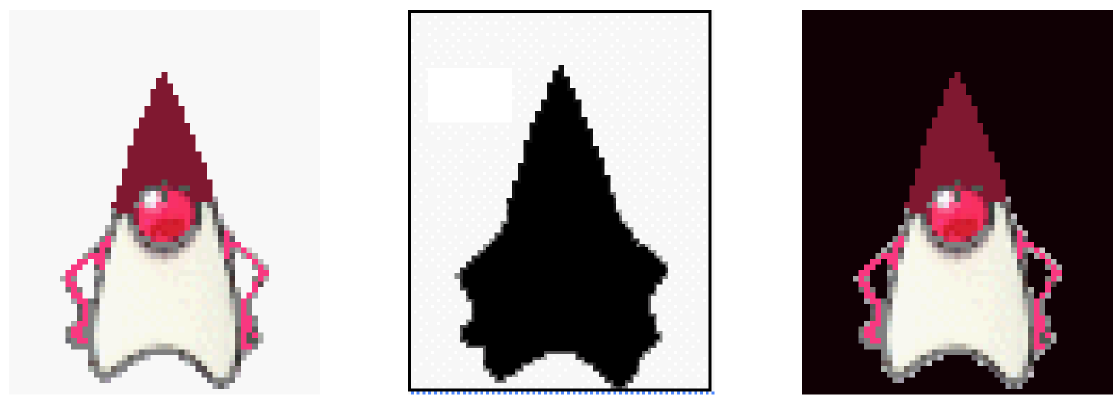


Fig. 2.7: Sprite creation: Original, mask image M, and sprite S (*"Duke" figure courtesy of Sun Microsystems.)*

- 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).

