Drag & Drop Scripting: How To Do Hypermedia Right

F. Hanisch and W. Straßer

WSI/GRIS University of Tübingen
{hanisch,strasser}gris.uni-tuebingen.de

Abstract

The rising interest in repositories for educational material consolidates efforts of the diversified educational community. Developers, teachers, and designers have recognized the need for collaboration in order to create the best-possible learning objects, and, moreover, to preserve and reuse them. Developing learning objects in the field of Computer Graphics is archetypical in several aspects; we naturally face the needs for complex visualizations, we bring along profound knowledge in human-computer interaction, and we are familiar with the underlying technology.

Nevertheless, interactive visualizations are supported little in current learning technology standards, especially with regards to metadata and interoperability. Standards enable educators to browse and search repositories, integrate the object of preference into their curriculum, and adapt it to their needs. To raise sympathies for interactivity and clarify its denotation, we identify three approaches: (1) the developer's view of interactivity as user interface characteristics, (2) the educator's view of interaction between internal and external knowledge representation, and (3) the communication theorist's view that provides a qualitative framework based on learning theories. We reformulate the results in terms of Computer Graphics principles and illustrate the impact of a consequent implementation of great interactivity in Web-based teaching with components of our own courses. Most notably, we propose a visual scripting paradigm that communicates not only a learning object's data, but functionality, through Drag & Drop operations on images.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [User Interfaces]: Interaction styles K.3.2 [Computers And Education]: Computer and Information Science Education I.3.6 [Computer Graphics]: Interaction techniques

1. Introduction

Web-based teaching is an interdisciplinary field. According to people's background, the denotation of terminology varies heavily. In the following, we sharpen the term interactivity in Web-based teaching, which has become an often-cited panacea for education[1] – all the more, when combined with hypermedia. In many cases, we find interactivity trivialized to menu selection, clickable objects, or linear sequencing[2]. We want to clarify the denotation and relevance of interactivity in learning by emphasizing sucessively the physical, perceptual-cognitive, and communication-theoretic points of view (see Section 2).

Section 2.1 starts with characterizing interactivity as ingredients of a graphical user interface (GUI). We contrast qualities of today's representatives of hypermedia and user interfaces — the Web and the desktop — and analyze the gap between. While design principles such as direct manipulation, building blocks, and Model View Controller (MVC) became integral part of the system, others, like e.g. programming and linking, got lost. As this is a rather developer-centered point of view, Section 2.2 deals with didactical aspects of interactivity. In exploring interactivity as perceptual or cognitive process between internal and external representation of information (integration and construction of knowledge) we result in design concepts for interactivity.

The growing demand for reusable educational material, so-called learning objects[3] or sharable content objects[4], has lead to learning technology standards such as SCORM[4], which puts together proposals of IEEE LOM[3], ARIADNE[5], IMS[6], and others. These standards specify learning object metadata and interoperability in order to enable educators searching/browsing repositories or digital libraries[7]. Learning management systems could further launch and track learning objects and software components, and adapt to the user's activity.

Although, current taxonomies of interactivity will result in subjective impressions only that rule out any international understanding[8]. SCORM differentiates the interactivity type between learning-by-doing (active) and learning-by-reading (expositive). For instance, while simulations, questionnaires, or exercises are active, video clips or hypertext are regarded as expositive (browsing is seen as navigation, not as interactivity, see Section 2.3). The interactivity level is denoted within an ordinal range from very low to very high. Currently, SCORM does not assign any characteristics to these ranges, neither physical nor cognitive activities. Therefore, Section 2.3 reviews a more suitable qualitative framework for modelling interactivity.

Subsequently, we propose a so-called MVC Interactivity (see Section 3.1) that reformulates the major concepts of interactivity in terms of Computer Graphics principles, respectively in terms of Model View Controller. We break down MVC's model into parameters and internal functionality (structure and components). In our notion, a highly interactive learning object allows for directly manipulating its view, parameters, and functionality.

As a major step towards direct manipulation of interactive Web content we introduce a visual scripting mechanism, Drag & Drop scripting, that communicates parameters and functionality beyond the browser barrier – between other hypermedia objects or native applications (see Section 3.2). Scripting instructions are encrypted into standard images. Our prototypes demonstrate that great interactivity[9] may increase not only a learning object's usability, but also didactical value.