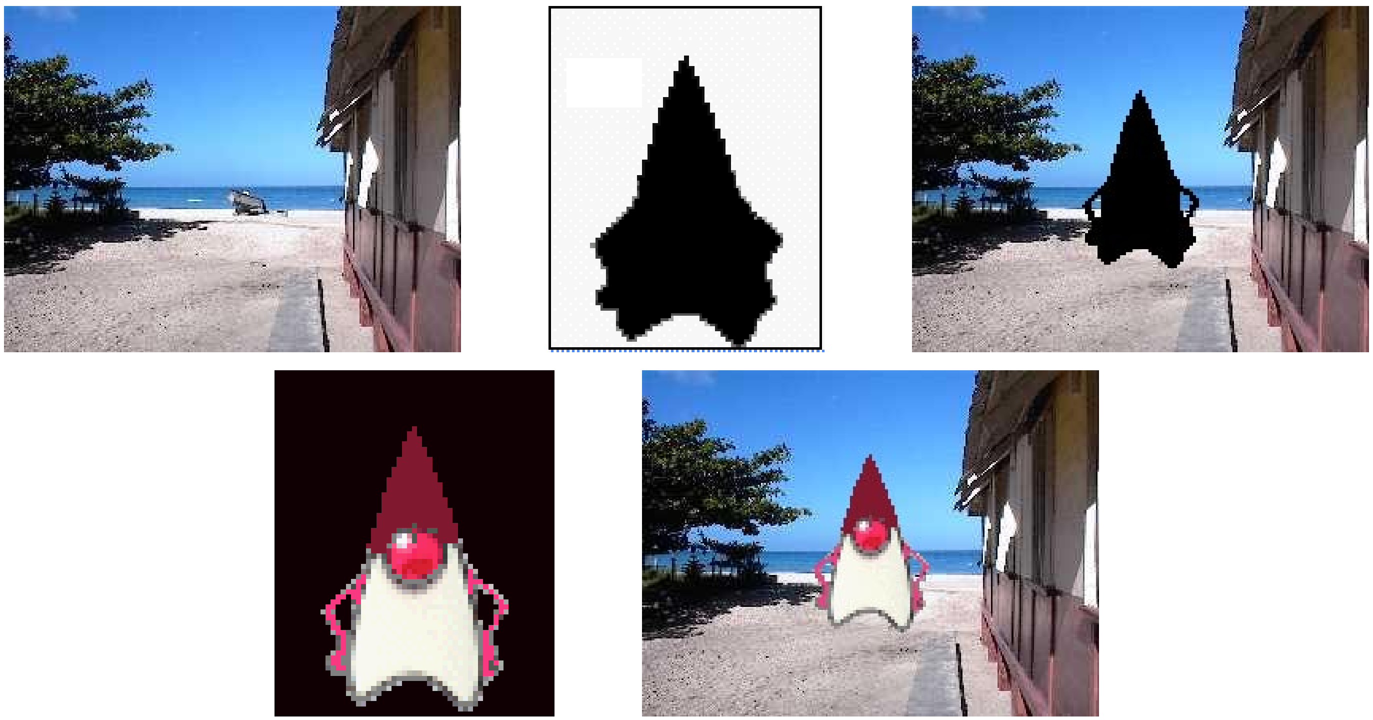


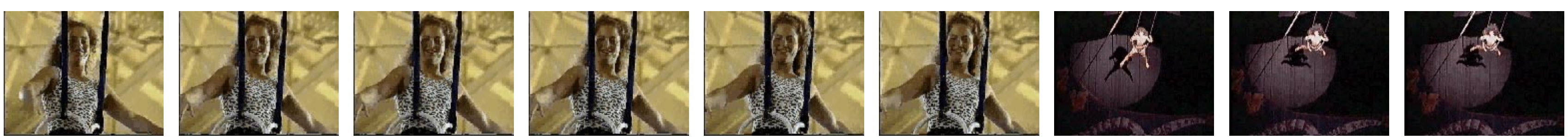
Fig. 2.8: Sprite animation: (a): Background B. (b): Mask M. (c): B AND M. (d): Sprite S. (e): B AND M OR S

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**.

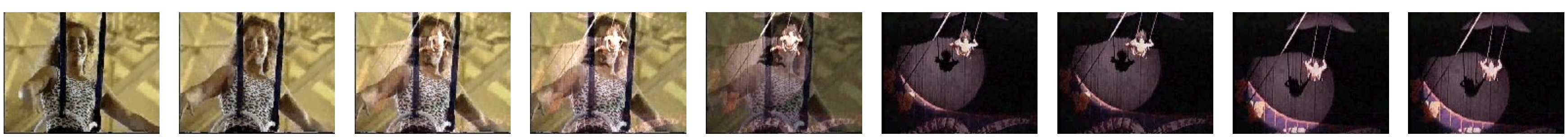
Type I: Cross Dissolve

- Every pixel is affected gradually. It can be defined by:

$$\mathbf{D} = (1 - \alpha(t)) \cdot \mathbf{A} + \alpha(t) \cdot \mathbf{B} \qquad (2.2)$$

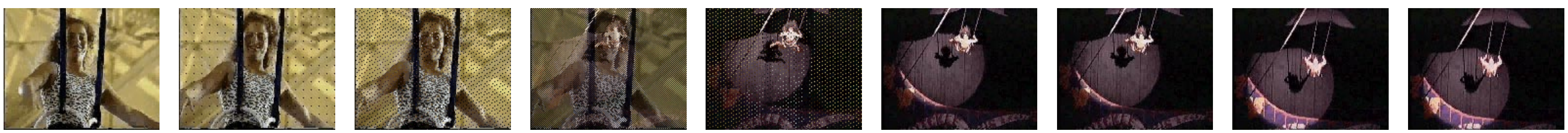
where **A** and **B** are the color 3-vectors for video A and video B. Here, $\alpha(t)$ is a transition function, which is often linear:

$$\alpha(t) = k \cdot t, \qquad \text{with } k \cdot t_{max} \equiv 1 \qquad (2.3)$$



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.

- 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 2.9 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 $x_T = x_{max}$ 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. 2.10 (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$_L$. 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. 2.10(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 $x_{max} - x_T$.

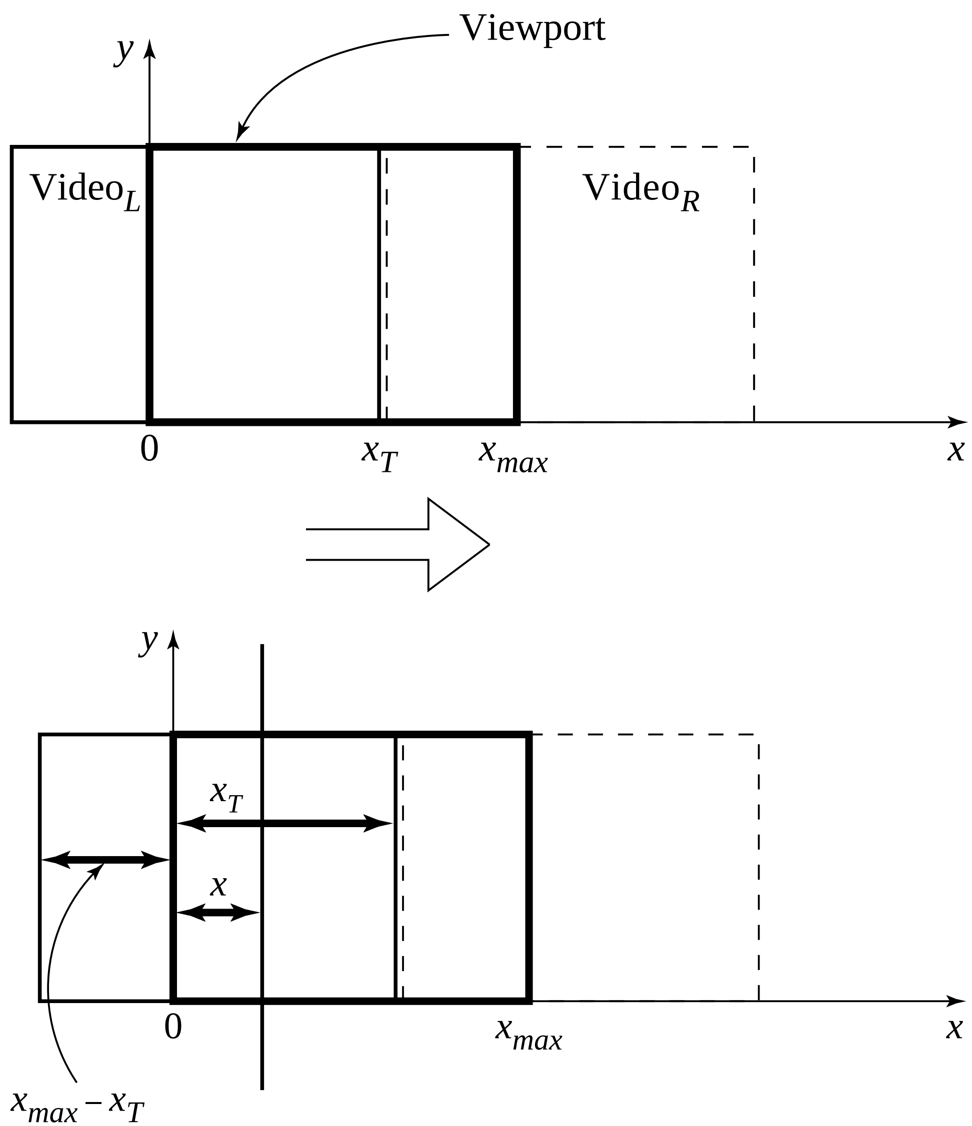


Fig. 2.10: (a): Geometry of Video$_L$ pushing out Video$_R$. (b): Calculating position in Video$_L$ from where pixels are copied to the viewport.

Slide Transition (Cont'd)

- Substituting the fact that the transition moves linearly with time, $x_T = x_{max}(t/t_{max})$, a pseudocode solution in shown in Fig. 2.11.

$$
\begin{array}{l}
\texttt{for } t \texttt{ in 0..} t_{max} \\
\quad \texttt{for } x \texttt{ in 0..} x_{max} \\
\quad\quad \texttt{if } \left(\frac{x}{x_{max}} < \frac{t}{t_{max}} \right) \\
\quad\quad R = R_L \left(x + x_{max} * [1 - \frac{t}{t_{max}}],\ t \right) \\
\quad\quad \texttt{else} \\
\quad\quad R = R_R \left(x - x_{max} * \frac{t}{t_{max}},\ t \right)
\end{array}
$$

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.

– Automatic Authoring

- **Hypermedia documents**: Generally, three steps:

  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.

- **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.