2. Interactivity

2.1. GUI Characteristics

Remember the first interactive computer graphics. Exactly 40 years ago, Ivan Sutherland presented his Sketchpad system[10]. Using a lightpen and a 40-button command box, he could create, manipulate, duplicate, and store engineering drawings directly on the display. Not only that he could zoom or scroll the drawing area, he could also apply constraints such as orthogonal lines. Think also of Douglas Engelbart's NLS system[11] in the 1960s and his notions of connectivity and multiple views of information. How far have we gone since then with interactivity in Web-based teaching?

Interactivity between humans and computers has become well-investigated in the field of Human-Computer-Interaction (HCI[12]). As we all know, graphical user interfaces (GUI) and hypermedia have the same origins[13]. Although, while GUI design has evolved into the desktop metaphor with its WIMP (windows, icons, menus, pointer device) and direct manipulation design, Web technology stayed inside the limits of a browsing metaphor. Today, the Web community tries to bridge the gap and realize the vision of a medium that focuses on structure, multimedia, collaboration, and personalization, - as it was thought by hypertext's pioneers[11]. Interactivity in Web-based education is mainly achieved by the use of Java, Macromedia Flash, or basic (D)HTML (forms, image maps, scripts, et al).

Following Ben Shneiderman, direct manipulation[14, 15] inheres a (1) continuous representation of the objects and actions of interest, (2) physical actions or button presses instead of complex syntax, and (3) rapid incremental reversible operations whose effect on the object of interest is immediately visible. Direct manipulation lowers initial hurdles for novices as well as it enables experts to work more efficiently. Users get immediate feedback and gain confidence and mastery, as they initiate and control actions and may predict system responses. The most prominent representative of direct manipulation is Drag & Drop (DnD), basically a shortcut for Copy & Paste. DnD is supported by all major platforms, e.g. OLE (Win32) DnD, CDE/Motif dynamic protocol, MacOS, OS/2, and JavaOS/Java.

Facing the request for a better adaptability (or extensionality), we advance from the DnD gesture layer to the programming layer. Alan Kay envisions the computer as a personal, dynamic medium. The benefits of computer technology for facilitating learning are, in his words[9], at first a "great interactivity", next, the hypermedia aspect of integrating all multimedia and representing information alternatively, and, last but not least, the capability of expressing and simulating dynamic models of ideas. Together with Daniell Ingall, he created Smalltalk[16]. Smalltalk, as a software architecture that has its core based on object-oriented programming with an uniform message system, enables users to interact with all aspects of the system. Objects can be adapted on system level and interlinked system-widely. Today, Java represents Kay's idea of a common programming platform.

Similar, objects could be interlinked as system service. Norman Meyrowitz, developer of Intermedia[11] at Brown and Shockwave (and today's president of Macromedia), presented a Start & Complete Link desktop metaphor[17] that acts exactly like Copy & Paste. Intermedia demonstrated impressively the use of a link database, e.g. back-linking, or pointing to any internal component of any multimedia object. Meyrowith realized that monolithic systems (including Smalltalk) demands users disown their present computing environment to use hypermedia functionality. In 1989, he proposed:

> Linking functionality must be incorporated, as a fundamental advance in application integration, into the heart of the standard computing toolboxes [...] and application developers must be provided with the tools that enable applications to 'link up'

> in a standard manner. Only when the paradigm is positioned as an integrating factor for all third-party applications, and not as a special attribute of a limited few, will knowledge workers accept and integrate hypertext and hypermedia into their daily work process.

Evolution has turned into just the opposite. We are faced today with insular, application-centered operation systems; at best, applications provide restricted and non-conform scripting or macro functionality. Desktop linking refers not to components, but documents. The Web environment exists only inside of a browser window. Interactive Web content is restricted either by security policies, or simply by the needs for staying compatible with browser diversity. We will discuss in Section 3.2 to what extend this gap can be bridged.

Other design principles behind Smalltalk became common practice. One of these is the use of "building blocks"[16]. Developing a larger number of interactive learning objects is time-consuming and expensive; therefore, the idea is to spend inital efforts on toolkits and reusable software components[18]. David Canfield Smith's thesis[19] at Xerox PARC introduced an appropriate visual programming paradigm that allows for connecting components by direct manipulation. The Java Swing package contains about 40 GUI components, including the whole range from buttons, menus, input fields, lists, to more advanced components such as WYSIWYG (what you see is what you get) styled text or HTML editors.