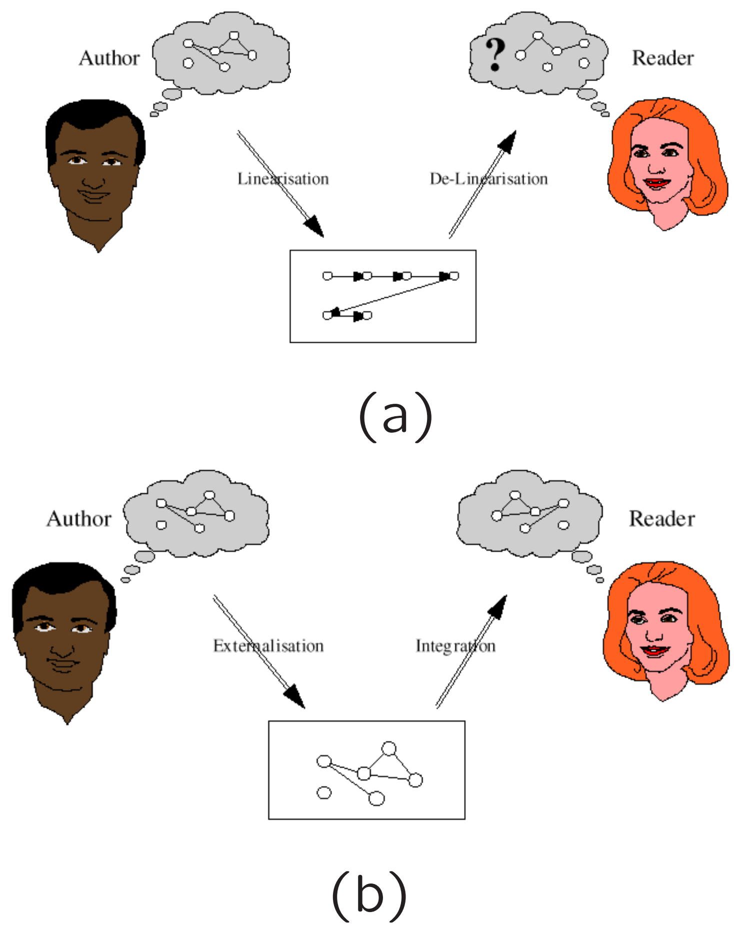


Fig. 2.12: Communication using hyperlinks [from David Lowe].