Another principle is a separation of an object's functionality, visual representation, and interactivity. Known today as Model View Controller (MVC[20]), it is still the design pattern of choice for interactive Java applications. More complex graphical scenes are represented by scene graph components. Until today, there is no built-in interaction mode for 2D/3D graphical scenes, neither in Java2D, nor in Java3D. Typically, the system (e.g. OpenGL) only provides picking support and developers build their own toolkits to support interaction modes such as zoom, pan, rotate, walk, or select. Based on such toolkits, we have implemented about 100 educational Java applets in the field of Computer Graphics[21] and Visualization[22].

Software components and composite learning objects come with several kinds of interactivity. Rod Sims[2] identifies the following ingredients: object interactivity (object activation via mouse clicks), linear interactivity (forward/backward movements through a predetermined linear sequence), hierarchical interactivity, support interactivity (general or context-sensitive help), update interactivity (dynamic responses, feedback), construct interactivity (manipulation of components), reflective interactivity, simulation interactivity, hyperlinked interactivity (traveling through a knowledge base), non-immersive contextual interactivity (microworlds), and immersive virtual interactivity (complete virtual worlds).

Some components can be directly mapped to these classes. Learning objects, as they are composites of components, will usually represent a combination of several kinds of interactivity. Statements about didactics or quality are, at best, subjective. We will incorporate these aspects in the next sections.

2.2. Perception And Cognition

Direct manipulation and the desktop metaphor intend to provide an intuitive interactivity, that is, to model familiar instances of everyday life. The central idea is to bridge the gap between abstraction and reality. Aldrich, Rogers, and Scaife[1] argue that we need to analyze the interactions between internal and external representations of information. Interactivity appears as perceptual or cognitive process when users utilize, adapt, or construct an external representation in a given activity. More precisely, interactions occur (1) from external to internal representation and (2) from internal to external representation.

In exploring the first direction, which describes the integration of information, we find out how to structure multimedia efficiently in order to "convey the appropriate kind, level, and abstraction of knowledge for a given domain"[23]. Different kinds of media and interactivity could be used in parallel to allow for a more effective way of understanding concepts. All representations should be dynamically interlinked to visualize the relationships between them. Learners should be allowed to modify (correct or incorrect) elements in any representation; resulting effects should be displayed simultaneously in all other representations.

The inverse cognitive process, construction of external representations, refers to classic annotation and re-representation methods such as highlighting objects and making notes or sketches. Having a better understanding on how to create content will enable users to have a better understanding of how the system works. Moreover, it enables learners to develop their understanding of the content by making changes to it for their own purposes. Therefore, a learning object should require users to work with it in order to refine their mental model of its function and structure.

Rogers and Scaife[23] describe cognitive properties of external representations and develop guidelines for different audiences such as developers, educators, or parents. They identified design concepts such as explicitness and visibility (how to direct learner's attention to key components, e.g. make visible what are normally "hidden" processes), cognitive tracing (how to allow users to manipulate and annotate dynamic representations), ease of production (how easy is it for users to create external representations), and combinability and modifiability (how to enable both system and users to combine different kinds of representations).

Several authors have concretized these concepts into usability guidelines. Although, we believe that usefulness of a learning object is mainly determined by the developer's or educator's mastery. To cultivate such competence, the presented catalog of questions appears to be more adequate than merely a list of bits and pieces.

2.3. A Qualitative Framework

In separating the physical aspects of interactivity from their symbolical meaning, we move towards a communication-theoretic point of view. The nature of interaction is described by methodologies based on theories of learning. In such a context, interactivity describes the learning process that occurs while modifying learning objects. Hyperlinked interactivity as described in Section 2.1 symbolizes no longer interactivity, but mere navigation.

Rolf Schulmeister depicts learning by direct manipulation (physical) as learning by constructing[24] (symbolical). As response to current standardization efforts, he proposes a qualitative framework[8] for modelling interactivity of multimedia. His taxonomy provides six ascending degrees of interactivity that can be directly mapped to SCORM's interactivity levels. With ascending degree, the related theory of learning alters from behaviourism to instructionalism to constructivism. The forth level corresponds to learning by discovering, the fifth level to learning by construction.