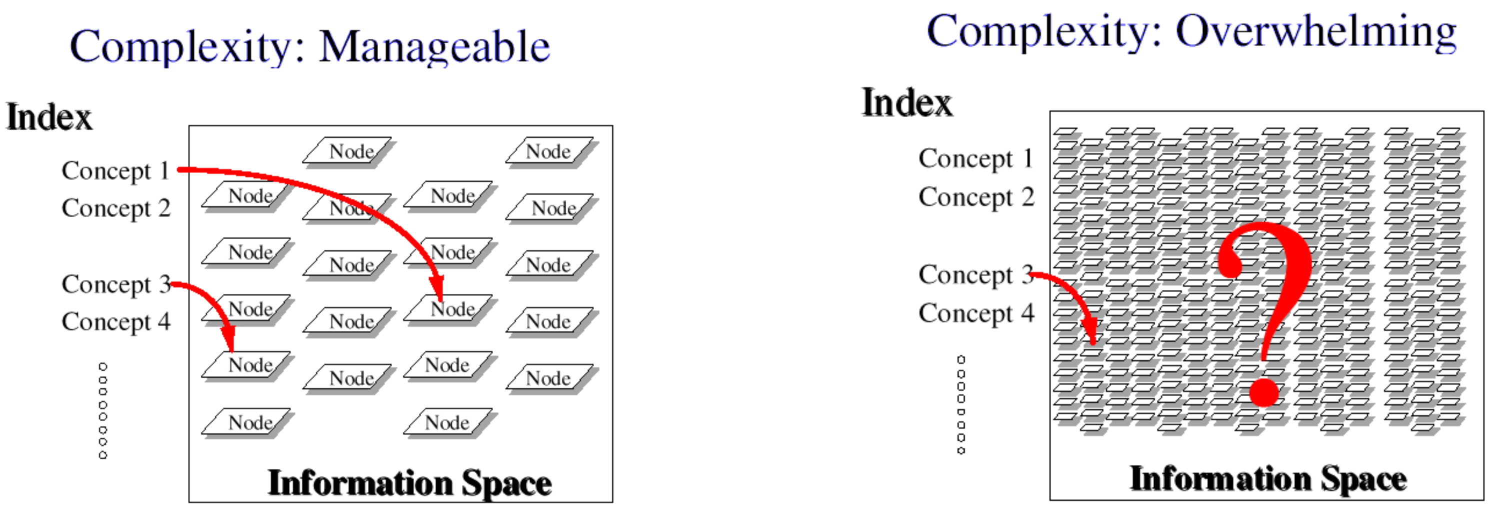


Fig. 2.13: Complex information space [from David Lowe].

(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.

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.

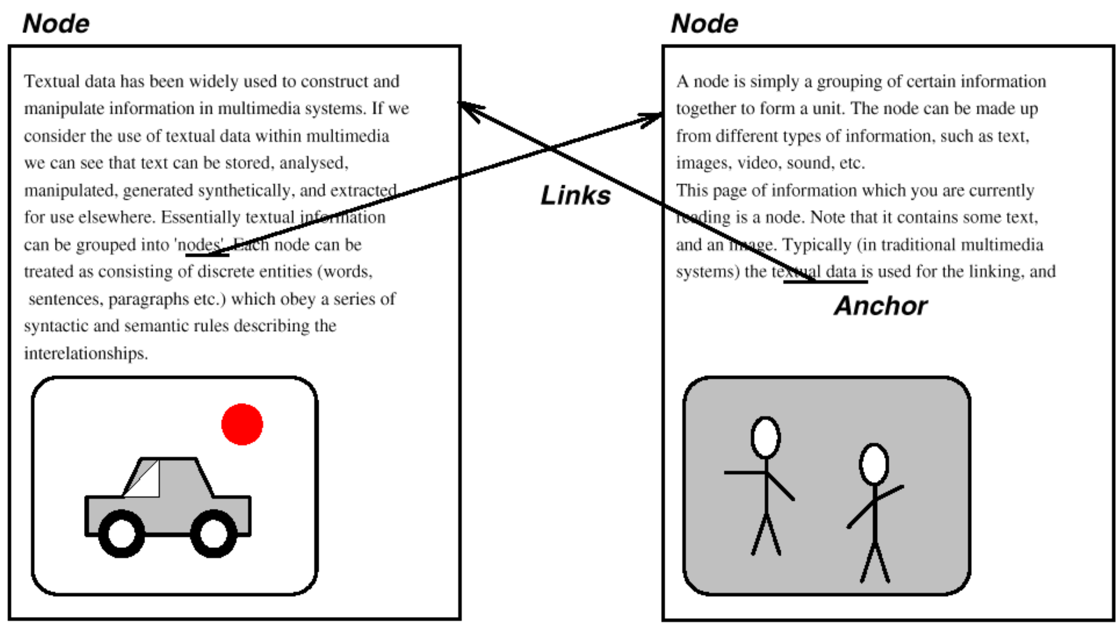


Fig. 2.14: Nodes and anchors in hypertext [from David Lowe].