Level one represents no interactivity. The user observes a multimedia object passively (image, video, sound, automated program) and performs necessary actions (start, stop). In the second level, interactions have only illustrative or informational character; the user may choose from a set of options and contemplate temporal (slow motion, step-wise) or spatial (point of view) versions of multimedia objects, for example a point & click slide show, or alternative data lists.

The third degree includes modifications of an object's visualization, but not content (e.g. pan, zoom, rotate graphical scenes or single geometric objects). Modification of an object's content leads to the forth level: content is no more pre-prepared, but generated as response to the user. The user may create different visualizations or visualize different relations by varying parameters. Interactions occur with cognitive concepts.

Level five covers the construction of new objects or design of models and processes – the user constructs his own microworld. Finally, the last level includes feedback, which consists of intelligent responses according to the user's actions.

According to SCORM's scaling, each level includes all aspects of the lower levels. Schulmeister implicitly assumes that each level is designed properly by the developers. In practice, such an ascending order occurs rarely; a learning object as we know it typically includes a mixture of some ingredients. Especially, feedback seems to be a long-winded term. Nevertheless, Schulmeister's proposition provides a far more elaborated taxonomy than current draft standards.

3. MVC Interactivity

3.1. Definition

We now combine the major concepts of interactivity and reformulate the terminology of Section 2 in terms of Computer Graphics principles.

The Model View Controller paradigm basically meets technological, didactical, and cognitive demands. MVC separates interface issues (controller, e.g. direct manipulation) from an object's visual appearance (view) and its state or functionality (model). Although, by their symbolical meaning, parameter interactivity differs from interactivity on structure/model layer. Therefore, we break down MVC's model into parameters and internal functionality; while the first one represents an object's state, the latter one holds the underlying structure and components.

We consider an object as 'highly interactive learning object', if it provides means for

1. representation of domain knowledge that may induce a learning process
2. illustrative actions
3. direct manipulation of object view
4. direct manipulation of all essential object parameters
5. direct manipulation of object functionality (structure, components)
6. adequate feedback and help

We have integrated all levels of Schulmeister's taxonomy, required direct manipulation in all aspects, and explicitly ask for a proper design of interactivity with respect to the range of interactive parameters. Note that we strive for great interactivity only – other degrees of interactivity can be derived. The first item gives a formal definition of a learning object.

We imply that a highly interactive learning object visualizes complicated topics and relationships graphically and allows for direct manipulation of all parameters that belong to the topic's core. Learners can modify internal components and get visual feedback or help, wherever needed. Modifying the model may require visual scripting (see below) or visual programming.

3.2. Visual Scripting

In Web-based teaching, browser plug-in technology prevents nearly all interoperability between hypermedia objects. Usually, a learning object that we embed into a Web page behaves as black box; we can hardly access its components, modify intercomponent issues or even link to them. The only established communication model between plug-in content and HTML environment is scripting (e.g. in JavaScript, VBScript, AppleScript).

Scripting instructions are embedded into HTML, and we trigger them e.g. by activating hyperlinks. Although, it is the browser application that executes scripts, and not the system. Typically, the script's scope is restricted to the current browser document, or to its siblings. We can not script local learning objects from Web pages, and, vice versa, scripts located outside the browser application (e.g. within a word processor, or a presentation program) will not work with Web content. More general, only hypermedia objects belonging to the same context may communicate by scripting.

We therefore introduce a visual scripting mechanism that we call Drag & Drop scripting. As discussed in Section 2.1, DnD represents a platform-independent direct manipulation paradigm that operates beyond application boundaries. It is also part of the user's familiar desktop environment. Nearly all of the Java Swing components support DnD natively, others have to implement a minimal DnD API[25]. We transfer all scripting functionality to the DnD action's source and destination object; that way, our approach works even if the user has disabled browser scripting functionality.

The basic idea of Drag & Drop scripting is as follows: (1) source out a learning object's functionality into scripting operations, (2) encrypt a script in an image that illustrates the result of the script, (3) permit the user to drag and drop the image onto the learning object, and (4) decrypt the scripting operation inside the learning object and execute it.

We use a framework that organizes scripting instructions within a script database and generates Web pages using templates[26]. Illustrating images are part of our script metadata. The template already converts scripts into a corresponding HTML sequence, creates a thumbnail, and embeds it into the final Web page. In addition, the script template now hides the script in the image. Currently, we perform a least significant bit (LSB) insertion that works only with lossless image formats. In fact, each pixel stores 2 bits of our data. In the case of 8-bit images, which are less forgiving to LSB manipulation, we simply colorize the pixels in the Web page's background color. An improved version would use watermarking or a steganographic system[27], and support JPEG images. We start with a header (magic number, learning object identifier, border color, et al.) that enables the learning object to identify script images, validate the drop action, or deny it. The subsequent data block contains the script.

Next, the learning object has to recognize DnD actions. Any of its GUI components that may receive a drop action must implement the DnD API and delegate work to our script interpreter (which is a member of our learning object's base class). We have done prototyping with central components (e.g. image browser, image viewer, video player), which cover the bigger part of our learning objects for Image Processing and Video Processing.

Figure 1 demonstrates a Java applet that teaches basics of color spaces in Video Processing, respectively RGB and YUV color spaces. It is available online at http://www.gris.uni-tuebingen.de/projects/vis, together with

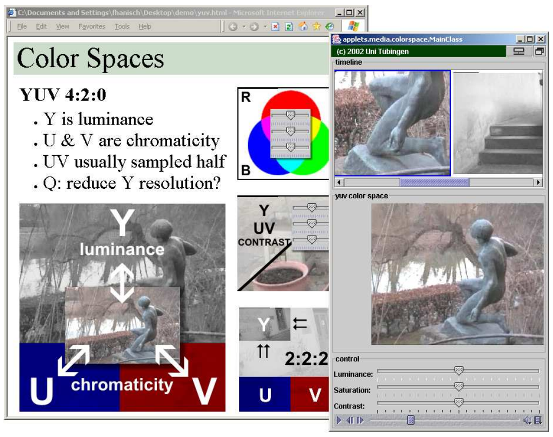

Figure 1: *This Java applet illustrates color spaces in Video Processing. A Web page provides theory and scripts (embedded into images) that may be operated on the applet via Drag & Drop. Video frames may be dropped to the timeline or any other location, including native applications.*

a video preview. We have provided scripts and images that modify parameters of the video renderer (to visualize YUV channels separated, or combined), rearrange the applet's layout (to insert controls for lightness/saturation/contrast or the amount of red/green/blue), and apply other YUV formats. The script action is performed by physically operating the corresponding image on the Java applet. For example, a course slide (Figure 1, left side) might enquire the learner about the effect of reducing not only color information (U and V), but also luminance (Y). By dragging the accompanied image (bottom center) into the applet (right side), the learner may experience in a simulation that the eye is more sensitive to luminance detail than color detail.

When the user starts a dragging action in our video renderer, we extract the current video frame and embed a script sequence that will prompt the video player to reposition the video stream according to the frame's timestamp. The user can drag the frame to the applet's timeline (Figure 1, top right) or any other application that supports DnD. Note that applications that resample DnD images (e.g. Microsoft Powerpoint) will distort a primitive LSB encryption.

4. Conclusions

Based on three complementary approaches to interactivity we provided a useful definition for great interactivity that includes technological, didactical, and cognitive aspects. In our notion, highly interactive learning objects visualize complicated topics and relationships graphically and permits direct manipulation of all essential parameters and functionality (structure and components). We therefore proposed a simple Drag & Drop metaphor that allows for interoperability beyond the browser barrier.

References

1. F. Aldrich, Y. Rogers, and M. Scaife. Getting to grips with 'interactivity': helping teachers evaluate the educational value of CD-ROMs. *British Journal of Educational Technology*, **29**(4):245-261, July 1990. 1, 3

2. R. Sims. Interactivity: A Forgotten Art? *ITFORUM*. 10, November 1995. 1, 3

3. IEEE Learning Technology Standards Committee (LTSC), Learning Object Metadata Working Group. Draft Standard for Learning Object Metadata (IEEE 1484.12.1-2002). *http://ltsc.ieee.org/wg12*, July 2002. 1

4. Advanced Distributed Learning (ADL). Sharable Content Object Reference Model (SCORM), Version 2, The SCORM Content Aggretation Model. *http://www.adlnet.org*, October 2001. 1