Hyperimages

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

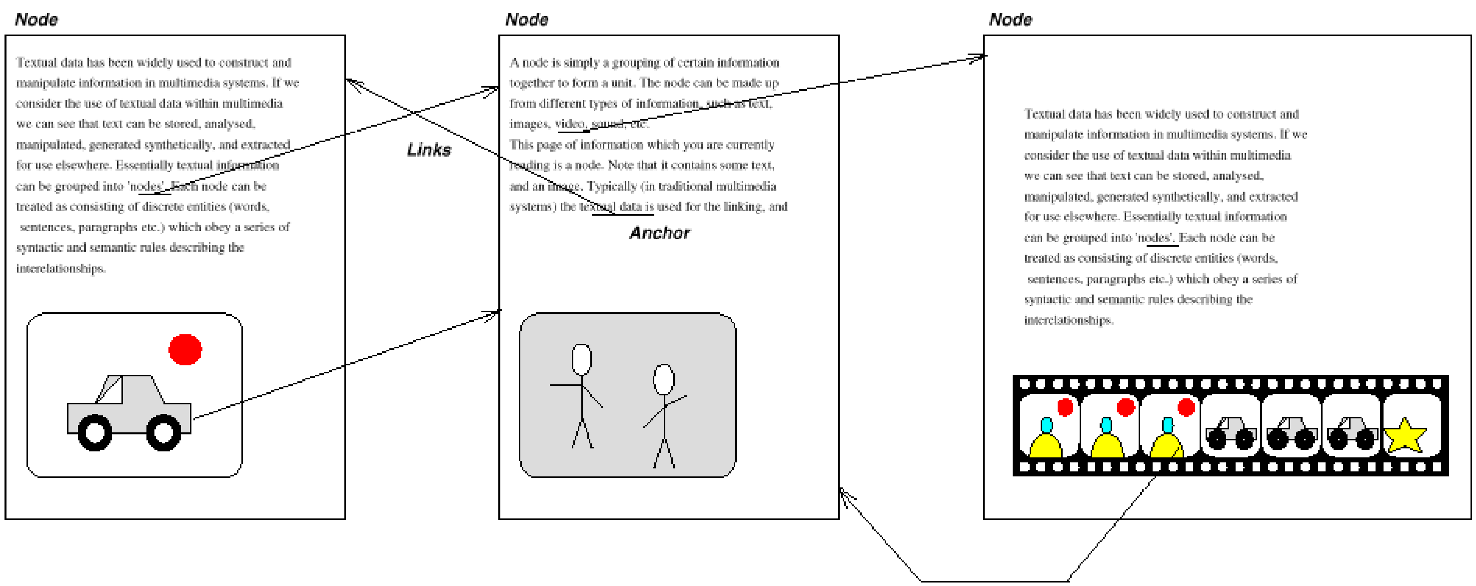


Fig. 2.15: Structure of hypermedia [from David Lowe].

- 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:

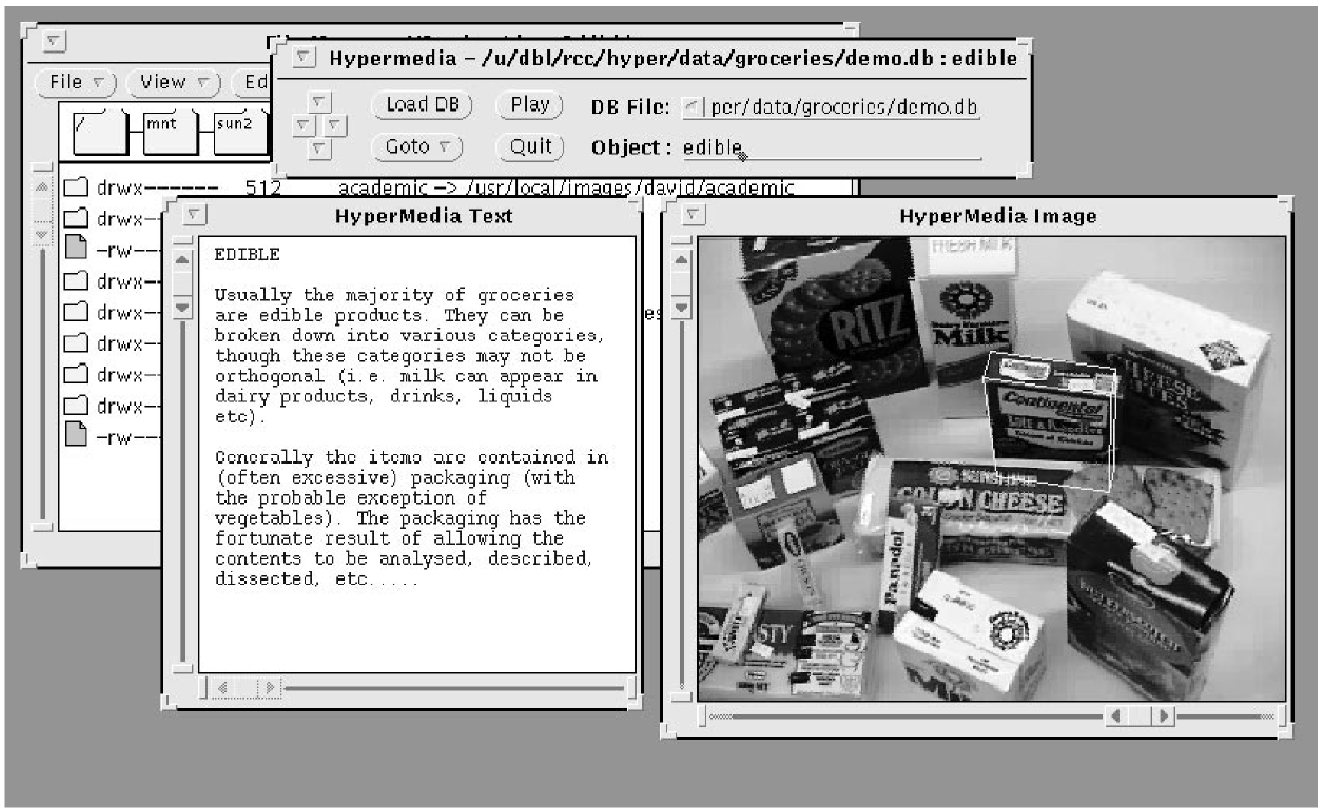


Fig. 2.16: Hyperimage [from David Lowe].

2.2 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 6**
  - **Macromedia Director 8 and MX**
  - **Flash 5 and MX**
  - **Dreamweaver MX**
- **Hint for Studying This Section**: Hands-on work in a Lab environment, with reference to the text.

2.2.1 Adobe Premiere

2.2.2 Macromedia Director

2.2.3 Macromedia Flash

2.2.4 Dreamweaver

Cakewalk Pro Audio

2.3 VRML (Virtual Reality Modelling Language)

Overview

(a) **VRML**: conceived in the first international conference of the World Wide Web as a platform-independent language that would be viewed on the Internet.

(b) **Objective of VRML**: capability to put colored objects into a 3D environment.

(c) VRML is an interpreted language; however it has been very influential since it was the first method available for displaying a 3D world on the World Wide Web.

History of VRML

- VRML 1.0 was created in May of 1995, with a revision for clarification called VRML 1.0C in January of 1996:
  - VRML is based on a subset of the file inventor format created by Silicon Graphics Inc.
  - VRML 1.0 allowed for the creation of many simple 3D objects such as a cube and sphere as well as user-defined polygons. Materials and textures can be specified for objects to make the objects more realistic.

- The last major revision of VRML was VRML 2.0, standardized by ISO as VRML97:
  - This revision added the ability to create an interactive world. VRML 2.0, also called "Moving Worlds", allows for animation and sound in an interactive virtual world.
  - New objects were added to make the creation of virtual worlds easier.
  - Java and Javascript have been included in VRML to allow for interactive objects and user-defined actions.
  - VRML 2.0 was a large change from VRML 1.0 and they are not compatible with each other. However, conversion utilities are available to convert VRML 1.0 to VRML 2.0 automatically.

VRML Shapes

- VRML contains basic geometric shapes that can be combined to create more complex objects. Fig. 2.28 displays some of these shapes:

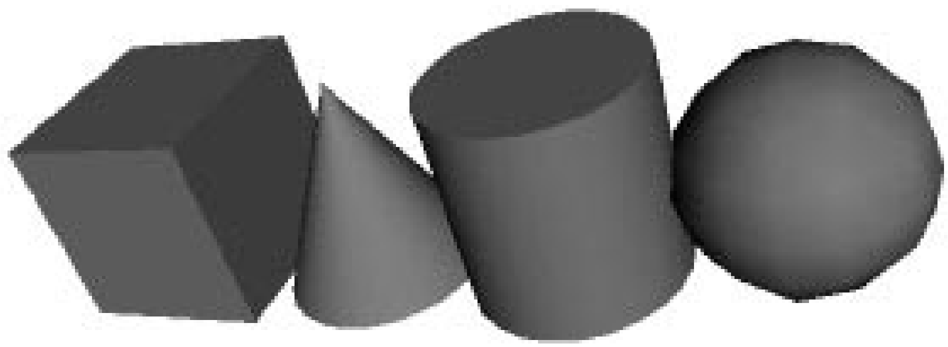


Fig. 2.28: Basic VRML shapes.

  - **Shape node** is a generic node for all objects in VRML.
  - **Material node** specifies the surface properties of an object. It can control what color the object is by specifying the red, green and blue values of the object.

- There are three kinds of texture nodes that can be used to map textures onto any object:

    1. **ImageTexture**: The most common one that can take an external JPEG or PNG image file and map it onto the shape.

    2. **MovieTexture**: allows the mapping of a movie onto an object; can only use MPEG movies.

    3. **PixelTexture**: simply means creating an image to use with ImageTexture within VRML.

VRML world

- Fig. 2.29 displays a simple VRML scene from one viewpoint:
  → Openable-book VRML simple world!:
  ⟶ Link to mmbook/examples/vrml.html.

  – The position of a viewpoint can be specified with the `position` node and it can be rotated from the default view with the `orientation` node.

  – Also the camera's angle for its field of view can be changed from its default 0.78 radians, with the `fieldOfView` node.

  – Changing the field of view can create a telephoto effect.

Fig. 2.29: A simple VRML scene.

- Three types of lighting can be used in a VRML world:
  - **DirectionalLight** node shines a light across the whole world in a certain direction.
  - **PointLight** shines a light from all directions from a certain point in space.
  - **SpotLight** shines a light in a certain direction from a point.
  - **RenderMan**: rendering package created by Pixar.
- The **background** of the VRML world can also be specified using the `Background` node.
- A **Panorama** node can map a texture to the sides of the world. A panorama is mapped onto a large cube surrounding the VRML world.

Animation and Interactions

- The only method of animation in VRML is by tweening — done by slowly changing an object that is specified in an interpolator node.

- This node will modify an object over time, based on the six types of interpolators: color, coordinate, normal, orientation, position, and scalar.

  (a) All interpolators have two nodes that must be specified: the **key** and **keyValue**.

  (b) The **key** consists of a list of two or more numbers starting with 0 and ending with 1, defines how far along the animation is.

  (c) Each key element must be complemented with a `keyValue` element: defines what values should change.

- To time an animation, a **TimeSensor** node should be used:

  (a) **TimeSensor** has no physical form in the VRML world and just keeps time.

  (b) To notify an interpolator of a time change, a `ROUTE` is needed to connect two nodes together.

  (c) Most animation can be accomplished through the method of routing a `TimeSensor` to an interpolator node, and then the interpolator node to the object to be animated.

- Two categories of sensors can be used in VRML to obtain input from a user:

  (a) **Environment sensors**: three kinds of environmental sensor nodes: `VisibilitySensor`, `ProximitySensor`, and `Collision`.

  (b) **Pointing device sensors**: touch sensor and drag sensors.

VRML Specifics

- Some VRML Specifics:

  (a) A VRML file is simply a text file with a "`.wrl`" extension.

  (b) VRML97 needs to include the line `#VRML V2.0 UTF8` in the first line of the VRML file — tells the VRML client what version of VRML to use.

  (c) VRML nodes are case sensitive and are usually built in a hierarchical manner.

  (d) All Nodes begin with "{" and end with "}" and most can contain nodes inside of nodes.

  (e) Special nodes called group nodes can cluster together multiple nodes and use the keyword "children" followed by "[...]".

(f) Nodes can be named using **DEF** and be used again later by using the keyword **USE**. This allows for the creation of complex objects using many simple objects.

- A simple VRML example to create a box in VRML: one can accomplish this by typing:

```
Shape {
       Geometry Box{}
      }
```

The Box defaults to a 2-meter long cube in the center of the screen. Putting it into a **Transform** node can move this box to a different part of the scene. We can also give the box a different color, such as red.

```
Transform { translation 0 10 0 children [
  Shape {
    Geometry Box{}
    appearance Appearance {
      material Material {
        diffuseColor 1 0 0
      }
    }
  }
]}
```

2.4 Further Exploration

⟶ Link to Further Exploration for Chapter 2.

- Good general references for multimedia authoring are introductory books [3,1] and Chapters 5-8 in [4].
- A link to the overall, and very useful, FAQ file for multimedia authoring is in the textbook website's "Further Exploration" section for Chapter 2.
- A link to a good FAQ collection for Director, plus a simple Director movie:

  ⟶ Link to mmbook/examples/director.html.