5. ARIADNE Foundation. ARIADNE Educational Metadata Recommendation - V3.2. *http://www.ariadne-eu.org*, February 2002. 1

6. IMS Global Learning Consortium, Inc. IMS Content Packaging Specification, Version 1.1.2. *http://www.imsproject.org/content/packaging*, August 2001. 1

7. S.G. Owen, R. Sunderraman, Y. Zhang. The development of a digital library to support the teaching of computer graphics and visualization. *Computer & Graphics* **24**(4):623-627, 2000. 1

8. R. Schulmeister. *Lernplattformen für das virtuelle Lernen: Evaluation und Didaktik*, Oldenbourg, München, 2003. 2, 4

9. A. Kay. Computers, Networks & Education. *Scientific American Magazine*, September 1991. 2

10. I.E. Sutherland. Sketchpad - A Man-Machine Graphical Communication System. *AFIPS Spring Joint Computer Conference (SJCC '63)*, pp. 329-346, 1963. 2

11. A. van Dam. Hypertext '87 Keynote Address. *Communications of the ACM*, **31**(7):887-895, 1988. 2

12. T. Hewett, R. Baecker, S. Card, T. Carey, J. Gasen, M. Mantei, G. Perlman, G. Strong, and W. Verplank. ACM SIGCHI Curricula for Human-Computer Interaction *Report of the ACM SIGCHI Curriculum Development Group*, ACM, 1992. 2

13. M. Müller-Prove. *Vision and Reality of Hypertext and Graphical User Interfaces*, Department of Informatics, University of Hamburg, February 2002. 2

14. B. Shneiderman. *Designing the User Interface*, Addison-Wesley, Reading, MA, 1992. 2

15. B. Shneiderman. Direct manipulation for comprehensible, predictable, and controllable user interfaces. *Proceedings of the ACM International Workshop on Intelligent User Interfaces '97*, pp. 33-39, New York, 1997. 2

16. D.H.H. Ingalls. Design Principles Behind Smalltalk. *BYTE Magazine*, August 1981. 2, 3

17. N.K. Meyrowitz. The Missing Link: Why We're All Doing Hypertext Wrong. *The society of text: Hypertext, hypermedia and the social construction of information*, pp. 107-114, MIT Press, 1989. 2

18. J.R. Laleuf, A.M. Spalter. A component repository for learning objects: a progress report. *Proceedings of the first ACM/IEEE-CS joint conference on Digital libraries.* pp. 33-40, ACM Press, 2001. 3

19. D.C. Smith. *Pygmalion: A Computer Program to Model and Simulate Creative Thought*, Basel and Stuttgart, Birkhauser Verlag, 1977. 3

20. G.E. Krasner, and S.T. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 system *Journal of Object Oriented Programming*, **1**(3), pp. 26-49, 1988. 3

21. R. Klein, F. Hanisch, W. Straßer. Web-based Teaching of Computer Graphics: Concepts and Realization of an Interactive Online Course. *SIGGRAPH 98 Conference Proceedings*, 1998. 3

22. F. Hanisch, and W. Straßer. Highly Interactive Web-based Courseware. *Proc. of CGE02 - Eurographics/SIGGRAPH Workshop on Computer Graphics Education*, pp. 31-35, 2002. 3

23. Y. Rogers, M. Scaife. How can interactive multimedia facilitate learning? *Intelligence and Multimodality in Multimedia Interfaces: Research and Applications*, AAAI, Menlo Park, CA, 1998. 3

24. R. Schulmeister. *Hypermedia Learning Systems: Theory - Didactics - Design.* http://www.izhd.uni-hamburg.de/paginae/Book/Frames/Start_FRAME.html. English online version of "Grundlagen hypermedialer Lernsysteme", 2nd ed. Oldenbourg, München, 1997. 4

25. Sun Microsystems, Inc. Proposal for a Drag and Drop subsystem for the Java Foundation Classes. *http://www.java.sun.com/products/javabeans/glasgow*, Final Draft v0.96, Aug. 24, 1998. 5

26. F. Hanisch. Authoring and Linking of Highly Interactive Content within Web-based Courseware. *Networked Learning in a Global Environment: Challenges and Solutions for Virtual Education (NL 2002)*, Technical University of Berlin, 2002. 5

27. N.F. Johnson, and S. Jajodia. Exploring Steganography: Seeing the Unseen. *IEEE Computer*, **31**(2):26-34, February 1998. 5